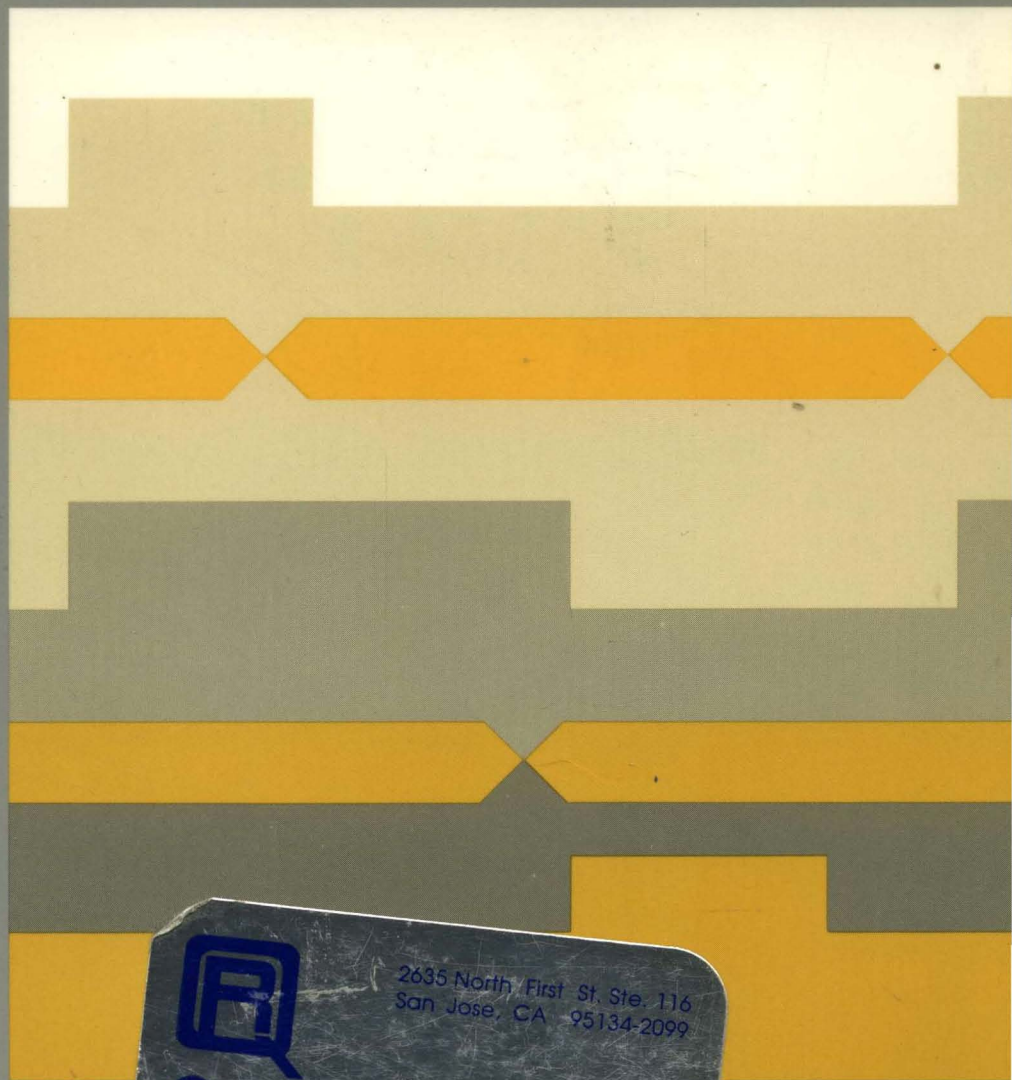
March, 1988

HITACHI®

HMCS400® SERIES HANDBOOK

- USER'S MANUAL
- SOFTWARE APPLICATION NOTES
- HARDWARE APPLICATION NOTES



#U01

HMCS400 SERIES HANDBOOK

- User's Manual
- Software Application Notes
- Hardware Application Notes

HITACHI®

AD-E00078

MEDICAL APPLICATIONS

Hitachi's products are not authorized for use in MEDICAL APPLICATIONS, including, but not limited to, use in life support devices without the written consent of the appropriate officer of Hitachi's sales company. Buyers of Hitachi's products are requested to notify Hitachi's sales offices when planning to use the products in MEDICAL APPLICATIONS.

When using this manual, the reader should keep the following in mind:

1. This manual may, wholly or partially, be subject to change without notice.
2. All rights reserved: No one is permitted to reproduce or duplicate, in any form, the whole or part of this manual without Hitachi's permission.
3. Hitachi will not be responsible for any damage to the user that may result from accidents or any other reasons during operation of his unit according to this manual.
4. This manual neither ensures the enforcement of any industrial properties or other rights, nor sanctions the enforcement right thereof.
5. Circuitry and other examples described herein are meant merely to indicate characteristics and performance of Hitachi semiconductor-applied products. Hitachi assumes no responsibility for any patent infringements or other problems resulting from applications based on the examples described herein.
6. No license is granted by implication or otherwise under any patents or other rights of any third party or Hitachi, Ltd.

March 1988

Printed in U.S.A.

HMCS400 SERIES

User's Manual

SECTION 1

Software Application Notes

SECTION 2

Hardware Application Notes

SECTION 3

HMCS400 SERIES

SECTION
1

Section One

User's Manual

PREFACE

The HMCS400 Series is a CMOS 4-bit single-chip microcomputer which contains variety of on-chip resources such as CPU, ROM, RAM, serial interface, and I/O.

The HMCS400 Series, advanced product of the HMCS40 Series, realizes high-speed speed operation, high-level function and program-productive efficiency. In addition, it adopts the latest CMOS high break-down process and can drive fluorescent display tube directly.

For additional information reference:

- **Section 2, HMCS400 Series Software Application Notes**
- **Section 3, HMCS400 Series Hardware Application Notes**

CONTENTS

Symbols and Abbreviations

PC	Program Counter
SP	Stack Pointer
I/E	Interrupt Enable Flag
IF0	INT0 Interrupt Flag
IF1	INT1 Interrupt Flag
IFTA	Timer A Interrupt Flag
IFTB	Timer B Interrupt Flag
IFS	Serial Interface Interrupt Flag
IM0	INT0 Interrupt Mask
IM1	INT1 Interrupt Mask
IMTA	Timer A Interrupt Mask
IMTB	Timer B Interrupt Mask
IMS	Serial Interface Interrupt Mask
PMR	Port Mode Register
SMR	Serial Mode Register
TMA	Timer Mode Register A
TMB	Timer Mode Register B
TCA	Timer Counter A
TCBL	Timer Counter B Lower Digits
TCBU	Timer Counter B Upper Digits
TLRL	Timer Load Register Lower Digits
TLRU	Timer Load Register Upper Digits
SRL	Serial Data Register Lower Digits
SRU	Serial Data Register Upper Digits
ST	Status
CA	Carry
A	Accumulator
B	B Register
W	W Register
X	X Register
SPX	SPX Register
Y	Y Register
SPY	SPY Register
M	Memory (RAM)
MR	Memory Register
RAM	Random Access Memory
ROM	Read Only Memory
R	Data I/O Pin or Data I/O Register
D	Discrete I/O Pin or Discrete Latch

1. OVERVIEW

1.1 Features

The new CMOS 4-bit microcomputer HMCS400 series satisfies the microcomputer system which needs a large program capacity and high-level functions to meet advanced applications. The HMCS400 series offers high software-productive architecture, enhanced peripheral functions, high speed instruction execution, and support tools. It also has the characteristics of low power dissipation with CMOS process, and it is applicable to the product which needs low power dissipation as portable machine.

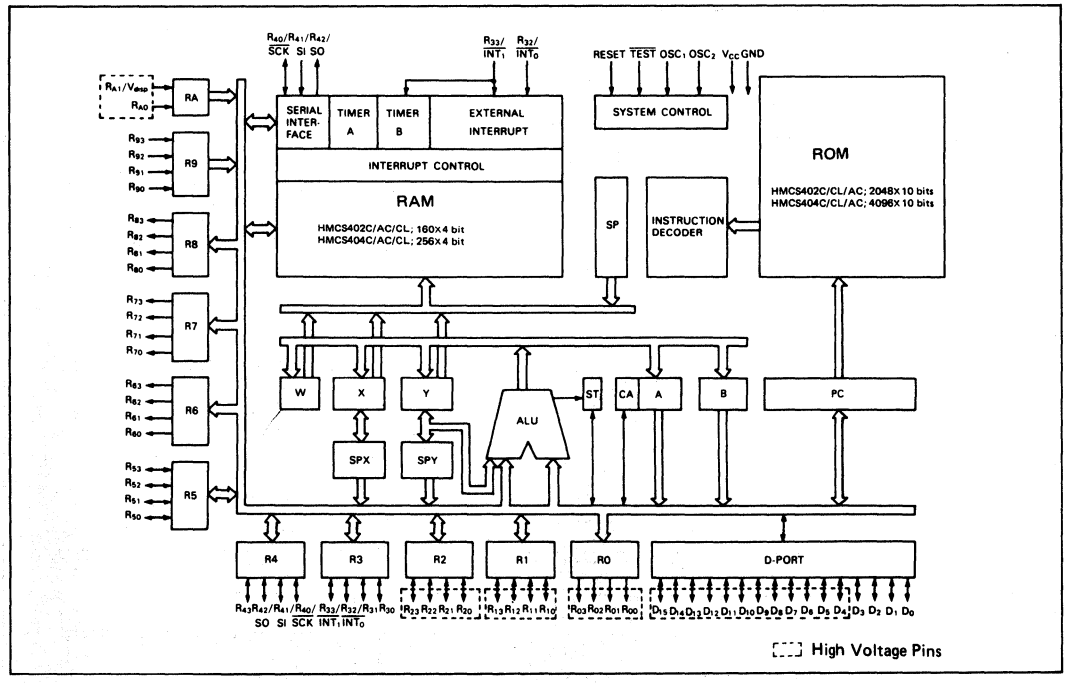
- Process : CMOS
- Architecture is compatible with the HMCS40 series for easy replacement.
- One cycle per instruction execution utilizing 10 bits per instruction
- Powerful ROM and RAM addressing capability
- 16 nesting levels
- Reinforced instruction system including logic arithmetic operating instruction, BCD arithmetic operating instruction, and pattern generating instruction
- Reinforced interrupt function ; Five interrupt levels (External : 2, Timer/Counter : 2, Serial Interface : 1)
- 8-bit serial interface
- Two timer/counters
  - 8-bit free running timer
  - 8-bit autoreload timer/event counter
- 58 I/O lines (including 26 High Voltage (40V) I/O Lines); HMCS402/404/408
- 36 I/O lines (including 24 High Voltage (40V) I/O Lines); HMCS412/414
- High-speed instruction execution HMCS408AC/412AC/414AC : 0.89µs
HMCS402AC/404AC : 1.29µs
- EPROM on-package type : HD614P080S/HD614P0160S
HD614P180/HD40P4181
- EPROM in-package type : HD4074008

Table 1-1 HMCS400 Series Function List

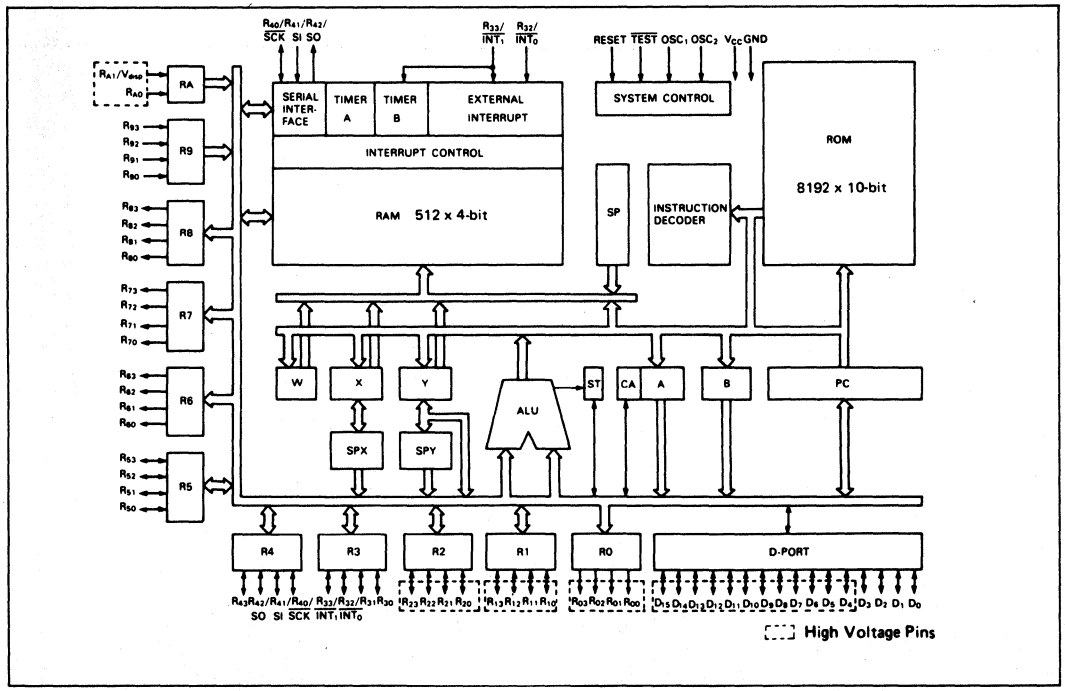
Type	Name		HMCS402C/CL/AC	HMCS404C/CL/AC	HMCS408C/CL/AC	HMCS412C/CL/AC	HMCS414C/CL/AC
LSI Characteristics	Supply Voltage (V)		5/3/5	5/3/5	5/3/5	5/3/5	5/3/5
	Max.I/O Terminal Voltage (V)		V_{CC}-40	V_{CC}-40	V_{CC}-40	V_{CC}-40	V_{CC}-40
	Operating Temperature Range (°C)		-20 to +75	-20 to +75	-20 to +75	-20 to +75	-20 to +75
	Package		DP-64S,FP-64	DP-64S,FP-64	DP-64S,FP-64	DP-42,DP-42S	DP-42,DP-42S
Functions	Memory	ROM (vits)	2,048 × 10	4,096 × 10	8,192 × 10	2,048 × 10	4,096 × 10
		RAM (bits)	160 × 4	256 × 4	512 × 4	160 × 4	160 × 4
	I/O Ports		58	58	58	36	36
	Interrupt	External	2	2	2	2	2
		Timer/Counter	2	2	2	1	1
		Serial Interface	1	1	1	-	-
	Instruction		99	99	99	98	98
	Timer		8 bit × 2	8 bit × 2	8 bit × 2	8 bit × 1	8 bit × 1
	SCI		8 bit × 1	8 bit × 1	8 bit × 1	-	-
	EPROM on the Package Type		HD614P080S	HD614P080S	HD614P080S HD614P0160S	HD614P180 HD40P4181	HD614P180 HD40P4181
	EPROM in the Package Type		HD4074008	HD4074008	HD4074008	—	—

1.2 Block Diagram

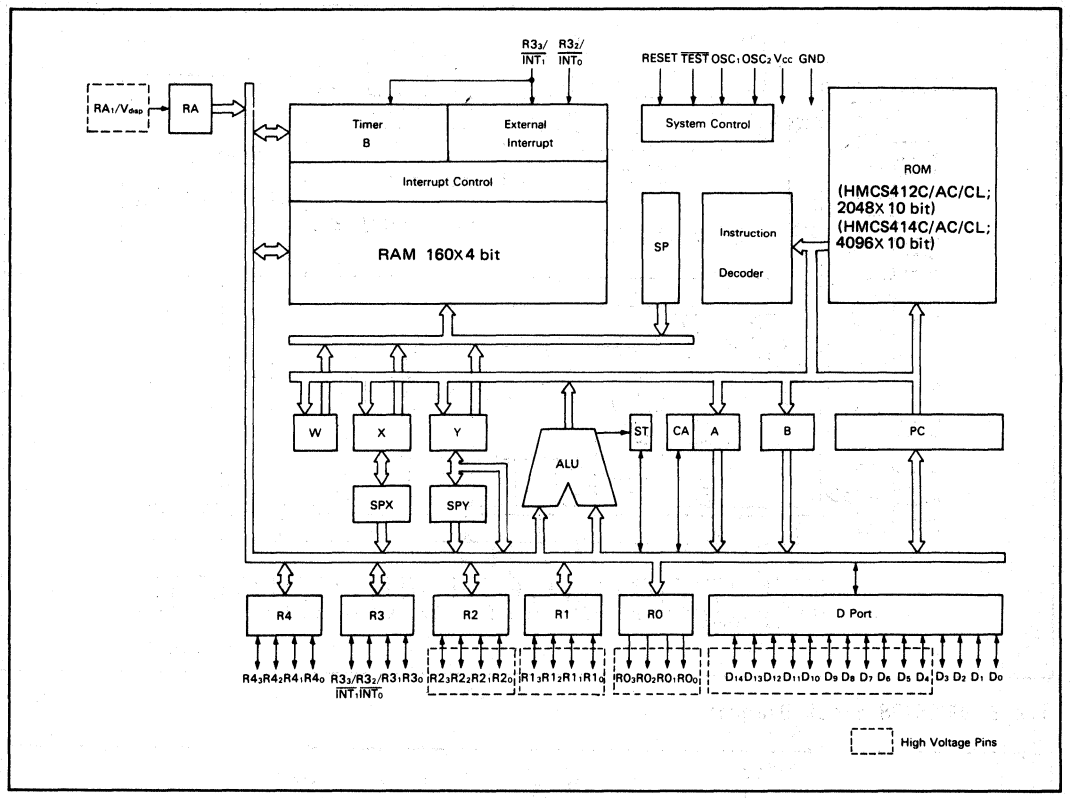
1.2.1 HMCS402/404 Block Diagram



1.2.2 HMCS408 Block Diagram



1.2.3 HMCS412/414 Block Diagram



1.3 Pin Description

The MCU input and output signals are described below.

- o GND, V_{CC}, V_{disp}

  These are the Power supply pins for the MCU. Connect the GND to the ground (0V) and apply the V_{CC} power supply voltage to the V_{CC} pin. The V_{disp} pin (multiplexed with R_{A1}) is a power supply for high voltage I/O pins with maximum voltage of 40V(V_{CC}). For details, see "2.7 Input/Output".

- o $\overline{TEST}$

  This pin is not for use by users. It should be connected to V_{CC} pin.

- o RESET

  This pin is used to reset the MCU. For details, see "2.8 Reset".

- o OSC_1, OSC_2

  These are input pins for the internal oscillator circuit. They can be connected to the crystal resonator, ceramic filter resonator, resistor (resistor oscillation is applied to the HMCS402C and HMCS404C) or external oscillator circuits. For details, see "2.9 Internal Oscillator Circuit".

- o D-port

  The D-port is input/output port addressed by one bit. The pins D_0 to D_3 are standard-type pins and D_4-D_{15} are high voltage pins. The Circuit type for each pin can be selected using a mask option. For details, see "2.7 Input/Output".

- o R-ports (R_0 to R_A)

  These are 4-bit I/O ports. (R_A however, is 2-bit construction.) R_0, R_6, R_7 and R_8 are output ports, R_9 and R_A are input ports, and R_1 to R_5 are I/O ports. R_0, R_1, R_2 and R_A are high voltage ports, and R_3-R_9 are standard ports. Each pin has a mask option which selects its circuit type. The pins R_{32}, R_{33}, R_{40}, R_{41}, and R_{42} are multiplexed with $\overline{INT_0}$, $\overline{INT_1}$, $\overline{SCK}$, SI, and SO respectively. For details, see "2.7 Input/Output".

- o $\overline{INT_0}$, $\overline{INT_1}$

  These are input pins with which MCU operations can be interrupted externally. $\overline{INT_1}$ can be used as and external event input pin for Timer B. $\overline{INT_0}$ and $\overline{INT_1}$ are multiplexed with R_{32}, R_{33} respectively. For details, see "2.4 Interrupt".

o $\overline{SCK}$, SI, SO

The Transfer Clock I/O pin ($\overline{SCK}$), Serial Data Input pin (SI), and Serial Data Output pin (SO) are used for serial interface. $\overline{SCK}$, SI, and SO are multiplexed with R_{40}, R_{41}, and R_{42} respectively. For details, see "2.5 Serial Interface".

2. INTERNAL STRUCTURE AND ITS OPERATION

2.1 ROM Memory Map

Table 2-1 shows the ROM capacity of each family. ROM memory map is illustrated in Fig. 2-1 and described in the following paragraphs.

Table 2-1 Capacity of HMCS400 Series ROM

Family	ROM Capacity
HMCS402C, HMCS402CL, HMCS402AC	2,048 words × 10 bits
HMCS404C, HMCS404CL, HMCS404AC	4,096 words × 10 bits
HMCS408C, HMCS408CL, HMCS408AC	8,192 words × 10 bits
HMCS412C, HMCS412CL, HMCS412AC	2,048 words × 10 bits
HMCS414C, HMCS414CL, HMCS414AC	4,096 words × 10 bits

(1) Vector Address Area --- $0000 to $000F

Locations $0000 through $000F are reserved for JMPL instructions to branch to the starting address of the initialization program and of the interrupt service programs. After reset of interrupt routine is serviced, the program is executed from the vector address.

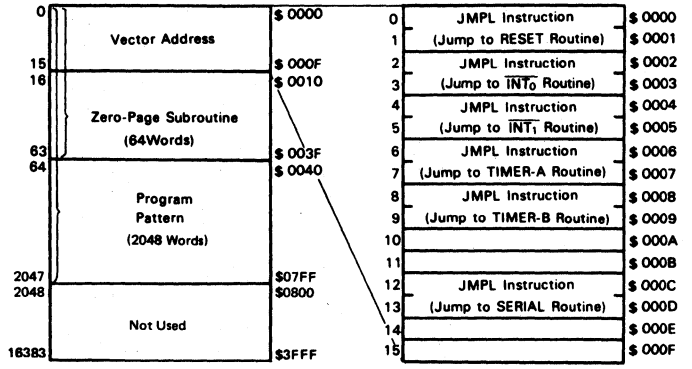
(2) Zero-Page Subroutine Area --- $0000 to $003F

Maximum Locations $0000 through $0FFF are reserved for ROM data. P instruction allows to branch to the subroutine.

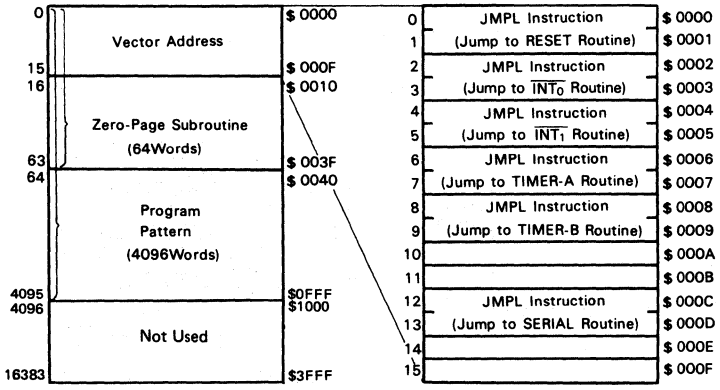
(3) Pattern Area --- $0000 to $07FF (HMCS402C/CL/AC, HMCS412C/CL/AC)
$0000 to $0FFF (HMCS404C/CL/AC, HMCS408C/CL/AC, HMCS414C/CL/AC)

Maximum locations $0000 through $0FFF are reserved for ROM data. P instruction allows referring to the ROM data as a pattern.

(4) Program Area --- $0000 to $07FF (HMCS402C/CL/AC, HMCS412C/CL/AC)
$0000 to $0FFF (HMCS404C/CL/AC, HMCS414C/CL/AC)
$0000 to $1FFF (HMCS408C/CL/AC)

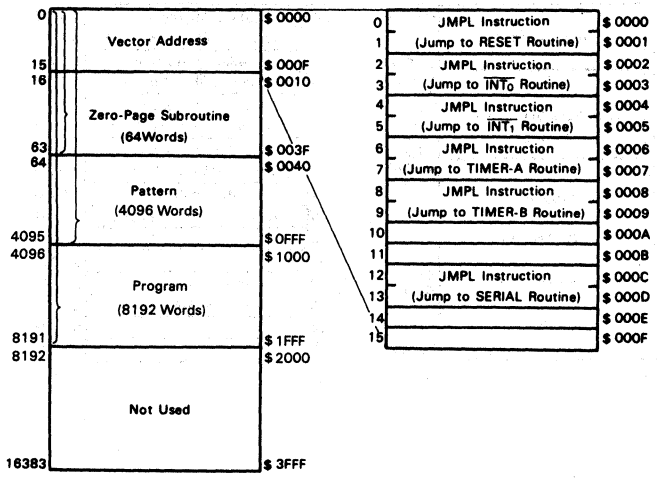


HMCS402C/AC/CL

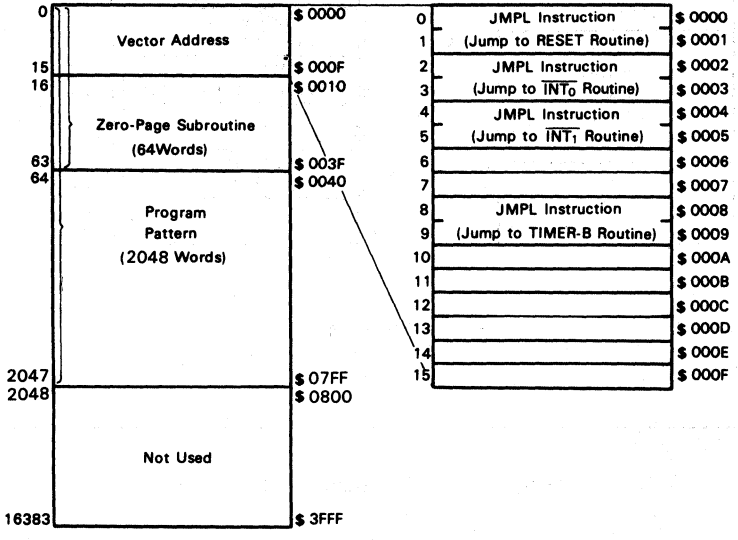


HMCS404C/CL/AC

Fig. 2-1 ROM Memory Map

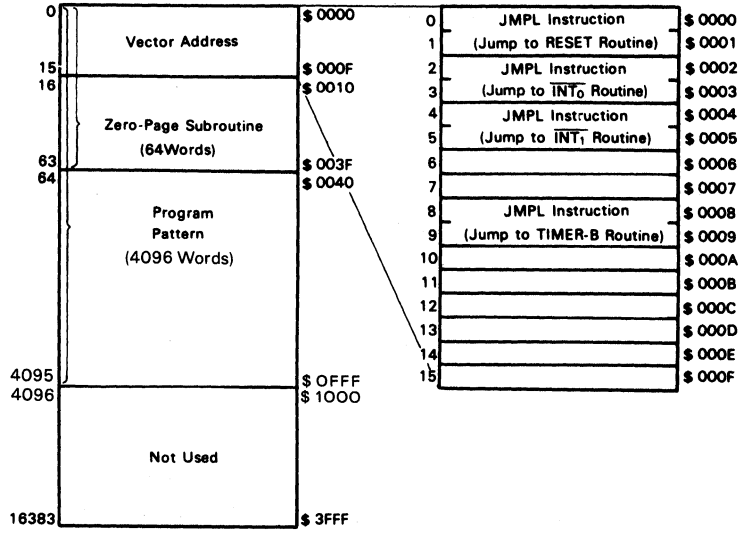


HMCS408C/CL/AC



HMCS412C/CL/AC

Fig. 2-1 ROM Memory Map



HMCS414C/CL/AC

Fig. 2-1 ROM Memory Map

2.2 RAM Memory Map

The MCU includes RAM as the data area and stack area. In addition to these areas, interrupt control bits and special function registers are also mapped on the RAM memory space. Table 2-2 shows the RAM capacity of each family. RAM memory map is illustrated in Fig. 2-2 and described in the following paragraphs.

Table 2-2 Capacity of HMCS400 Series RAM

Family	RAM Capacity
HMCS402C, HMCS402CL, HMCS402AC	160 digits × 4 bits
HMCS404C, HMCS404CL, HMCS404AC	256 digits × 4 bits
HMCS408C, HMCS408CL, HMCS408AC	512 digits × 4 bits
HMCS412C, HMCS412CL, HMCS412AC	160 digits × 4 bits
HMCS414C, HMCS414CL, HMCS414AC	160 digits × 4 bits

(1) Interrupt Control Bit Area --- $000 to $003

This area is used for interrupt controls, and is illustrated in Fig. 2-3. It is accessable only by RAM bit manipulation instruction. However, the interrupt request flag cannot be set by software.

(2) Special Function Registers Area --- $004 to $00B

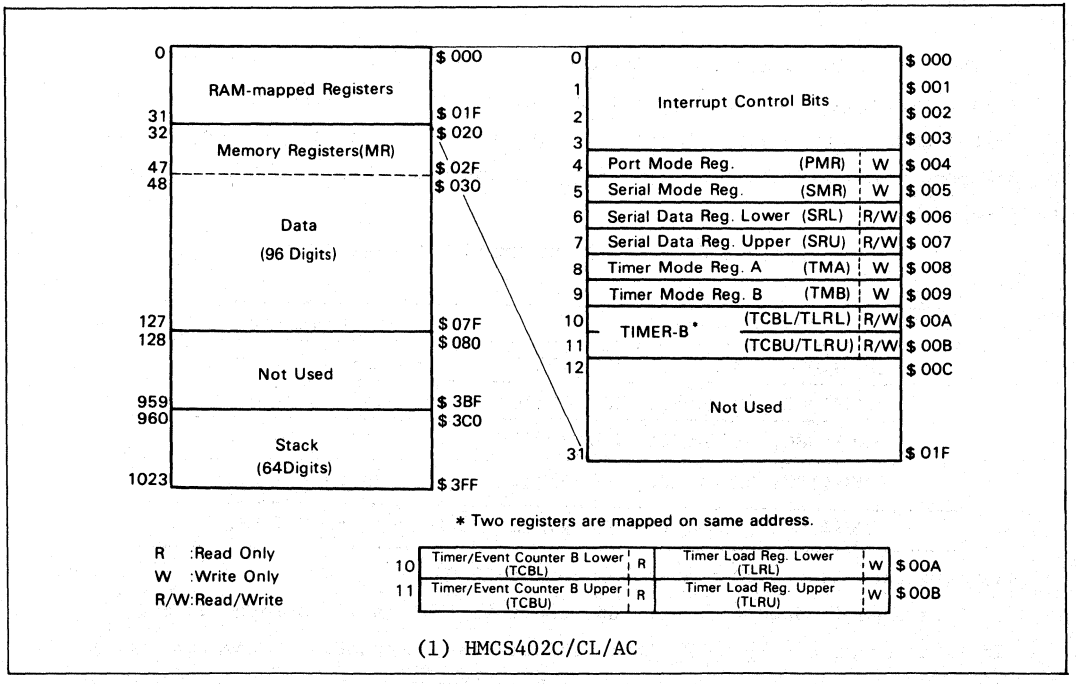
The Special Function Registers are the mode or data registers for the external interrupt, the serial interface, and the timer/counter. These registers are classified into three types: Write-only, Read-only, and Read/Write as shown in Fig. 2-2. These registers cannot be accessed by RAM bit manipulation instruction.

(3) Data Area --- $020 to $07F (HMCS402C/CL/AC, HMCS412C/CL/AC, HMCS414C/CL/AC)
$020 to $0DF (HMCS404C/CL/AC)
$020 to $1DF (HMCS408C/CL/AC)

16 digits of $020 through $02F are called memory register (MR) and accessable by LAMR and XMRA instructions. The configuration is shown in Fig. 2-4.

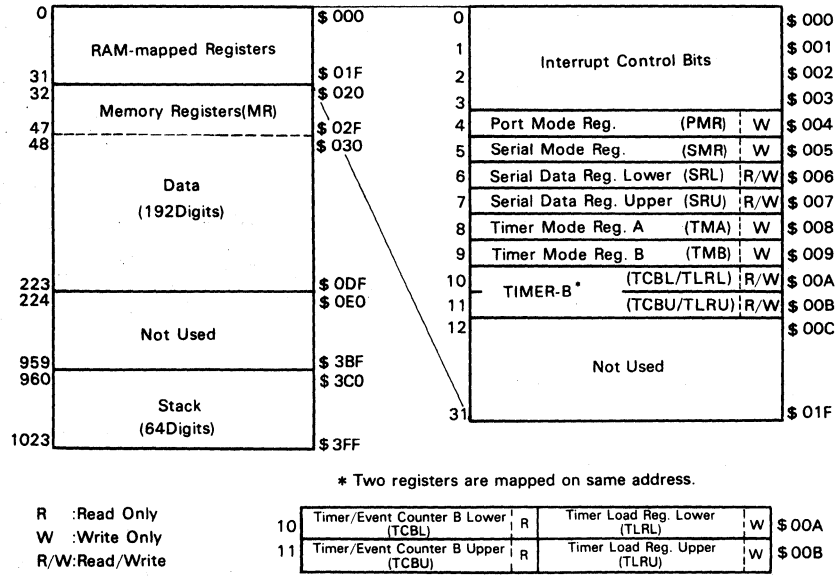
(4) Stack Area --- $3C0 to $3FF

Locations $3C0 through $3FF are reserved for LIFO stacks to save the contents of the program counter (PC), status (ST) and carry (CA) when interruption is serviced. This area can be used as 16 nesting level stack which one level requires 4 digits. A save condition is shown in Fig. 2-4. The program counter is restored by RTN and RTNI instructions. Status and Carry are restored only by RTNI instruction. The area, not used for stacking, is available as a data area.



(1) HMCS402C/CL/AC

Fig. 2-2 RAM Memory Map

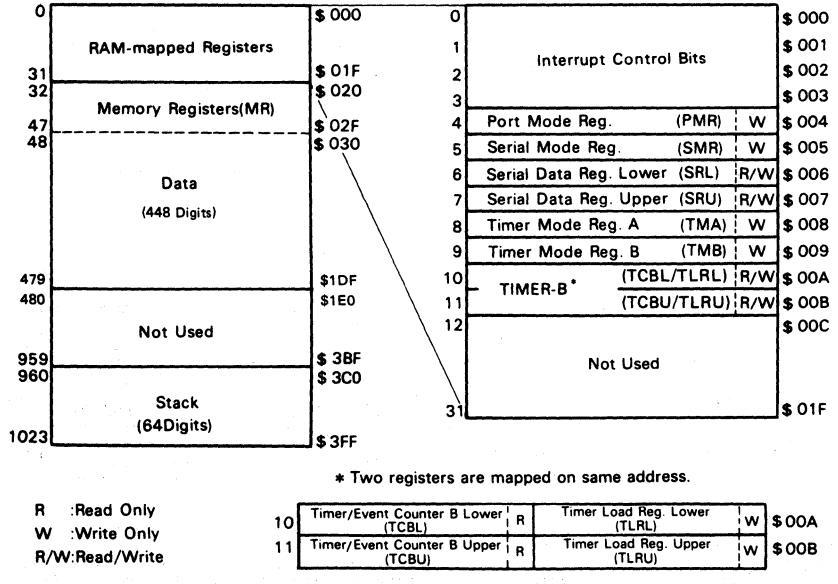


* Two registers are mapped on same address.

R :Read Only
W :Write Only
R/W:Read/Write

10	Timer/Event Counter B Lower (TCBL)	R	Timer Load Reg. Lower (TLRL)	W	$ 00A
11	Timer/Event Counter B Upper (TCBU)	R	Timer Load Reg. Upper (TLRU)	W	$ 00B

(2) HMCS404C/CL/AC



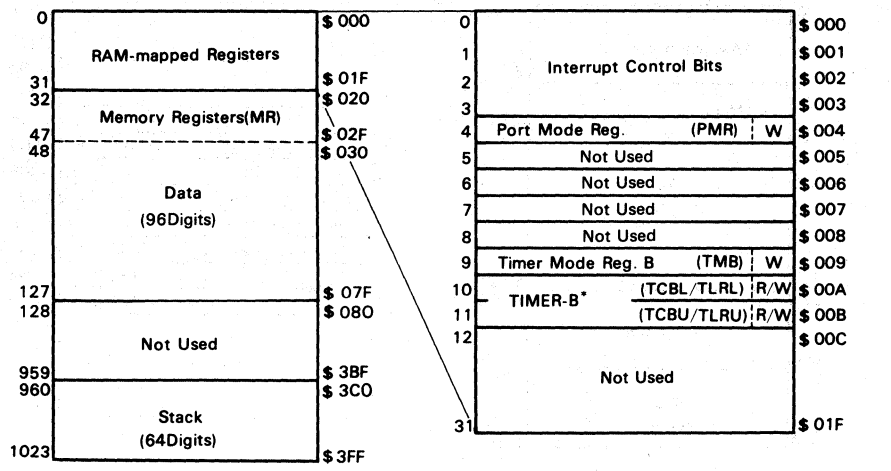
* Two registers are mapped on same address.

R :Read Only
W :Write Only
R/W:Read/Write

10	Timer/Event Counter B Lower (TCBL)	R	Timer Load Reg. Lower (TLRL)	W	$ 00A
11	Timer/Event Counter B Upper (TCBU)	R	Timer Load Reg. Upper (TLRU)	W	$ 00B

(3) HMCS408C/CL/AC

Fig. 2-2 RAM Memory Map



* Two registers are mapped on same address.

R :Read Only
W :Write Only
R/W:Read/Write

10	Timer/Event Counter B Lower (TCBL)	R	Timer Load Reg. Lower (TLRL)	W	$ 00A
11	Timer/Event Counter B Upper (TCBU)	R	Timer Load Reg. Upper (TLRU)	W	$ 00B

(4) HMCS412C/CL/AC, HMCS414C/CL/AC

Fig. 2-2 RAM Memory Map

	bit 3	bit 2	bit 1	bit 0	
0	IMO (IM of $\overline{INT_0}$)	IFO (IF of $\overline{INT_0}$)	RSP (Reset SP Bit)	I/E (Interrupt Enable Flag)	$000
1	IMTA (IM of TIMER-A)	IFTA (IF of TIMER-A)	IM1 (IM of $\overline{INT_1}$)	IF1 (IF of $\overline{INT_1}$)	$001
2	Not Used	Not Used	IMTB (IM of TIMER-B)	IFTB (IF of TIMER-B)	$002
3	Not Used	Not Used	IMS (IM of SERIAL)	IFS (IF of SERIAL)	$003

IF : Interrupt Request Flag
IM : Interrupt Mask
I/E : Interrupt Enable Flag
SP : Stack Pointer

(Note) Each bit in Interrupt Control Bits Area is set by SEM/SEMD instruction, is reset by REM/REMD instruction and is tested by TM/TMD instruction. It is not affected by other instructions. Furthermore, Interrupt Request Flag is not affected by SEM/SEMD instruction. The content of Status becomes invarid when "Not Used" bit is tested.

(1) HMCS402, HMCS404, HMCS408

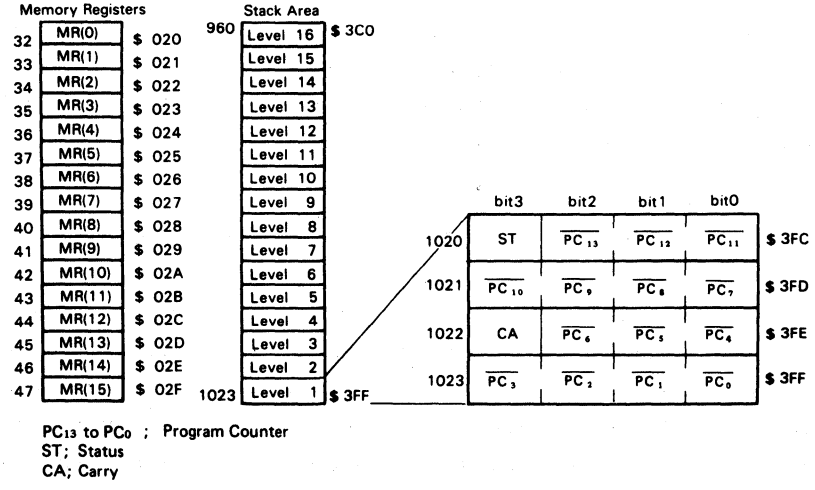
Fig. 2-3 Configuration of Interrupt Control Bit Area

	bit 3	bit 2	bit 1	bit 0	
0	IM0 (IM of $\overline{INT_0}$)	IF0 (IF of $\overline{INT_0}$)	RSP (Reset SP Bit)	I/E (Interrupt Enable Flag)	$000
1	Not Used	Not Used	IM1 (IM of $\overline{INT_1}$)	IF1 (IF of $\overline{INT_1}$)	$001
2	Not Used	Not Used	IMTB (IM of Timer B)	IFTB (IF of Timer B)	$002
3	Not Used	Not Used	Not Used	Not Used	$003

IF: Interrupt Request Flag
IM: Interrupt Mask
I/E: Interrupt Enable Flag
SP: Stack Pointer
Note: Each bit in the interrupt control bits area is set by the SEM/SEMD instruction, is reset by the REM/REMD instruction, and is tested by the TM/TMD instruction. It is not affected by other instructions. Furthermore the interrupt request flag is not affected by the SEM/SEMD instruction.
The content of status becomes invalid when "Not Used" bit and RSP bit are tested by a TM or TMD instruction.

(2) HMCS412, HMCS414

Fig. 2-3 Configuration of Interrupt Control Bit Area



Note) 1. As the HMCS402/HMCS412 have 2k ROM version, $\overline{PC_{11}}$, $\overline{PC_{12}}$, $\overline{PC_{13}}$ are not used.

2. As the HMCS404/HMCS414 have 4k ROM version, $\overline{PC_{12}}$, $\overline{PC_{13}}$ are not used.

3. As the HMCS408 have 8k ROM version, $\overline{PC_{13}}$ is not used.

Fig. 2-4 Configuration of Memory Register, Stack Area and Stack Position

2.3 Registers and Flags

The MCU has nine registers and two flags for the CPU operations. They are illustrated in Fig. 2-5 and described in the following paragraphs.

(1) Accumulator (A), B Register (B)

The 4-bit registers Accumulator and B Register are used to hold the results of Arithmetic Logic Unit (ALU), and to transfer data to/from memories, I/O, and other registers.

(2) W Register (W), X Register (X), Y Register (Y)

W Register is 2-bit, and X and Y Registers are 4-bit registers used for indirect addressing of RAM. Y Register is also used for D-port addressing. W Register is a write-only register. For details, see "8.1 Precautions on Using W Register".

(3) SPX Register (SPX), SPY Register (SPY)

The 4-bit registers SPX and SPY Registers are used to assist X and Y Register respectively.

(4) Carry (CA)

The Carry (CA) stores the overflow of ALU generated by the arithmetic operation. It is also affected by SEC, REC, ROTL and ROTR instructions.

During interrupt is serviced, Carry is pushed onto the stack and restored by RTNI instruction (not by RTN instruction).

(5) Status (ST)

The Status (ST) latches and overflow and Not Zero generated from ALU, results of bit test. It is a branch condition of BR, BRL, CAL or CALL instructions. The value of the Status remains unchanged until the next arithmetic compare of bit test instruction is executed. Status becomes "1" after the BR, BRL, CAL or CALL instruction is executed irrespectively whether it is executed or skipped. During the interrupt servicing, Status is pushed onto the stack and restored back from the stack by RTNI instruction (not by RTN instruction).

(6) Program Counter (PC)

The Program Counter is a 14-bit binary counter which controls the sequence in which the instructions stored in ROM are executed.

(7) Stack Pointer (SP)

The 10-bit Stack Pointer contains the address at which the last data was pushed onto the stack.

The Stack Pointer is initialized to locate $3FF on the RAM address, and is decremented by 4 when data is pushed onto the stack, and incremented by 4 when data is restored back from there. Upper 4 bits of the Stack Pointer are fixed to "1111", so that the stack can be used up to 16 levels.

The Stack Pointer is initialized to $3FF in two ways; one is MCU reset and the other is to reset RSP bit by REM or REMD instruction.

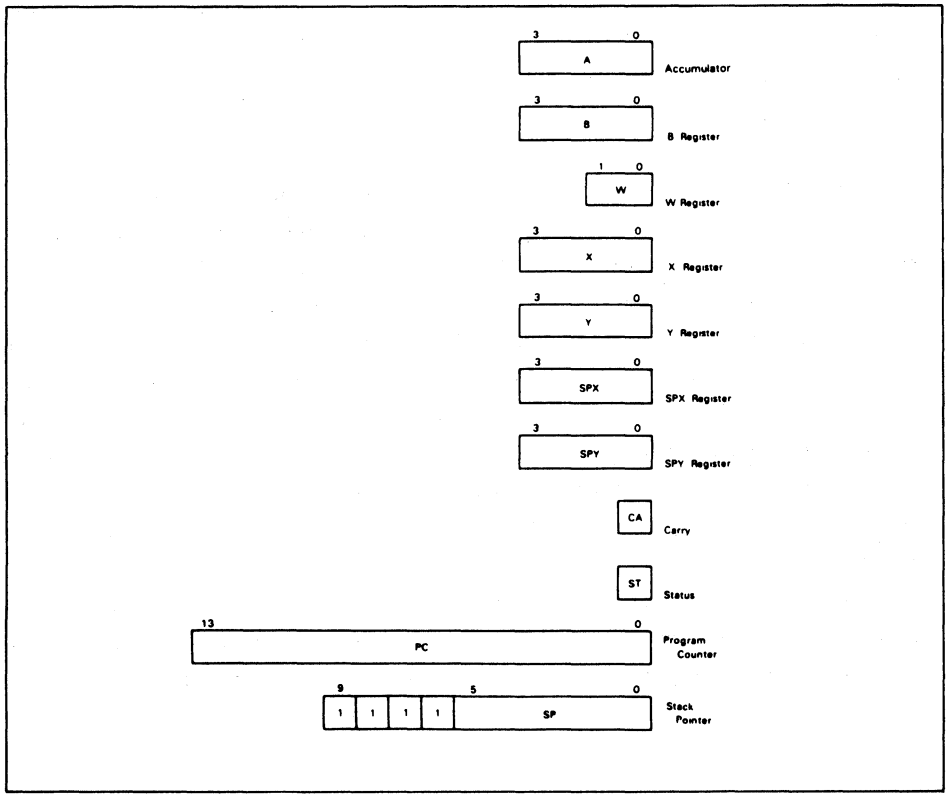


Fig. 2-5 Registers and Flags

2.4 Interrupt

Maximum interrupt sources are five available on the MCU: External Request ($\overline{INT_0}$, $\overline{INT_1}$), Timer/Counter (TIMER-A, TIMER-B), and Serial Port (SERIAL). For each source, the Interrupt Request Flag (IF), Interrupt Mask (IM) and interrupt vector addresses are provided to control and maintain and interrupt request. The Interrupt Enable Flag (I/E) is also used to control the total interrupt operations.

(1) Interrupt Control Bits and Interrupt Service

The interrupt control bits are mapped on $000 through $003 of the RAM space and are accessable by RAM bit manipulation instruction. (The Interrupt Request Flag (IF) cannot be set by software.) The Interrupt Enable Flag (I/E) and IF are set to "0", and the Interrupt Mask (IM) is set to "1" at initialization by MCU reset.

Fig. 2-6 is a block diagram of the interrupt control circuits. Table 2-3 shows the interrupt priority and vector addresses, and Table 2-4 shows the interrupt conditions corresponding to each interrupt source. The interrupt request is generated when the IF is set to "1" and IM is "0". If the I/E is "1" at this time, the interrupt will be activated and vector addresses will be generated from the priority PLA corresponding to the five interrupt sources.

Fig. 2-7 shows the interrupt service sequence, and Fig. 2-8 shows the interrupt service flowchart. If the interrupt is requested, the instruction being executed finishes in the first cycle. The I/E is reset in the second cycle. In the second and third cycles, the Carry, Status and Program Counter are pushed onto the stack. In the third cycle, the instruction is re-executed after jumping to the vector address.

In each vector address, program JMPL instruction to branch to a starting address of the interrupt service program. The IF which caused the interrupt service has to be reset by software in the interrupt service program.

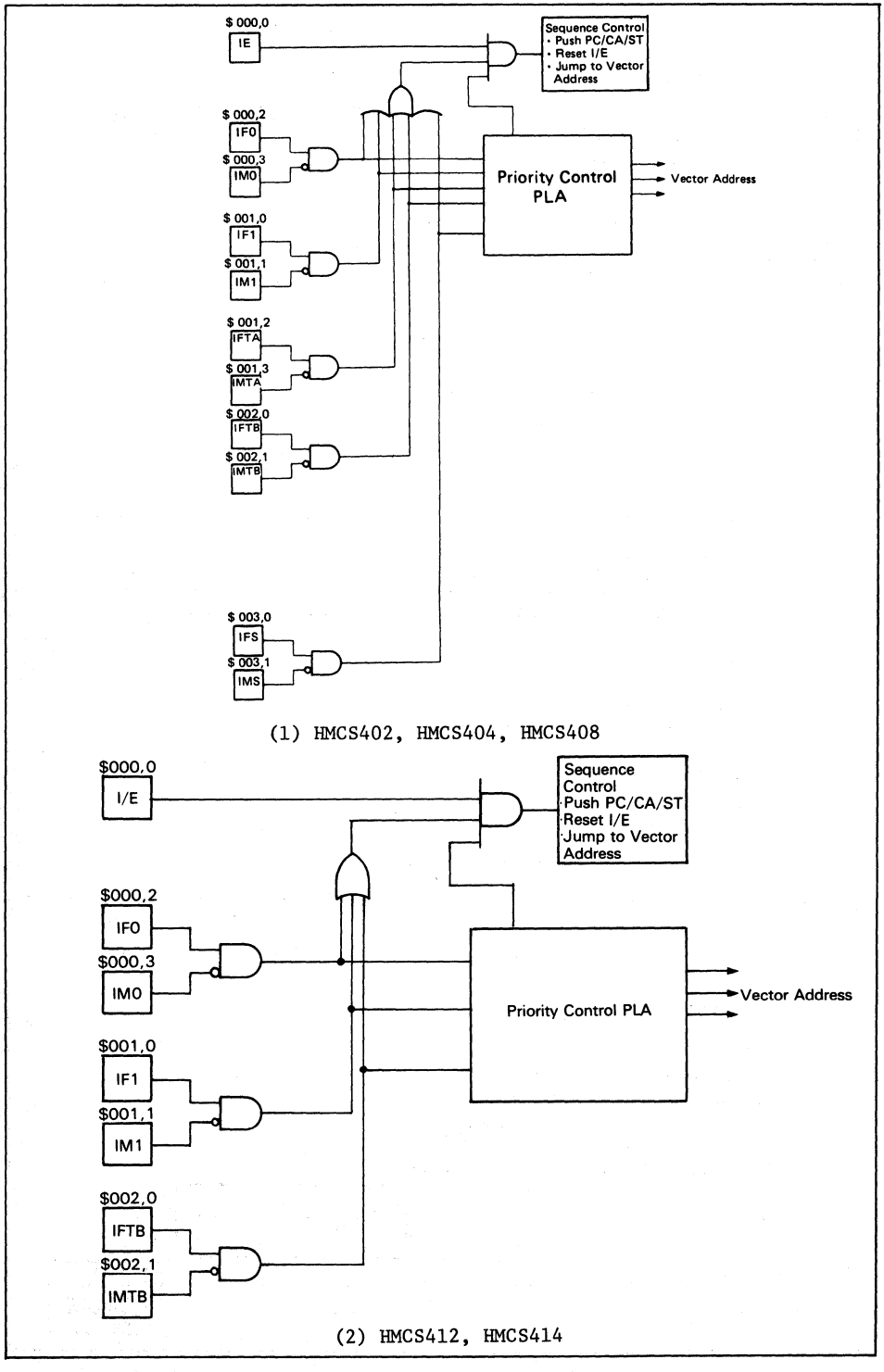


Fig. 2-6 Interrupt Circuit Block Diagram

Table 2-3 Vector Address and Interrupt Priority

(1) HMCS402, HMCS404, HMCS408

Reset · Interrupt	Priority	Vector addresses
RESET	–	$0000
$\overline{INT_0}$	1	$0002
$\overline{INT_1}$	2	$0004
TIMER-A	3	$0006
TIMER-B	4	$0008
SERIAL	5	$000C

(2) HMCS412, HMCS414

Reset, Interrupt	Priority	Vector addresses
RESET	–	$0000
$\overline{INT_0}$	1	$0002
$\overline{INT_1}$	2	$0004
Timer B	3	$0008

Table 2-4 Conditions of Interrupt Service

(1) HMCS402, HMCS404, HMCS408

Interrupt source / Interrupt control bits	$\overline{INT_0}$	$\overline{INT_1}$	TIMER-A	TIMER-B	SERIAL
I/E	1	1	1	1	1
IF0·$\overline{IM0}$	1	0	0	0	0
IF1·$\overline{IM1}$	*	1	0	0	0
IFTA·$\overline{IMTA}$	*	*	1	0	0
IFTB·$\overline{IMTB}$	*	*	*	1	0
IFS·$\overline{IMS}$	*	*	*	*	1

* Don't care

(2) HMCS412, HMCS414

Interrupt Control Bit	$\overline{INT_0}$	$\overline{INT_1}$	Timer B
I/E	1	1	1
IF0·$\overline{IM0}$	1	0	0
IF1·$\overline{IM1}$	*	1	0
IFTB·$\overline{IMTB}$	*	*	1

* Don't care

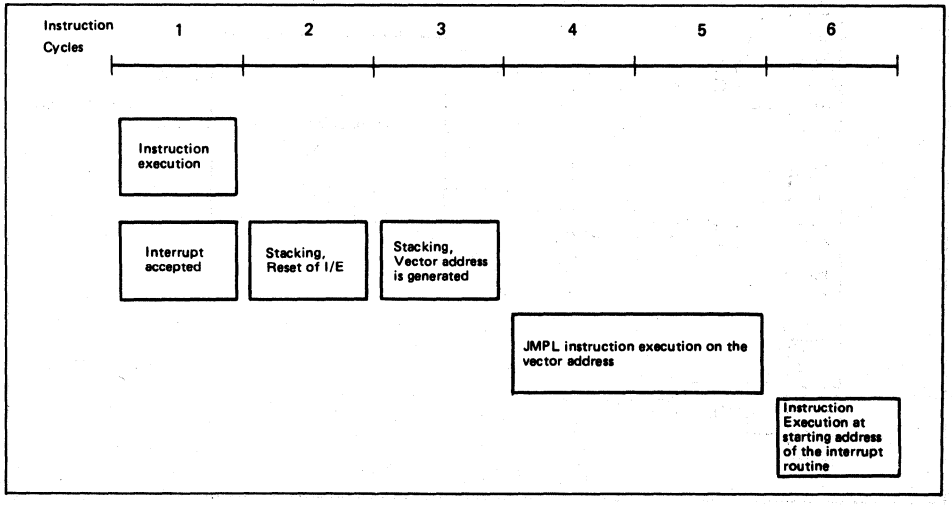
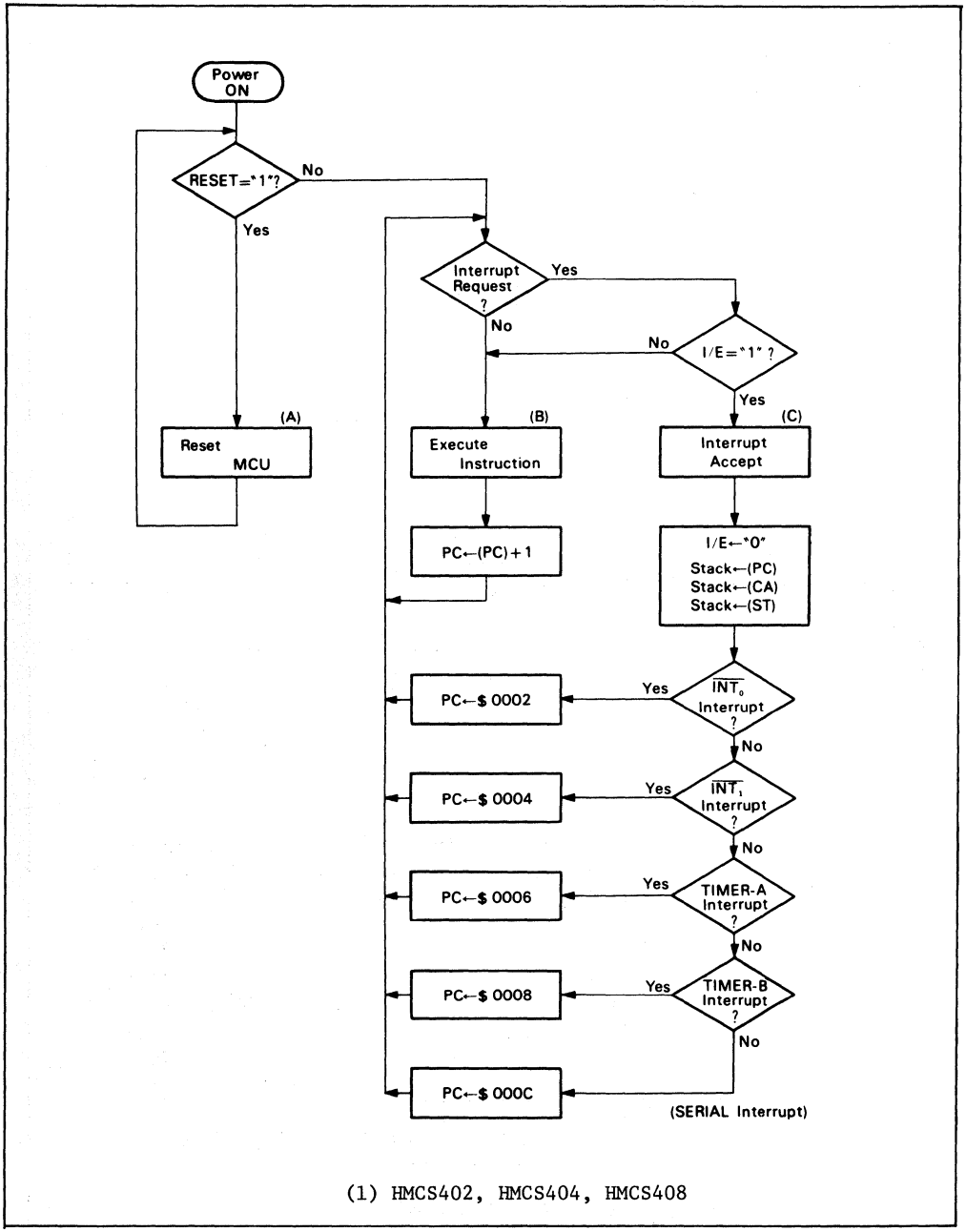
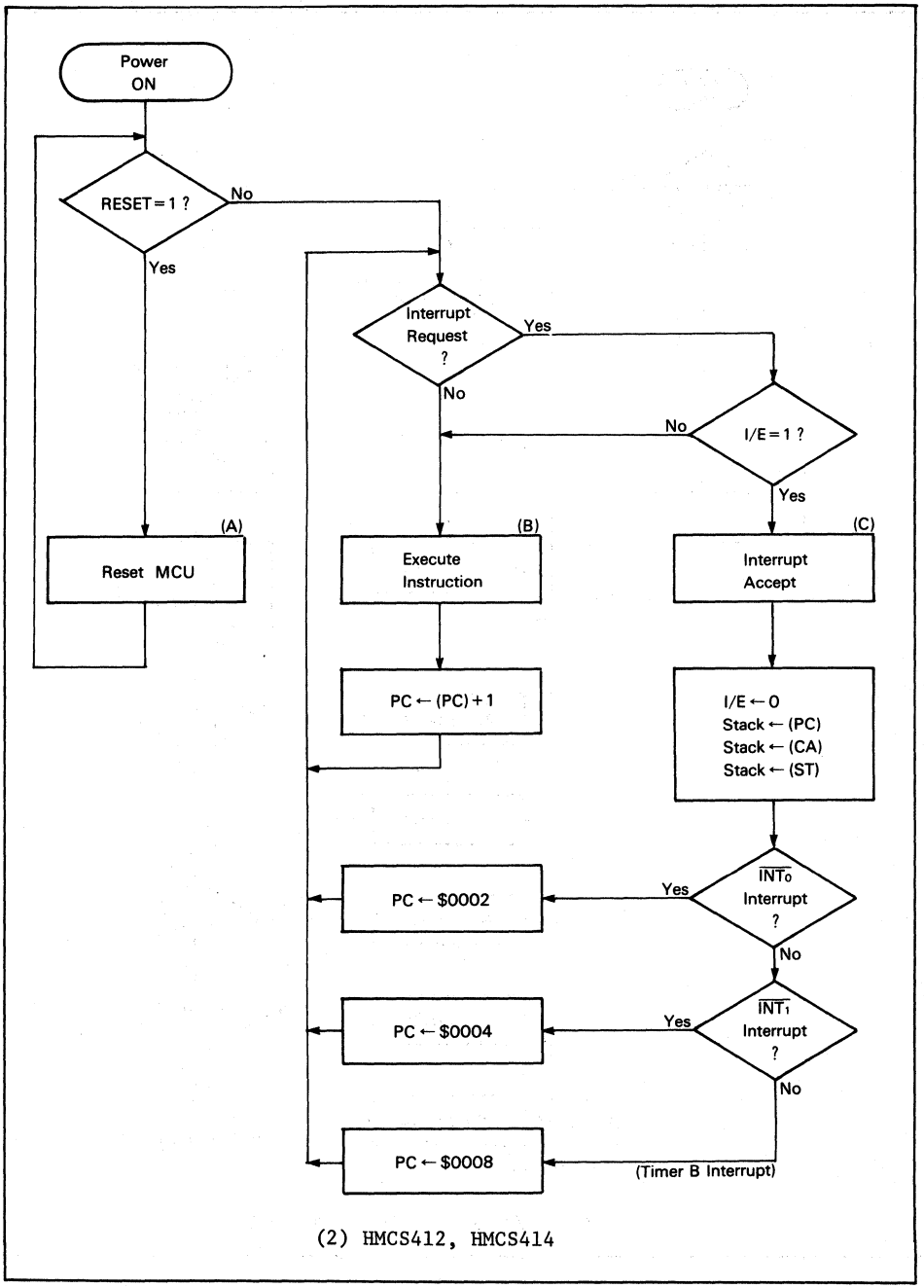


Fig. 2-7 Interrupt Servicing Sequence



(1) HMCS402, HMCS404, HMCS408

Fig. 2-8 Interrupt Servicing Flowchart



(2) HMCS412, HMCS414

Fig. 2-8 Interrupt Servicing Flowchart

(2) Interrupt Enable Flag (I/E:$000, bit 0)

The Interrupt Enable Flag controls enable/disable of interrupt requests from the sources as shown in Table 2-5. It is reset by the interrupt servicing and set by RTNI instruction.

Table 2-5 Interrupt Enable Flag

Interrupt Enable Flag	Interrupt Enable/Disable
0	Disable
1	Enable

(3) External Interrupts ($\overline{INT_0}$, $\overline{INT_1}$)

The external interrupt request inputs ($\overline{INT_0}$, $\overline{INT_1}$) can be selected by the Port Mode Register (PMR:$004). Setting the bit 3 and bit 2 of PMR causes $R_{33}/\overline{INT_1}$ pin and $R_{32}/\overline{INT_0}$ pin to be used as $\overline{INT_1}$ pin and $\overline{INT_0}$ pin respectively.

The External Interrupt Request Flags (IF0, IF1) are set at the falling edge of $\overline{INT_0}$, $\overline{INT_1}$ inputs. (Refer to Table 2-6).

$\overline{INT_1}$ input can be used as a clock signal input of TIMER-B. Then, TIMER-B counts up at each falling edge of input. When using $\overline{INT_1}$ as TIMER-B external event, and External Interrupt Mask (IM1) has to be set so that the interrupt request by $\overline{INT_1}$ will not be accepted. (Refer to Table 2-7.)

Table 2-6 External Interrupt Request Flag

External Interrupt Request Flags	Interrupt Requests
0	No
1	Yes

Table 2-7 External Interrupt Mask

External Interrupt Masks	Interrupt Requests
0	Enable
1	Disable (masks)

(4) External Interrupt Request Flags (IF0:$000 bit 2, $001 bit 0)

The External Interrupt Request Flags (IF0, IF1) are set at the falling edge of $\overline{INT_0}$, $\overline{INT_1}$ inputs respectively.

(5) External Interrupt Masks (IM0:$000 bit 3, $001 bit 1)

The External Interrupt Masks are used to mask the external interrupt requests.

(6) Port Mode Register (PMR:$004)

The Port Mode Register is a 4-bit write-only register which controls the $R_{32}/\overline{INT_0}$ pin, $R_{33}/\overline{INT_1}$ pin, R_{41}/SI pin and R_{42}/SO pin as shown in Table 2-8. The Port Mode Register will be initialized to $0 by MCU reset, all these pins are therefore used as ports.

Table 2-8 Port Mode Register

PMR bit 3	$R_{33}/\overline{INT_1}$ pin
0	Used as R_{33} port input/output pin
1	Used as $\overline{INT_1}$ input pin

PMR bit 2	$R_{32}/\overline{INT_0}$ pin
0	Used as R_{32} port input/output pin
1	Used as $\overline{INT_0}$ input pin

PMR bit 1	R_{41}/SI pin
0	Used as R_{41} port input/output pin
1	Used as SI input pin

PMR bit 0	R_{42}/SO pin
0	Used as R_{42} port input/output pin
1	Used as SO output pin

2.5 Serial Interface

The serial interface is used to transmit/receive 8-bit data serially. This consists of the Serial Data Register, the Serial Mode Register, the Octal Counter and the multiplexer as illustrated in Fig. 2-9. Pin $R_{40}/\overline{SCK}$ and the transfer clock signal are controlled by the Serial Mode Register. The contents of the Serial Data Register can be written into or read out by software. The data in the Serial Data Register can be shifted synchronously with the transfer clock signal.

STS instruction is used to initiate serial interface operations and to reset the Octal Counter to $0. The counter starts to count at the falling edge of the transfer clock ($\overline{SCK}$) signal and increments by one at the rising edge of the $\overline{SCK}$. When the Octal Counter is reset to $0 after eight transfer clock signals, or when a transmit/receive operation is discontinued by resetting the Octal Counter, the SERIAL Interrupt Request Flag will be set.

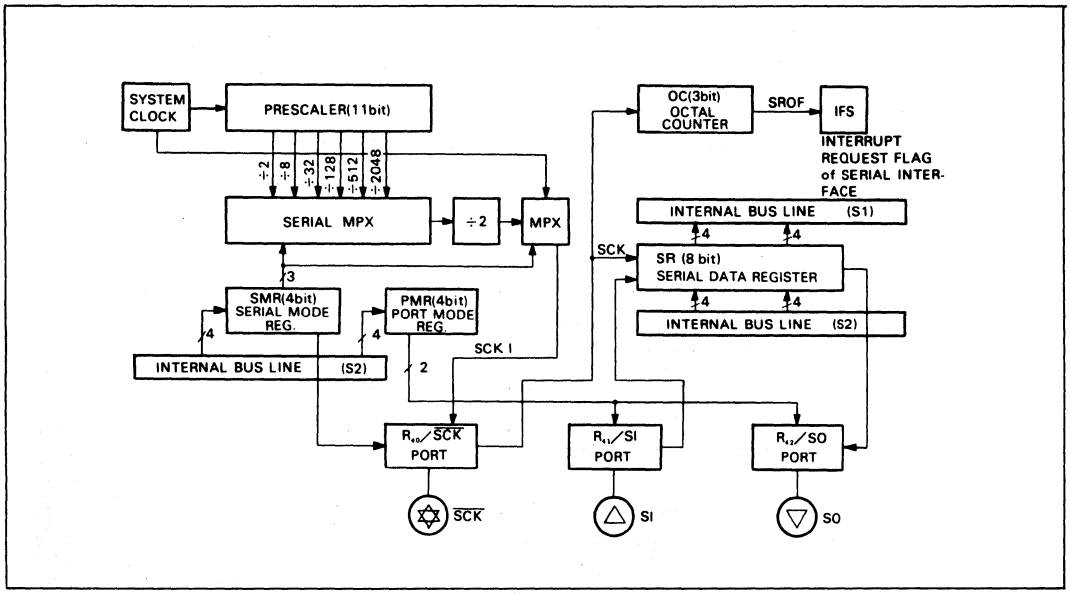


Fig. 2-9 Serial Interface Block Diagram

(1) Serial Mode Register (SMR:$005)

The 4-bit write-only Serial Mode Register controls the $R_{40}/\overline{SCK}$, prescaler divide ratio, and transfer clock source as shown in Table 2-9.

The Write Signal to the Serial Mode Register controls the operating state of the serial interface.

The Write Signal to the Serial Mode Register stops the Serial Data Register and Octal Counter from applying transfer clock, and it also resets the Octal Counter to $0 simultaneously. Therefore, when the Serial Interface is in the "Transfer State", the Write Signal causes the Serial Mode Register to case the data transfer and to set the SERIAL Interrupt Request Flag.

Contents of the Serial Mode Register will be changed on the second instruction cycle after writing into the Serial Mode Register. Therefore, it will be necessary to execute the STS instruction after the data in the Serial Mode Register has been changed completely. The Serial Mode Register will be reset to $0 by MCU reset.

(2) Serial Data Register (SRL:$006, SRU:$007)

The 8-bit read/write Serial Data Register consists of a low-order digit (SRL:$006) and a high-order digit (SRU:$007).

The data in the Serial Data Register will be output from the SO pin, from LSB to MSB, synchronously with the falling edge of the transfer clock signal. At the same time, external data will be input from the SI pin to the Serial

Data Register, to MSB first, synchronously with the rising edge of the transfer clock. Fig. 2-10 shows the I/O timing chart for the transfer clock signal and the data.

The read/write operations of the Serial Data Register should be performed after the completion of data transmit/receive. Otherwise the data may not be guaranteed.

Table 2-9 Serial Mode Register

SMR Bit 3	$R_{40}/\overline{SCK}$
0	Used as R_{40} port input/output pin
1	Used as $\overline{SCK}$ input/output pin

SMR			Transfer Clock			
Bit 2	Bit 1	Bit 0	$R_{40}/\overline{SCK}$ Port	Clock Source	Prescaler Divide Ratio	System Clock Divide Ratio
0	0	0	$\overline{SCK}$ Output	Prescaler	÷ 2048	÷ 4096
0	0	1	$\overline{SCK}$ Output	Prescaler	÷ 512	÷ 1024
0	1	0	$\overline{SCK}$ Output	Prescaler	÷ 128	÷ 256
0	1	1	$\overline{SCK}$ Output	Prescaler	÷ 32	÷ 64
1	0	0	$\overline{SCK}$ Output	Prescaler	÷ 8	÷ 16
1	0	1	$\overline{SCK}$ Output	Prescaler	÷ 2	÷ 4
1	1	0	$\overline{SCK}$ Output	System Clock	–	÷ 1
1	1	1	$\overline{SCK}$ Input	External Clock	–	–

(3) Serial Interrupt Request Flag (IFS:$003 bit 0)

The Serial Interrupt Request Flag will be set when the Octal Counter counts eight transfer clock signals, or when data transfer is discontinued by resetting the Octal Counter. Refer to Table 2-10.

(4) Serial Interrupt Mask (IMS:$003 bit 1)

The Serial Interrupt Mask masks the interrupt request. Refer to Table 2-11.

(5) Selection and Change of the Operation Mode

Table 2-12 shows the serial interface operation modes which are determined by a combination of the value in the Port Mode Register and that in the Serial Mode Register.

Initialize the serial interface by the Write Signal to the Serial Mode Register, when the Operation Mode is changed.

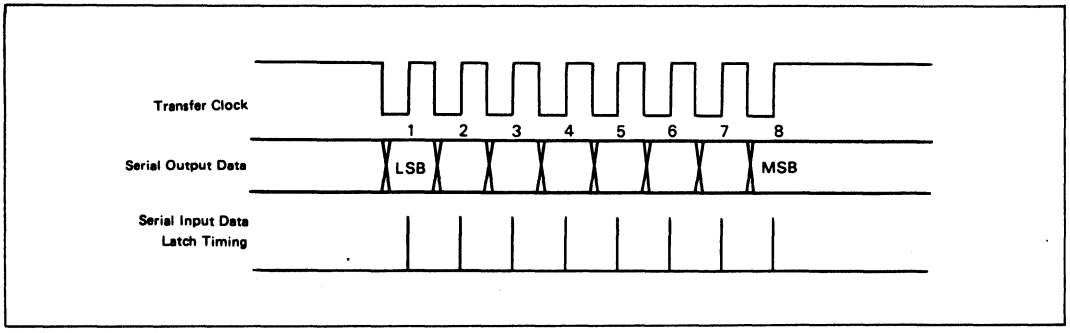


Fig. 2-10 Serial Interface I/O Timing Chart

Table 2-10 SERIAL Interrupt Request Flag

SERIAL Interrupt Request Flag	Interrupt Request
0	No
1	Yes

Table 2-11 SERIAL Interrupt Mask

SERIAL Interrupt Mask	Interrupt Request
0	Enable
1	Disable (mask)

Table 2-12 Serial Interface Operation Mode

SMR	PMR		Serial Interface Operating Mode
Bit 3	Bit 1	Bit 0	
1	0	0	Clock Continuous Output Mode
1	0	1	Transmit Mode
1	1	0	Receive Mode
1	1	1	Transmit/Receive Mode

(6) Operating State of Serial Interface

The serial interface has three operating states, the STS waiting state, SCK waiting state, and Transfer state, as shown in Fig. 2-11.

The STS waiting state is the initialization state of the serial interface internal state. The serial interface enters this state in one of two ways: either by changing the operation mode through a change in the data in the Port Mode Register, or by writing data into the Serial Mode Register. In this state, the serial interface does not operate even if the transfer clock is applied. If an STS instruction is executed, the serial interface shifts to "SCK waiting state".

In this state the falling edge of the first transfer clock causes the serial interface shift to "transfer state", while the Octal Counter counts-up and the Serial Data Register shifts simultaneously. As an exception, if the clock continuous output mode is selected, the serial interface stays in "SCK waiting state" while the transfer clock outputs continuously.

The Octal Counter becomes "000" again by 8 transfer clocks or by execution of STS instruction, so that the serial interface returns to "SCK waiting state", and the Serial Interrupt Request Flag is set simultaneously.

When the internal transfer clock is selected, the transfer clock output is triggered by the execution of an STS instruction, and stops after 8 clocks.

(7) Example of Transfer Clock Error Detection

The serial interface functions abnormally when the transfer clock is disturbed by external noises. In this case, transfer clock error can be detected by the procedure shown in Fig. 2-12.

If more than 8 transfer clocks are applied in the "SCK waiting state", the state of the serial interface shifts as the following sequence: first, "transfer state", second, "SCK waiting state" and third, "transfer state" again. The Serial Interrupt Flag should be reset before entering into the STS state by writing data to SMR. This procedure causes the serial IRF to be set again.

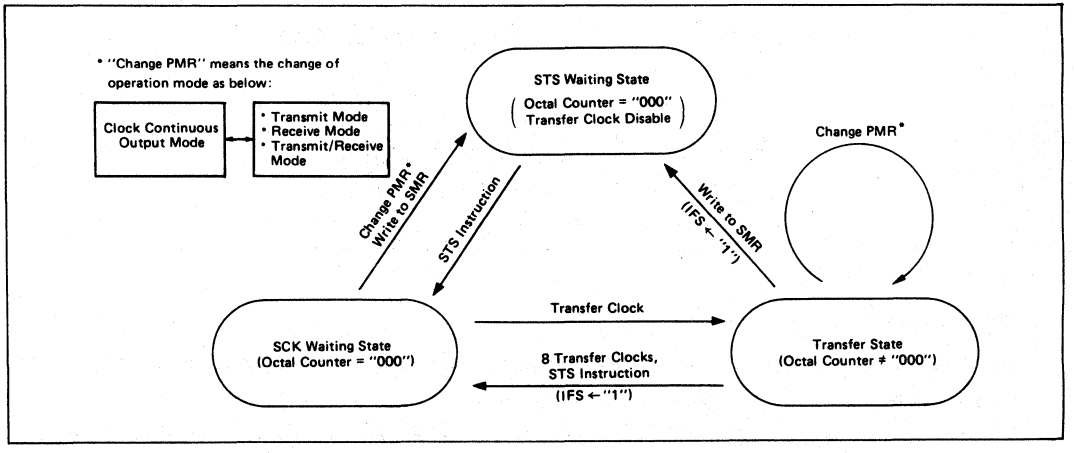


Fig. 2-11 Serial Interface Operation State

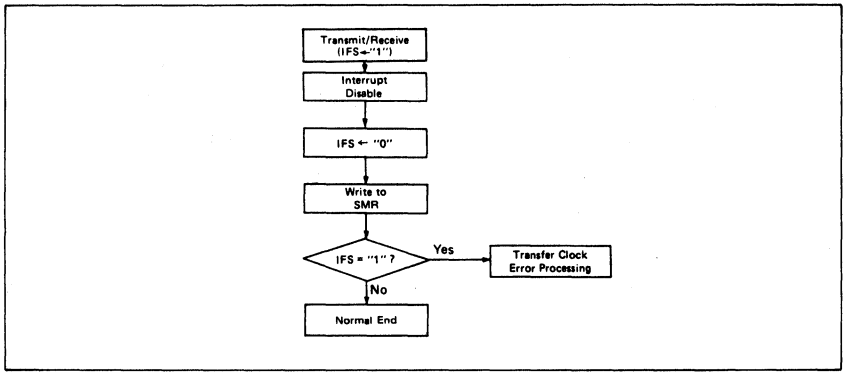


Fig. 2-12 Example of Transfer Clock Error Detection

2.6 Timer

The MCU contains a prescaler and two timer/counters (TIMER-A, TIMER-B). A block diagram is shown in Fig. 2-13. The prescaler is an 11-bit binary counter, TIMER-A an 8-bit free-running timer, and TIMER-B an 8-bit auto-reload timer/event counter.

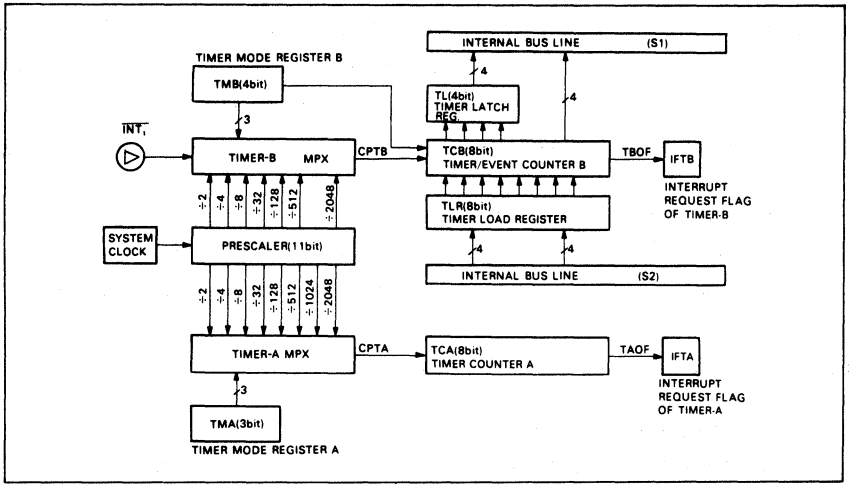


Fig. 2-13 Timer/Counter Block Diagram

(1) Prescaler

The input to the prescaler is a system clock signal. The prescaler is initialized to $0000 by MCU reset, and it starts to count up the system clock signal as soon as RESET input goes to logic "0". The prescaler keeps counting up except in MCU reset and stop mode. The prescaler provides clock signals to TIMER-A, TIMER-B and the serial interface. The prescaler divide ratio of the clock signals are selected according to the contents of the mode registers, e.g. Timer Mode Register A (TMA), Timer Mode Register B (TMB), Serial Mode Register (SMR).

(2) TIMER-A Operation

After TIMER-A is initialized to $00 by MCU reset, it counts up at every clock input signal. When the next clock signal is applied after TIMER-A becomes $FF, it will generate an overflow and become $00. This overflow causes the TIMER-A Interrupt Request Flag (IFTA:$001 bit 2) to go to "1". This timer can function as an interval timer periodically generating overflow output at every 256th clock signal input.

The clock input signals to TIMER-A are selected by the Timer Mode Register A (TMA:$008).

(3) TIMER-B Operation

Timer Mode Register B (TMB:$009) is used to select the auto-reload function, input clock source, and the prescaler divide ratio of TIMER-B. When the external event input is used as an input clock signal to TIMER-B, select the $R_{33}/\overline{INT_1}$ as $\overline{INT_1}$ and set the External Interrupt Mask (IM1) to prevent an external interrupt request from occurring.

TIMER-B is initialized according to the value written into the Timer Load Register by software. TIMER-B counts up at every clock input signal. When the next clock signal is applied to TIMER-B after it is set to $FF, TIMER-B will generate overflow output. In this case, if the auto-reload function is selected TIMER-B is initialized according to the value of the Timer Load Register, and if it is not selected, TIMER-B goes to $00. The TIMER-B Interrupt Request Flag (IFTB:$002 bit 0) will be set at this overflow output.

(4) Timer Mode Register A (TMA:$008)

The Timer Mode Register A (TMA) is a 3-bit write-only register. The TMA controls the prescaler divide ratio of TIMER-A clock input, as shown in Table 2-13.

The TMA is initialized to $0 by MCU reset.

(5) Timer Mode Register B (TMB:$009)

The Timer Mode Register B (TMB) is a 4-bit write-only register which controls the selection of the auto-reload function of TIMER-B and the prescaler divide ratio, and the source of the clock input signal, as shown in Table 2-14.

The Timer Mode Register B is initialized to $0 by MCU reset.

The operation mode of TIMER-B is changed at the second instruction cycle after writing into the Timer Mode Register B. Initialization of TIMER-B by the write instruction to Timer Load Register should be performed after the contents of TMB are changed. Configuration and function of Timer Mode Register is shown in Fig. 2-14.

Table 2-13 Timer Mode Register A

TMA			Prescaler Divide Ratio
Bit 2	Bit 1	Bit 0	
0	0	0	÷2048
0	0	1	÷1024
0	1	0	÷ 512
0	1	1	÷ 128
1	0	0	÷ 32
1	0	1	÷ 8
1	1	0	÷ 4
1	1	1	÷ 2

Table 2-14 Timer Mode Register B

TMB	Auto-reload Function
Bit 3	
0	No
1	Yes

TMB			Prescaler Divide Ratio, Clock Input Source
Bit 2	Bit 1	Bit 0	
0	0	0	÷2048
0	0	1	÷ 512
0	1	0	÷ 128
0	1	1	÷ 32
1	0	0	÷ 8
1	0	1	÷ 4
1	1	0	÷ 2
1	1	1	$\overline{INT_1}$ (External Event Input)

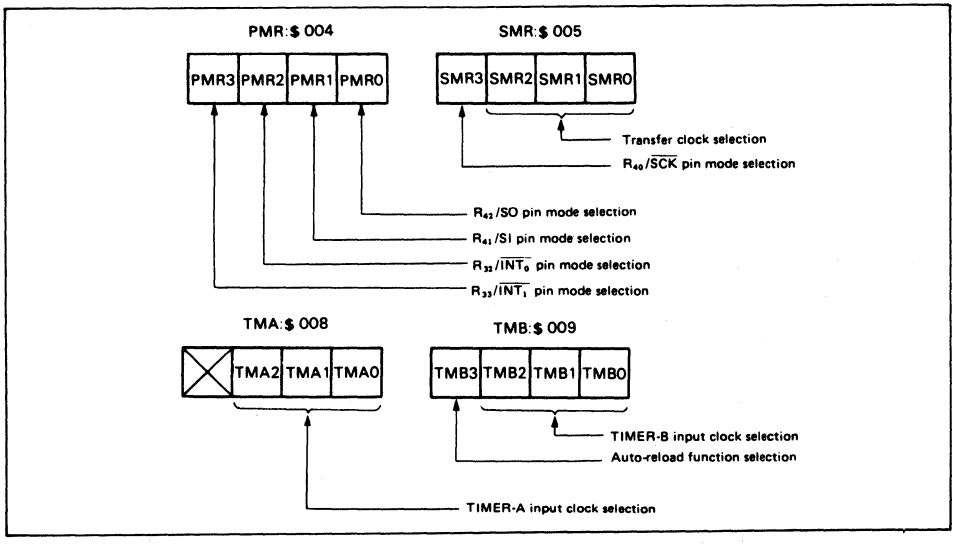


Fig. 2-14 Mode Register Configuration and Function

(6) TIMER-B (TCBL:$00A, TCBU:$00B, TLRL.$00A, TLRU:$00B)

TIMER-B consists of an 8-bit write-only Timer Load Register, and an 8-bit read-only Timer/Event Counter. Each of them has a low-order digit (TCBL:$00A, TLRL:$00A) and a high-order digit (TCBU:$00B, TLRU.$00B). (Refer to Fig. 2-2.)

The Timer/Event Counter can be initialized by writing data into the Timer Load Register. In this case, write the low-order digit first, and then the high-order digit. The Timer/Event Counter is initialized at the time when the high-order digit is written. The Timer Load Register has been initialized to $00 by the MCU reset.

The counter value of TIMER-B can be obtained by reading the Timer/Event Counter. In this case, read the high-order digit first, and then the low-order digit. The count value of the low-order digit is latched at the time when the high-order digit is read.

(7) TIMER-A Interrupt Request Flag (IFTA:$001 bit 2)

The TIMER-A Interrupt Request Flag is set by the overflow output of TIMER-A. Refer to Table 2-15.

(8) TIMER-A Interrupt Mask (IMTA:$001 bit 3)

The TIMER-A Interrupt Mask prevents an interrupt request from being generated by TIMER-A Interrupt Request Flag. Refer to Table 2-16.

(9) TIMER-B Interrupt Request Flag (IFTB:$002 bit 0)

The TIMER-B Interrupt Request Flag is set by the overflow output of TIMER-B. Refer to Table 2-17.

(10) TIMER-B Interrupt Mask (IMTB:$002 bit 1)

The TIMER-B Interrupt Mask prevents an interrupt request from being generated by TIMER-B Interrupt Request Flag. Refer to Table 2-18.

Table 2-15 TIMER-A Interrupt Request Flag

TIMER-A Interrupt Request Flag	Interrupt Request
0	No
1	Yes

Table 2-16 TIMER-A Interrupt Mask

TIMER-A Interrupt Mask	Interrupt Request
0	Enable
1	Disable (Mask)

Table 2-17 TIMER-B Interrupt Request Flag

TIMER-B Interrupt Request Flag	Interrupt Request
0	No
1	Yes

Table 2-18 TIMER-B Interrupt Mask

TIMER-B Interrupt Mask	Interrupt Request
0	Enable
1	Disable (Mask)

2.7 Input/Output

The MCU has 58 I/O pins, 32 standard pins and 26 high voltage pins. One of these circuit types, CMOS, with pull-up MOS, and without pull-up MOS (NMOS open drain) can be selected for each standard pin, and one of two circuit types, with pull-down MOS and without pull-down MOS (PMOS open drain), can be selected for each high voltage pins. Since the pull-down MOS is connected to the internal V_{disp} line, the V_{disp} line must be selected from the RA_1/V_{disp} pin via mask option when at least one high voltage pin is selected as "With pull-down MOS" option. See Table 2-19 as for I/O pin circuit types.

When every input/output pin is used as an input pin, the mask option and output data must be selected in the manner specified in Table 2-20.

(1) Output Circuit Operation of "With pull-up MOS" Standard Pins

In the "with pull-up MOS" standard pin option, the circuit shown in Fig. 2-15 is used to shorten rise time of output.

When an output instruction is executed, a write pulse will be generated and applied to the R port addressed by this instruction. This pulse will switch the PMOS (B) to ON and shorten the rise time. In this case, the write pulse keeps PMOS in the ON state for one-eighth of the instruction cycle time. While the write pulse is "0", a high output level is maintained by the pull-up MOS (C).

As the $\overline{HLT}$ signal becomes "0" in stop mode, MOS (A) (B) (C) turn OFF.

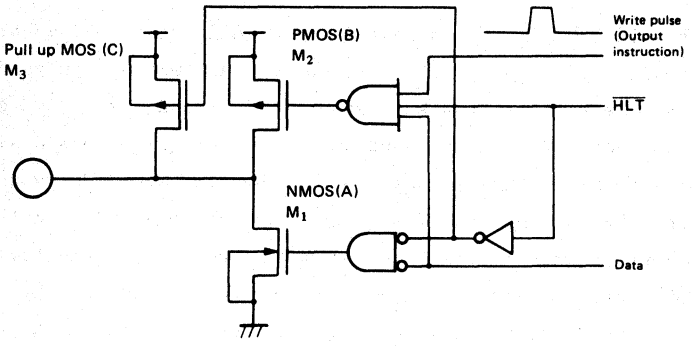
(2) D-port

The D-port is an I/O port which has 16 descrete I/O pins, each of which can be addressed independently. It can be set/reset through SED/RED and SEDD/REDD instructions, and can be tested through TD and TDD instructions. See Table 2-19 as for the classification of standard pin, high voltage pin and the I/O pin circuit types.

(3) R-ports

The R-ports are 4-bit I/O ports. The eleven R-ports in HMCS408 are composed of 20 I/O pins, 16 output-only pins, and 6 input-only pins. Data is input through LAR and LBR instructions and output through LRA and LRB instructions. The MCU will not be affected by writing into the input-only and/or non-existing ports, while invalid data will be read by reading from the output-only and/or non-existing ports.

The R_{32}, R_{33}, R_{40}, R_{41}, and R_{42} pins are multiplexed with the $\overline{INT_0}$, $\overline{INT_1}$, $\overline{SCK}$, SI, and SO pins respectively. See Table 2-19 as for the classification of standard pins, high voltage pins and selectable circuit types of I/O pins.



Features / MOS Buffer	ON Resistance Value	
	HMCS402C/AC, HMCS404C/AC	HMCS402CL, HMCS404CL
M_1	approx. 250Ω	approx. 1kΩ
M_2	approx. 1kΩ	approx. 5kΩ
M_3	approx. 40kΩ to 160kΩ (V_{CC}=5V)	approx. 75kΩ to 1MΩ(V_{CC}=3V) approx. 40kΩ to 160kΩ(V_{CC}=5V)

Features / MOS Buffer	ON Resistance Value	
	HMCS408C, HMCS408AC HMCS412C, HMCS412AC HMCS414C, HMCS414AC	HMCS408CL HMCS412CL HMCS414CL
M_1	approx. 250Ω	approx. 1kΩ
M_2	approx. 1kΩ	approx. 1.7kΩ
M_3	approx. 30kΩ to 160kΩ (V_{CC}=5V)	approx. 60kΩ to 1MΩ(V_{CC}=3V) approx. 30kΩ to 160kΩ(V_{CC}=5V)

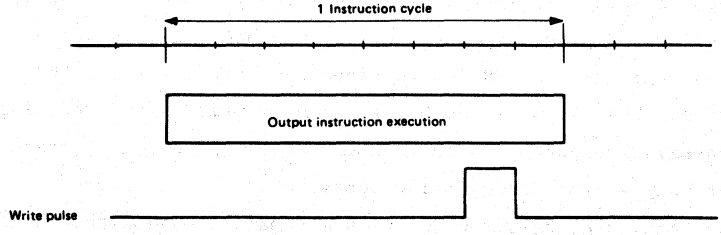
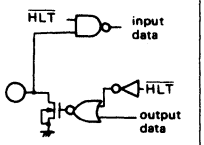
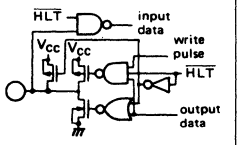
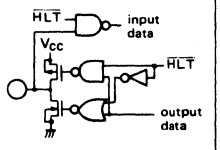
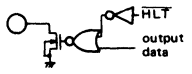
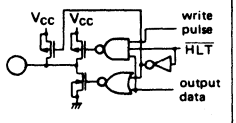
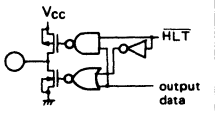
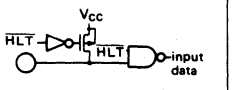
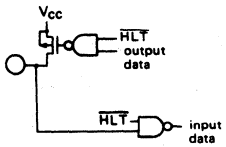
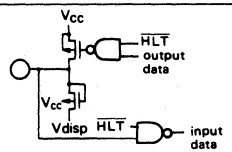
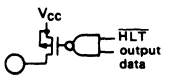
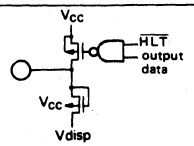
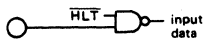
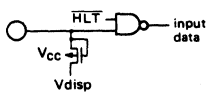
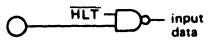


Fig. 2-15 Output Circuit Operation of Standard Pins with "with Pull-up MOS" Option

Table 2-19 I/O Pin Circuit Type

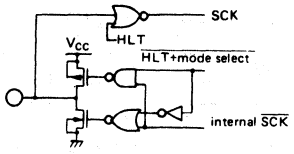
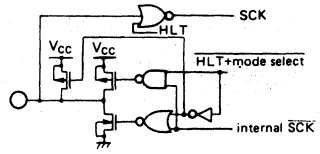
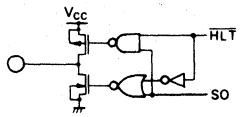
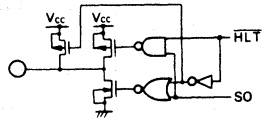
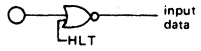
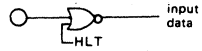
(1) HMCS402/HMCS404/HMCS408 I/O Pin Circuit Type

		Without pull-up MOS (NMOS open drain) (A)	With pull-up MOS (B)	CMOS (C)	Applied pins
Standard pins	I/O common pins				D_0 ~D_3, R_{30} ~R_{33}, R_{40} ~R_{43}, R_{50} ~R_{53}
	Output pins				R_{60}~R_{63}, R_{70}~R_{73}, R_{80}~R_{83}
	Input pins				R_{90}~R_{93}

		Without pull-down MOS (PMOS open drain) (D)	With pull-down MOS (E)	Applied pins
High voltage pins	I/O common pins			D_4~D_{15}, R_{10}~R_{13}, R_{20}~R_{23}
	Output pins			R_{00}~R_{03}
	Input pins			R_{A0}
				R_{A1}

(Note) In the stop mode, $\overline{HLT}$ signal is "0" and I/O pins are in high impedance state.

(to be continued)

		Without pull-up MOS (NMOS open drain) or CMOS (A or C)	With pull-up MOS (B)	Applied pins
Standard pins	I/O common pins			SCK (Note 2) (OUTPUT MODE)
	Output pins			SO
	Input pins			$\overline{INT_0}$, $\overline{INT_1}$, SI SCK (Note 2) (INPUT MODE)

(Note 1) In the stop mode, $\overline{HLT}$ signal is "0", HLT signal is "1" and I/O pins are in high impedance state.

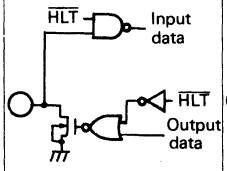
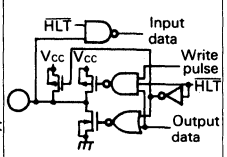
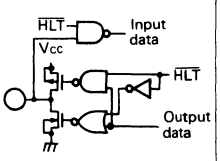
(Note 2) If the MCU is interrupted by serial interface in the external clock input mode, the $\overline{SCK}$ terminal becomes input only.

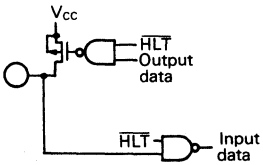
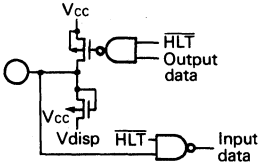
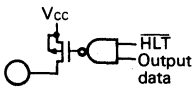
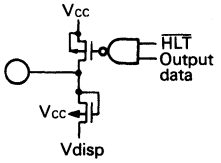
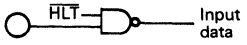
Table 2-20 Data Input from Input/Output Common Pins

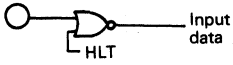
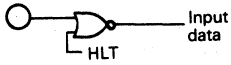
I/O pin circuit type		Possibility of Input	Available pin condition for input
Standard pins	CMOS	No	–
	Without pull-up MOS (NMOS open drain)	Yes	"1"
	With pull-up MOS	Yes	"1"
High voltage pins	Without pull-down MOS (PMOS open drain)	Yes	"0"
	With pull-down MOS	Yes	"0"

Table 2-19 I/O Pin Circuit Type

(2) HMCS412/HMCS414 I/O Pin Circuit Type

		Without pull-up MOS (NMOS open drain) (A)	With pull-up MOS (B)	CMOS (C)	Applicable pins
Standard Pins	I/O Common Pins				$D_0 - D_3$ $R3_0 - R3_3$ $R4_0 - R4_3$

		Without pull-down MOS (PMOS open drain) (D)	With pull-down MOS (E)	Applicable pins
High Voltage Pins	I/O Common Pins			$D_4 - D_{14}$ $R1_0 - R1_3$ $R2_0 - R2_3$
	Output Pins			$R0_0 - R0_3$
	Input Pins			RA_1

		Without pull-up MOS (NMOS open drain) or CMOS (A or C)	With pull-up MOS (B)	Applicable pins
Standard Pins	Input Pins			$\overline{INT_0}$ $\overline{INT_1}$

Note: In the stop mode, $\overline{HLT}$ signal is 0, HLT signal is 1 and I/O pins are in high impedance state.

I/O Pin Circuit Type		Input Possible	Input Pin State
Standard Pins	CMOS	No	—
	Without pull-up MOS (NMOS open drain)	Yes	1
	With pull-up MOS	Yes	1
High Voltage Pins	Without pull-down MOS (PMOS open drain)	Yes	0
	With pull-down MOS	Yes	0

2.8 Reset

The MCU is reset by bringing the RESET pin high. At power ON, when cancelling Stop Mode, the reset must satisfy t_{RC} for the oscillator to stabilize. In all other cases, at least two instructions cycles are required for the MCU to be reset.

Table 2-21 shows the parts to be initialized by MCU reset and the status of each after the reset has been carried out.

Table 2-21 Initial Value by MCU Reset

Items			Initial value by MCU reset	Contents
Program counter (PC)			$0000	Execute program from the top of ROM address.
Status (ST)			"1"	Enable to branch with conditional branch instructions.
Stack pointer (SP)			$3FF	Stack level is 0.
I/O pin output register	Standard pin	(A) Without pull-up MOS	"1"	Enable to input.
		(B) With pull-up MOS	"1"	Enable to input
		(C) CMOS	"1"	–
	High voltage pin	(D) Without pull-down MOS	"0"	Enable to input.
		(E) With pull-down MOS	"0"	Enable to input.
Interrupt flag	Interrupt Enable Flag (I/E)		"0"	Inhibit all interrupts.
	Interrupt Request Flag (IF)		"0"	No interrupt request.
	Interrupt Mask (IM)		"1"	Mask interrupt request.
Mode register	Port Mode Register (PMR)		"0000"	See Item "Port Mode Register".
	Serial Mode Register (SMR)		"0000"	See Item "Serial Mode Register".
	Timer Mode Register A (TMA)		"000"	See Item "Timer Mode Register A".
	Timer Mode Register B (TMB)		"0000"	See Item "Timer Mode Register B".
Timer/Counter, Serial interface	Prescaler		$000	–
	Timer/Counter A (TCA)		$00	–
	Timer/Event Counter B (TCB)		$00	–
	Timer Load Register (TLR)		$00	–
	Octal Counter		"000"	–

(Note) MCU reset affects to the rest of registers as follows:

Item		After recovering from STOP mode by MCU reset	After MCU reset except for the left condition
Carry	(CA)	The contents of the items before MCU reset are not retained. It is necessary to intialize them by software again.	The contents of the items before MCU reset are not retained. It is necessary to initialize them by software again.
Accumulator	(A)		
B Register	(B)		
W Register	(W)		
X/SPX Registers	(X/SPX)		
Y/SPY Registers	(Y/SPY)		
Serial Data Register	(SR)	Same as above	Same as above
RAM		The contents of RAM before MCU reset (just before STOP instruction) are retained.	Same as above

2.9 Internal Oscillator Circuit

Fig. 2-16 outlines the internal oscillator circuit. Through mask option, either crystal oscillator or ceramic filter oscillator can be selected as the oscillator type, as shown in Table 2-22, and refer to Table 2-23 for selection of the type. In addition, see Fig. 2-17 for selection of layout of crystal or ceramic filter. In all cases, external clock operation is available. On the HMCS408C, three divide ratios, 1/16, 1/8, and 1/4, are selectable via mask option.

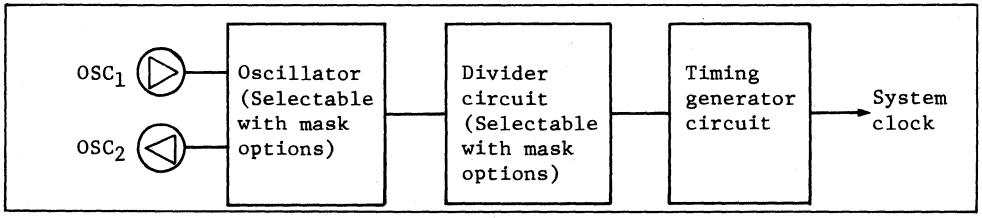


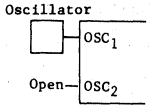
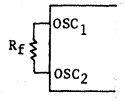
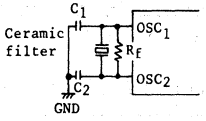
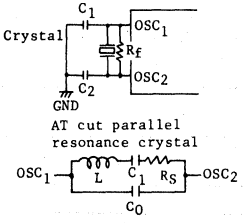
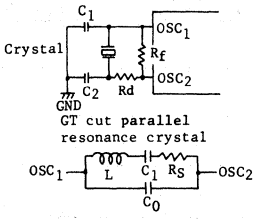
Fig. 2-16 Internal Oscillator Circuit

Table 2-22 Internal Oscillation Circuit Mask Option

		HMCS402C, HMCS404C	HMCS402AC, HMCS404AC	HMCS402CL, HMCS404CL
Oscillator	Crystal	o	o	o
	Ceramic	o	o	o
	Resistor	o	-	-
Divider	1/8	o	o	o

		HMCS408C HMCS412C HMCS414C	HMCS408CL HMCS412CL HMCS414CL	HMCS408AC HMCS412AC HMCS414AC
Oscillator	Crystal	o	o	o
	Ceramic	o	o	o
Divider	1/16	-	o	-
	1/8	o	o	-
	1/4	o	o	o

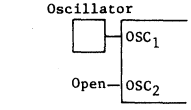
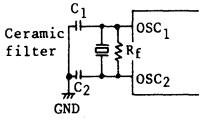
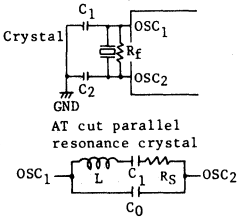
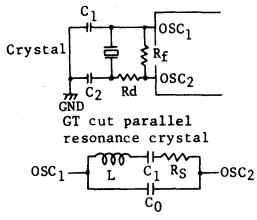
Table 2-23 Examples of Oscillator Circuit

	Circuit configuration	Circuit constants		
		HMCS402C, HMCS404C	HMCS402CL, HMCS404CL	HMCS402AC, HMCS404AC
External clock operation				
Resistor oscillator		R_f=20kΩ±20%		
Ceramic filter oscillator		Ceramic filter CSA4.00MG (Murata) R_f:1MΩ±20% C_1:30pF±20% C_2:30pF±20%	Ceramic filter CSA2.000MK (Murata) R_f:1MΩ±20% C_1:30pF±20% C_2:30pF±20%	Ceramic filter CSA6.00MG (Murata) R_f:1MΩ±20% C_1:30pF±20% C_2:30pF±20%
Crystal oscillator		R_f:1MΩ±20% C_1:10∿22pF±20% C_2:10∿22pF±20% Crystal: equivalent circuit to the left C_0:7pF max. R_S:60Ω max. f:2.0∿4.5MHz		R_f:1MΩ±20% C_1:10∿22pF±20% C_2:10∿22pF±20% Crystal: equivalent circuit to the left C_0:7pF max. R_S:100Ω max. f:2.0∿6.2MHz
			R_f:2MΩ±20% C_1:10∿22pF±20% C_2:10∿22pF±20% Crystal: equivalent circuit to the left C_0:7pF max. R_S:100Ω max. f:2.0∿2.25MHz	

(Note 1) On the crystal and ceramic filter resonator, the upper circuit parameters are the one recommended by crystal or ceramic filter maker. The circuit parameters are changed by crystal, ceramic filter resonator and the floated capacitance in designing the board. In employing the resonator, please consult with the engineers of crystal or ceramic filter maker to determine the circuit parameter.

(Note 2) Wiring among OSC_1, OSC_2 and elements should be as short as possible, and never cross the other wirings. Refer to the recommendable layout of crystal and ceramic filter.

Table 2-23 Examples of Oscillator Circuit

	Circuit configuration	Circuit constants		
		HMCS408C HMCS412C HMCS414C	HMCS408CL HMCS412CL HMCS414CL	HMCS408AC HMCS412AC HMCS414AC
External clock operation				
Ceramic filter oscillator		Ceramic filter CSA4.00MG CSA2.000MK (Murata) R_f:1MΩ±20% C_1: 30pF±20% C_2: 30pF±20%	Ceramic filter CSA2.000MK CSA4.00MG (Murata) R_f:1MΩ±20% C_1:30pF±20% C_2:30pF±20%	Ceramic filter CSA6.00MG (Murata) R_f:1MΩ±20% C_1:30pF±20% C_2:30pF±20%
Crystal oscillator		R_f:1MΩ±20% C_1:10∿22pF±20% C_2:10∿22pF±20% Crystal: equivalent circuit to the left C_0:7pF max. R_S:100Ω max. f:1.0-4.5MHz		R_f:1MΩ±20% C_1:10∿22pF±20% C_2:10∿22pF±20% Crystal: equivalent circuit to the left C_0:7pF max. R_S:100Ω max. f:1.0-4.5MHz
			R_f:2MΩ±20% C_1:10∿22pF±20% C_2:10∿22pF±20% Crystal: equivalent circuit to the left C_0:7pF max. R_S:100Ω max. f:1.0-2.25MHz	

(Note 1) On the crystal and ceramic filter resonator, the upper circuit parameters are the one recommended by crystal or ceramic filter maker. The circuit parameters are changed by crystal, ceramic filter resonator and the floated capacitance in designing the board. In employing the resonator, please consult with the engineers of crystal or ceramic filter maker to determine the circuit parameter.

(Note 2) Wiring among OSC_1, OSC_2 and elements should be as short as possible, and never cross the other wirings. Refer to the recommendable layout of crystal and ceramic filter.

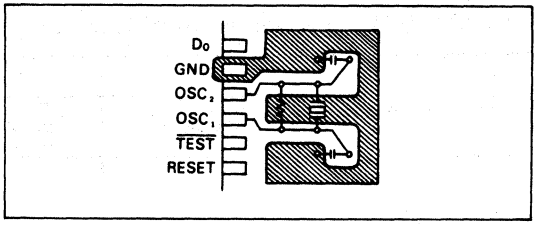


Fig. 2-17 Recommended Layout of Crystal and Ceramic Filter

2.10 Low Power Dissipation Mode

The MCU has two low power dissipation modes, Standby Mode and Stop Mode. Their functions are shown in Table 2-24, and a mode transition is shown in Fig. 2-18.

Table 2-24 Low Power Dissipation Mode Function

Low Power Dissipation Mode	Instruction	Condition							Recovering method
		Oscillator circuit	Instruction execution	Register, Flag	Interrupt function	RAM	Input/ Output pin	Timer/ Counter, Serial Interface	
Standby mode	SBY instruction	Active	Stop	Retained	Active	Retained	Retained*3)	Active	RESET Input, Interrupt request
Stop mode	STOP instruction	Stop	Stop	RESET*1)	Stop	Retained	High*2) impedance	Stop	RESET Input

*1) The MCU recovers from STOP mode by RESET input. Refer to Table 2-21 for the contents of the flags and registers.

*2) A high voltage pin with a pull-down MOS is tied to the V_{disp} power supply through the pull-down MOS. As the pull-down MOS keeps ON, a pull-down MOS current flows when a difference between the pin voltage and the V_{disp} voltage exists. This is the additional current to the current dissipation in Stop Mode (Istop).

*3) As an I/O circuit is active, an I/O current possibly flows according to the state of I/O pin in Standby Mode. This is the additional current to the current dissipation in Standby Mode (I_{SBY1}, I_{SBY2}).

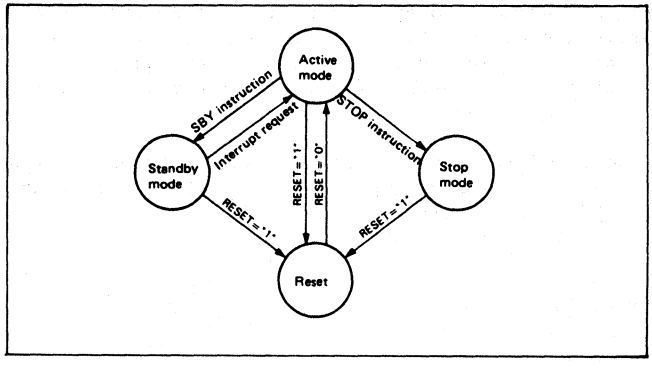


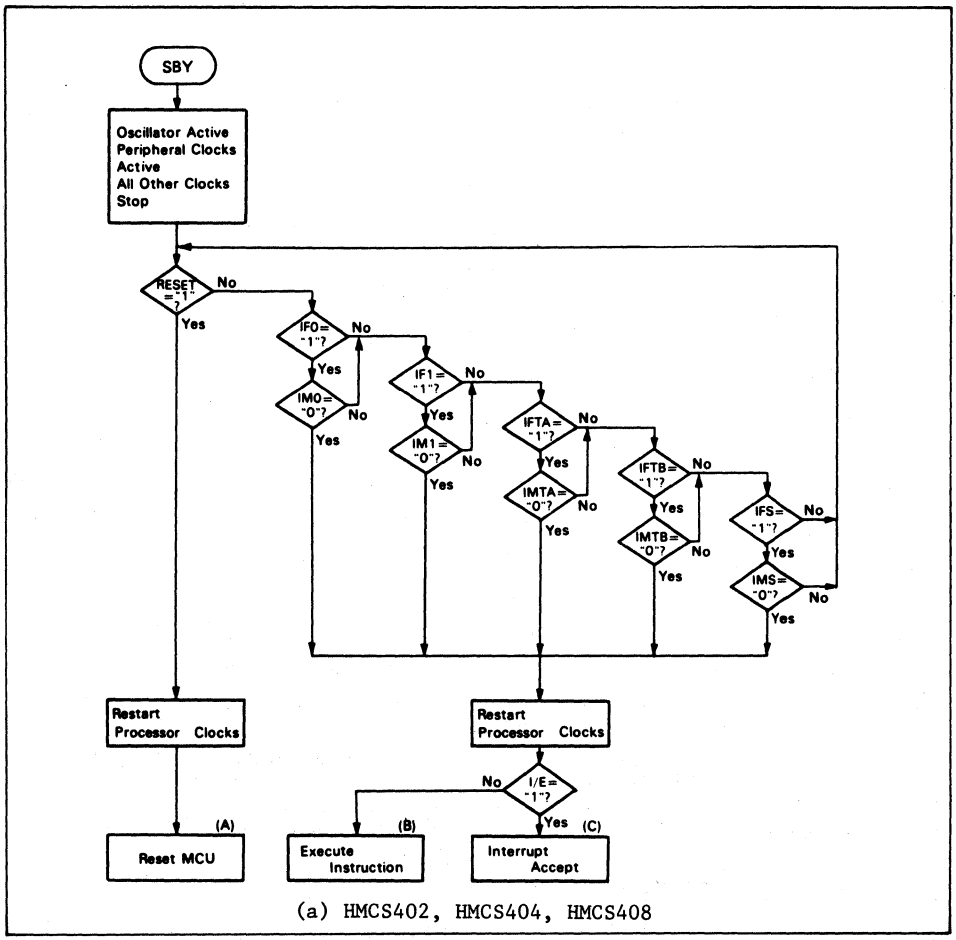
Fig. 2-18 MCU Operation Mode Transition

(1) Standby Mode

Executing an SBY instruction puts the MCU into Standby Mode. In Standby Mode, the oscillator circuit is active and the timer, counter and serial interface continue working. On the other hand, the CPU stops since the clock related to the instruction execution stops. Registers, RAM and I/O pins retain the state they were in just before the MCU went into Standby Mode.

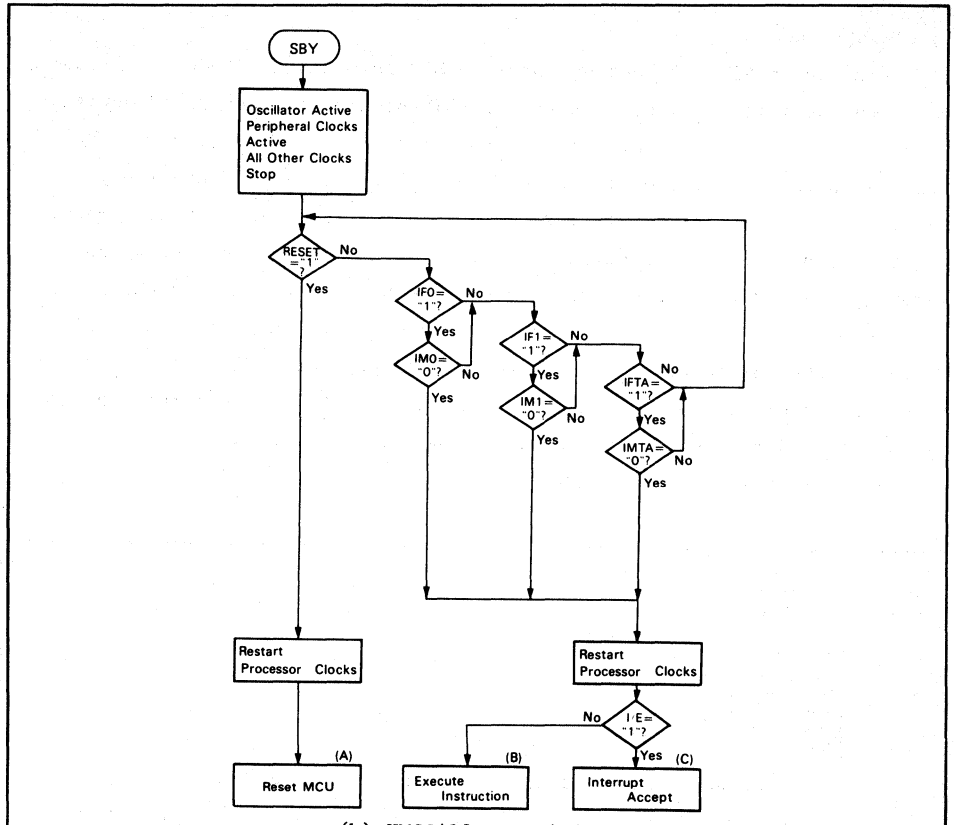
Standby Mode may be cancelled by inputting RESET or by executing an interrupt request. In the former case the MCU is reset, and in the later case, the MCU becomes an active mode and executes the next instruction following the SBY instruction. If the Interrupt Enable Flag is "1" at this time, the interrupt is executed, while if it is "0", the interrupt request is put on hold and normal instruction execution continues.

Fig. 2-19 shows the flowchart of the Standby Mode.



(a) HMCS402, HMCS404, HMCS408

Fig. 2-19 MCU Operating Flowchart



(b) HMCS412, HMCS414

Fig. 2-19 MCU Operating Flowchart

(2) Stop Mode

Executing a STOP instruction brings the MCU into Stop Mode, in which the oscillator circuit and every function of the MCU stop.

Stop Mode may be cancelled by resetting the MCU. At this time, as shown in Fig. 2-20, reset input must be applied at least to t_{RC} for oscillation to stabilize. (Refer to "AC CHARACTERISTICS"). After Stop Mode is cancelled, RAM retains the state it was in just before the MCU went into Stop Mode, but the Accumulator, B Register, W Register, X/SPX Registers, Y/SPY Registers, Carry and Serial Data Register may not retain their contents.

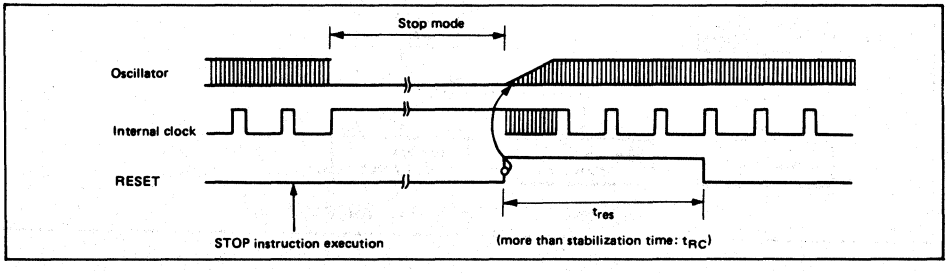


Fig. 2-20 Timing Chart of Recovering from Stop Mode

3. INSTRUCTION SYSTEM

3.1 RAM Addressing Mode

As shown in Fig. 3-1, the MCU has three RAM addressing modes, i.e. Register Indirect Addressing, Direct Addressing and Memory Register Addressing.

(1) Register Indirect Addressing

The total of the W Register, X Register, and Y Register contents (10 bits) is used as the RAM address.

(2) Direct Addressing

The direct addressing instruction consists of two words, with the second word (10 bits) following the Op-code (the first word) used as the RAM address.

(3) Memory Register Addressing

The Memory Register (16 digits from $020 to $02F) is accessed by executing LAMR and XMRA instructions.

3.2 ROM Addressing Mode and P Instructions

The MCU has four kinds of ROM addressing modes as shown in Fig. 3-2.

(1) Direct Addressing Mode

The program can be branched to any address in the ROM memory space by executing JMPL, BRL or CALL instructions. These instructions replace the 14 program counter bits (PC_{13} to PC_0) with 14-bit immediate data.

(2) Current Page Addressing Mode

The MCU has 8 pages of ROM (256 words per page). By executing a BR instruction, the program can be branched to an address in current page. This instruction replaces the low-order eight bits of the program counter (PC_7 to PC_0) with 8-bit immediate data.

However, when BR is on page boundary (256n + 255), executing a BR instruction transfers the PC contents to the next page according to the hardware architecture. Consequently, the program is branched to the next page when the BR on a page boundary is used. The outline is given in Fig. 3-3. The HMCS400 series cross macro assembler has an automatic paging facility for ROM pages.

(3) Zero Page Addressing Mode

By executing a CAL instruction, the program can be branched to the zero page subroutine area, which is located in the $0000-$003F address area. When a CAL instruction is executed, 6-bit immediate data is placed in the low-order six bits of the program counter (PC_5 to PC_0) and "0s" are placed in the high-order eight bits (PC_{13} to PC_6).

(4) Table Data Addressing

By executing a TBR instruction, the program can be branched to the address determined by the contents of the 4-bit immediate data, accumulator and B register.

(5) P Instruction

By executing a P instruction, ROM data addressed by Table Data Addressing can be referred to as shown in Fig. 3-4. When bit 8 in the referred ROM data is "1", 8 bits of referred ROM data are written into the accumulator and B Register. When bit 9 is "1", 8 bits of referred ROM data are written into the R_1 and R_2 port output register. When both bits 8 and 9 are "1", ROM data are written into the accumulator and B Register and also to the R_1 and R_2 port output register at the same time.

The P instruction has no effect on the program counter.

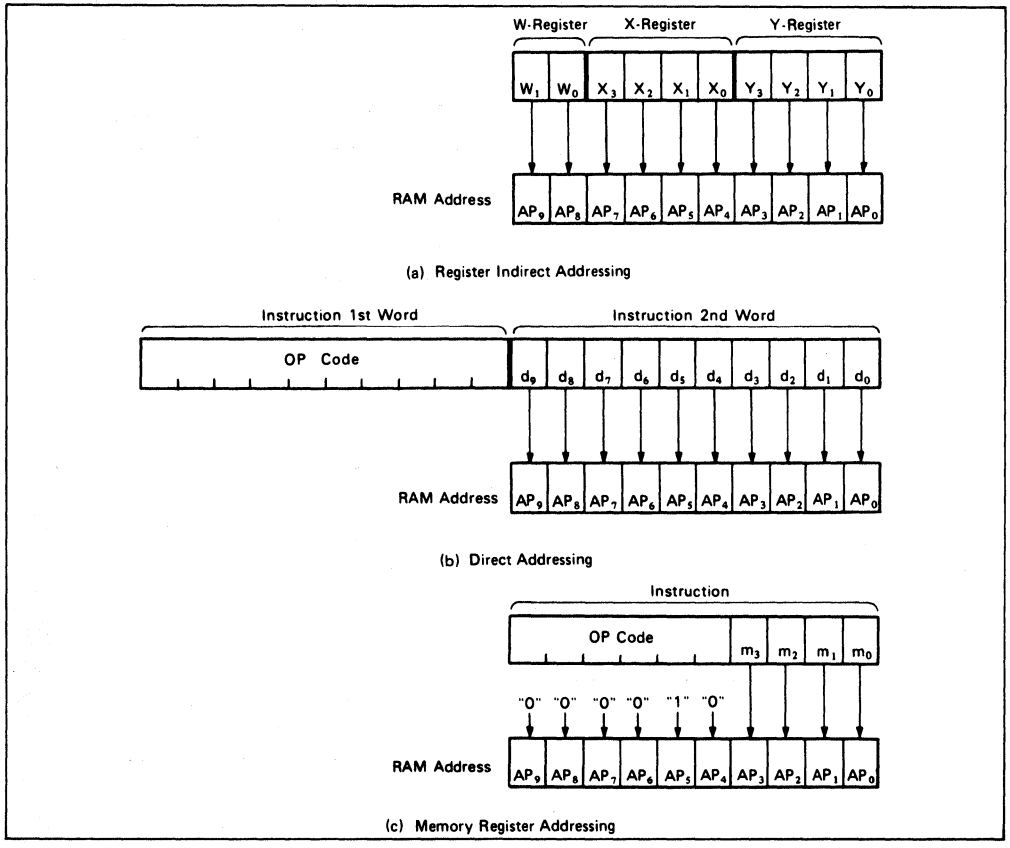


Fig. 3-1 RAM Addressing Mode

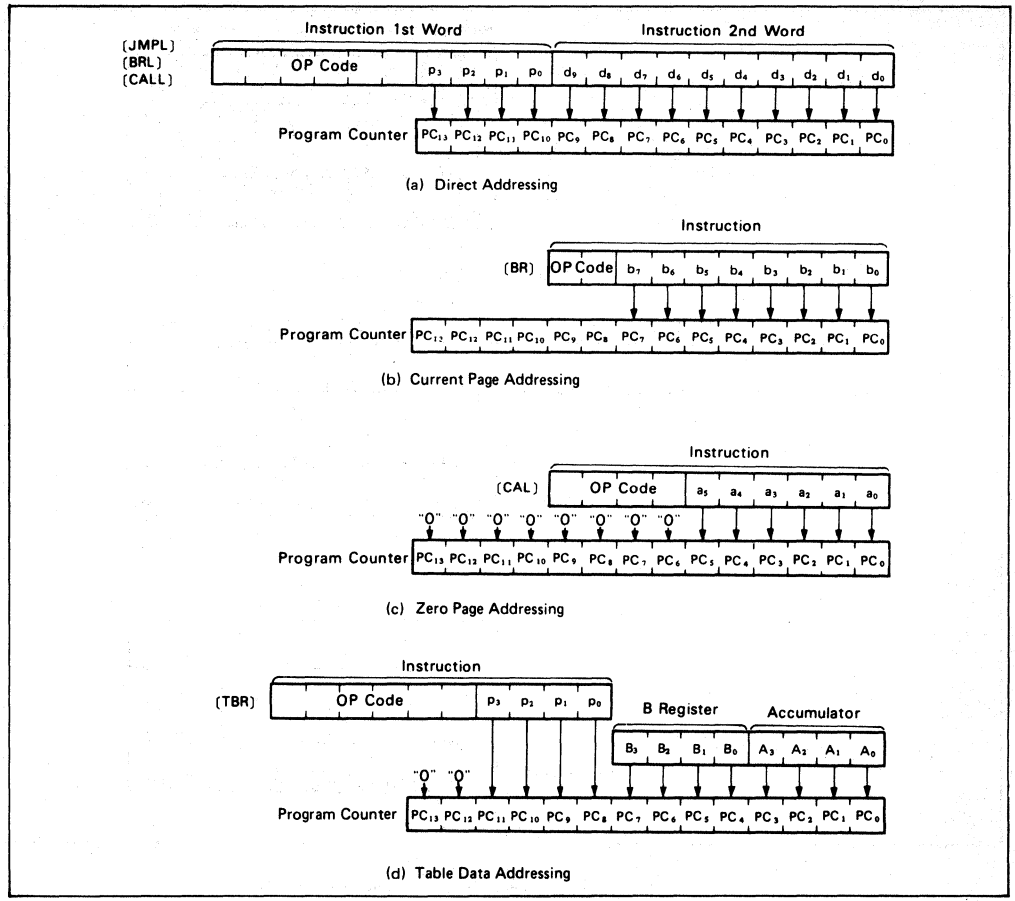


Fig. 3-2 ROM Addressing Mode

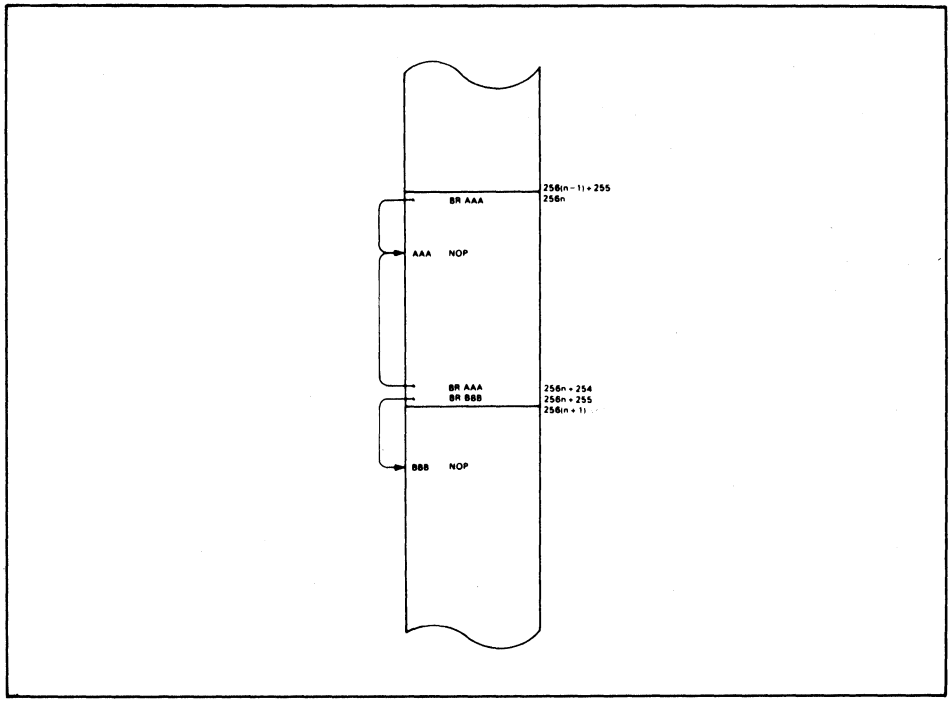


Fig. 3-3 The Branch Destination by BR Instruction on the Boundary between Pages

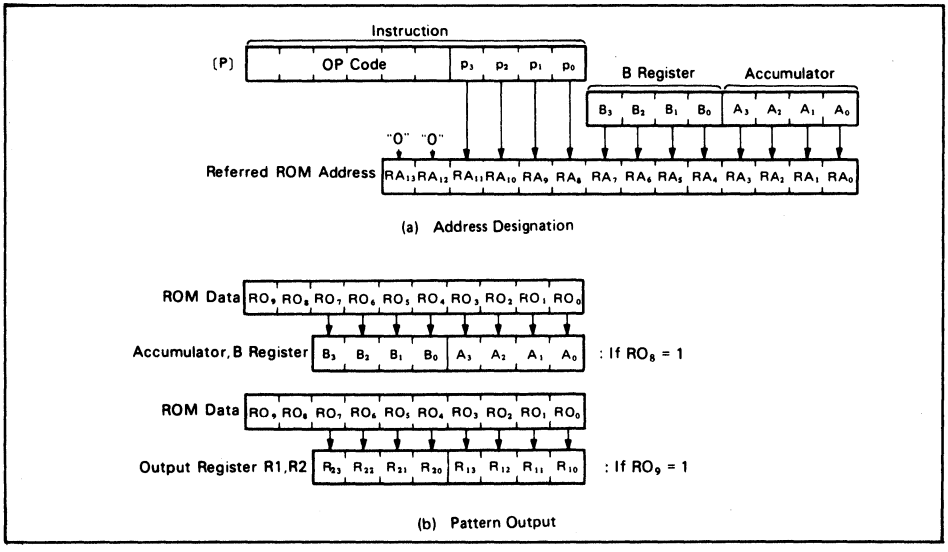


Fig. 3-4 P Instruction

3.3 Instruction Set

The HMCS400 series microcomputers provide 99 instructions which are classified into 10 groups as follows;

(1) Immediate Instruction(Table 3-1)
(2) Register-to-Register Instruction(Table 3-2)
(3) RAM Address Instruction(Table 3-3)
(4) RAM Register Instruction(Table 3-4)
(5) Arithmetic Instruction(Table 3-5)
(6) Compare Instruction(Table 3-6)
(7) RAM Bit Manipulation Instruction(Table 3-7)
(8) ROM Address Instruction(Table 3-8)
(9) Input/Output Instruction(Table 3-9)
(10) Control Instruction (Table 3-10)

(Note) In the HMCS412 and HMCS414, there is not the STS instructions, because these ones have not the serial interface.

Table 3-1 Immediate Instruction

OPERATION	MNEMONIC	OPERATION CODE	FUNCTION	STATUS	WORD/CYCLE
Load A from Immediate	LAI i	1 0 0 0 1 1 i_3 i_2 i_1 i_0	i→A		1/1
Load B from Immediate	LBI i	1 0 0 0 0 0 i_3 i_2 i_1 i_0	i→B		1/1
Load Memory from Immediate	LMID i,d	0 1 1 0 1 0 i_3 i_2 i_1 i_0 d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	i→M		2/2
Load Memory from Immediate, Increment Y	LMIIY i	1 0 1 0 0 1 i_3 i_2 i_1 i_0	i→M,Y+1→Y	NZ	1/1

Table 3-2 Register-to-Register Instruction

OPERATION	MNEMONIC	OPERATION CODE	FUNCTION	STATUS	WORD/CYCLE
Load A from B	LAB	0 0 0 1 0 0 1 0 0 0	B→A		1/1
Load B from A	LBA	0 0 1 1 0 0 1 0 0 0	A→B		1/1
Load A from Y	LAY	0 0 1 0 1 0 1 1 1 1	Y→A		1/1
Load A from SPX	LASPX	0 0 0 1 1 0 1 0 0 0	SPX→A		1/1
Load A from SPY	LASPY	0 0 0 1 0 1 1 0 0 0	SPY→A		1/1
Load A from MR	LAMR m	1 0 0 1 1 1 m_3 m_2 m_1 m_0	MR(m)→A		1/1
Exchange MR and A	XMRA m	1 0 1 1 1 1 m_3 m_2 m_1 m_0	MR(m)↔A		1/1

Table 3-3 RAM Address Instruction

OPERATION	MNEMONIC	OPERATION CODE	FUNCTION	STATUS	WORD/CYCLE
Load W from Immediate	LWI i	0 0 1 1 1 1 0 0 i_1 i_0	i→W		1/1
Load X from Immediate	LXI i	1 0 0 0 1 0 i_3 i_2 i_1 i_0	i→X		1/1
Load Y from Immediate	LYI i	1 0 0 0 0 1 i_3 i_2 i_1 i_0	i→Y		1/1
Load X from A	LXA	0 0 1 1 1 0 1 0 0 0	A→X		1/1
Load Y from A	LYA	0 0 1 1 0 1 1 0 0 0	A→Y		1/1
Increment Y	IY	0 0 0 1 0 1 1 1 0 0	Y+1→Y	NZ	1/1
Decrement Y	DY	0 0 1 1 0 1 1 1 1 1	Y−1→Y	NB	1/1
Add A to Y	AYY	0 0 0 1 0 1 0 1 0 0	Y+A→Y	OVF	1/1
Subtract A from Y	SYY	0 0 1 1 0 1 0 1 0 0	Y−A→Y	NB	1/1
Exchange X and SPX	XSPX	0 0 0 0 0 0 0 0 0 1	X↔SPX		1/1
Exchange Y and SPY	XSPY	0 0 0 0 0 0 0 0 1 0	Y↔SPY		1/1
Exchange X and SPX,Y and SPY	XSPXY	0 0 0 0 0 0 0 0 1 1	X↔SPX,Y↔SPY		1/1

Table 3-4 RAM Register Instruction

OPERATION	MNEMONIC	OPERATION CODE	FUNCTION	STATUS	WORD/CYCLE
Load A from Memory	LAM(XY)	0 0 1 0 0 1 0 0 y x	M→A, $(^{X:SPX}_{Y:SPY})$		1/1
Load A from Memory	LAMD d	0 1 1 0 0 1 0 0 0 0 d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	M→A		2/2
Load B from Memory	LBM(XY)	0 0 0 1 0 0 0 0 y x	M→B, $(^{X:SPX}_{Y:SPY})$		1/1
Load Memory from A	LMA(XY)	0 0 1 0 0 1 0 1 y x	A→M, $(^{X:SPX}_{Y:SPY})$		1/1
Load Memory from A	LMAD d	0 1 1 0 0 1 0 1 0 0 d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	A→M		2/2
Load Memory from A, Increment Y	LMAIY(X)	0 0 0 1 0 1 0 0 0 x	A→M,Y+1→Y(X:SPX)	NZ	1/1
Load Memory from A, Decrement Y	LMADY(X)	0 0 1 1 0 1 0 0 0 x	A→M,Y−1→Y(X:SPX)	NB	1/1
Exchange Memory and A	XMA(XY)	0 0 1 0 0 0 0 0 y x	M↔A, $(^{X:SPX}_{Y:SPY})$		1/1
Exchange Memory and A	XMAD d	0 1 1 0 0 0 0 0 0 0 d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	M↔A		2/2
Exchange Memory and B	XMB(XY)	0 0 1 1 0 0 0 0 y x	M↔B, $(^{X:SPX}_{Y:SPY})$		1/1

Note) (XY) and (X) have the meaning as follows:

(1) The instructions with (XY) have 4 mnemonics and 4 object codes for each. (example of LAM (XY) is given below.) The op-code X or Y is assembled as follows.

MNEMONIC	y	x	FUNCTION
LAM	0	0	
LAMX	0	1	X↔SPX
LAMY	1	0	Y↔SPY
LAMXY	1	1	X↔SPX, Y↔SPY

(2) The instructions with (X) have 2 mnemonics and 2 object codes for each. (example of LMAIY(X) is given below.) The op-code X is assembled as follows.

MNEMONIC	x	FUNCTION
LMAIY	0	
LMAIYX	1	X↔SPX

Table 3-5 Arithmetic Instruction

OPERATION	MNEMONIC	OPERATION CODE	FUNCTION	STATUS	WORD/CYCLE
Add Immediate to A	AI i	101000i_3 i_2 i_1 i_0	A+i→A	OVF	1/1
Increment B	IB	000100110 0	B+1→B	NZ	1/1
Decrement B	DB	001100111 1	B−1→B	NB	1/1
Decimal Adjust for Addition	DAA	001010011 0			1/1
Decimal Adjust for Subtraction	DAS	001010101 0			1/1
Negate A	NEGA	000110000 0	$\bar{A}$+1→A		1/1
Complement B	COMB	010100000 0	$\bar{B}$→B		1/1
Rotate Right A with Carry	ROTR	001010000 0			1/1
Rotate Left A with Carry	ROTL	001010000 1			1/1
Set Carry	SEC	001111011 1	1→CA		1/1
Reset Carry	REC	001111011 0	0→CA		1/1
Test Carry	TC	000110111 1		CA	1/1
Add A to Memory	AM	000000010 0	M+A→A	OVF	1/1
Add A to Memory	AMD d	010000010 00 d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	M+A→A	OVF	2/2
Add A to Memory with Carry	AMC	000001100 0	M+A+CA→A OVF→CA	OVF	1/1
Add A to Memory with Carry	AMCD d	010001100 00 d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	M+A+CA→A OVF→CA	OVF	2/2
Subtract A from Memory with Carry	SMC	001001100 0	M−A−$\overline{CA}$→A NB→CA	NB	1/1
Subtract A from Memory with Carry	SMCD d	011001100 00 d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	M−A−$\overline{CA}$→A NB→CA	NB	2/2
OR A and B	OR	010100010 0	A∪B →A		1/1
AND Memory with A	ANM	001001111 0	A∩M→A	NZ	1/1
AND Memory with A	ANMD d	011001111 00 d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	A∩M→A	NZ	2/2
OR Memory with A	ORM	000000110 0	A∪M→A	NZ	1/1
OR Memory with A	ORMD d	010000110 00 d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	A∪M→A	NZ	2/2
EOR Memory with A	EORM	000001110 0	A⊕M→A	NZ	1/1
EOR Memory with A	EORMD d	010001110 00 d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	A⊕M→A	NZ	2/2

Table 3-6 Compare Instruction

OPERATION	MNEMONIC	OPERATION CODE	FUNCTION	STATUS	WORD/CYCLE
Immediate Not Equal to Memory	INEM i	000010 i_3 i_2 i_1 i_0	i / M	NZ	1/1
Immediate Not Equal to Memory	INEMD i,d	010010 i_3 i_2 i_1 i_0 / d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	i / M	NZ	2/2
A Not Equal to Memory	ANEM	0000000100	A ≠ M	NZ	1/1
A Not Equal to Memory	ANEMD d	0100000100 / d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	A / M	NZ	2/2
B Not Equal to Memory	BNEM	0001000100	B / M	NZ	1/1
Y Not Equal to Immediate	YNEI i	000111 i_3 i_2 i_1 i_0	Y ≠ i	NZ	1/1
Immediate Less or Equal to Memory	ILEM i	000011 i_3 i_2 i_1 i_0	i ≤ M	NB	1/1
Immediate Less or Equal to Memory	ILEMD i,d	010011 i_3 i_2 i_1 i_0 / d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	i ≤ M	NB	2/2
A Less or Equal to Memory	ALEM	0000010100	A ≤ M	NB	1/1
A Less or Equal to Memory	ALEMD d	0100010100 / d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	A ≤ M	NB	2/2
B Less or Equal to Memory	BLEM	0011000100	B ≤ M	NB	1/1
A Less or Equal to Immediate	ALEI i	101011 i_3 i_2 i_1 i_0	A ≤ i	NB	1/1

Table 3-7 RAM Bit Manipulation Instruction

OPERATION	MNEMONIC	OPERATION CODE	FUNCTION	STATUS	WORD/CYCLE
Set Memory Bit	SEM n	0010000 1 n_1 n_0	1→M(n)		1/1
Set Memory Bit	SEMD n,d	0110000 1 n_1 n_0 / d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	1→M(n)		2/2
Reset Memory Bit	REM n	0010000 0 n_1 n_0	0→M(n)		1/1
Reset Memory Bit	REMD n,d	0110000 0 n_1 n_0 / d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	0→M(n)		2/2
Test Memory Bit	TM n	0010001 1 n_1 n_0		M(n)	1/1
Test Memory Bit	TMD n,d	0110001 1 n_1 n_0 / d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0		M(n)	2/2

Table 3-8 ROM Address Instruction

OPERATION	MNEMONIC	OPERATION CODE	FUNCTION	STATUS	WORD/CYCLE
Branch on Status 1	BR b	1 1 b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0		1	1/1
Long Branch on Status 1	BRL u	0101111 p_3 p_2 p_1 p_0 / d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0		1	2/2
Long Jump Unconditionally	JMPL u	0101101 p_3 p_2 p_1 p_0 / d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0			2/2
Subroutine Jump on Status 1	CAL a	0111 a_5 a_4 a_3 a_2 a_1 a_0		1	1/2
Long Subroutine Jump on Status 1	CALL u	0101110 p_3 p_2 p_1 p_0 / d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0		1	2/2
Table Branch	TBR p	001011 p_3 p_2 p_1 p_0			1/1
Return from Subroutine	RTN	0000010000			1/3
Return from Interrupt	RTNI	0000010001	1→I/E CA RESTORE	ST	1/3

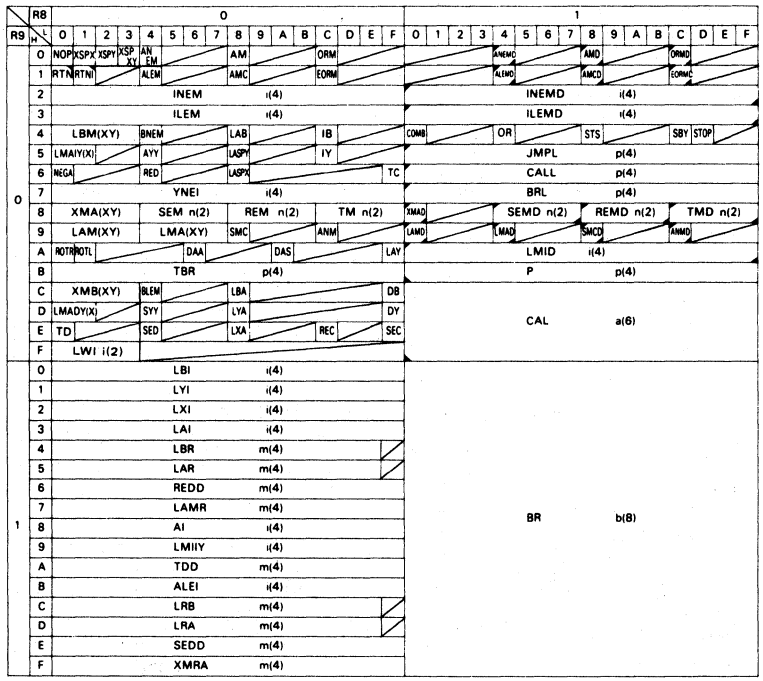
Table 3-9 Input/Output Instruction

OPERATION	MNEMONIC	OPERATION CODE	FUNCTION	STATUS	WORD/CYCLE
Set Discrete I/O Latch	SED	0011100100	1→D(Y)		1/1
Set Discrete I/O Latch Direct	SEDD m	101110 m_3 m_2 m_1 m_0	1→D(m)		1/1
Reset Discrete I/O Latch	RED	0001100100	0→D(Y)		1/1
Reset Discrete I/O Latch Direct	REDD m	100110 m_3 m_2 m_1 m_0	0→D(m)		1/1
Test Discrete I/O Latch	TD	0011100000		D(Y)	1/1
Test Discrete I/O Latch Direct	TDD m	101010 m_3 m_2 m_1 m_0		D(m)	1/1
Load A from R-Port Register	LAR m	100101 m_3 m_2 m_1 m_0	R(m)→A		1/1
Load B from R-Port Register	LBR m	100100 m_3 m_2 m_1 m_0	R(m)→B		1/1
Load R-Port Register from A	LRA m	101101 m_3 m_2 m_1 m_0	A→R(m)		1/1
Load R-Port Register from B	LRB m	101100 m_3 m_2 m_1 m_0	B→R(m)		1/1
Pattern Generation	P p	011011 p_3 p_2 p_1 p_0			1/2

Table 3-10 Control Instruction

OPERATION	MNEMONIC	OPERATION CODE	FUNCTION	STATUS	WORD / CYCLE
No Operation	NOP	0000000000			1/1
Start Serial	STS	0101001000			1/1
Stand-by Mode	SBY	0101001100			1/1
Stop Mode	STOP	0101001101			1/1

Table 3-11 Op-Code Map



3.4 Instruction Table

Three kinds of instruction tables are shown to explain the instructions which construct Instruction System of the HMCS400 series.

3.4.1 Functional Table

Instructions are classified by function. The table shows mnemonic code and simple explanation of the function, and shows functional comparison between HMCS40 series and HMCS400 series.

CATEGORY	MNEMONIC	FUNCTION	STATUS	400	40
REGISTER TO REGISTER	LAB	B → A		*	*
	LBA	A → B		*	*
	LAY	Y → A		*	*
	LASPX	SPX → A		*	*
	LASPY	SPY → A		*	*
	LAMR m	MR(m) → A		*	
	XMRA m	MR(m) ⟷ A		*	*
RAM ADDRESS	LXA	A → X		*	*
	LYA	A → Y		*	*
	LXI i	i → X		*	*
	LYI i	i → Y		*	*
	LWI i	i → W		*	
	IY	Y + 1 → Y	NZ	*	*
	DY	Y - 1 → Y	NB	*	*
	AYY	Y + A → Y	OVF	*	*
	SYY	Y - A → Y	NB	*	*
	XSPX	X ⟷ SPX		*	*
	XSPY	Y ⟷ SPY		*	*
	XSPXY	X ⟷ SPX, Y ⟷ SPY		*	*
REGISTER TO RAM	LAM(XY)	M → A, (X⟷SPX, Y⟷SPY)		*	*
	LAMD d	M(d) → A		*	
	LBM(XY)	M → B, (X⟷SPX, Y⟷SPY)		*	*
	XMA(XY)	M → A, (X⟷SPX, Y⟷SPY)		*	*
	XMAD d	M(d) → A		*	
	XMB(XY)	M ⟷ B, (X⟷SPX, Y⟷SPY)		*	*
	LMA(XY)	A → M, (X⟷SPX, Y⟷SPY)		*	
	LMAD d	A → M(d)		*	
	LMAIY(X)	A → M, Y+1 → Y, (X⟷SPX)	NZ	*	*
	LMADY(X)	A → M, Y-1 → Y, (X⟷SPX)	NB	*	*
IMMEDIATE	LMIIY i	i → M, Y+1 → Y	NZ	*	*
	LMID i,d	i → M(d)		*	
	LAI i	i → A		*	*
	LBI i	i → B		*	*
ARITHMETIC	AI i	A + i → A	OVF	*	*
	IB	B + 1 → B	NZ	*	*
	DB	B - I → B	NB	*	*
	AMC	M + A + CA → A, OVF → CA	OVF	*	*
	AMCD d	M(d) + A + CA → A, OVF → CA	OVF	*	
	SMC	M - A - $\overline{CA}$ → A, NB → CA	NB	*	*
	SMCD d	M(d) - A - $\overline{CA}$ → A, NB → CA	NB	*	
	AM	M + A → A	OVF	*	*
	AMD d	M(d) + A → A	OVF	*	
	DAA	Decimal Adjust (Add)		*	*
	DAS	Decimal Adjust (Subtract)		*	*
	NEGA	$\overline{A}$ + 1 → A		*	*
	COMB	$\overline{B}$ → B		*	*
	SEC	1 → CA		*	*
	REC	0 → CA		*	*
	TC	Test CA	CA	*	*
	ROTR	Rotate Right A with Carry		*	*
	ROTL	Rotate Left A with Carry		*	*
	OR	A ∪ B → A		*	*
	ANM	A ∩ M → A	NZ	*	
	ANMD d	A ∩ M(d) → A	NZ	*	
	ORM	A ∪ M → A	NZ	*	
	ORMD d	A ∪ M(d) → A	NZ	*	
	EORM	A ⊕ M → A	NZ	*	
	EORMD d	A ⊕ M(d) → A	NZ	*	

(to be continued)

CATEGORY	MNEMONIC	FUNCTION	STATUS	400	40
	INEM i	i ≠ M	NZ	*	*
	INEMD i,d	i ≠ M(d)	NZ	*	
	ANEM	A ≠ M	NZ	*	*
	ANEMD d	A ≠ M(d)	NZ	*	
	BNEM	B ≠ M	NZ	*	*
	YNEI i	Y ≠ i	NZ	*	*
COMPARE	ILEM i	i ≦ M	NB	*	
	ILEMD i,d	i ≦ M(d)	NB	*	
	ALEM	A ≦ M	NB	*	*
	ALEMD d	A ≦ M(d)	NB	*	
	BLEM	B ≦ M	NB	*	*
	ALEI i	A ≦ i	NB	*	*
	SEM n	1 → M(n)		*	*
	SEMD n,d	1 → M(d,n)		*	
RAM BIT	REM n	0 → M(n)		*	*
MANIPULATION	REMD n,d	0 → M(d, n)		*	
	TM n	Test M(n)	M(n)	*	*
	TMD n,d	Test M(d, n)	M(d, n)	*	
	BR b	Branch on Status 1	1	*	*
	BRL u	Long Branch on Status 1	1	*	
	JMPL u	Long Jump Unconditionally		*	
ROM ADDRESS	CAL a	Subroutine Jump on Status 1	1	*	*
	CALL u	Long Subroutine Jump on Status 1	1	*	
	TBR p	Table Branch		*	*
	RTN	Return from Subroutine		*	*
	LPU u	Load Program Counter Upper on Status 1			*
	SELE	1 → I/E			*
	SEIF0	1 → IF0			*
	SEIF1	1 → IF1			*
	SETF	1 → TF			*
	SECF	1 → CF			*
	REIE	0 → I/E			*
	REIF0	0 → IF0			*
	REIF1	0 → IF1			*
	RETF	0 → TF			*
INTERRUPT	RECF	0 → CF			*
	TI0	Test INT0	INT0		*
	TI1	Test INT1	INT1		*
	TIF0	Test IF0	IF0		*
	TIF1	Test IF1	IF1		*
	TTF	Test TF	TF		*
	LTI i	i → Timer/Counter			*
	LTA	A → Timer/Counter			*
	LAT	Timer/Counter → A			*
	RTNI	Return from Interrupt	ST	*	*
	SED	1 → D(Y)		*	*
	RED	0 → D(Y)		*	*
	TD	Test Discrete I/O Latch D(Y)	D(Y)	*	*
	SEDD m	1 → D(m)		*	*
	REDD m	0 → D(m)		*	*
I/O	TDD m	Test Discrete I/O Latch D(m)	D(m)	*	
	LAR m	R(m) → A		*	*
	LBR m	R(m) → B		*	*
	LRA m	A → R(m)		*	*
	LRB m	B → R(m)		*	*
	P p	Pattern Generation		*	*
	NOP	No Operation		*	*
CONTROL	STS	Start Serial		*	
	SBY	Standby Mode		*	
	STOP	Stop Mode		*	

3.4.2 Alphabetical Order Table

Instructions are arranged in its mnemonic code's alphabetical order.

MNEMONIC	OP CODE	FUNCTION	STATUS	W/C
AI i	10-1000-....	A + i → A	OVF	1/1
ALEI i	10-1011-....	A ≦ i	NB	1/1
ALEM	00-0001-0100	A ≦ M	NB	1/1
ALEMD d	01-0001-0100	A → M(d)	NB	2/2
AM	00-0000-1000	M + A → A	OVF	1/1
AMD d	01-0000-1000	M(d) + A → A	OVF	2/2
AMC	00-0001-1000	M + A + CA → A, OVF → CA	OVF	1/1
AMCD d	01-0001-1000	M(d) + A + CA → A, OVF → CA	OVF	2/2
ANEM	00-0000-0100	A ≠ M	NZ	1/1
ANEMD d	01-0000-0100	A ≠ M(d)	NZ	2/2
ANM	00-1001-1100	A ∩ M → A	NZ	1/1
ANMD d	01-1001-1100	A ∩ M(d) → A	NZ	2/2
AYY	00-0101-0100	Y + A → Y	OVF	1/1
BLEM	00-1100-0100	B ≦ M	NB	1/1
BNEM	00-0100-0100	B ≠ M	NZ	1/1
BR d	11-....-....	Branch on Status 1	1	1/1
BRL u	01-0111-....	Long Branch on Status 1	1	2/2
CAL a	01-11..-....	Subroutine Jump on Status 1	1	1/2
CALL u	01-0110-....	Long Subroutine Jump on Status 1	1	2/2
COMB	01-0100-0000	$\overline{B}$ → B		1/1
DAA	00-1010-0110	Decimal Adjust (Add)		1/1
DAS	00-1010-1010	Decimal Adjust (Subtract)		1/1
DB	00-1100-1111	B - 1 → B	NB	1/1
DY	00-1101-1111	Y - 1 → Y	NB	1/1
EORM	00-0001-1100	A ⊕ M → A	NZ	1/1
EORMD d	01-0001-1100	A ⊕ M(d) → A	NZ	2/2
IB	00-0100-1100	B + 1 → B	NZ	1/1
ILEM	00-0011-....	i ≦ M	NB	1/1
ILEMD i,d	01-0011-....	i ≦ M(d)	NB	2/2
INEM i	00-0010-....	i ≠ M	NZ	1/1
INEMD i,d	01-0010-....	i ≠ M(d)	NZ	2/2
IY	00-0101-1100	Y + 1 → Y	NZ	1/1
JMPL u	01-0101-....	Long Jump Unconditionally		2/2
LAB	00-0100-1000	B → A		1/1
LAI i	10-0011-....	i → A		1/1
LAM(XY)	00-1001-00..	M → A, (X←→SPX, Y←→SPY)		1/1
LAMD d	01-1001-0000	M(d) → A		2/2
LAMR m	10-0111-....	MR(m) → A		1/1
LAR m	10-0101-....	R(m) → A		1/1
LASPX	00-0110-1000	SPX → A		1/1
LASPY	00-0101-1000	SPY → A		1/1
LAY	00-1010-1111	Y → A		1/1
LBA	00-1100-1000	A → B		1/1
LBI i	10-0000-....	i → B		1/1
LBM(XY)	00-0100-00..	M → B, (X←→SPX, Y←→SPY)		1/1
LBR m	10-0100-....	R(m) → B		1/1
LMA(XY)	00-1001-01..	A → M, (X←→SPX, Y←→SPY)		1/1
LMAD d	01-1001-0100	A → M(d)		2/2
LMADY(X)	00-1101-000.	A → M, Y-1 ←→ Y, (X ←→ SPX)	NB	1/1
LMAIY(X)	00-0101-000.	A → M, Y+1 ←→ Y, (X ←→ SPX)	NZ	1/1
LMID i,d	01-1010-....	i → M(d)		2/2
LMIIY i	10-1001-....	i → M, Y+1 ←→ Y	NZ	1/1
LRA m	10-1101-....	A → R(m)		1/1
LRB m	10-1100-....	B → R(m)		1/1
LWI i	00-1111-00..	i → W		1/1

W/C --- Word/Cycle

(to be continued)

MNEMONIC	OP CODE	FUNCTION	STATUS	W/C
LXA	00-1110-1000	A → X		1/1
LXI i	10-0010-....	i → X		1/1
LYA	00-1101-1000	A → Y		1/1
LYI i	10-0001-....	i → Y		1/1
NEGA	00-0110-0000	$\overline{A}$ + 1 → A		1/1
NOP	00-0000-0000	No Operation		1/1
OR	01-0100-0100	A ∪ B → A		1/1
ORM	00-0000-1100	A ∪ M → A	NZ	1/1
ORMD d	01-0000-1100	A ∪ M(d) → A	NZ	2/2
P p	01-1011-....	Pattern Generation		1/2
REC	00-1110-1100	0 → CA		1/1
RED	00-0110-0100	0 → D(Y)		1/1
REDD m	10-0110-....	0 → D(m)		1/1
REM n	00-1000-10..	0 → M(n)		1/1
REMD n,d	01-1000-10..	0 → M(d,n)		2/2
ROTL	00-1010-0001	Rotate Left A with Carry		1/1
ROTR	00-1010-0000	Rotate Right A with Carry		1/1
RTN	00-0001-0000	Return from Subroutine		1/3
RTNI	00-0001-0001	Return from Interrupt	ST	1/3
SBY	01-0100-1100	Standby Mode		1/1
SEC	00-1110-1111	1 → CA		1/1
SED	00-1110-0100	1 → D(Y)		1/1
SEDD m	10-1110-....	1 → D(m)		1/1
SEM n	00-1000-01..	1 → M(n)		1/1
SEMD n,d	01-1000-01..	1 → M(d,n)		2/2
SMC	00-1001-1000	M - A - $\overline{CA}$ → A, NB → CA	NB	1/1
SMCD d	01-1001-1000	M(d) - A - $\overline{CA}$ → A, NB → CA	NB	2/2
STOP	01-0100-1101	Stop Mode		1/1
STS	01-0100-1000	Start Serial		1/1
SYY	00-1101-0100	Y - A → Y	NB	1/1
TBR p	00-1011-....	Table Branch		1/1
TC	00-0110-1111	Test Carry	CA	1/1
TD	00-1110-0000	Test Discrete I/O Latch D(Y)	D(Y)	1/1
TDD m	10-1010-....	Test Discrete I/O Latch D(m)	D(m)	1/1
TM n	00-1000-11..	Test Memory Bit M(n)	M(n)	1/1
TMD n,d	01-1000-11..	Test Memory Bit M(d,n)	M(d,n)	2/2
XMA(XY)	00-1000-00..	M ⟷ A, (X⟷SPX, Y ↔ SPY)		1/1
XMAD d	01-1000-0000	M(d) ↔ A		2/2
XMB(XY)	00-1100-00..	M ⟷ B, (X⟷SPX, Y⟷SPY)		1/1
XMRA m	10-1111-....	MR(m) ⟷ A		1/1
XSPX	00-0000-0001	X ⟷ SPX		1/1
XSPXY	00-0000-0011	X ⟷ SPX, Y ⟷ SPY		1/1
XSPY	00-0000-0010	Y ⟷ SPY		1/1
YNEI i	00-0111-....	Y ≠ i	NZ	1/1

W/C---Word/cycle

3.4.3 Object Code Table

Instructions are arranged in object code order.

OP-CODE	MNEMONIC	FUNCTION	STATUS	W/C
00-0000-0000	NOP	No Operation		1/1
00-0000-0001	XSPX	X ⟷ SPX		1/1
00-0000-0010	XSPY	Y ⟷ SPY		1/1
00-0000-0011	XSPXY	X ⟷ SPX, Y ⟷ SPY		1/1
00-0000-0100	ANEM	A ≠ M	NZ	1/1
00-0000-1000	AM	M + A → A	OVF	1/1
00-0000-1100	ORM	A ∪ M → A	NZ	1/1
00-0001-0000	RTN	Return from Subroutine		1/3
00-0001-0001	RTNI	Return from Interrupt	ST	1/3
00-0001-0100	ALEM	A ≦ M	NB	1/1
00-0001-1000	AMC	M + A + CA → A, OVF → CA	OVF	1/1
00-0001-1100	EORM	A ⊕ M → A	NZ	1/1
00-0010-....	INEM i	i ≠ M	NZ	1/1
00-0011-....	ILEM i	i ≦ M	NB	1/1
00-0100-00..	LBM(XY)	M → B, (X⟷SPX, Y ⟷SPY)		1/1
00-0100-0100	BNEM	B ≠ M	NZ	1/1
00-0100-1000	LAB	B → A		1/1
00-0100-1100	IB	B + 1 → B	NZ	1/1
00-0101-000.	LMAIY(X)	A → M, Y+1 → Y, (X ⟷ SPX)	NZ	1/1
00-0101-0100	AYY	Y + A → Y	OVF	1/1
00-0101-1000	LASPY	SPY → A		1/1
00-0101-1100	IY	Y + 1 → Y	NZ	1/1
00-0110-0000	NEGA	$\overline{A}$ + 1 → A		1/1
00-0110-0100	RED	0 → D(Y)		1/1
00-0110-1000	LASPX	SPX → A		1/1
00-0110-1111	TC	Test Carry	CA	1/1
00-0111-....	YNEI i	Y ≠ i	NZ	1/1
00-1000-00..	XMA(XY)	M ⟷ A, (X⟷SPX, Y⟷SPY)		1/1
00-1000-01..	SEM n	1 → M(n)		1/1
00-1000-10..	REM n	0 → M(n)		1/1
00-1000-11..	TM n	Test Memory Bit M(n)	M(n)	1/1
00-1001-00..	LAM(XY)	M → A, (X ↔ SPX, Y ↔ SPY)		1/1
00-1001-01..	LMA(XY)	A → M, (X ↔ SPX, Y ↔ SPY)		1/1
00-1001-1000	SMC	M - A - $\overline{CA}$ → A, NB → CA	NB	1/1
00-1001-1100	ANM	A ∩ M → A	NZ	1/1
00-1010-0000	ROTR	Rotate Right A with Carry		1/1
00-1010-0001	ROTL	Rotate Left A with Carry		1/1
00-1010-0110	DAA	Decimal Adjust (Add)		1/1
00-1010-1010	DAS	Decimal Adjust (Subtract)		1/1
00-1010-1111	LAY	Y → A		1/1
00-1011-....	TBR P	Table Branch		1/1
00-1100-00..	XMB(XY)	M ⟷ B, (X⟷SPX, Y⟷SPY)		1/1
00-1100-0100	BLEM	B ≦ M	NB	1/1
00-1100-1000	LBA	A → B		1/1
00-1100-1111	DB	B - 1 → B	NB	1/1
00-1101-000.	LMADY(X)	A → M, Y-1 → Y, (X ⟷ SPX)	NB	1/1
00-1101-0100	SYY	Y - A → Y	NB	1/1
00-1101-1000	LYA	A → Y		1/1
00-1101-1111	DY	Y - 1 → Y	NB	1/1
00-1110-0000	TD	Test Discrete I/O Latch D(Y)	D(Y)	1/1
00-1110-0100	SED	1 → D(Y)		1/1
00-1110-1000	LXA	A → X		1/1
00-1110-1100	REC	0 → CA		1/1
00-1110-1111	SEC	1 → CA		1/1

W/C --- Word/Cycle

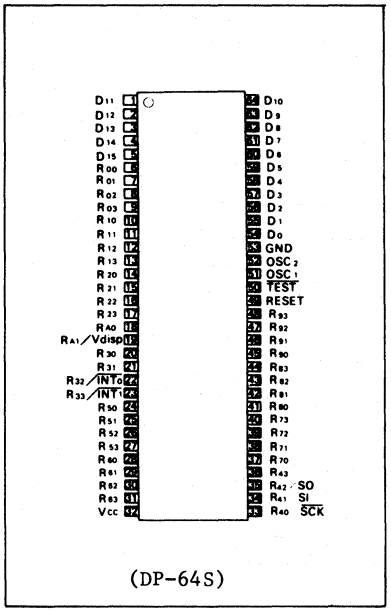
(to be continued)

OP-CODE	MNEMONIC		FUNCTION	STATUS	W/C
00-1111-00..	LWI	i	i → W		1/1
01-0000-0100	ANEMD	d	A ≠ M(d) → A	NZ	2/2
01-0000-1000	AMD	d	A + M(d) → A	OVF	2/2
01-0000-1100	ORMD	d	A ∪ M(d) → A	NZ	2/2
01-0001-0100	ALEMD	d	A ≦ M(d)	NB	2/2
01-0001-1000	AMCD	d	M(d) + A + CA → A, OVF → CA	OVF	2/2
01-0001-1100	EORMD	d	A ⊕ M(d) → A	NZ	2/2
01-0010-....	INEMD	i,d	i ≠ M(d)	NZ	2/2
01-0011-....	ILEMD	i,d	i ≦ M(d)	NB	2/2
01-0100-0000	COMB		$\overline{B}$ → B		1/1
01-0100-0100	OR		A ∪ B → A		1/1
01-0100-1000	STS		Start Serial		1/1
01-0100-1100	SBY		Standby Mode		1/1
01-0100-1101	STOP		Stop Mode		1/1
01-0101-....	JMPL	u	Long Jump Unconditionally		2/2
01-0110-....	CALL	u	Long Subroutine Jump on Status 1	1	2/2
01-0111-....	BRL	u	Long Branch on Status 1	1	2/2
01-1000-0000	XMAD	d	M(d) ⟷ A		2/2
01-1000-01..	SEMD	n,d	1 → M(d,n)		2/2
01-1000-10..	REMD	n,d	0 → M(d,n)		2/2
01-1000-11..	TMD	n,d	Test Memory Bit M(d,n)	M(d,n)	2/2
01-1001-0000	LAMD	d	M(d) → A		2/2
01-1001-0100	LMAD	d	A → M(d)		2/2
01-1001-1000	SMCD	d	M(d) - A - $\overline{CA}$ → A, NB → CA	NB	2/2
01-1001-1100	ANMD	d	A ∩ M(d) → A	NZ	2/2
01-1010-....	LMID	i,d	i → M(d)		2/2
01-1011-....	P	P	Pattern Generation		1/2
01-11..-....	CAL	a	Subroutine Jump on Status 1	1	1/2
10-0000-....	LBI	i	i → B		1/1
10-0001-....	LYI	i	i → Y		1/1
10-0010-....	LXI	i	i → X		1/1
10-0011-....	LAI	i	i → A		1/1
10-0100-....	LBR	m	R(m) → B		1/1
10-0101-....	LAR	m	R(m) → A		1/1
10-0110-....	REDD	m	0 → D(m)		1/1
10-0111-....	LAMR	m	MR(m) → A		1/1
10-1000-....	AI	i	A + i → A	OVF	1/1
10-1001-....	LMIIY	i	i → M, Y+1 → Y	NZ	1/1
10-1010-....	TDD	m	Test Discrete I/O Latch D(m)	D(m)	1/1
10-1011-....	ALEI	i	A ≦ i	NB	1/1
10-1100-....	LRB	m	B → R(m)		1/1
10-1101-....	LRA	m	A → R(m)		1/1
10-1110-....	SEDD	m	1 → D(m)		1/1
10-1111-....	XMRA	m	MR(m) ⟷ A		1/1
11-....-....	BR	b	Branch on Status 1	1	1/1

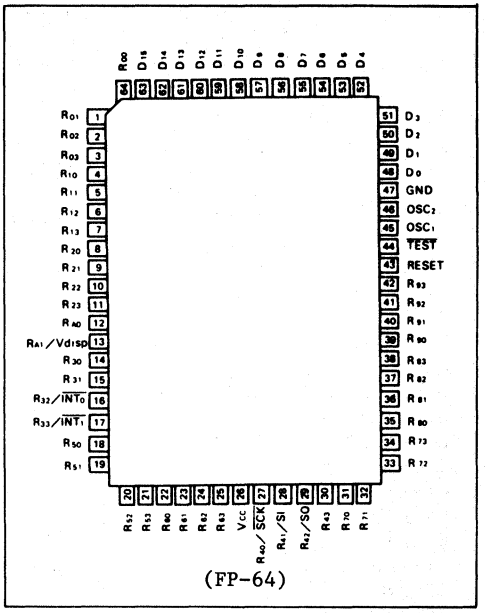
W/C --- Word/Cycle

4. PIN ARRANGEMENT AND PACKAGE DIMENSION

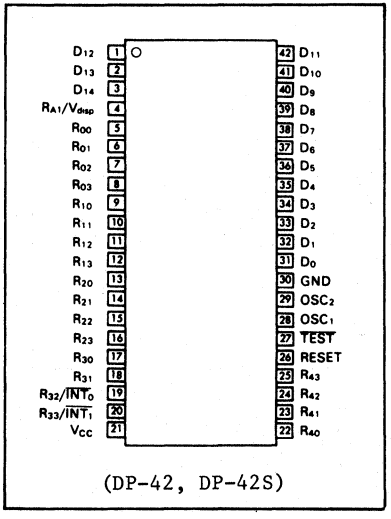
4.1 Pin Arrangement (Top View)



(DP-64S)



(FP-64)

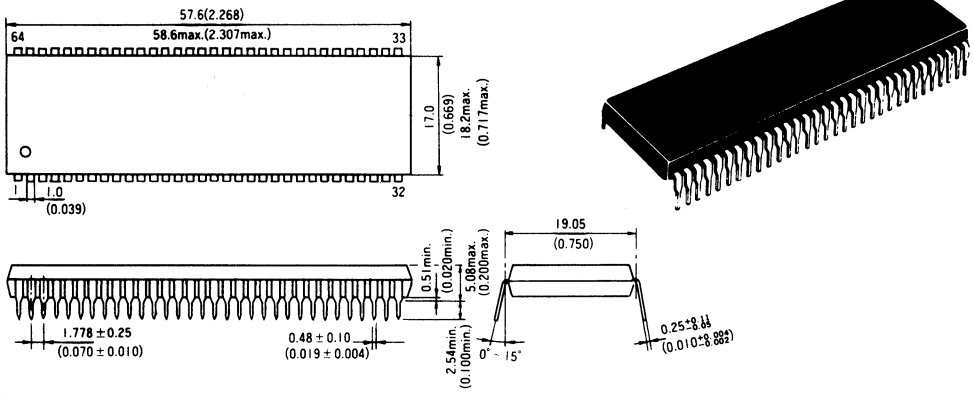


(DP-42, DP-42S)

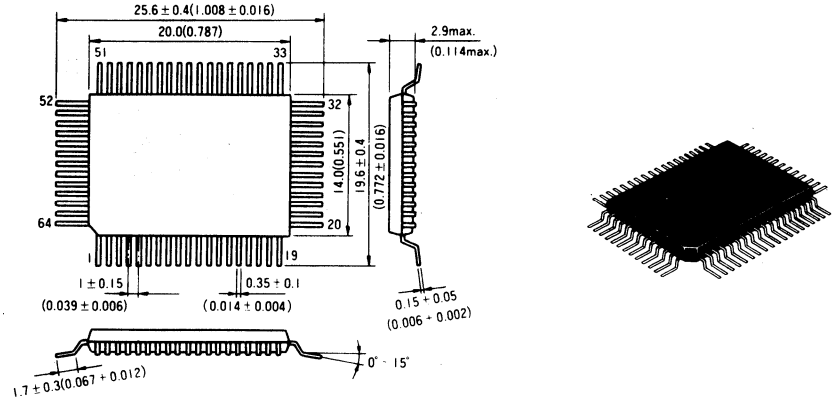
4.2 Package Dimension

Unit : mm (inch)

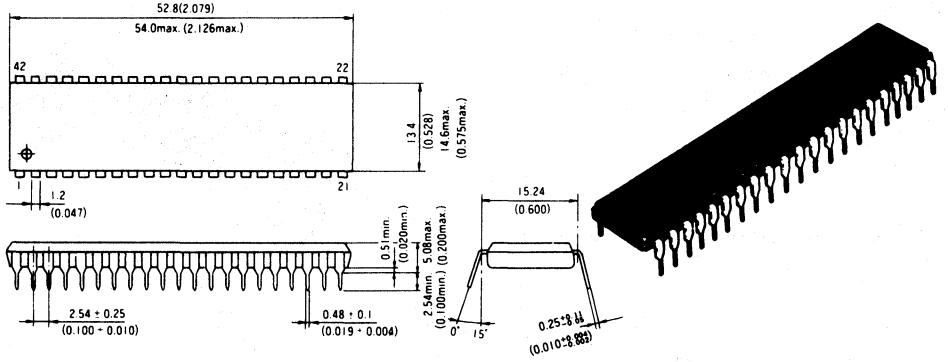
DP-64S



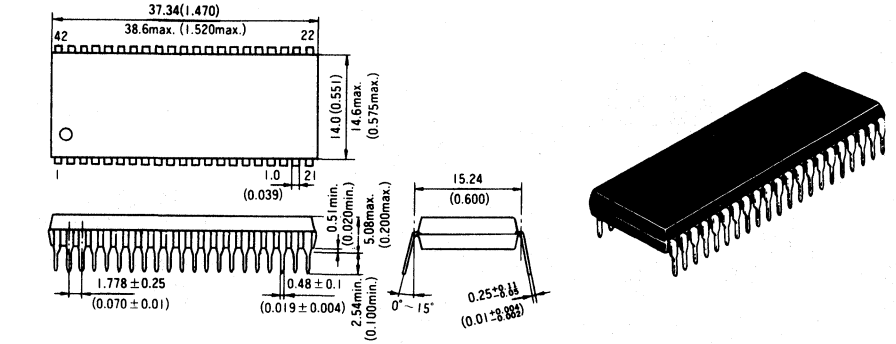
FP-64



DP-42


52.8(2.079)
54.0max. (2.126max.)
42
22
1
21
13.4
(0.528)
14.6max.
(0.575max.)
1.2
(0.047)
0.51min.
(0.020min.)
5.08max.
(0.200max.)
2.54min.
(0.100min.)
15.24
(0.600)
2.54 ± 0.25
(0.100 ± 0.010)
0.48 ± 0.1
(0.019 ± 0.004)
0°
15°

DP-42S


37.34(1.470)
38.6max. (1.520max.)
42
22
1
21
14.0(0.551)
14.6max.
(0.575max.)
1.0
(0.039)
0.51min.
(0.020min.)
5.08max.
(0.200max.)
2.54min.
(0.100min.)
15.24
(0.600)
1.778 ± 0.25
(0.070 ± 0.01)
0.48 ± 0.1
(0.019 ± 0.004)
0° ~ 15°

5. ELECTRICAL CHARACTERISTICS

5.1 HMCS402/404/408 Absolute Maximum Ratings

Item	Symbol	Value	Unit	Note
Supply Voltage	V_{CC}	−0.3 to +7.0	V	
Terminal Voltage	V_T	−0.3 to V_{CC}+0.3	V	3
		V_{CC} −45 to V_{CC} +0.3	V	4
Total Allowance of Input Currents	ΣI_O	50	mA	5
Total Allowance of Output Currents	$-\Sigma I_O$	150	mA	6
Maximum Input Current	I_O	15	mA	7, 8
Maximum Output Current	$-I_O$	4	mA	9, 10
		6	mA	9, 11
		30	mA	9, 12
Operating Temperature	T_{opr}	−20 to +75	°C	
Storage Temperature	T_{stg}	−55 to +125	°C	

(Note 1) Permanent damage may occur if "Absolute Maximum Ratings" are exceeded. Normal operation should be under the conditions of "Electrical Characteristics". If these conditions are exceeded, it may cause the malfunction and affect the reliability of LSI.
(Note 2) All voltages are with respect to GND.
(Note 3) Applied to standard pins.
(Note 4) Applied to high voltage pins.
(Note 5) Total allowance of input current is the total sum of input current which flow in from all I/O pins to GND simultaneously.
(Note 6) Total allowance of output current is the total sum of the output current which flow out from V_{CC} to all I/O pins simultaneously.
(Note 7) Maximum input current is the maximum amount of input current from each I/O pin to GND.
(Note 8) Applied to $D_0 - D_3$ and R3 − R8.
(Note 9) Maximum output current is the maximum amount of output current from V_{CC} to each I/O pin.
(Note 10) Applied to $D_0 - D_3$ and R3 − R8.
(Note 11) Applied to to R0 − R2.
(Note 12) Applied to $D_4 - D_{15}$.

5.2 HMCS402C Electrical Characteristics

(1) DC Characteristics

(V_{CC}=4V to 6V, GND=0V, V_{disp}=V_{CC}-40V to V_{CC}, Ta=-20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions		Value min	Value typ	Value max	Unit	Note
Input "High" Voltage	V_{IH}	$\overline{RESET}$, SCK, $\overline{INT_0}$, $\overline{INT_1}$			0.7V_{CC}	–	V_{CC}+0.3	V	
		SI			0.7V_{CC}	–	V_{CC}+0.3	V	
		OSC_1			V_{CC}- 0.5	–	V_{CC}+0.3	V	
Input "Low" Voltage	V_{IL}	$\overline{RESET}$, SCK, $\overline{INT_0}$, $\overline{INT_1}$			-0.3	–	0.22V_{CC}	V	
		SI			-0.3	–	0.22V_{CC}	V	
		OSC_1			-0.3	–	0.5	V	
Output "High" Voltage	V_{OH}	$\overline{SCK}$, SO	$-I_{OH}$ = 1.0 mA		V_{CC}-1.0	–	–	V	
			$-I_{OH}$ = 0.01 mA		V_{CC}-0.3	–	–	V	
Output "Low" Voltage	V_{OL}	$\overline{SCK}$, SO	I_{OL} = 1.6 mA		–	–	0.4	V	
Input/Output Leakage Current	$\|I_{IL}\|$	$\overline{RESET}$, SCK, $\overline{INT_0}$, $\overline{INT_1}$, SI, SO, OSC_1	V_{in} = 0V to V_{CC}		–	–	1	μA	1
Current Dissipation in Active Mode	I_{CC}	V_{CC}	V_{CC}=5V	Crystal or Ceramic Filter Oscillator Option f_{osc} = 4MHz	–	–	2.0	mA	2, 6
				Resistor Oscillator Option f_{osc} = 4MHz	–	–	2.4	mA	2, 6
Current Dissipation in Standby Mode	I_{SBY1}	V_{CC}	Maximum Logic Operation V_{CC} = 5V	Crystal or Ceramic Filter Oscillator Option f_{osc} = 4MHz	–	–	1.2	mA	3, 6
				Resistor Oscillator Option f_{osc} = 4MHz	–	–	1.6	mA	3, 6
	I_{SBY2}	V_{CC}	Minimum Logic Operation V_{CC} = 5V	Crystal or Ceramic Filter Oscillator Option f_{osc} =4MHz	–	–	0.9	mA	4, 6
				Resistor Oscillator Option f_{osc} = 4MHz	–	–	1.3	mA	4, 6
Current Dissipation in Stop Mode	I_{stop}	V_{CC}	$V_{in}(\overline{TEST})$ = V_{CC}-0.3V to V_{CC} V_{in}(RESET) = 0V to 0.3V		–	–	10	μA	5
Stop Mode Retain Voltage	V_{stop}	V_{CC}			2	–	–	V	

(Note 1) Pull-up MOS current and output buffer current are excluded.
(Note 2) The MCU is in the reset state. The input/output current does not flow.
Test Conditions: MCU state; • Reset state in Operation Mode
Pin state; • RESET, $\overline{TEST}$ ... V_{CC} voltage
• D_0-D_3, R3–R9 ··· V_{CC} voltage
• D_4-D_{15}, R0–R2, R_{A0}, R_{A1} ... V_{disp} voltage
(Note 3) The timer/counter operate with the fastest clock and input/output current does not flow.
Test Conditions: MCU state; • Standby Mode
• Input/Output; Reset state
• TIMER-A; ÷2 prescaler divide ratio
• TIMER-B; ÷2 prescaler divide ratio
• SERIAL Interface ; Stop
Pin state; • RESET ... GND voltage
• $\overline{TEST}$ ... V_{CC} voltage
• D_0-D_3, R3–R9 ··· V_{CC} voltage
• D_4-D_{15}, R0–R2, R_{A0}, R_{A1} ··· V_{disp} voltage
(Note 4) The timer/counter operate with the slowest clock and input/output current does not flow.
Test Conditions: MCU state; • Standby Mode
• Input/Output; Reset state
• TIMER-A; ÷2048 prescaler divide ratio
• TIMER-B; ÷2048 prescaler divide ratio
• SERIAL Interface ; Stop
Pin state; • RESET ··· GND voltage
• $\overline{TEST}$ ... V_{CC} voltage
• D_0-D_3, R3~R9 ··· V_{CC} voltage
• D_4-D_{15}, R0~R2, R_{A0}, R_{A1} ··· V_{disp} voltage
(Note 5) Pull-down MOS current is excluded.
(Note 6) When $f_{osc}=x$ [MHz], the Current Dissipation in Operation mode and Standby mode are estimated as follows:

$$\text{max. value } (f_{osc}=x\text{[MHz]}) = \frac{x}{4} \times \text{max. value } (f_{osc}=4\text{[MHz]})$$

(2) Input/output characteristics for standard pin

(V_{CC}=4V to 6V, GND=0V, $V_{disp}=V_{CC}$-40V to V_{CC}, Ta=-20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Input "High" Voltage	V_{IH}	D_0 – D_3, R3 – R5, R9		0.7V_{CC}	–	V_{CC}+0.3	V	
Input "Low" Voltage	V_{IL}	D_0 – D_3, R3 – R5, R9		–0.3	–	0.22V_{CC}	V	
Output "High" Voltage	V_{OH}	D_0 – D_3, R3 – R8	$-I_{OH}$ = 1.0 mA	V_{CC}–1.0	–	–	V	1
		D_0 – D_3, R3 – R8	$-I_{OH}$ = 0.01 mA	V_{CC}–0.3	–	–	V	1
Output "Low" Voltage	V_{OL}	D_0 – D_3, R3 – R8	I_{OL} = 1.6 mA	–	–	0.4	V	
Input/Output Leakage Current	$\|I_{IL}\|$	D_0 – D_3, R3 – R9	V_{in} = 0V to V_{CC}	–	–	1	μA	2
Pull-Up MOS Current	$-I_p$	D_0 – D_3, R3 – R9	V_{CC} = 5V, V_{in} = 0V	30	60	120	μA	3

(Note 1) Applied to I/O pins with "CMOS" Output selected by mask option.
(Note 2) Pull-up MOS current and output buffer current are excluded.
(Note 3) Applied to I/O pins with "with Pull-up MOS" selected by mask option.

(3) Input/output characteristics for high voltage pin
(V_{CC}=4V to 6V, GND=0V, V_{disp}=V_{CC}-40V to V_{CC}, Ta=-20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Input "High" Voltage	V_{IH}	D_4 – D_{15}, R1 R2, R_{A0}, R_{A1}		0.7V_{CC}	–	V_{CC}+0.3	V	
Input "Low" Voltage	V_{IL}	D_4–D_{15}, R1 R2, R_{A0}, R_{A1}		V_{CC}–40	–	0.22V_{CC}	V	
Output "High" Voltage	V_{OH}	D_4 – D_{15}	$-I_{OH}$=15mA, V_{CC}=5V±10%	V_{CC}–3.0	–	–	V	
			$-I_{OH}$=9 mA	V_{CC}–2.0	–	–	V	
		R0 – R2	$-I_{OH}$=3 mA, V_{CC}=5V±10%	V_{CC}–3.0	–	–	V	
			$-I_{OH}$=1.8 mA	V_{CC}–2.0	–	–	V	
Output "Low" Voltage	V_{OL}	D_4–D_{15} R0–R2	V_{disp} = V_{CC}–40V	–	–	V_{CC}–37	V	1
		D_4–D_{15} R0–R2	150kΩ to V_{CC}–40V	–	–	V_{CC}–37	V	2
Input/Output Leakage Current	$\|I_{IL}\|$	D_4–D_{15} R0–R2 R_{A0}, R_{A1}	V_{in} = V_{CC}–40V to V_{CC}	–	–	20	μA	3
Pull Down MOS Current	I_d	D_4–D_{15} R0–R2 R_{A0}, R_{A1}	V_{disp} = V_{CC}–35V V_{in} = V_{CC}	125	250	500	μA	4

(Note 1) Applied to I/O pins with "with Pull-down MOS" selected by mask option.
(Note 2) Applied to I/O pins with "without Pull-down MOS (PMOS Open Drain)" selected by mask option.
(Note 3) Pull-down MOS current and output buffer current are excluded.
(Note 4) Applied to I/O pins with "with Pull-down MOS" selected by mask option.

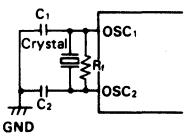
(4) AC characteristics

(V_{CC}=4V to 6V, GND=0V, V_{disp}=V_{CC}-40V to V_{CC}, Ta=-20 to +75°C, if not specified.)

	Item	Symbol	Pin Name	Test Conditions	Value min	Value typ	Value max	Unit	Note
Crystal or Ceramic Filter Oscillator	Oscillation Frequency	f_{osc}	OSC_1, OSC_2		0.4	4	4.5	MHz	
	Instruction Cycle Time	t_{cyc}			1.78	2	20	μs	
	Oscillator Stabilization Time	t_{RC}	OSC_1, OSC_2		–	–	20	ms	1
Resistor Oscillator	Oscillation Frequency	f_{osc}	OSC_1, OSC_2	R_f=20kΩ±2%	1.8	3.0	4.2	MHz	
	Instruction Cycle Time	t_{cyc}		R_f=20kΩ±2%	1.9	2.66	4.44	μs	
	Oscillator Stabilization Time	t_{RC}	OSC_1, OSC_2	R_f=20kΩ±2%	–	–	0.5	ms	1
External Clock	External Clock Frequency	f_{CP}	OSC_1		0.4	–	4.5	MHz	2
	External Clock "High" Level Width	t_{CPH}	OSC_1		92	–	–	ns	2
	External Clock "Low" Level Width	t_{CPL}	OSC_1		92	–	–	ns	2
	External Clock Rise Time	t_{CPr}	OSC_1		–	–	20	ns	2
	External Clock Fall Time	t_{CPf}	OSC_1		–	–	20	ns	2
	Instruction Cycle Time	t_{cyc}			1.78	–	20	μs	2
$\overline{INT_0}$ "High" Level Width		t_{I0H}	$\overline{INT_0}$		2	–	–	t_{cyc}	3
$\overline{INT_0}$ "Low" Level Width		t_{I0L}	$\overline{INT_0}$		2	–	–	t_{cyc}	3
$\overline{INT_1}$ "High" Level Width		t_{I1H}	$\overline{INT_1}$		2	–	–	t_{cyc}	3
$\overline{INT_1}$ "Low" Level Width		t_{I1L}	$\overline{INT_1}$		2	–	–	t_{cyc}	3
RESET "High" Level Width		t_{RSTH}	RESET		2	–	–	t_{cyc}	4
Input Capacitance		C_{in}	all pins	f=1MHz V_{in} = 0V	–	–	15	pF	
RESET Fall Time		t_{RSTf}			–	–	20	ms	4

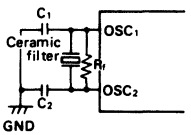
(Note 1) Oscillator stabilization time is the time until the oscillator stabilizes after V_{CC} reaches 4.0V at "Power-on", or after RESET input level goes to "High" by resetting to quit the stop mode by MCU reset on the circuits below. At power ON or recovering from stop mode, apply RESET input more than t_{RC} to obtain the necessary time for oscillator stabilization. When using crystal or ceramic filter oscillator, please ask a crystal oscillator maker's or ceramic filter maker's advice because oscillator stabilization time depends on the circuit constant and stray capacity.

Crystal oscillator



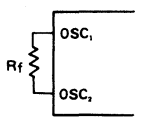
Crystal: 4.194304MHz NC-18C(Nihon Denpa Kogyo)
Rf : 1MΩ ±2%
C_1 : 22pF±20%
C_2 : 22pF±20%

Ceramic filter oscillator



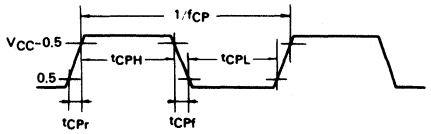
Ceramic filter: CSA4.00MG (Murata)
Rf : 1MΩ±2%
C_1 : 30pF±20%
C_2 : 30pF±20%

Resistor oscillator

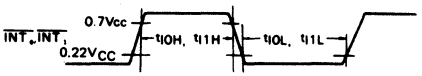


Rf : 20kΩ±2%

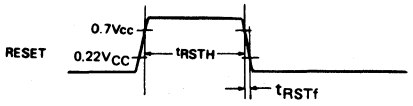
(Note 2)



(Note 3)



(Note 4)



(5) Serial interface timing characteristics

(V_{CC}=4V to 6V, GND=0V, V_{disp}=V_{CC}-40V to V_{CC}, Ta=-20 to +75°C, if not specified.)

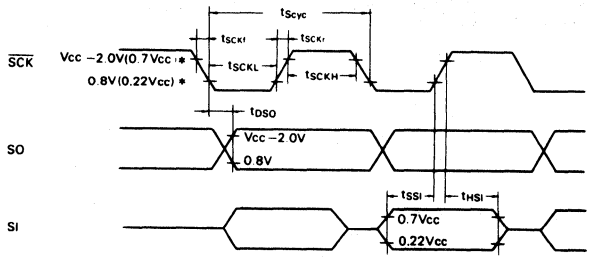
- At Transfer Clock Output

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Transfer Clock Cycle Time	t_{Scyc}	$\overline{SCK}$	(Note 2)	1	–	–	t_{cyc}	1, 2
Transfer Clock "High" Level Width	t_{SCKH}	$\overline{SCK}$	(Note 2)	0.5	–	–	t_{Scyc}	1, 2
Transfer Clock "Low" Level Width	t_{SCKL}	$\overline{SCK}$	(Note 2)	0.5	–	–	t_{Scyc}	1, 2
Transfer Clock Rise Time	t_{SCKr}	$\overline{SCK}$	(Note 2)	–	–	100	ns	1, 2
Transfer Clock Fall Time	t_{SCKf}	$\overline{SCK}$	(Note 2)	–	–	100	ns	1, 2
Serial Output Data Delay Time	t_{DSO}	SO	(Note 2)	–	–	300	ns	1, 2
Serial Input Data Set-up Time	t_{SSI}	SI		500	–	–	ns	1
Serial Input Data Hold Time	t_{HSI}	SI		150	–	–	ns	1

- At Transfer Clock Input

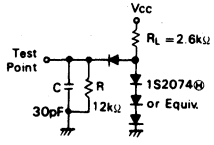
Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Transfer Clock Cycle Time	t_{Scyc}	$\overline{SCK}$		1	–	–	t_{cyc}	1
Transfer Clock "High" Level Width	t_{SCKH}	$\overline{SCK}$		0.5	–	–	t_{Scyc}	1
Transfer Clock "Low" Level Width	t_{SCKL}	$\overline{SCK}$		0.5	–	–	t_{Scyc}	1
Transfer Clock Rise Time	t_{SCKr}	$\overline{SCK}$		–	–	100	ns	1
Transfer Clock Fall Time	t_{SCKf}	$\overline{SCK}$		–	–	100	ns	1
Serial Output Data Delay Time	t_{DSO}	SO	(Note 2)	–	–	300	ns	1, 2
Serial Input Data Set-up Time	t_{SSI}	SI		500	–	–	ns	1
Serial Input Data Hold Time	t_{HSI}	SI		150	–	–	ns	1

(Note 1) Timing Diagram of Serial Interface

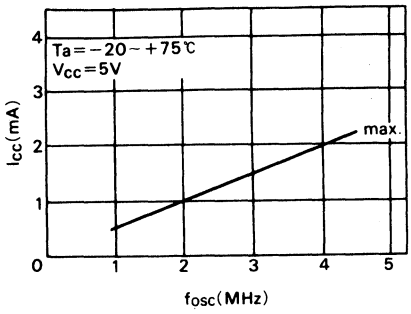


* V_{CC} − 2.0V and 0.8V are the threshold voltage for transfer clock output.
0.7 V_{CC} and 0.22 V_{CC} are the threshold voltage for transfer clock input.

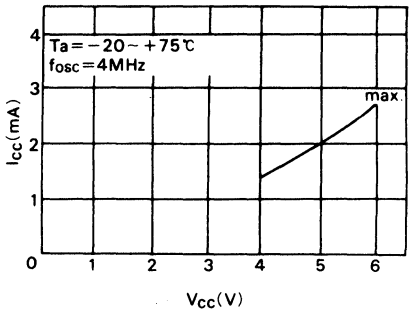
(Note 2) Timing Load Circuit



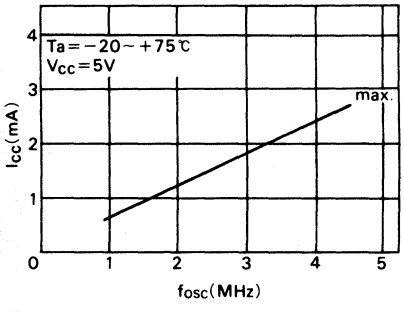
(6) Characteristics Curve (Reference Data)



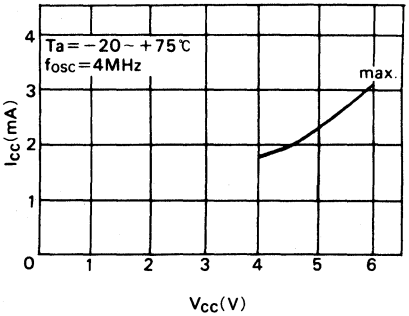
I_{CC} vs. f_{osc} Characteristics
(Crystal, Ceramic Filter Oscillator Option)



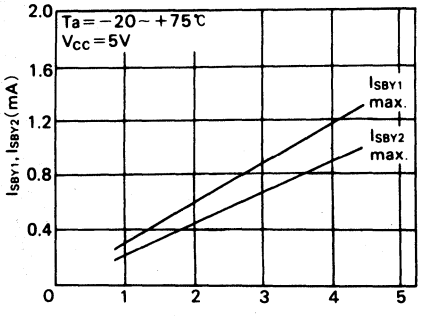
I_{CC} vs. V_{CC} Characteristics
(Crystal, Ceramic Filter Oscillator Option)



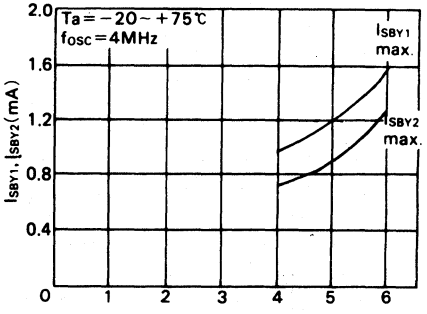
I_{CC} vs. f_{osc} Characteristics
(Resistor Oscillator Option)



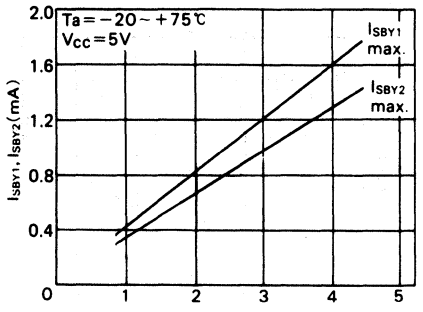
I_{CC} vs. V_{CC} Characteristics
(Resistor Oscillator Option)



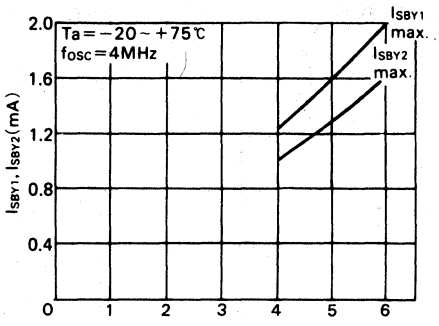
I_{SBY} vs. f_{osc} Characteristics
(Crystal, Ceramic Filter Oscillator Option)



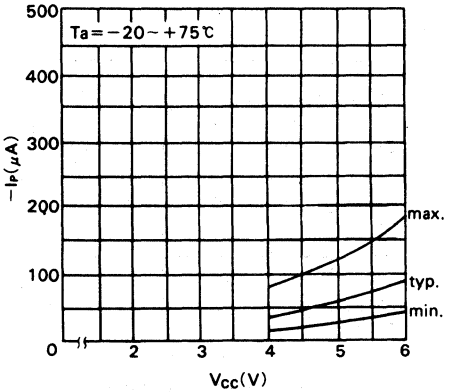
I_{SBY} vs. V_{CC} Characteristics
(Crystal, Ceramic Filter Oscillator Option)



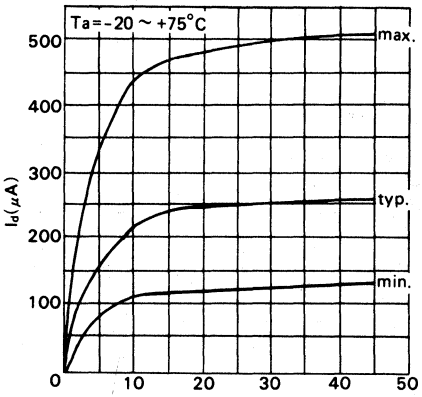
I_{SBY} vs. f_{osc} Characteristics
(Resistor Oscillator Option)



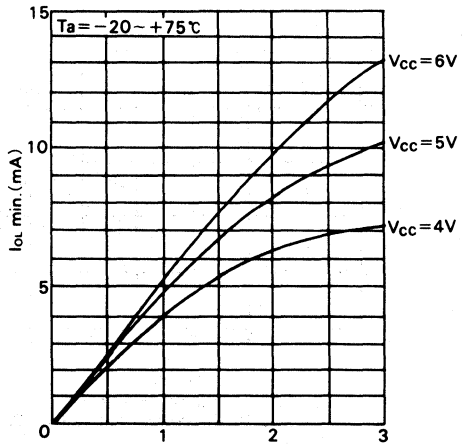
I_{SBY} vs. V_{CC} Characteristics
(Resistor Oscillator Option)



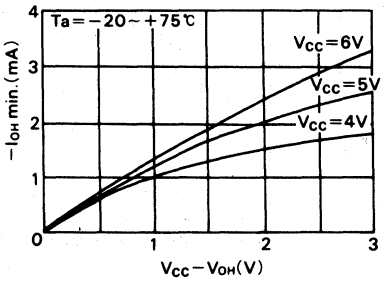
$-I_P$ (Pull-up MOS Current) vs.
V_{CC} Characteristics



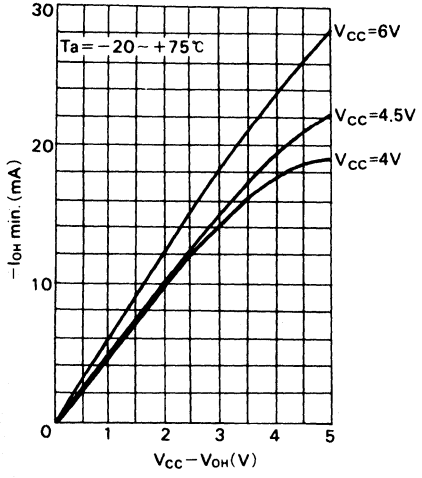
I_d (Pull-down MOS Current) vs.
($V_{CC} - V_{disp}$) Characteristics



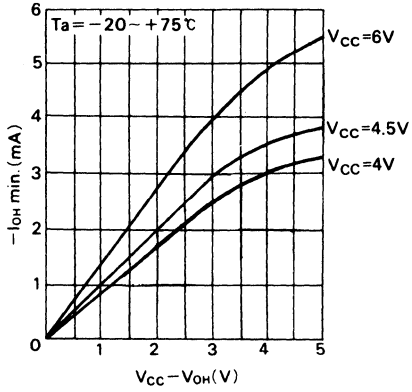
I_{OL} min. vs. V_{OL} Characteristics
(Standard Pin)



$-I_{OH}$ min vs. ($V_{CC} - V_{OH}$) Characteristics
(Standard Pin "CMOS")



$-I_{OH}$ min. vs. ($V_{CC} - V_{OH}$) Characteristics
($D_4 \sim D_{15}$ Pins)



$-I_{OH}$ min. vs. ($V_{CC} - V_{OH}$) Characteristics
(RO ~ R2 Pins)

5.3 HMCS402CL Electrical Characteristics

(1) DC characteristics (V_{CC}=2.7V to 6V, GND=0V, V_{disp}=V_{CC}-40V to V_{CC}, Ta=-20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Input "High" Voltage	V_{IH}	RESET, SCK, INT0, INT1		0.85V_{CC}	–	V_{CC}+0.3	V	
		SI		0.85V_{CC}	–	V_{CC}+0.3	V	
		OSC1		V_{CC}−0.3	–	V_{CC}+0.3	V	
Input "Low" Voltage	V_{IL}	RESET, SCK, INT0, INT1		-0.3	–	0.15V_{CC}	V	
		SI		-0.3	–	0.15V_{CC}	V	
		OSC1		-0.3	–	0.3	V	
Output "High" Voltage	V_{OH}	SCK, SO	$-I_{OH}$ = 0.1 mA	V_{CC}−0.5	–	–	V	
Output "Low" Voltage	V_{OL}	SCK, SO	I_{OL} = 0.4mA	–	–	0.4	V	
Input/Output Leakage Current	$\|I_{IL}\|$	RESET, SCK, INT0, INT1, SI, SO, OSC1	V_{in} = 0 V to V_{CC}	–	–	1	μA	1
Current Dissipation in Active Mode	I_{CC}	V_{CC}	V_{CC} = 3 V f_{osc} = 2 MHz	–	–	0.6	mA	2, 6
Current Dissipation in Standby Mode	I_{SBY1}	V_{CC}	Maximum Logic Operation V_{CC} = 3 V f_{osc} = 2 MHz	–	–	0.5	mA	3, 6
	I_{SBY2}	V_{CC}	Minimum Logic Operation V_{CC} = 3 V f_{osc} = 2MHz	–	–	0.4	mA	4, 6
Current Dissipation in Stop Mode	I_{stop}	V_{CC}	V_{in} ($\overline{TEST}$) = V_{CC}-0.2V to V_{CC} V_{in} (RESET) = 0V to 0.2 V	–	–	10	μA	5
Stop Mode Retain Voltage	V_{stop}	V_{CC}		2	–	–	V	

(Note 1) Pull-up MOS current and output buffer current are excluded.
(Note 2) The MCU is in the reset state. The input/output current does not flow.
Test Conditions: MCU state; • Reset state in Operation Mode
Pin state; • RESET, $\overline{TEST}$ ··· V_{CC} voltage
• D_0−D_3, R3−R9 ··· V_{CC} voltage
• D_4−D_{15}, R0−R2, R_{A0}, R_{A1} ··· V_{disp} voltage
(Note 3) The timer/counter operate with the fastest clock and input/output current does not flow.
Test Conditions: MCU state; • Standby Mode
• Input/Output; Reset state
• TIMER-A; ÷2 prescaler divide ratio
• TIMER-B; ÷2 prescaler divide ratio
• SERIAL Interface ; Stop
Pin state; • RESET ···GND voltage
• $\overline{TEST}$ ··· V_{CC} voltage
• D_0−D_3, R3−R9 ··· V_{CC} voltage
• D_4−D_{15}, R0−R2, R_{A0}, R_{A1} ··· V_{disp} voltage
(Note 4) The timer/counter operate with the slowest clock and input/output current does not flow.
Test Conditions: MCU state; • Standby Mode
• Input/Output; Reset state
• TIMER-A; ÷2048 prescaler divide ratio
• TIMER-B; ÷2048 prescaler divide ratio
• SERIAL Interface ; Stop
Pin state; • RESET ··· GND voltage
• $\overline{TEST}$ ··· V_{CC} voltage
• D_0−D_3, R3−R9 ··· V_{CC} voltage
• D_4−D_{15}, R0−R2, R_{A0}, R_{A1} ··· V_{disp} voltage
(Note 5) Pull-down MOS current is excluded.
(Note 6) When f_{osc}=x[MHz], the Current Dissipation in Operation mode and Standby mode are estimated as follows:

[When Divide-by-8 (D-8) option is selected.] max. value (f_{osc}=x[MHz]) = $\frac{x}{2}$ x max. value (f_{osc}=2[MHz])

(2) Input/output characteristics for standard pin

(V_{CC}=2.7V to 6V, GND=0V, V_{disp}=V_{CC}-40V to V_{CC}, Ta=-20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Input "High" Voltage	V_{IH}	$D_0 - D_3$, R3 – R5, R9		0.85V_{CC}	–	V_{CC}+0.3	V	
Input "Low" Voltage	V_{IL}	$D_0 - D_3$, R3 – R5, R9		-0.3	–	0.15 V_{CC}	V	
Output "High" Voltage	V_{OH}	$D_0 - D_3$, R3 – R8	$-I_{OH}$ = 0.1 mA	V_{CC}-0.5	–	–	V	1
Output "Low" Voltage	V_{OL}	$D_0 - D_3$, R3 – R8	I_{OL} = 0.4 mA	–	–	0.4	V	
Input/Output Leakage Current	$\|I_{IL}\|$	$D_0 - D_3$, R3 – R9	V_{in}= 0V to V_{CC}	–	–	1	μA	2
Pull-Up MOS Current	$-I_p$	$D_0 - D_3$, R3 – R9	V_{CC} = 3V, V_{in}= 0V	3	15	40	μA	3
		$D_0 - D_3$, R3 – R9	V_{CC} = 5V, V_{in} = 0V	30	60	120	μA	3

(Note 1) Applied to I/O pins with "CMOS" output selected by mask option.
(Note 2) Pull-up MOS current and output buffer current are excluded.
(Note 3) Applied to I/O pins "with Pull-up MOS" selected by mask option.

(3) Input/output characteristics for high voltage pin

(V_{CC}=2.7V to 6V, GND=0V, V_{disp}=V_{CC}-40V to V_{CC}, Ta=-20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Input "High" Voltage	V_{IH}	$D_4 - D_{15}$, R1 R2, R_{A0}, R_{A1}		0.85V_{CC}	–	V_{CC}+0.3	V	
Input "Low" Voltage	V_{IL}	$D_4 - D_{15}$, R1 R2, R_{A0}, R_{A1}		V_{CC}–40	–	0.15V_{CC}	V	
Output "High" Voltage	V_{OH}	$D_4 \sim D_{15}$	$-I_{OH}$=15mA, V_{CC}=5V±10%	V_{CC}–3.0	–	–	V	
			$-I_{OH}$=2.5mA	V_{CC}–1.0	–	–	V	
		R0 ~ R2	$-I_{OH}$=3 mA, V_{CC}=5V±10%	V_{CC}–3.0	–	–	V	
			$-I_{OH}$=0.5 mA	V_{CC}–1.0	–	–	V	
Output "Low" Voltage	V_{OL}	$D_4 - D_{15}$ R0 – R2	V_{disp} = V_{CC}–40V	–	–	V_{CC}–37	V	1
		$D_4 - D_{15}$ R0 – R2	150kΩ to V_{CC}–40V	–	–	V_{CC}–37	V	2
Input/Output Leakage Current	$\|I_{IL}\|$	$D_4 - D_{15}$ R0 – R2 R_{A0}, R_{A1}	V_{in} = V_{CC}–40V to V_{CC}	–	–	20	μA	3
Pull Down MOS Current	I_d	$D_4 - D_{15}$ R0 – R2 R_{A0}, R_{A1}	V_{disp} = V_{CC}–35V, V_{in} = V_{CC}	125	250	500	μA	4

(Note 1) Applied to I/O pins "with Pull-down MOS" selected by mask option.
(Note 2) Applied to I/O pins "without Pull-down MOS (PMOS Open Drain)" selected by mask option.
(Note 3) Pull-down MOS current and output buffer current are excluded.
(Note 4) Applied to I/O pins "with Pull-down MOS" selected by mask option.

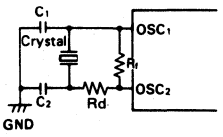
(4) AC characteristics

(V_{CC}=2.7V to 6V, GND=0V, V_{disp}=V_{CC}-40V to V_{CC}, Ta=-20 to +75°C, if not specified.)

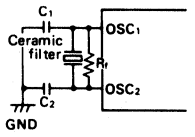
Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Oscillation Frequency	f_{osc}	OSC_1, OSC_2		0.4	2	2.25	MHz	
Instruction Cycle Time	t_{cyc}			3.55	4	20	μs	
Oscillator Stabilization Time	t_{RC}	OSC_1, OSC_2		–	–	60	ms	1
External Clock "High" Level Width	t_{CPH}	OSC_1		203	–	–	ns	2
External Clock "Low" Level Width	t_{CPL}	OSC_1		203	–	–	ns	2
External Clock Rise Time	t_{CPr}	OSC_1		–	–	20	ns	2
External Clock Fall Time	t_{CPf}	OSC_1		–	–	20	ns	2
$\overline{INT_0}$ "High" Level Width	t_{I0H}	$\overline{INT_0}$		2	–	–	t_{cyc}	3
$\overline{INT_0}$ "Low" Level Width	t_{I0L}	$\overline{INT_0}$		2	–	–	t_{cyc}	3
$\overline{INT_1}$ "High" Level Width	t_{I1H}	$\overline{INT_1}$		2	–	–	t_{cyc}	3
$\overline{INT_1}$ "Low" Level Width	t_{I1L}	$\overline{INT_1}$		2	–	–	t_{cyc}	3
RESET "High" Level Width	t_{RSTH}	RESET		2	–	–	t_{cyc}	4
Input Capacitance	C_{in}	all pins	f = 1 MHz V_{in} = 0 V	–	–	15	pF	
RESET Fall Time	t_{RSTf}			–	–	15	ms	4

(Note 1) Oscillator stabilization time is the time until the oscillator stabilizes after V_{CC} reaches 2.7V at "Power-on", or after RESET input level goes "High" by resetting to quit the stop mode by MCU reset. At power ON or recovering from stop mode, apply RESET input more than t_{RC} to obtain the necessary time for oscillator stabilization. The circuits used to measure the value are described below. When using crystal or ceramic filter oscillator, please ask a crystal oscillator maker's or ceramic filter maker's advice because oscillator stabilization time depends on the circuit constant and stray capacity.

Crystal oscillator



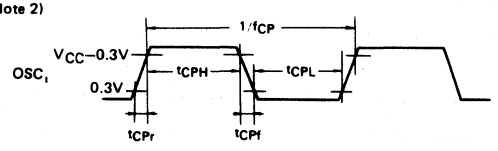
Ceramic filter oscillator



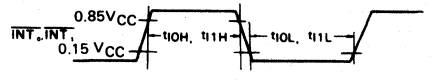
Crystal: 2.097152MHz DS-MGQ308 (Seiko Denshi)
Rf = 2MΩ ± 2%, Rd = 2.2kΩ ± 2%
C_1 = 10pF ± 20%
C_2 = 10pF ± 20%

Ceramic filter: CSA2.000MK (Murata)
Rf = 1MΩ ± 2%, C_1=C_2 = 30pF ±20%

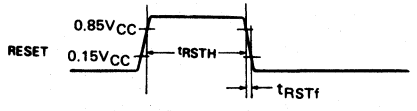
(Note 2)



(Note 3)



(Note 4)



(5) Serial interface timing characteristics

(V_{CC}=2.7V to 6V, GND=0V, V_{disp}=V_{CC}-40V to V_{CC}, Ta=-20 to +75°C, if not specified.)

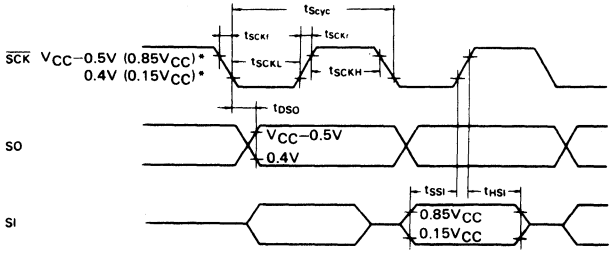
- At Transfer Clock Output

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Transfer Clock Cycle Time	t_{Scyc}	$\overline{SCK}$	(Note 2)	1	–	–	t_{cyc}	1, 2
Transfer Clock "High" Level Width	t_{SCKH}	$\overline{SCK}$	(Note 2)	0.5	–	–	t_{Scyc}	1, 2
Transfer Clock "Low" Level Width	t_{SCKL}	$\overline{SCK}$	(Note 2)	0.5	–	–	t_{Scyc}	1, 2
Transfer Clock Rise Time	t_{SCKr}	$\overline{SCK}$	(Note 2)	–	–	300	ns	1, 2
Transfer Clock Fall Time	t_{SCKf}	$\overline{SCK}$	(Note 2)	–	–	300	ns	1, 2
Serial Output Data Delay Time	t_{DSO}	SO	(Note 2)	–	–	600	ns	1, 2
Serial Input Data Set-up Time	t_{SSI}	SI		1000	–	–	ns	1
Serial Input Data Hold Time	t_{HSI}	SI		500	–	–	ns	1

- At Transfer Clock Input

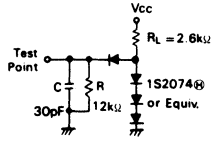
Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Transfer Clock Cycle Time	t_{Scyc}	$\overline{SCK}$		1	–	–	t_{cyc}	1
Transfer Clock "High" Level Width	t_{SCKH}	$\overline{SCK}$		0.5	–	–	t_{Scyc}	1
Transfer Clock "Low" Level Width	t_{SCKL}	$\overline{SCK}$		0.5	–	–	t_{Scyc}	1
Transfer Clock Rise Time	t_{SCKr}	$\overline{SCK}$		–	–	300	ns	1
Transfer Clock Fall Time	t_{SCKf}	$\overline{SCK}$		–	–	300	ns	1
Serial Output Data Delay Time	t_{DSO}	SO	(Note 2)	–	–	600	ns	1, 2
Serial Input Data Set-up Time	t_{SSI}	SI		1000	–	–	ns	1
Serial Input Data Hold Time	t_{HSI}	SI		500	–	–	ns	1

(Note 1) Timing Diagram of Serial Interface

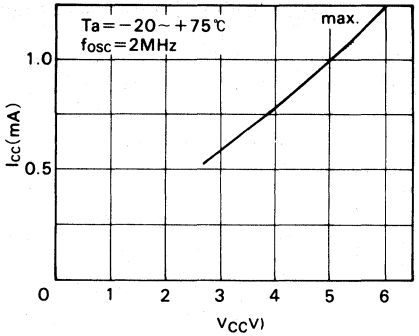


*V_{CC}–0.5V and 0.4V are the threshold voltage for transfer clock output.
0.85V_{CC} and 0.15V_{CC} are the threshold voltage for transfer clock input.

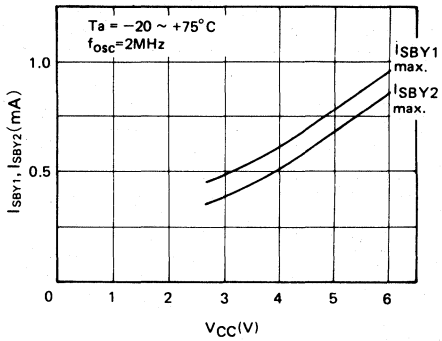
(Note 2) Timing Load Circuit



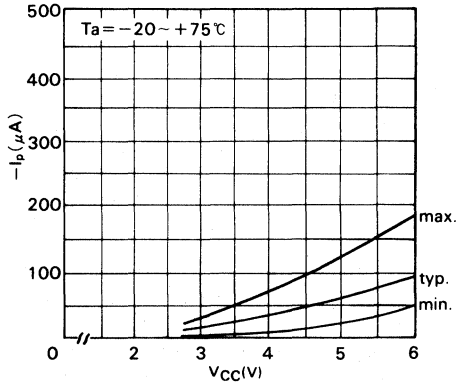
(6) Characteristics curve (Reference data)



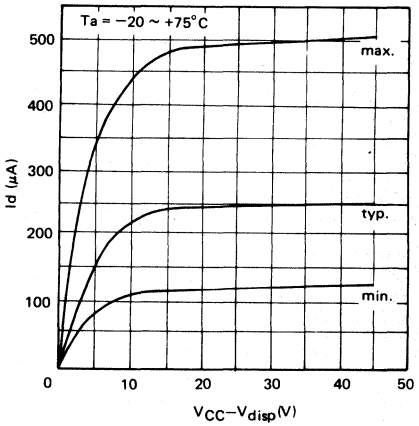
I_{CC} vs. V_{CC} Characteristics
(Crystal, Ceramic Filter Oscillator)



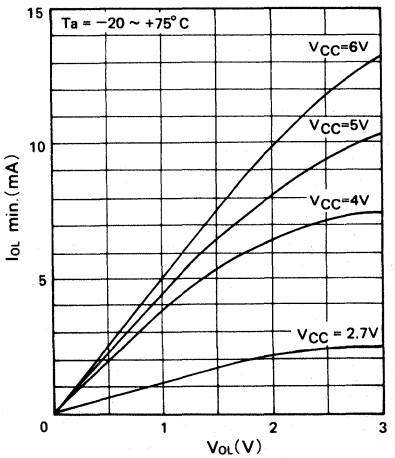
I_{SBY} vs. V_{CC} Characteristics
(Crystal, Ceramic Filter Oscillator)



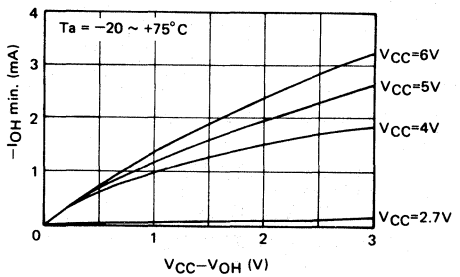
$-I_p$ (Pull-up MOS Current) vs.
V_{CC} Characteristics



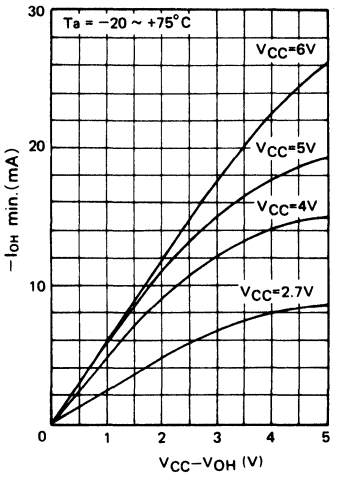
I_d (Pull-down MOS Current) vs.
($V_{CC}-V_{disp}$) Characteristics



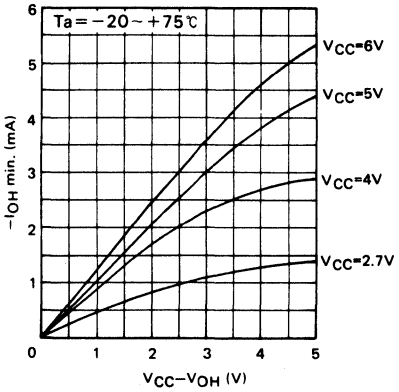
I_{OL} min. vs. V_{OL} Characteristics
(Standard Pin)



$-I_{OH}$ min. vs. ($V_{CC}-V_{OH}$) Characteristics
(Standard Pin "CMOS")



$-I_{OH}$ min. vs. ($V_{CC}-V_{OH}$) Characteristics
($D_4 - D_{15}$ Pins)



$-I_{OH}$ min. vs. ($V_{CC}-V_{OH}$) Characteristics
(R0 – R2 Pins)

5.4 HMCS402AC Electrical Characteristics

(1) DC characteristics (V_{CC}=4.5V to 6V, GND=0V, V_{disp}=V_{CC}-40V to V_{CC}, Ta=-20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Input "High" Voltage	V_{IH}	$\overline{RESET}$, $\overline{SCK}$, $\overline{INT_0}$, $\overline{INT_1}$		0.7V_{CC}	–	V_{CC}+0.3	V	
		SI		0.7V_{CC}	–	V_{CC}+0.3	V	
		OSC_1		V_{CC}-0.5	–	V_{CC}+0.3	V	
Input "Low" Voltage	V_{IL}	$\overline{RESET}$, $\overline{SCK}$, $\overline{INT_0}$, $\overline{INT_1}$		-0.3	–	0.22V_{CC}	V	
		SI		-0.3	–	0.22V_{CC}	V	
		OSC_1		-0.3	–	0.5	V	
Output "High" Voltage	V_{OH}	$\overline{SCK}$, SO	$-I_{OH}$ = 1.0 mA	V_{CC}-1.0	–	–	V	
			$-I_{OH}$ = 0.01 mA	V_{CC}-0.3	–	–	V	
Output "Low" Voltage	V_{OL}	$\overline{SCK}$, SO	I_{OL} = 1.6 mA	–	–	0.4	V	
Input/Output Leakage Current	$\|I_{IL}\|$	$\overline{RESET}$, $\overline{SCK}$, $\overline{INT_0}$, $\overline{INT_1}$ SI, SO, OSC_1	V_{in} = 0 V to V_{CC}	–	–	1	μA	1
Current Dissipation in Active Mode	I_{CC}	V_{CC}	V_{CC} = 5 V f_{osc} = 6 MHz	–	–	3.0	mA	2, 6
Current Dissipation in Standby Mode	I_{SBY1}	V_{CC}	Maximum Logic Operation V_{CC} = 5 V f_{osc} = 6 MHz	–	–	1.8	mA	3, 6
	I_{SBY2}	V_{CC}	Minimum Logic Operation V_{CC} = 5 V f_{osc} = 6MHz	–	–	1.35	mA	4, 6
Current Dissipation in Stop Mode	I_{stop}	V_{CC}	V_{in} ($\overline{TEST}$) = V_{CC} –0.3 V to V_{CC} V_{in} (RESET) = 0V to 0.3 V	–	–	10	μA	5
Stop Mode Retain Voltage	V_{stop}	V_{CC}		2	–	–	V	

(Note 1) Pull-up MOS current and output buffer current are excluded.
(Note 2) The MCU is in the reset state. The input/output current does not flow.
Test Conditions: MCU state;
- Reset state in Operation Mode

Pin state;
- RESET, $\overline{TEST}$ ... V_{CC} voltage
- D_0–D_3, R3–R9 ... V_{CC} voltage
- D_4–D_{15}, R0–R2, R_{A0}, R_{A1} ... V_{disp} voltage

(Note 3) The timer/counter operate with the fastest clock and input/output current does not flow.
Test Conditions: MCU state;
- Standby Mode
- Input/Output; Reset state
- TIMER-A; ÷2 prescaler divide ratio
- TIMER-B; ÷2 prescaler divide ratio
- SERIAL Interface ; Stop

Pin state;
- RESET ... GND voltage
- $\overline{TEST}$ ... V_{CC} voltage
- D_0–D_3, R3–R9 ... V_{CC} voltage
- D_4–D_{15}, R0–R2, R_{A0}, R_{A1} ... V_{disp} voltage

(Note 4) The timer/counter operate with the slowest clock and input/output current does not flow.
Test Conditions: MCU state;
- Standby Mode
- Input/Output; Reset state
- TIMER-A; ÷2048 prescaler divide ratio
- TIMER-B; ÷2048 prescaler divide ratio
- SERIAL Interface ; Stop

Pin state;
- RESET ... GND voltage
- $\overline{TEST}$ ... V_{CC} voltage
- D_0–D_3, R3–R9 ... V_{CC} voltage
- D_4–D_{15}, R0–R2, R_{A0}, R_{A1} ... V_{disp} voltage

(Note 5) Pull-down MOS current is excluded.
(Note 6) When f_{osc}=x [MHz], the Current Dissipation in Operation mode and Standby mode are estimated as follows:

$$\text{max. value } (f_{osc}=x\,[\text{MHz}]) = \frac{x}{6} \times \text{max. value } (f_{osc}=6\,[\text{MHz}])$$

(2) Input/output characteristics for standard pin

(V_{CC}=4.5V to 6V, GND=0V, V_{disp}=V_{CC}-40V to V_{CC}, Ta=-20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Input "High" Voltage	V_{IH}	$D_0 - D_3$, R3 – R5, R9		0.7V_{CC}	–	V_{CC}+0.3	V	
Input "Low" Voltage	V_{IL}	$D_0 - D_3$, R3 – R5, R9		–0.3	–	0.22V_{CC}	V	
Output "High" Voltage	V_{OH}	$D_0 - D_3$, R3 – R8	$-I_{OH}$ = 1.0 mA	V_{CC}–1.0	–	–	V	1
		$D_0 - D_3$, R3 – R8	$-I_{OH}$ = 0.01 mA	V_{CC}–0.3	–	–	V	1
Output "Low" Voltage	V_{OL}	$D_0 - D_3$, R3 – R8	I_{OL} = 1.6 mA	–	–	0.4	V	
Input/Output Leakage Current	$\|I_{IL}\|$	$D_0 - D_3$, R3 – R9	V_{in} = 0V to V_{CC}	–	–	1	μA	2
Pull-Up MOS Current	$-I_p$	$D_0 - D_3$, R3 – R9	V_{CC} = 5V, V_{in} = 0V	30	60	120	μA	3

(Note 1) Applied to I/O pins with "CMOS" Output selected by mask option.
(Note 2) Pull-up MOS current and output buffer current are excluded.
(Note 3) Applied to I/O pins "with Pull-up MOS" selected by mask option.

(3) Input/output characteristics for high voltage pin

(V_{CC}=4.5V to 6V, GND=0V, V_{disp}=V_{CC}-40V to V_{CC}, Ta=-20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Input "High" Voltage	V_{IH}	$D_4 - D_{15}$, R1 R2, R_{A0}, R_{A1}		0.7V_{CC}	–	V_{CC}+0.3	V	
Input "Low" Voltage	V_{IL}	$D_4 - D_{15}$, R1 R2, R_{A0}, R_{A1}		V_{CC}–40	–	0.22V_{CC}	V	
Output "High" Voltage	V_{OH}	$D_4 - D_{15}$	$-I_{OH}$=15mA, V_{CC}=5V ± 10%	V_{CC}–3.0	–	–	V	
			$-I_{OH}$=9mA	V_{CC}–2.0	–	–	V	
		R0 – R2	$-I_{OH}$=3mA, V_{CC}=5V ± 10%	V_{CC}–3.0	–	–	V	
			$-I_{OH}$=1.8 mA	V_{CC}–2.0	–	–	V	
Output "Low" Voltage	V_{OL}	$D_4 - D_{15}$ R0 – R2	V_{disp} = V_{CC}–40V	–	–	V_{CC}–37	V	1
		$D_4 - D_{15}$ R0 – R2	150kΩ to V_{CC}–40V	–	–	V_{CC}–37	V	2
Input/Output Leakage Current	$\|I_{IL}\|$	$D_4 - D_{15}$ R0 – R2 R_{A0}, R_{A1}	V_{in} = V_{CC}–40V to V_{CC}	–	–	20	μA	3
Pull Down MOS Current	I_d	$D_4 - D_{15}$ R0 – R2 R_{A0}, R_{A1}	V_{disp} = V_{CC}–35V, V_{in} = V_{CC}	125	250	500	μA	4

(Note 1) Applied to I/O pins "with Pull-down MOS" selected by mask option.
(Note 2) Applied to I/O pins "without Pull-down MOS (PMOS Open Drain)" selected by mask option.
(Note 3) Pull-down MOS current and output buffer current are excluded.
(Note 4) Applied to I/O pins "with Pull-down MOS" selected by mask option.

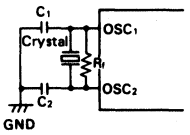
(4) AC characteristics

(V_{CC}=4.5V to 6V, GND=0V, V_{disp}=V_{CC}-40V to V_{CC}, Ta=-20 to +75°C, if not specified.)

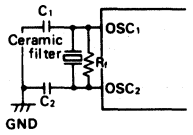
Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Oscillation Frequency	f_{osc}	OSC_1, OSC_2		0.4	6	6.2	MHz	
Instruction Cycle Time	t_{cyc}			1.29	1.33	20	μs	
Oscillator Stabilization Time	t_{RC}	OSC_1, OSC_2		–	–	20	ms	1
External Clock "High" Level Width	t_{CPH}	OSC_1		61	–	–	ns	2
External Clock "Low" Level Width	t_{CPL}	OSC_1		61	–	–	ns	2
External Clock Rise Time	t_{CPr}	OSC_1		–	–	20	ns	2
External Clock Fall Time	t_{CPf}	OSC_1		–	–	20	ns	2
$\overline{INT_0}$ "High" Level Width	t_{I0H}	$\overline{INT_0}$		2	–	–	t_{cyc}	3
$\overline{INT_0}$ "Low" Level Width	t_{I0L}	$\overline{INT_0}$		2	–	–	t_{cyc}	3
$\overline{INT_1}$ "High" Level Width	t_{I1H}	$\overline{INT_1}$		2	–	–	t_{cyc}	3
$\overline{INT_1}$ "Low" Level Width	t_{I1L}	$\overline{INT_1}$		2	–	–	t_{cyc}	3
RESET "High" Level Width	t_{RSTH}	RESET		2	–	–	t_{cyc}	4
Input Capacitance	C_{in}	all pins	f = 1 MHz V_{in} = 0 V	–	–	15	pF	
RESET Fall Time	t_{RSTf}			–	–	20	ms	4

(Note 1) Oscillator stabilization time is the time until the oscillator stabilizes after V_{CC} reaches 4.5V at "Power-on", or after RESET input level goes "High" by resetting to quit the stop mode by MCU reset. At power ON or recovering from stop mode, apply RESET input more than t_{RC} to obtain the necessary time for oscillator stabilization. The circuits used to measure the value are described below. When using crystal or ceramic filter oscillator, please ask a crystal oscillator maker's or ceramic filter maker's advice because oscillator stabilization time depends on the circuit constant and stray capacity.

Crystal oscillator



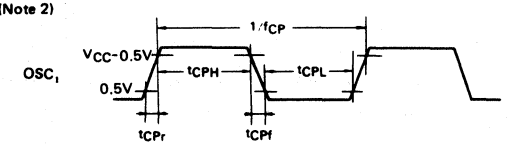
Ceramic filter oscillator



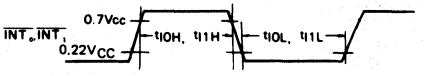
Crystal: 6.0MHz NC-18C (Nihon Denpa Kogyo)
Rf : 1MΩ ±2%
C_1 : 20pF ±20%
C_2 : 20pF ±20%

Ceramic filter : CSA6.00MG (Murata)
Rf : 1MΩ ±2%
C_1 : 30pF ±20%
C_2 : 30pF ±20%

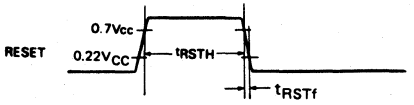
(Note 2)



(Note 3)



(Note 4)



(5) Serial interface timing characteristics

(V_{CC}=4.5V to 6V, GND=0V, V_{disp}=V_{CC}-40V to V_{CC}, Ta=-20 to +75°C, if not specified.)

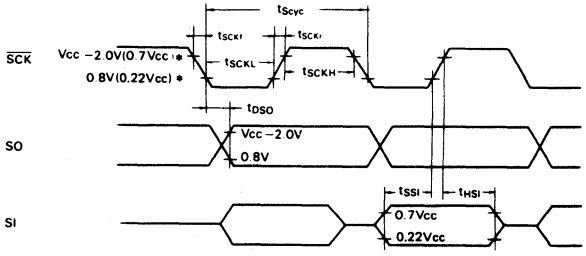
• At Transfer Clock Output

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Transfer Clock Cycle Time	t_{Scyc}	$\overline{SCK}$	(Note 2)	1	–	–	t_{cyc}	1, 2
Transfer Clock "High" Level Width	t_{SCKH}	$\overline{SCK}$	(Note 2)	0.5	–	–	t_{Scyc}	1, 2
Transfer Clock "Low" Level Width	t_{SCKL}	$\overline{SCK}$	(Note 2)	0.5	–	–	t_{Scyc}	1, 2
Transfer Clock Rise Time	t_{SCKr}	$\overline{SCK}$	(Note 2)	–	–	100	ns	1, 2
Transfer Clock Fall Time	t_{SCKf}	$\overline{SCK}$	(Note 2)	–	–	100	ns	1, 2
Serial Output Data Delay Time	t_{DSO}	SO	(Note 2)	–	–	250	ns	1, 2
Serial Input Data Set-up Time	t_{SSI}	SI		300	–	–	ns	1
Serial Input Data Hold Time	t_{HSI}	SI		150	–	–	ns	1

• At Transfer Clock Input

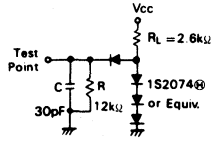
Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Transfer Clock Cycle Time	t_{Scyc}	$\overline{SCK}$		1	–	–	t_{cyc}	1
Transfer Clock "High" Level Width	t_{SCKH}	$\overline{SCK}$		0.5	–	–	t_{Scyc}	1
Transfer Clock "Low" Level Width	t_{SCKL}	$\overline{SCK}$		0.5	–	–	t_{Scyc}	1
Transfer Clock Rise Time	t_{SCKr}	$\overline{SCK}$		–	–	100	ns	1
Transfer Clock Fall Time	t_{SCKf}	$\overline{SCK}$		–	–	100	ns	1
Serial Output Data Delay Time	t_{DSO}	SO	(Note 2)	–	–	250	ns	1, 2
Serial Input Data Set-up Time	t_{SSI}	SI		300	–	–	ns	1
Serial Input Data Hold Time	t_{HSI}	SI		150	–	–	ns	1

(Note 1) Timing Diagram of Serial Interface

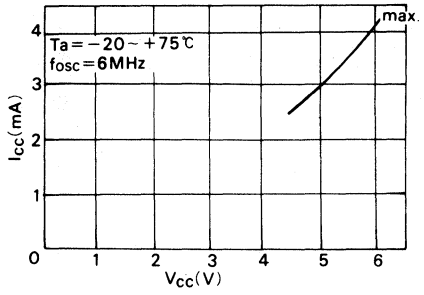


* V_{CC} − 2.0V and 0.8V are the threshold voltage for transfer clock output.
0.7 V_{CC} and 0.22 V_{CC} are the threshold voltage for transfer clock input.

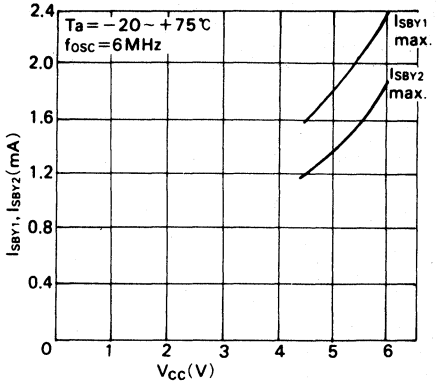
(Note 2) Timing Load Circuit



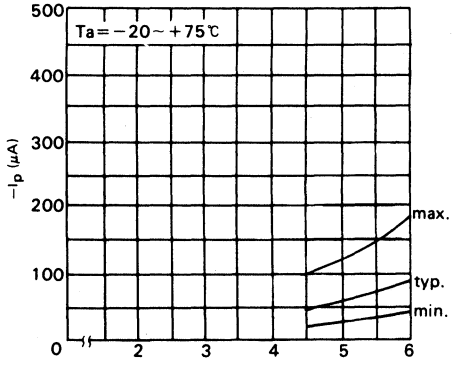
(6) Characteristics Curve (Reference data)



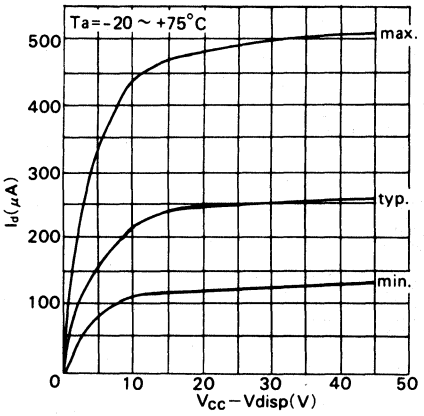
I_{CC} vs. V_{CC} Characteristics
(Crystal, Ceramic Filter Oscillator)



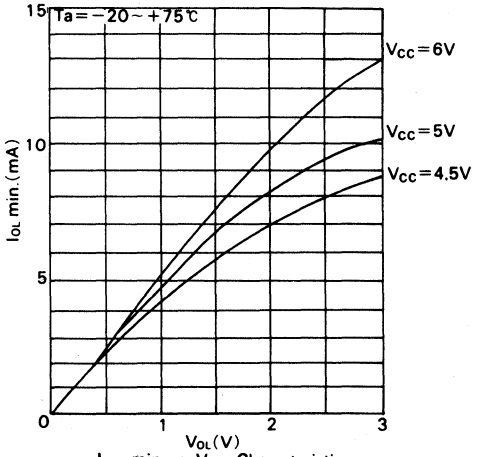
I_{SBY} vs. V_{CC} Characteristics
(Crystal, Ceramic Filter Oscillator)



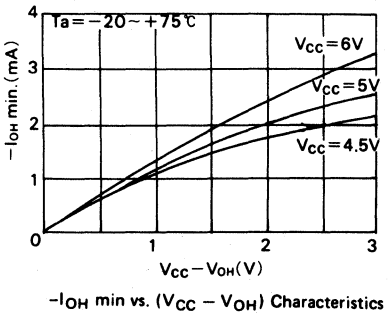
$-I_p$ (Pull-up MOS Current) vs.
V_{CC} Characteristics



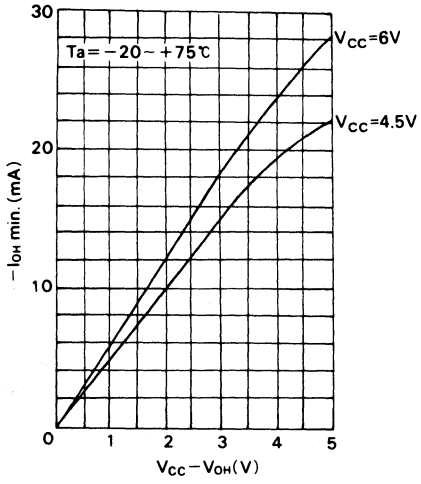
I_d (Pull-down MOS Current) vs.
(V_{CC} − V_{disp}) Characteristics



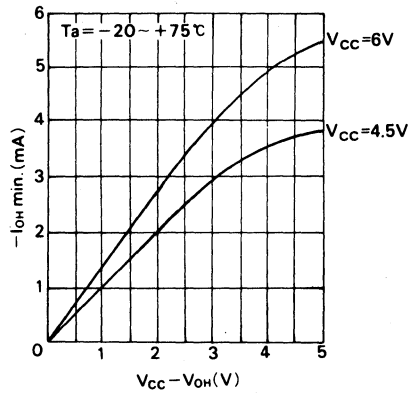
I_{OL} min. vs. V_{OL} Characteristics
(Standard Pin)



$-I_{OH}$ min vs. (V_{CC} − V_{OH}) Characteristics
(Standard Pin "CMOS")



$-I_{OH}$ min. vs. ($V_{CC} - V_{OH}$) Characteristics
($D_4 - D_{15}$ Pins)



$-I_{OH}$ min. vs. ($V_{CC} - V_{OH}$) Characteristics
(R0 – R2 Pins)

5.5 HMCS404C Electrical Characteristics

(1) DC characteristics

(V_{CC}=4V to 6V, GND=0V, V_{disp}=V_{CC}-40V to V_{CC}, Ta=-20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions		Value min	Value typ	Value max	Unit	Note
Input "High" Voltage	V_{IH}	$\overline{RESET}$, $\overline{SCK}$, $\overline{INT_0}$, $\overline{INT_1}$			0.7V_{CC}	–	V_{CC}+0.3	V	
		SI			0.7V_{CC}	–	V_{CC}+0.3	V	
		OSC_1			V_{CC}- 0.5	–	V_{CC}+0.3	V	
Input "Low" Voltage	V_{IL}	$\overline{RESET}$, $\overline{SCK}$, $\overline{INT_0}$, $\overline{INT_1}$			-0.3	–	0.22V_{CC}	V	
		SI			-0.3	–	0.22V_{CC}	V	
		OSC_1			-0.3	–	0.5	V	
Output "High" Voltage	V_{OH}	$\overline{SCK}$, SO	$-I_{OH}$ = 1.0 mA		V_{CC}-1.0	–	–	V	
			$-I_{OH}$ = 0.01 mA		V_{CC}-0.3	–	–	V	
Output "Low" Voltage	V_{OL}	$\overline{SCK}$, SO	I_{OL} = 1.6 mA		–	–	0.4	V	
Input/Output Leakage Current	$\|I_{IL}\|$	$\overline{RESET}$, $\overline{SCK}$, $\overline{INT_0}$, $\overline{INT_1}$, SI, SO, OSC_1	V_{in} = 0V to V_{CC}		–	–	1	μA	1
Current Dissipation in Active Mode	I_{CC}	V_{CC}	V_{CC}=5V	Crystal or Ceramic Filter Oscillator Option f_{osc} = 4MHz	–	–	2.0	mA	2, 6
				Resistor Oscillator Option f_{osc} = 4MHz	–	–	2.4	mA	2, 6
Current Dissipation in Standby Mode	I_{SBY1}	V_{CC}	Maximum Logic Operation V_{CC} = 5V	Crystal or Ceramic Filter Oscillator Option f_{osc} = 4MHz	–	–	1.2	mA	3, 6
				Resistor Oscillator Option f_{osc} = 4MHz	–	–	1.6	mA	3, 6
	I_{SBY2}	V_{CC}	Minimum Logic Operation V_{CC} = 5V	Crystal or Ceramic Filter Oscillator Option f_{osc} =4MHz	–	–	0.9	mA	4, 6
				Resistor Oscillator Option f_{osc} = 4MHz	–	–	1.3	mA	4, 6
Current Dissipation in Stop Mode	I_{stop}	V_{CC}	$V_{in}(\overline{TEST})$ = V_{CC}-0.3V to V_{CC} V_{in}(RESET) = 0V to 0.3V		–	–	10	μA	5
Stop Mode Retain Voltage	V_{stop}	V_{CC}			2	–	–	V	

(Note 1) Pull-up MOS current and output buffer current are excluded.
(Note 2) The MCU is in the reset state. The input/output current does not flow.
Test Conditions: MCU state; • Reset state in Operation Mode
Pin state; • RESET, $\overline{TEST}$ ··· V_{CC} voltage
• D_0– D_3, R3–R9 ··· V_{CC} voltage
• D_4–D_{15}, R0–R2, R_{A0}, R_{A1} ··· V_{disp} voltage
(Note 3) The timer/counter operate with the fastest clock and input/output current does not flow.
Test Conditions: MCU state; • Standby Mode
• Input/Output; Reset state
• TIMER-A; ÷2 prescaler divide ratio
• TIMER-B; ÷2 prescaler divide ratio
• SERIAL Interface ; Stop
Pin state; • RESET ··· GND voltage
• $\overline{TEST}$ ··· V_{CC} voltage
• D_0–D_3, R3–R9 ··· V_{CC} voltage
• D_4–D_{15}, R0–R2, R_{A0}, R_{A1} ··· V_{disp} voltage
(Note 4) The timer/counter operate with the slowest clock and input/output current does not flow.
Test Conditions: MCU state; • Standby Mode
• Input/Output; Reset state
• TIMER-A; ÷2048 prescaler divide ratio
• TIMER-B; ÷2048 prescaler divide ratio
• SERIAL Interface ; Stop
Pin state; • RESET ··· GND voltage
• $\overline{TEST}$ ··· V_{CC} voltage
• D_0–D_3, R3–R9 ··· V_{CC} voltage
• D_4–D_{15}, R0–R2, R_{A0}, R_{A1} ··· V_{disp} voltage
(Note 5) Pull-down MOS current is excluded.
(Note 6) When $f_{osc}=x$[MHz], the Current Dissipation in Operation mode and Standby mode are estimated as follows:

$$\text{max. value } (f_{osc}=x\text{[MHz]}) = \frac{x}{4} \times \text{max. value } (f_{osc}=4\text{[MHz]})$$

(2) Input/output characteristics for standard pin

(V_{CC}=4V to 6V, GND=0V, $V_{disp}=V_{CC}$-40V to V_{CC}, Ta=-20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Input "High" Voltage	V_{IH}	D_0 – D_3, R3 – R5, R9		0.7V_{CC}	–	V_{CC}+0.3	V	
Input "Low" Voltage	V_{IL}	D_0 – D_3, R3 – R5, R9		–0.3	–	0.22V_{CC}	V	
Output "High" Voltage	V_{OH}	D_0 – D_3, R3 – R8	$-I_{OH}$ = 1.0 mA	V_{CC}–1.0	–	–	V	1
		D_0 – D_3, R3 – R8	$-I_{OH}$ = 0.01 mA	V_{CC}–0.3	–	–	V	1
Output "Low" Voltage	V_{OL}	D_0 – D_3, R3 – R8	I_{OL} = 1.6 mA	–	–	0.4	V	
Input/Output Leakage Current	$\|I_{IL}\|$	D_0 – D_3, R3 – R9	V_{in} = 0V to V_{CC}	–	–	1	μA	2
Pull-Up MOS Current	$-I_P$	D_0 – D_3, R3 – R9	V_{CC} = 5V V_{in} = 0V	30	60	120	μA	3

(Note 1) Applied to I/O pins with "CMOS" Output selected by mask option.
(Note 2) Pull-up MOS current and output buffer current are excluded.
(Note 3) Applied to I/O pins with "with Pull-up MOS" selected by mask option.

(3) Input/output characteristics for high voltage pin

(V_{CC}=4V to 6V, GND=0V, V_{disp}=V_{CC}-40V to V_{CC}, Ta=-20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Input "High" Voltage	V_{IH}	$D_4 - D_{15}$, R1 R2, R_{A0}, R_{A1}		0.7V_{CC}	–	V_{CC}+0.3	V	
Input "Low" Voltage	V_{IL}	$D_4 - D_{15}$, R1 R2, R_{A0}, R_{A1}		V_{CC}−40	–	0.22V_{CC}	V	
Output "High" Voltage	V_{OH}	$D_4 - D_{15}$	$-I_{OH}$=15mA, V_{CC}=5V±10%	V_{CC}−3.0	–	–	V	
			$-I_{OH}$=9 mA	V_{CC}−2.0	–	–	V	
		R0 – R2	$-I_{OH}$=3 mA, V_{CC}=5V±10%	V_{CC}−3.0	–	–	V	
			$-I_{OH}$=1.8 mA	V_{CC}−2.0	–	–	V	
Output "Low" Voltage	V_{OL}	$D_4 - D_{15}$ R0 – R2	V_{disp} = V_{CC}−40V	–	–	V_{CC}−37	V	1
		$D_4 - D_{15}$ R0 – R2	150kΩ to V_{CC}−40V	–	–	V_{CC}−37	V	2
Input/Output Leakage Current	$\|I_{IL}\|$	$D_4 - D_{15}$ R0 – R2 R_{A0}, R_{A1}	V_{in} = V_{CC}−40V to V_{CC}	–	–	20	μA	3
Pull Down MOS Current	I_d	D_4−D_{15} R0−R2 R_{A0}, R_{A1}	V_{disp} = V_{CC}−35V V_{in} = V_{CC}	125	250	500	μA	4

(Note 1) Applied to I/O pins with "with Pull-down MOS" selected by mask option.
(Note 2) Applied to I/O pins with "without Pull-down MOS (PMOS Open Drain)" selected by mask option.
(Note 3) Pull-down MOS current and output buffer current are excluded.
(Note 4) Applied to I/O pins with "with Pull-down MOS" selected by mask option.

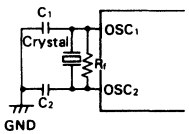
(4) AC characteristics

(V_{CC}=4V to 6V, GND=0V, V_{disp}=V_{CC}-40V to V_{CC}, Ta=-20 to +75°C, if not specified.)

	Item	Symbol	Pin Name	Test Conditions	Value min	Value typ	Value max	Unit	Note
Crystal or Ceramic Filter Oscillator	Oscillation Frequency	f_{osc}	OSC_1, OSC_2		0.4	4	4.5	MHz	
	Instruction Cycle Time	t_{cyc}			1.78	2	20	μs	
	Oscillator Stabilization Time	t_{RC}	OSC_1, OSC_2		–	–	20	ms	1
Resistor Oscillator	Oscillation Frequency	f_{osc}	OSC_1, OSC_2	R_f=20kΩ±2%	1.8	3.0	4.2	MHz	
	Instruction Cycle Time	t_{cyc}		R_f=20kΩ±2%	1.9	2.66	4.44	μs	
	Oscillator Stabilization Time	t_{RC}	OSC_1, OSC_2	R_f=20kΩ±2%	–	–	0.5	ms	1
External Clock	External Clock Frequency	f_{CP}	OSC_1		0.4	–	4.5	MHz	2
	External Clock "High" Level Width	t_{CPH}	OSC_1		92	–	–	ns	2
	External Clock "Low" Level Width	t_{CPL}	OSC_1		92	–	–	ns	2
	External Clock Rise Time	t_{CPr}	OSC_1		–	–	20	ns	2
	External Clock Fall Time	t_{CPf}	OSC_1		–	–	20	ns	2
	Instruction Cycle Time	t_{cyc}			1.78	–	20	μs	2
$\overline{INT_0}$ "High" Level Width		t_{I0H}	$\overline{INT_0}$		2	–	–	t_{cyc}	3
$\overline{INT_0}$ "Low" Level Width		t_{I0L}	$\overline{INT_0}$		2	–	–	t_{cyc}	3
$\overline{INT_1}$ "High" Level Width		t_{I1H}	$\overline{INT_1}$		2	–	–	t_{cyc}	3
$\overline{INT_1}$ "Low" Level Width		t_{I1L}	$\overline{INT_1}$		2	–	–	t_{cyc}	3
RESET "High" Level Width		t_{RSTH}	RESET		2	–	–	t_{cyc}	4
Input Capacitance		C_{in}	all pins	f=1MHz V_{in} = 0V	–	–	15	pF	
RESET Fall Time		t_{RSTf}			–	–	20	ms	4

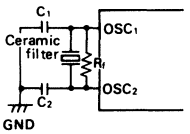
(Note 1) Oscillator stabilization time is the time until the oscillator stabilizes after V_{CC} reaches 4.0V at "Power-on", or after RESET input level goes to "High" by resetting to quit the stop mode by MCU reset on the circuits below. At power ON or recovering from stop mode, apply RESET input more than t_{RC} to obtain the necessary time for oscillator stabilization. When using crystal or ceramic filter oscillator, please ask a crystal oscillator maker's or ceramic filter maker's advice because oscillator stabilization time depends on the circuit constant and stray capacity.

Crystal oscillator



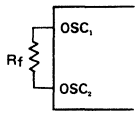
Crystal: 4.194304MHz NC-18C(Nihon Denpa Kogyo)
Rf : 1MΩ ±2%
C_1 : 22pF±20%
C_2 : 22pF±20%

Ceramic filter oscillator



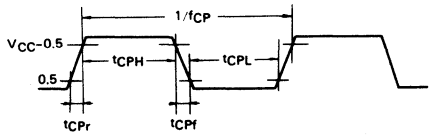
Ceramic filter: CSA4.00MG (Murata)
Rf : 1MΩ±2%
C_1 : 30pF±20%
C_2 : 30pF±20%

Resistor oscillator

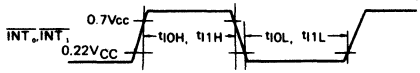


Rf : 20kΩ±2%

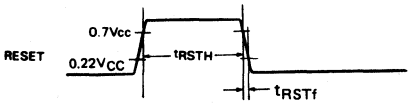
(Note 2)



(Note 3)



(Note 4)



(5) Serial interface timing characteristics

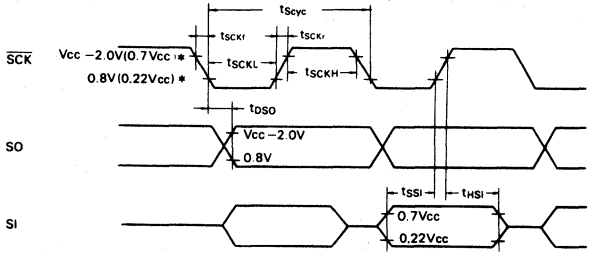
(V_{CC}=4V to 6V, GND=0V, V_{disp}=V_{CC}-40V to V_{CC}, Ta=-20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Transfer Clock Cycle Time	t_{Scyc}	$\overline{SCK}$	(Note 2)	1	–	–	t_{cyc}	1, 2
Transfer Clock "High" Level Width	t_{SCKH}	$\overline{SCK}$	(Note 2)	0.5	–	–	t_{Scyc}	1, 2
Transfer Clock "Low" Level Width	t_{SCKL}	$\overline{SCK}$	(Note 2)	0.5	–	–	t_{Scyc}	1, 2
Transfer Clock Rise Time	t_{SCKr}	$\overline{SCK}$	(Note 2)	–	–	100	ns	1, 2
Transfer Clock Fall Time	t_{SCKf}	$\overline{SCK}$	(Note 2)	–	–	100	ns	1, 2
Serial Output Data Delay Time	t_{DSO}	SO	(Note 2)	–	–	300	ns	1, 2
Serial Input Data Set-up Time	t_{SSI}	SI		500	–	–	ns	1
Serial Input Data Hold Time	t_{HSI}	SI		150	–	–	ns	1

• At Transfer Clock Input

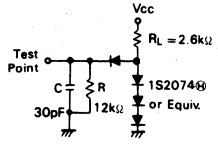
Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Transfer Clock Cycle Time	t_{Scyc}	$\overline{SCK}$		1	–	–	t_{cyc}	1
Transfer Clock "High" Level Width	t_{SCKH}	$\overline{SCK}$		0.5	–	–	t_{Scyc}	1
Transfer Clock "Low" Level Width	t_{SCKL}	$\overline{SCK}$		0.5	–	–	t_{Scyc}	1
Transfer Clock Rise Time	t_{SCKr}	$\overline{SCK}$		–	–	100	ns	1
Transfer Clock Fall Time	t_{SCKf}	$\overline{SCK}$		–	–	100	ns	1
Serial Output Data Delay Time	t_{DSO}	SO	(Note 2)	–	–	300	ns	1, 2
Serial Input Data Set-up Time	t_{SSI}	SI		500	–	–	ns	1
Serial Input Data Hold Time	t_{HSI}	SI		150	–	–	ns	1

(Note 1) Timing Diagram of Serial Interface

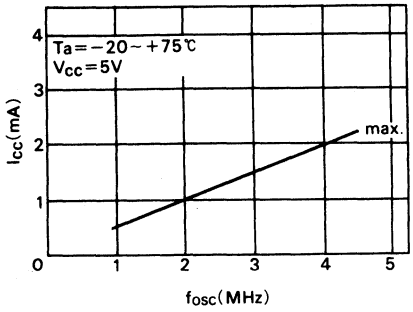


* V_{CC} − 2.0V and 0.8V are the threshold voltage for transfer clock output.
0.7 V_{CC} and 0.22 V_{CC} are the threshold voltage for transfer clock input.

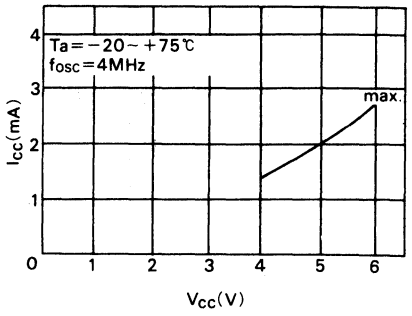
(Note 2) Timing Load Circuit



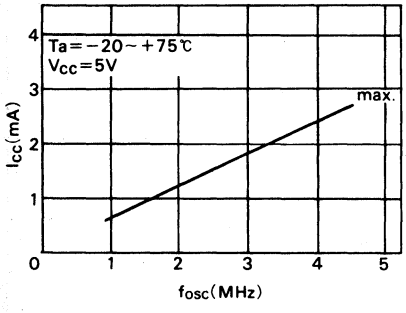
(6) Characteristics Curve (Reference data)



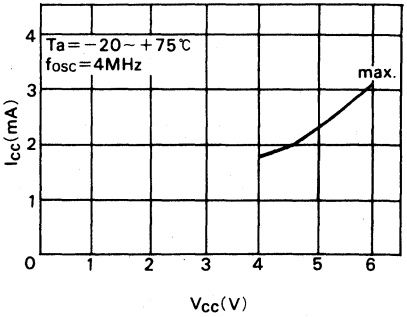
I_{CC} vs. f_{osc} Characteristics
(Crystal, Ceramic Filter Oscillator Option)



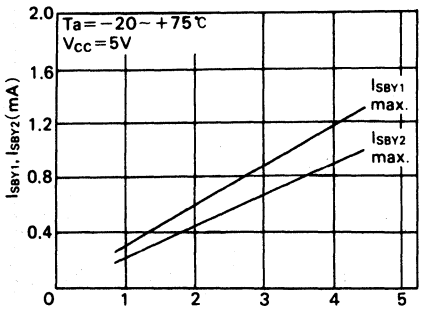
I_{CC} vs. V_{CC} Characteristics
(Crystal, Ceramic Filter Oscillator Option)



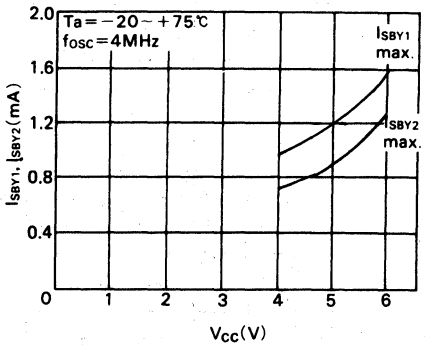
I_{CC} vs. f_{osc} Characteristics
(Resistor Oscillator Option)



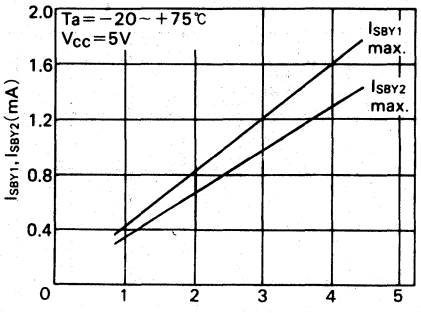
I_{CC} vs. V_{CC} Characteristics
(Resistor Oscillator Option)



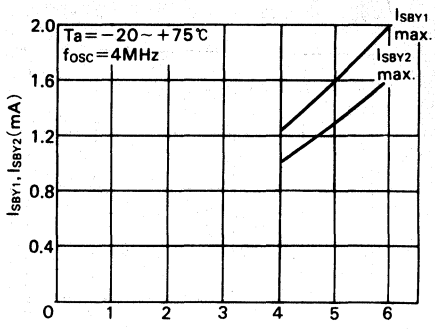
I_{SBY} vs. f_{osc} Characteristics
(Crystal, Ceramic Filter Oscillator Option)



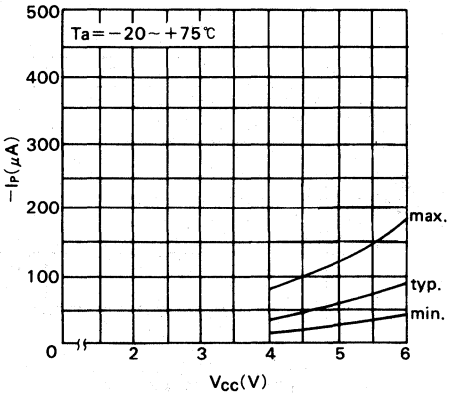
I_{SBY} vs. V_{CC} Characteristics
(Crystal, Ceramic Filter Oscillator Option)



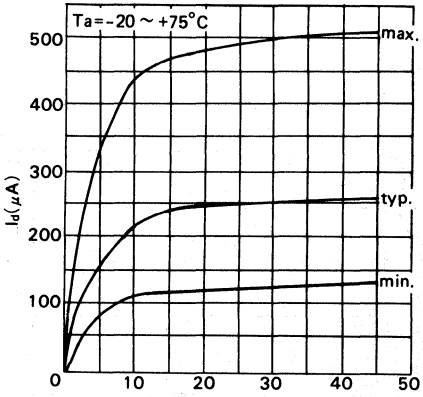
I_{SBY} vs. f_{osc} Characteristics
(Resistor Oscillator Option)



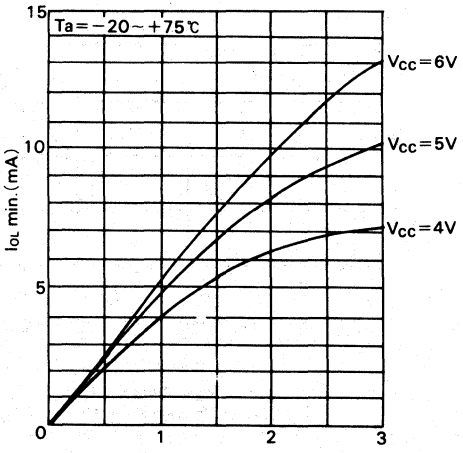
I_{SBY} vs. V_{CC} Characteristics
(Resistor Oscillator Option)



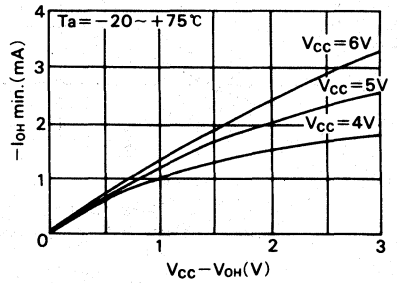
$-I_P$ (Pull-up MOS Current) vs.
V_{CC} Characteristics



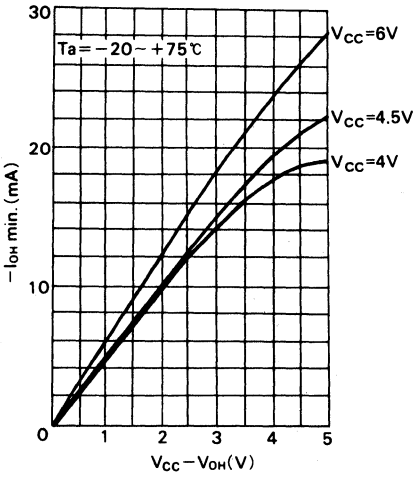
I_d (Pull-down MOS Current) vs.
($V_{CC} - V_{disp}$) Characteristics



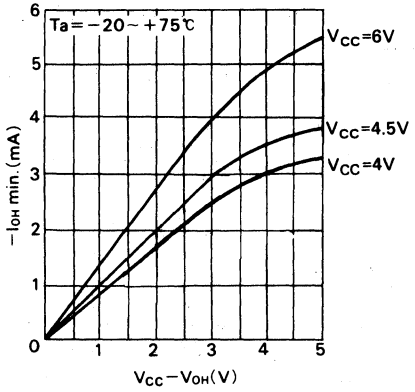
I_{OL} min. vs. V_{OL} Characteristics
(Standard Pin)



$-I_{OH}$ min vs. ($V_{CC} - V_{OH}$) Characteristics
(Standard Pin "CMOS")



$-I_{OH}$ min. vs. ($V_{CC} - V_{OH}$) Characteristics
($D_4 - D_{15}$ Pins)



$-I_{OH}$ min. vs. ($V_{CC} - V_{OH}$) Characteristics
(R0 – R2 Pins)

5.6 HMCS404CL Electrical Characteristics

(1) DC characteristics

(V_{CC}=2.7V to 6V, GND=0V, V_{disp}=V_{CC}-40V to V_{CC}, Ta=-20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions	Value min	Value typ	Value max	Unit	Note
Input "High" Voltage	V_{IH}	RESET, $\overline{SCK}$, $\overline{INT_0}$, $\overline{INT_1}$		0.85V_{CC}	–	V_{CC}+0.3	V	
		SI		0.85V_{CC}	–	V_{CC}+0.3	V	
		OSC_1		V_{CC}–0.3	–	V_{CC}+0.3	V	
Input "Low" Voltage	V_{IL}	RESET, $\overline{SCK}$, $\overline{INT_0}$, $\overline{INT_1}$		-0.3	–	0.15V_{CC}	V	
		SI		-0.3	–	0.15V_{CC}	V	
		OSC_1		-0.3	–	0.3	V	
Output "High" Voltage	V_{OH}	$\overline{SCK}$, SO	$-I_{OH}$ = 0.1 mA	V_{CC}–0.5	–	–	V	
Output "Low" Voltage	V_{OL}	$\overline{SCK}$, SO	I_{OL} = 0.4mA	–	–	0.4	V	
Input/Output Leakage Current	$\|I_{IL}\|$	RESET, $\overline{SCK}$, $\overline{INT_0}$, $\overline{INT_1}$, SI, SO, OSC_1	V_{in} = 0 V to V_{CC}	–	–	1	μA	1
Current Dissipation in Active Mode	I_{CC}	V_{CC}	V_{CC} = 3 V f_{osc} = 2 MHz	–	–	0.6	mA	2, 6
• Current Dissipation in Standby Mode	I_{SBY1}	V_{CC}	Maximum Logic Operation V_{CC} = 3 V f_{osc} = 2 MHz	–	–	0.5	mA	3, 6
	I_{SBY2}	V_{CC}	Minimum Logic Operation V_{CC} = 3 V f_{osc} = 2MHz	–	–	0.4	mA	4, 6
Current Dissipation in Stop Mode	I_{stop}	V_{CC}	V_{in} ($\overline{TEST}$) = V_{CC}-0.2V to V_{CC} V_{in} (RESET) = 0V to 0.2 V	–	–	10	μA	5
Stop Mode Retain Voltage	V_{stop}	V_{CC}		2	–	–	V	

(Note 1) Pull-up MOS current and output buffer current are excluded.

(Note 2) The MCU is in the reset state. The input/output current does not flow.

Test Conditions: MCU state;
- Reset state in Operation Mode

Pin state;
- RESET, $\overline{TEST}$ ... V_{CC} voltage
- D_0–D_3, R3–R9 ... V_{CC} voltage
- D_4–D_{15}, R0–R2, R_{A0}, R_{A1} ... V_{disp} voltage

(Note 3) The timer/counter operate with the fastest clock and input/output current does not flow.

Test Conditions: MCU state;
- Standby Mode
- Input/Output; Reset state
- TIMER-A; ÷2 prescaler divide ratio
- TIMER-B; ÷2 prescaler divide ratio
- SERIAL Interface ; Stop

Pin state;
- RESET ... GND voltage
- $\overline{TEST}$ ... V_{CC} voltage
- D_0–D_3, R3–R9 ... V_{CC} voltage
- D_4–D_{15}, R0–R2, R_{A0}, R_{A1} ... V_{disp} voltage

(Note 4) The timer/counter operate with the slowest clock and input/output current does not flow.

Test Conditions: MCU state;
- Standby Mode
- Input/Output; Reset state
- TIMER-A; ÷2048 prescaler divide ratio
- TIMER-B; ÷2048 prescaler divide ratio
- SERIAL Interface ; Stop

Pin state;
- RESET ... GND voltage
- $\overline{TEST}$ ... V_{CC} voltage
- D_0–D_3, R3–R9 ... V_{CC} voltage
- D_4–D_{15}, R0–R2, R_{A0}, R_{A1} ... V_{disp} voltage

(Note 5) Pull-down MOS current is excluded.

(Note 6) When f_{osc}=x[MHz], the Current Dissipation in Operation mode and Standby mode are estimated as follows:

[When Divide-by-8 (D-8) option is selected.] max. value (f_{osc}=x[MHz]) = $\frac{x}{2}$ x max. value (f_{osc}=2[MHz])

(2) Input/output characteristics for standard pin

(V_{CC}=2.7V to 6V, GND=0V, V_{disp}=V_{CC}-40V to V_{CC}, Ta=-20 to 75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Input "High" Voltage	V_{IH}	$D_0 - D_3$, R3 – R5, R9		0.85V_{CC}	–	V_{CC}+0.3	V	
Input "Low" Voltage	V_{IL}	$D_0 - D_3$, R3 – R5, R9		-0.3	–	0.15 V_{CC}	V	
Output "High" Voltage	V_{OH}	$D_0 - D_3$, R3 – R8	$-I_{OH}$ = 0.1 mA	V_{CC}–0.5	–	–	V	1
Output "Low" Voltage	V_{OL}	$D_0 - D_3$, R3 – R8	I_{OL} = 0.4 mA	–	–	0.4	V	
Input/Output Leakage Current	$\|I_{IL}\|$	$D_0 - D_3$, R3 – R9	V_{in}= 0V to V_{CC}	–	–	1	μA	2
Pull-Up MOS Current	$-I_p$	$D_0 - D_3$, R3 – R9	V_{CC} = 3V, V_{in}= 0V	3	15	40	μA	3
		$D_0 - D_3$, R3 – R9	V_{CC} = 5V, V_{in} = 0V	30	60	120	μA	3

(Note 1) Applied to I/O pins with "CMOS" output selected by mask option.
(Note 2) Pull-up MOS current and output buffer current are excluded.
(Note 3) Applied to I/O pins "with Pull-up MOS" selected by mask option.

(3) Input/output characteristics for high voltage pin

(V_{CC}=2.7V to 6V, GND=0V, V_{disp}=V_{CC}-40V to V_{CC}, Ta=-20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Input "High" Voltage	V_{IH}	$D_4 - D_{15}$, R1 R2, R_{A0}, R_{A1}		0.85V_{CC}	–	V_{CC}+0.3	V	
Input "Low" Voltage	V_{IL}	$D_4 - D_{15}$, R1 R2, R_{A0}, R_{A1}		V_{CC}–40	–	0.15V_{CC}	V	
Output "High" Voltage	V_{OH}	$D_4 - D_{15}$	$-I_{OH}$=15mA, V_{CC}=5V±10%	V_{CC}–3.0	–	–	V	
			$-I_{OH}$=2.5mA	V_{CC}–1.0	–	–	V	
		R0 – R2	$-I_{OH}$=3 mA, V_{CC}=5V±10%	V_{CC}–3.0	–	–	V	
			$-I_{OH}$=0.5 mA	V_{CC}–1.0	–	–	V	
Output "Low" Voltage	V_{OL}	$D_4 - D_{15}$ R0 – R2	V_{disp} = V_{CC}–40V	–	–	V_{CC}–37	V	1
		$D_4 - D_{15}$ R0 – R2	150kΩ to V_{CC}–40V	–	–	V_{CC}–37	V	2
Input/Output Leakage Current	$\|I_{IL}\|$	$D_4 - D_{15}$ R0 – R2 R_{A0}, R_{A1}	V_{in} = V_{CC}–40V to V_{CC}	–	–	20	μA	3
Pull Down MOS Current	I_d	$D_4 - D_{15}$ R0 – R2 R_{A0}, R_{A1}	V_{disp} = V_{CC}–35V, V_{in} = V_{CC}	125	250	500	μA	4

(Note 1) Applied to I/O pins "with Pull-down MOS" selected by mask option.
(Note 2) Applied to I/O pins "without Pull-down MOS (PMOS Open Drain)" selected by mask option.
(Note 3) Pull-down MOS current and output buffer current are excluded.
(Note 4) Applied to I/O pins "with Pull-down MOS" selected by mask option.

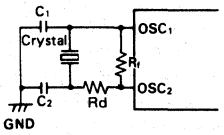
(4) AC characteristics

(V_{CC}=2.7V to 6V, GND=0V, V_{disp}=V_{CC}-40V to V_{CC}, Ta=-20 to +75°C, if not specified.)

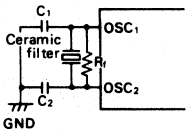
Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Oscillation Frequency	f_{osc}	OSC_1, OSC_2		0.4	2	2.25	MHz	
Instruction Cycle Time	t_{cyc}			3.55	4	20	μs	
Oscillator Stabilization Time	t_{RC}	OSC_1, OSC_2		–	–	60	ms	1
External Clock "High" Level Width	t_{CPH}	OSC_1		203	–	–	ns	2
External Clock "Low" Level Width	t_{CPL}	OSC_1		203	–	–	ns	2
External Clock Rise Time	t_{CPr}	OSC_1		–	–	20	ns	2
External Clock Fall Time	t_{CPf}	OSC_1		–	–	20	ns	2
$\overline{INT_0}$ "High" Level Width	t_{I0H}	$\overline{INT_0}$		2	–	–	t_{cyc}	3
$\overline{INT_0}$ "Low" Level Width	t_{I0L}	$\overline{INT_0}$		2	–	–	t_{cyc}	3
$\overline{INT_1}$ "High" Level Width	t_{I1H}	$\overline{INT_1}$		2	–	–	t_{cyc}	3
$\overline{INT_1}$ "Low" Level Width	t_{I1L}	$\overline{INT_1}$		2	–	–	t_{cyc}	3
RESET "High" Level Width	t_{RSTH}	RESET		2	–	–	t_{cyc}	4
Input Capacitance	C_{in}	all pins	f = 1 MHz V_{in} = 0 V	–	–	15	pF	
RESET Fall Time	t_{RSTf}			–	–	15	ms	4

(Note 1) Oscillator stabilization time is the time until the oscillator stabilizes after V_{CC} reaches 2.7V at "Power-on", or after RESET input level goes "High" by resetting to quit the stop mode by MCU reset. At power ON or recovering from stop mode, apply RESET input more than t_{RC} to obtain the necessary time for oscillator stabilization. The circuits used to measure the value are described below. When using crystal or ceramic filter oscillator, please ask a crystal oscillator maker's or ceramic filter maker's advice because oscillator stabilization time depends on the circuit constant and stray capacity.

Crystal oscillator



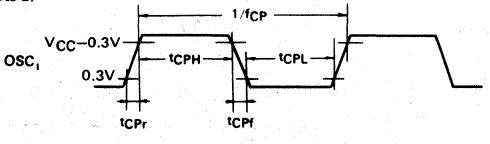
Ceramic filter oscillator



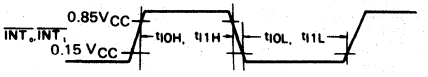
Crystal: 2.097152MHz DS-MGQ308 (Seiko Denshi)
Rf = 2MΩ ± 2%, Rd = 2.2kΩ ± 2%
C_1 = 10pF ± 20%
C_2 = 10pF ± 20%

Ceramic filter: CSA2.000MK (Murata)
Rf = 1MΩ ± 2%, C_1=C_2 = 30pF ±20%

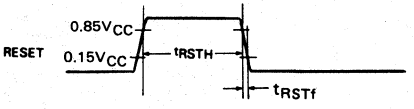
(Note 2)



(Note 3)



(Note 4)



(5) Serial interface timing characteristics

(V_{CC}=2.7V to 6V, GND=0V, V_{disp}=V_{CC}-40V to V_{CC}, Ta=-20 to +75°C, if not specified.)

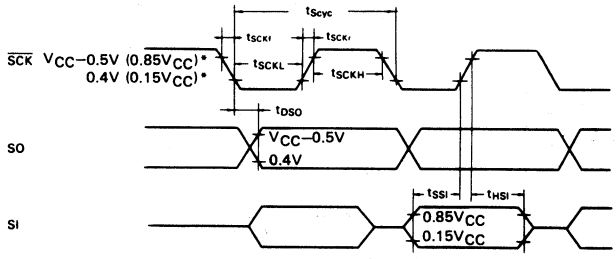
• At Transfer Clock Output

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Transfer Clock Cycle Time	t_{Scyc}	$\overline{SCK}$	(Note 2)	1	–	–	t_{cyc}	1, 2
Transfer Clock "High" Level Width	t_{SCKH}	$\overline{SCK}$	(Note 2)	0.5	–	–	t_{Scyc}	1, 2
Transfer Clock "Low" Level Width	t_{SCKL}	$\overline{SCK}$	(Note 2)	0.5	–	–	t_{Scyc}	1, 2
Transfer Clock Rise Time	t_{SCKr}	$\overline{SCK}$	(Note 2)	–	–	300	ns	1, 2
Transfer Clock Fall Time	t_{SCKf}	$\overline{SCK}$	(Note 2)	–	–	300	ns	1, 2
Serial Output Data Delay Time	t_{DSO}	SO	(Note 2)	–	–	600	ns	1, 2
Serial Input Data Set-up Time	t_{SSI}	SI		1000	–	–	ns	1
Serial Input Data Hold Time	t_{HSI}	SI		500	–	–	ns	1

• At Transfer Clock Input

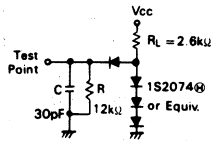
Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Transfer Clock Cycle Time	t_{Scyc}	$\overline{SCK}$		1	–	–	t_{cyc}	1
Transfer Clock "High" Level Width	t_{SCKH}	$\overline{SCK}$		0.5	–	–	t_{Scyc}	1
Transfer Clock "Low" Level Width	t_{SCKL}	$\overline{SCK}$		0.5	–	–	t_{Scyc}	1
Transfer Clock Rise Time	t_{SCKr}	$\overline{SCK}$		–	–	300	ns	1
Transfer Clock Fall Time	t_{SCKf}	$\overline{SCK}$		–	–	300	ns	1
Serial Output Data Delay Time	t_{DSO}	SO	(Note 2)	–	–	600	ns	1, 2
Serial Input Data Set-up Time	t_{SSI}	SI		1000	–	–	ns	1
Serial Input Data Hold Time	t_{HSI}	SI		500	–	–	ns	1

(Note 1) Timing Diagram of Serial Interface

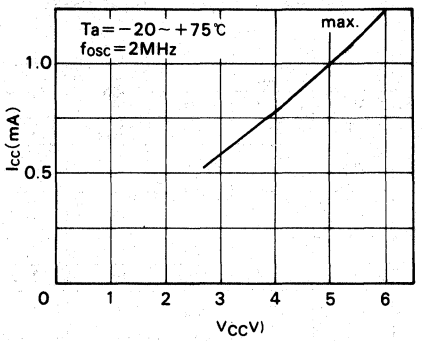


*V_{CC}–0.5V and 0.4V are the threshold voltage for transfer clock output.
0.85V_{CC} and 0.15V_{CC} are the threshold voltage for transfer clock input.

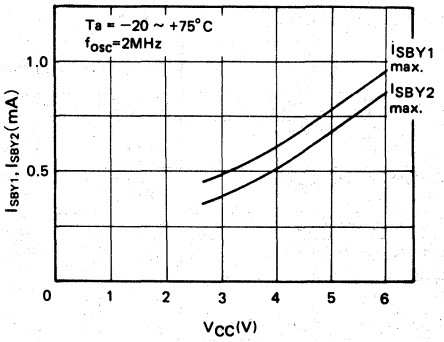
(Note 2) Timing Load Circuit



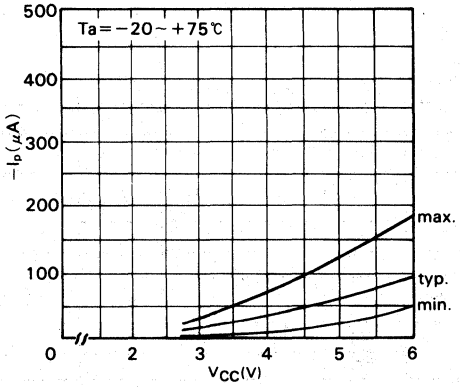
(6) Characteristics Curve (Reference data)



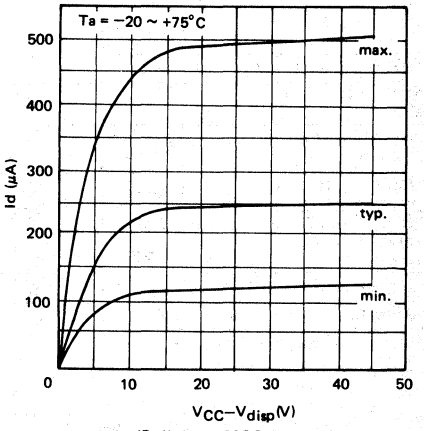
I_{CC} vs. V_{CC} Characteristics
(Crystal, Ceramic Filter Oscillator)



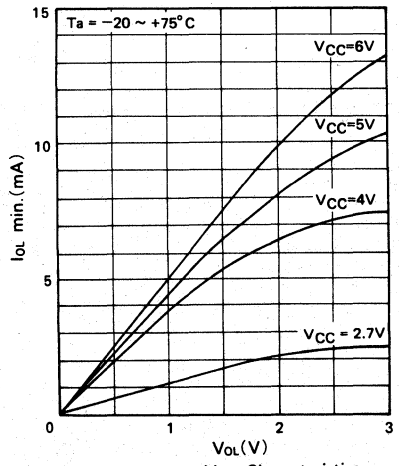
I_{SBY} vs. V_{CC} Characteristics
(Crystal, Ceramic Filter Oscillator)



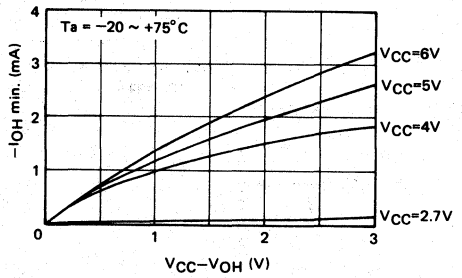
$-I_P$ (Pull-up MOS Current) vs.
V_{CC} Characteristics



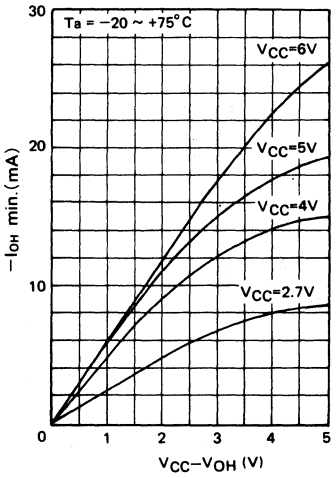
I_d (Pull-down MOS Current) vs.
($V_{CC}-V_{disp}$) Characteristics



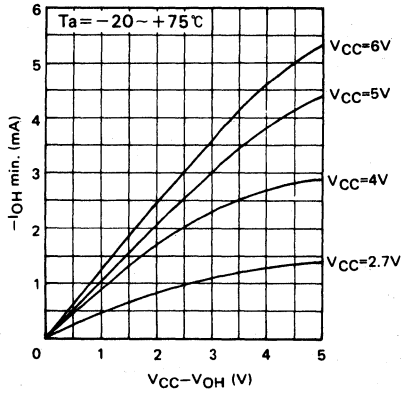
I_{OL} min. vs. V_{OL} Characteristics
(Standard Pin)



$-I_{OH}$ min. vs. ($V_{CC}-V_{OH}$) Characteristics
(Standard Pin "CMOS")



$-I_{OH}$ min. vs. ($V_{CC}-V_{OH}$) Characteristics
(D_4 – D_{15} Pins)



$-I_{OH}$ min. vs. ($V_{CC}-V_{OH}$) Characteristics
(R0 – R2 Pins)

5.7 HMCS404AC Electrical Characteristics

(1) DC characteristics

(V_{CC}=4.5V to 6V, GND=0V, V_{disp}=V_{CC}-40V to V_{CC}, Ta=-20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions	Value min	Value typ	Value max	Unit	Note
Input "High" Voltage	V_{IH}	RESET, SCK, INT0, INT1		0.7V_{CC}	–	V_{CC}+0.3	V	
		SI		0.7V_{CC}	–	V_{CC}+0.3	V	
		OSC1		V_{CC}–0.5	–	V_{CC}+0.3	V	
Input "Low" Voltage	V_{IL}	RESET, SCK, INT0, INT1		-0.3	–	0.22V_{CC}	V	
		SI		-0.3	–	0.22V_{CC}	V	
		OSC1		-0.3	–	0.5	V	
Output "High" Voltage	V_{OH}	SCK, SO	$-I_{OH}$ = 1.0 mA	V_{CC}–1.0	–	–	V	
			$-I_{OH}$ = 0.01 mA	V_{CC}–0.3	–	–	V	
Output "Low" Voltage	V_{OL}	SCK, SO	I_{OL} = 1.6 mA	–	–	0.4	V	
Input/Output Leakage Current	$\|I_{IL}\|$	RESET, SCK, INT0, INT1 SI, SO, OSC1	V_{in} = 0 V to V_{CC}	–	–	1	μA	1
Current Dissipation in Active Mode	I_{CC}	V_{CC}	V_{CC} = 5 V f_{osc} = 6 MHz	–	–	3.0	mA	2, 6
Current Dissipation in Standby Mode	I_{SBY1}	V_{CC}	Maximum Logic Operation V_{CC} = 5 V f_{osc} = 6 MHz	–	–	1.8	mA	3, 6
	I_{SBY2}	V_{CC}	Minimum Logic Operation V_{CC} = 5 V f_{osc} = 6MHz	–	–	1.35	mA	4, 6
Current Dissipation in Stop Mode	I_{stop}	V_{CC}	V_{in} ($\overline{TEST}$) = V_{CC} –0.3 V to V_{CC} V_{in} (RESET) = 0V to 0.3 V	–	–	10	μA	5
Stop Mode Retain Voltage	V_{stop}	V_{CC}		2	–	–	V	

(Note 1) Pull-up MOS current and output buffer current are excluded.
(Note 2) The MCU is in the reset state. The input/output current does not flow.
Test Conditions: MCU state; • Reset state in Operation Mode
Pin state; • RESET, $\overline{TEST}$ ... V_{CC} voltage
• D_0 – D_3, R3 – R9 ... V_{CC} voltage.
• D_4–D_{15}, R0–R2, R_{A0}, R_{A1} ... V_{disp} voltage

(Note 3) The timer/counter operate with the fastest clock and input/output current does not flow.
Test Conditions: MCU state; • Standby Mode
• Input/Output; Reset state
• TIMER-A; ÷2 prescaler divide ratio
• TIMER-B; ÷2 prescaler divide ratio
• SERIAL Interface ; Stop
Pin state; • RESET ... GND voltage
• $\overline{TEST}$ ... V_{CC} voltage
• D_0–D_3, R3–R9 ... V_{CC} voltage
• D_4–D_{15}, R0 – R2, R_{A0}, R_{A1} ... V_{disp} voltage

(Note 4) The timer/counter operate with the slowest clock and input/output current does not flow.
Test Conditions: MCU state; • Standby Mode
• Input/Output; Reset state
• TIMER-A; ÷2048 prescaler divide ratio
• TIMER-B; ÷2048 prescaler divide ratio
• SERIAL Interface ; Stop
Pin state; • RESET ... GND voltage
• $\overline{TEST}$ ... V_{CC} voltage
• D_0–D_3, R3–R9 ... V_{CC} voltage
• D_4–D_{15}, R0–R2, R_{A0}, R_{A1} ... V_{disp} voltage

(Note 5) Pull-down MOS current is excluded.
(Note 6) When f_{osc}=x [MHz], the Current Dissipation in Operation mode and Standby mode are estimated as follows:

$$\text{max. value } (f_{osc}=x\,[\text{MHz}]) = \frac{x}{6} \times \text{max. value } (f_{osc}=6\,[\text{MHz}])$$

(2) Input/output characteristics for standard pin

(V_{CC}=4.5V to 6V, GND=0V, V_{disp}=V_{CC}-40V to V_{CC}, Ta=-20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Input "High" Voltage	V_{IH}	$D_0 - D_3$, R3 – R5, R9		0.7V_{CC}	–	V_{CC}+0.3	V	
Input "Low" Voltage	V_{IL}	$D_0 - D_3$, R3 – R5, R9		–0.3	–	0.22V_{CC}	V	
Output "High" Voltage	V_{OH}	$D_0 - D_3$, R3 – R8	$-I_{OH}$ = 1.0 mA	V_{CC}–1.0	–	–	V	1
		$D_0 - D_3$, R3 – R8	$-I_{OH}$ = 0.01 mA	V_{CC}–0.3	–	–	V	1
Output "Low" Voltage	V_{OL}	$D_0 - D_3$, R3 – R8	I_{OL} = 1.6 mA	–	–	0.4	V	
Input/Output Leakage Current	$\|I_{IL}\|$	$D_0 - D_3$, R3 – R9	V_{in} = 0V to V_{CC}	–	–	1	μA	2
Pull-Up MOS Current	$-I_p$	$D_0 - D_3$, R3 – R9	V_{CC} = 5V, V_{in} = 0V	30	60	120	μA	3

(Note 1) Applied to I/O pins with "CMOS" Output selected by mask option.
(Note 2) Pull-up MOS current and output buffer current are excluded.
(Note 3) Applied to I/O pins "with Pull-up MOS" selected by mask option.

(3) Input/output characteristics for high voltage pin

(V_{CC}=4.5V to 6V, GND=0V, V_{disp}=V_{CC}-40V to V_{CC}, Ta=-20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Input "High" Voltage	V_{IH}	$D_4 - D_{15}$, R1 R2, R_{A0}, R_{A1}		0.7V_{CC}	–	V_{CC}+0.3	V	
Input "Low" Voltage	V_{IL}	$D_4 - D_{15}$, R1 R2, R_{A0}, R_{A1}		V_{CC}–40	–	0.22V_{CC}	V	
Output "High" Voltage	V_{OH}	$D_4 - D_{15}$	$-I_{OH}$=15mA, V_{CC}=5V ± 10%	V_{CC}–3.0	–	–	V	
			$-I_{OH}$=9mA	V_{CC}–2.0	–	–	V	
		R0 – R2	$-I_{OH}$=3mA, V_{CC}=5V ± 10%	V_{CC}–3.0	–	–	V	
			$-I_{OH}$=1.8 mA	V_{CC}–2.0	–	–	V	
Output "Low" Voltage	V_{OL}	$D_4 - D_{15}$ R0 – R2	V_{disp} = V_{CC}–40V	–	–	V_{CC}–37	V	1
		$D_4 - D_{15}$ R0 – R2	150kΩ to V_{CC}–40V	–	–	V_{CC}–37	V	2
Input/Output Leakage Current	$\|I_{IL}\|$	D_4–D_{15} R0–R2 R_{A0}, R_{A1}	V_{in} = V_{CC}–40V to V_{CC}	–	–	20	μA	3
Pull Down MOS Current	I_d	D_4–D_{15} R0–R2 R_{A0}, R_{A1}	V_{disp} = V_{CC}–35V, V_{in} = V_{CC}	125	250	500	μA	4

(Note 1) Applied to I/O pins "with Pull-down MOS" selected by mask option.
(Note 2) Applied to I/O pins "without Pull-down MOS (PMOS Open Drain)" selected by mask option.
(Note 3) Pull-down MOS current and output buffer current are excluded.
(Note 4) Applied to I/O pins "with Pull-down MOS" selected by mask option.

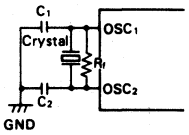
(4) AC characteristics

(V_{CC}=4.5V to 6V, GND=0V, V_{disp}=V_{CC}-40V to V_{CC}, Ta=-20 to +75°C, if not specified.)

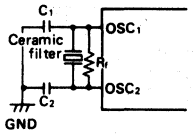
Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Oscillation Frequency	f_{osc}	OSC_1, OSC_2		0.4	6	6.2	MHz	
Instruction Cycle Time	t_{cyc}			1.29	1.33	20	μs	
Oscillator Stabilization Time	t_{RC}	OSC_1, OSC_2		–	–	20	ms	1
External Clock "High" Level Width	t_{CPH}	OSC_1		61	–	–	ns	2
External Clock "Low" Level Width	t_{CPL}	OSC_1		61	–	–	ns	2
External Clock Rise Time	t_{CPr}	OSC_1		–	–	20	ns	2
External Clock Fall Time	t_{CPf}	OSC_1		–	–	20	ns	2
$\overline{INT_0}$ "High" Level Width	t_{I0H}	$\overline{INT_0}$		2	–	–	t_{cyc}	3
$\overline{INT_0}$ "Low" Level Width	t_{I0L}	$\overline{INT_0}$		2	–	–	t_{cyc}	3
$\overline{INT_1}$ "High" Level Width	t_{I1H}	$\overline{INT_1}$		2	–	–	t_{cyc}	3
$\overline{INT_1}$ "Low" Level Width	t_{I1L}	$\overline{INT_1}$		2	–	–	t_{cyc}	3
RESET "High" Level Width	t_{RSTH}	RESET		2	–	–	t_{cyc}	4
Input Capacitance	C_{in}	all pins	f = 1 MHz V_{in} = 0 V	–	–	15	pF	
RESET Fall Time	t_{RSTf}			–	–	20	ms	4

(Note 1) Oscillator stabilization time is the time until the oscillator stabilizes after V_{CC} reaches 4.5V at "Power-on", or after RESET input level goes "High" by resetting to quit the stop mode by MCU reset. At power ON or recovering from stop mode, apply RESET input more than t_{RC} to obtain the necessary time for oscillator stabilization. The circuits used to measure the value are described below. When using crystal or ceramic filter oscillator, please ask a crystal oscillator maker's or ceramic filter maker's advice because oscillator stabilization time depends on the circuit constant and stray capacity.

Crystal oscillator



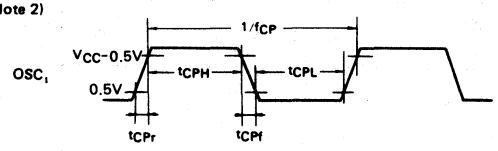
Ceramic filter oscillator



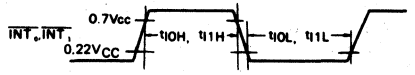
Crystal: 6.0MHz NC-18C (Nihon Denpa Kogyo)
Rf : 1MΩ ±2%
C_1 : 20pF ±20%
C_2 : 20pF ±20%

Ceramic filter : CSA6.00MG (Murata)
Rf : 1MΩ ±2%
C_1 : 30pF ±20%
C_2 : 30pF ±20%

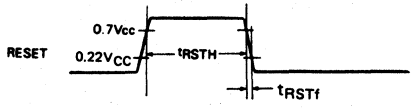
(Note 2)



(Note 3)



(Note 4)



(5) Serial interface timing characteristics

(V_{CC}=4.5V to 6V, GND=0V, V_{disp}=V_{CC}-40V to V_{CC}, Ta=-20 to +75°C, if not specified.)

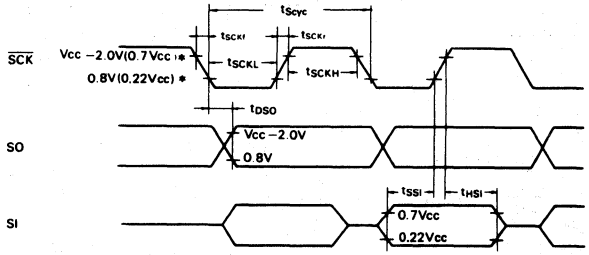
- At Transfer Clock Output

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Transfer Clock Cycle Time	t_{Scyc}	$\overline{SCK}$	(Note 2)	1	–	–	t_{cyc}	1, 2
Transfer Clock "High" Level Width	t_{SCKH}	$\overline{SCK}$	(Note 2)	0.5	–	–	t_{Scyc}	1, 2
Transfer Clock "Low" Level Width	t_{SCKL}	$\overline{SCK}$	(Note 2)	0.5	–	–	t_{Scyc}	1, 2
Transfer Clock Rise Time	t_{SCKr}	$\overline{SCK}$	(Note 2)	–	–	100	ns	1, 2
Transfer Clock Fall Time	t_{SCKf}	$\overline{SCK}$	(Note 2)	–	–	100	ns	1, 2
Serial Output Data Delay Time	t_{DSO}	SO	(Note 2)	–	–	250	ns	1, 2
Serial Input Data Set-up Time	t_{SSI}	SI		300	–	–	ns	1
Serial Input Data Hold Time	t_{HSI}	SI		150	–	–	ns	1

- At Transfer Clock Input

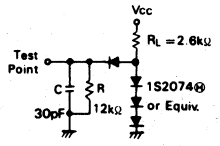
Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Transfer Clock Cycle Time	t_{Scyc}	$\overline{SCK}$		1	–	–	t_{cyc}	1
Transfer Clock "High" Level Width	t_{SCKH}	$\overline{SCK}$		0.5	–	–	t_{Scyc}	1
Transfer Clock "Low" Level Width	t_{SCKL}	$\overline{SCK}$		0.5	–	–	t_{Scyc}	1
Transfer Clock Rise Time	t_{SCKr}	$\overline{SCK}$		–	–	100	ns	1
Transfer Clock Fall Time	t_{SCKf}	$\overline{SCK}$		–	–	100	ns	1
Serial Output Data Delay Time	t_{DSO}	SO	(Note 2)	–	–	250	ns	1, 2
Serial Input Data Set-up Time	t_{SSI}	SI		300	–	–	ns	1
Serial Input Data Hold Time	t_{HSI}	SI		150	–	–	ns	1

(Note 1) Timing Diagram of Serial Interface

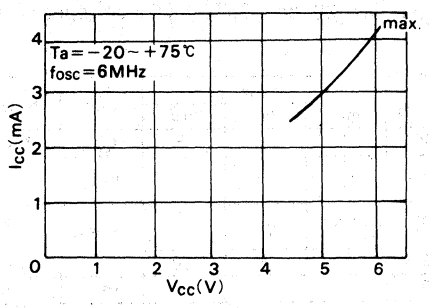


* V_{CC} − 2.0V and 0.8V are the threshold voltage for transfer clock output.
0.7 V_{CC} and 0.22 V_{CC} are the threshold voltage for transfer clock input.

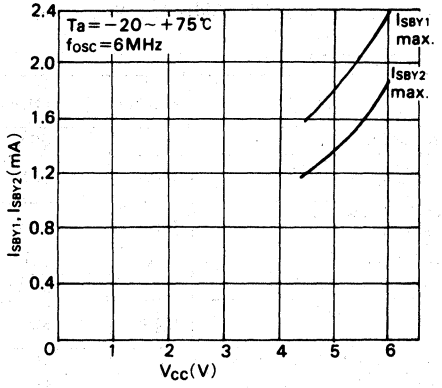
(Note 2) Timing Load Circuit



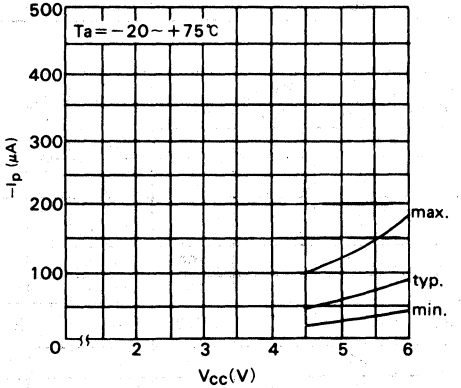
(6) Characteristics Curve (Reference data)



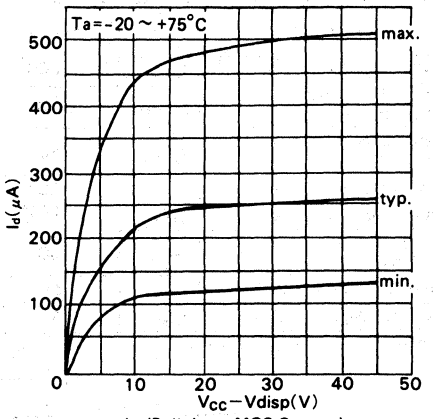
I_{CC} vs. V_{CC} Characteristics
(Crystal, Ceramic Filter Oscillator)



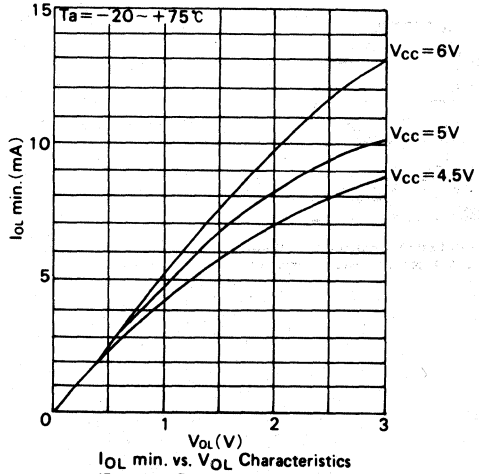
I_{SBY} vs. V_{CC} Characteristics
(Crystal, Ceramic Filter Oscillator)



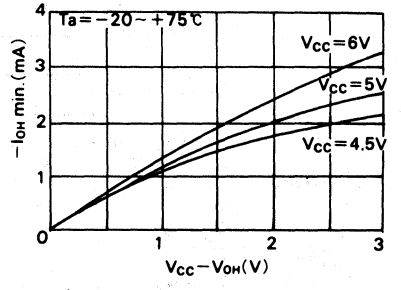
$-I_p$ (Pull-up MOS Current) vs.
V_{CC} Characteristics



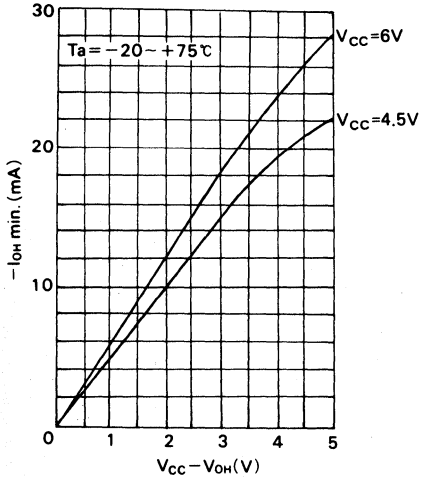
I_d (Pull-down MOS Current) vs.
($V_{CC} - V_{disp}$) Characteristics



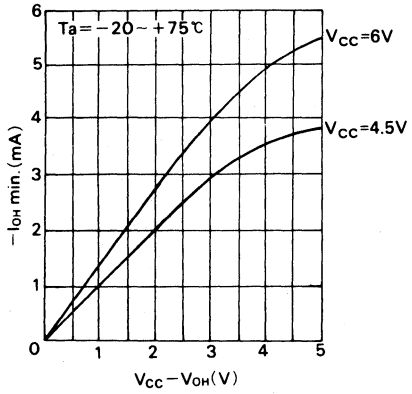
I_{OL} min. vs. V_{OL} Characteristics
(Standard Pin)



$-I_{OH}$ min vs. ($V_{CC} - V_{OH}$) Characteristics
(Standard Pin "CMOS")



$-I_{OH}$ min. vs. (V_{CC} − V_{OH}) Characteristics
(D_4 − D_{15} Pins)



$-I_{OH}$ min. vs. (V_{CC} − V_{OH}) Characteristics
(R0 − R2 Pins)

5.8 HMCS408C Electrical Characteristics

(1) DC characteristics

(V_{CC}=3.5V to 6V, GND=0V, V_{disp}=V_{CC}-40V to V_{CC}, Ta=-20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions	Value min	Value typ	Value max	Unit	Note
Input "High" Voltage	V_{IH}	RESET, SCK, $\overline{INT_0}$, $\overline{INT_1}$		0.8V_{CC}	–	V_{CC}+0.3	V	
		SI		0.7V_{CC}	–	V_{CC}+0.3	V	
		OSC_1		V_{CC}–0.5	–	V_{CC}+0.3	V	
Input "Low" Voltage	V_{IL}	RESET, SCK, $\overline{INT_0}$, $\overline{INT_1}$		–0.3	–	0.2 V_{CC}	V	
		SI		–0.3	–	0.3 V_{CC}	V	
		OSC_1		–0.3	–	0.5	V	
Output "High" Voltage	V_{OH}	$\overline{SCK}$, SO	$-I_{OH}$ = 1.0 mA	V_{CC}–1.0	–	–	V	
			$-I_{OH}$ = 0.5 mA	V_{CC}–0.5	–	–	V	
Output "Low" Voltage	V_{OL}	$\overline{SCK}$, SO	I_{OL} = 1.6 mA	–	–	0.4	V	
Input/Output Leakage Current	$\lvert I_{IL} \rvert$	RESET, $\overline{SCK}$, $\overline{INT_0}$, $\overline{INT_1}$, SI, SO, OSC_1	V_{in} = 0V to V_{CC}	–	–	1	μA	1
Current Dissipation in Active Mode	I_{CC}	V_{CC}	V_{CC} = 5V, fosc = 4MHz Divided by 8	–	–	2.3	mA	2.5
			V_{CC} = 5V, fosc = 2MHz Divided by 4	–	–	2.3	mA	2.5
Current Dissipation in Standby Mode	I_{SBY}	V_{CC}	V_{CC} = 5V, fosc = 4MHz Divided by 8	–	–	1.2	mA	3.5
			V_{CC} = 5V, fosc = 2MHz Divided by 4	–	–	1.2	mA	3.5
Current Dissipation in Stop Mode	I_{stop}	V_{CC}	V_{in} ($\overline{TEST}$) = V_{CC}-0.3V to V_{CC} V_{in} (RESET) = 0V to 0.3V	–	–	10	μA	4
Stop Mode Retain Voltage	V_{stop}	V_{CC}		2	–	–	V	–

(Note 1) Pull-up MOS current and output buffer current are excluded.

(Note 2) The MCU is in the reset state. The input/output current does not flow.

Test Conditions: MCU state; • Reset state in Operation Mode

Pin state; • RESET, $\overline{TEST}$ ··· V_{CC} voltage
• D_0–D_3, R3–R9 ··· V_{CC} voltage
• D_4–D_{15}, R0–R2, R_{A0}, R_{A1} ··· V_{disp} voltage

(Note 3) The timer/counter operate and input/output current does not flow.

Test Conditions: MCU state; • Standby Mode
• Input/Output; Reset state
• SERIAL Interface ; Stop

Pin state; • RESET ···GND voltage
• $\overline{TEST}$ ··· V_{CC} voltage
• D_0 – D_3, R3 – R9 ... V_{CC} voltage
• D_4 – D_{15}, R0 – R2, R_{A0}, R_{A1} ··· V_{disp} voltage

(Note 4) Pull-down MOS current is excluded.

(Note 5) When f_{osc}=x[MHz], the Current Dissipation in Operation mode and Standby mode are estimated as follows:

$$\text{max. value } (f_{osc}=x\text{[MHz]}) = \frac{x}{4} \times \text{max. value } (f_{osc}=4\text{[MHz]})$$

(2) Input/output characteristics for standard pin

(V_{CC}=3.5V to 6V, GND=0V, V_{disp}=V_{CC}-40V to V_{CC}, Ta=-20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Input "High" Voltage	V_{IH}	$D_0 - D_3$, R3 – R5, R9		0.7V_{CC}	–	V_{CC}+0.3	V	
Input "Low" Voltage	V_{IL}	$D_0 - D_3$, R3 – R5, R9		–0.3	–	0.3V_{CC}	V	
Output "High" Voltage	V_{OH}	$D_0 - D_3$, R3 – R8	$-I_{OH}$ = 1.0 mA	V_{CC}–1.0	–	–	V	1
		$D_0 - D_3$, R3 – R8	$-I_{OH}$ = 0.5 mA	V_{CC}–0.5	–	–	V	1
Output "Low" Voltage	V_{OL}	$D_0 - D_3$, R3 – R8	I_{OL} = 1.6 mA	–	–	0.4	V	
Input/Output Leakage Current	$\|I_{IL}\|$	$D_0 - D_3$, R3 – R9	V_{in} = 0V to V_{CC}	–	–	1	μA	2
Pull-Up MOS Current	$-I_p$	$D_0 - D_3$, R3 – R9	V_{CC} = 5V, V_{in} = 0V	30	60	150	μA	3

(Note 1) Applied to I/O pins specified as "CMOS Output".
(Note 2) Pull-up MOS current and output buffer current are excluded.
(Note 3) Applied to I/O pins specified as "with Pull-up MOS".

(3) Input/output characteristics for high voltage pin

(V_{CC}=3.5 to 6V, GND=0V, V_{disp}=V_{CC}-40V to V_{CC}, Ta=-20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Input "High" Voltage	V_{IH}	$D_4 - D_{15}$, R1, R2, R_{A0}, R_{A1}		0.7V_{CC}	–	V_{CC}+0.3	V	
Input "Low" Voltage	V_{IL}	$D_4 - D_{15}$, R1, R2, R_{A0}, R_{A1}		V_{CC}–40	–	0.3 V_{CC}	V	
Output "High" Voltage	V_{OH}	$D_4 - D_{15}$	$-I_{OH}$=15mA, V_{CC}=5V±20% $-I_{OH}$=10mA, V_{CC}=5V±20% $-I_{OH}$=4mA	V_{CC} –3.0 V_{CC} –2.0 V_{CC} –1.0	–	–	V	
		R0 – R2	$-I_{OH}$=3mA, V_{CC}=5V±20% $-I_{OH}$=2mA, V_{CC}=5V±20% $-I_{OH}$=0.8mA	V_{CC} –3.0 V_{CC} –2.0 V_{CC} 1.0	–	–	V	
Output "Low" Voltage	V_{OL}	$D_4 - D_{15}$, R0 – R2	V_{disp} = V_{CC}–40V	–	–	V_{CC}–37	V	1
		$D_4 - D_{15}$ R0 – R2	150kΩ to V_{CC}–40V	–	–	V_{CC}–37	V	2
Input/Output Leakage Current	$\|I_{IL}\|$	$D_4 - D_{15}$ R0 – R2, R_{A0}, R_{A1}	V_{in} = V_{CC}–40V to V_{CC}	–	–	20	μA	3
Pull Down MOS Current	I_d	$D_4 - D_{15}$, R0 – R2, R_{A0}	V_{disp} = V_{CC}–35V, V_{in} = V_{CC}	125	250	600	μA	4

(Note 1) Applied to I/O pins specified as "with Pull-down MOS."
(Note 2) Applied to I/O pins specified as "without Pull-down MOS (PMOS Open Drain)".
(Note 3) Pull-down MOS current and output buffer current are excluded.
(Note 4) Applied to I/O pins specified as "with Pull-down MOS."

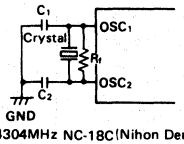
(4) AC characteristics

(V_{CC}=4V to 6V, GND=0V, V_{disp}=V_{CC}-40V to V_{CC}, Ta=-20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Clock Oscillation Frequency	f_{osc}	OSC_1, OSC_2	Divide-by-8	0.4	4	4.5	MHz	
			Divide-by-4	0.2	2	2.25	MHz	
Instruction Cycle Time	t_{cyc}			1.78	2	20	μs	
Oscillation Stabilization Time	t_{RC}	OSC_1, OSC_2		–	–	20	ms	1
External Clock "High" Level Width	t_{CPH}	OSC_1	Divide-by-8	92	–	–	ns	2
			Divide-by-4	203	–	–	ns	2
External Clock "Low" Level Width	t_{CPL}	OSC_1	Divide-by-8	92	–	–	ns	2
			Divide-by-4	203	–	–		2
External Clock Rise Time	t_{cpr}	OSC_1		–	–	20	ns	2
External Clock Fall Time	t_{cpf}	OSC_1		–	–	20	ns	2
$\overline{INT_0}$ "High" Level Width	t_{I0H}	$\overline{INT_0}$		2	–	–	t_{cyc}	3
$\overline{INT_0}$ "Low" Level Width	t_{I0L}	$\overline{INT_0}$		2	–	–	t_{cyc}	3
$\overline{INT_1}$ "High" Level Width	t_{I1H}	$\overline{INT_1}$		2	–	–	t_{cyc}	3
$\overline{INT_1}$ "Low" Level Width	t_{I1L}	$\overline{INT_1}$		2	–	–	t_{cyc}	3
RESET "High" Level Width	t_{RSTH}	RESET		2	–	–	t_{cyc}	4
Input Capacitance	C_{in}	all pins	f=1MHz V_{in} = 0V	–	–	15	pF	
RESET Fall Time	t_{RSTf}			–	–	20	ms	4

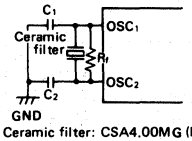
(Note 1) Oscillator stabilization time is the time until the oscillator stabilizes after V_{CC} reaches 3.5V at "Power-on", or after RESET input level goes to "High" by resetting to quit the stop mode by MCU reset on the circuits below. At power ON or recovering from stop mode, apply RESET input more than t_{RC} to obtain the necessary time for oscillator stabilization. When using crystal or ceramic filter oscillator, please ask a crystal oscillator maker's or ceramic filter maker's advice because oscillator stabilization time depends on the circuit constant and stray capacity.

Crystal oscillator



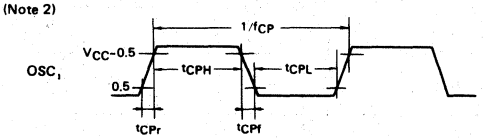
Crystal: 4.194304MHz NC-18C (Nihon Denpa Kogyo)
Rf : 1MΩ ±20%
C_1 : 22pF±20%
C_2 : 22pF±20%

Ceramic filter oscillator

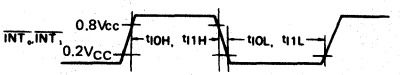


Ceramic filter: CSA4.00MG (Murata)
Rf : 1MHz±20%
C_1 : 30pF±20%
C_2 : 30pF±20%

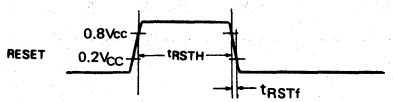
(Note 2)



(Note 3)



(Note 4)



(5) Serial interface timing characteristics

(V_{CC}=3.5V to 6V, GND=0V, V_{disp}=V_{CC}-40V to V_{CC}, Ta=-20 to +75°C, if not specified.)

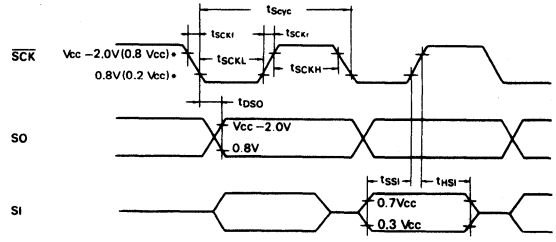
- At Transfer Clock Output

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Transfer Clock Cycle Time	t_{Scyc}	$\overline{SCK}$	(Note 2)	1	–	–	t_{cyc}	1, 2
Transfer Clock "High" Level Width	t_{SCKH}	$\overline{SCK}$	(Note 2)	0.5	–	–	t_{Scyc}	1, 2
Transfer Clock "Low" Level Width	t_{SCKL}	$\overline{SCK}$	(Note 2)	0.5	–	–	t_{Scyc}	1, 2
Transfer Clock Rise Time	t_{SCKr}	$\overline{SCK}$	(Note 2)	–	–	100	ns	1, 2
Transfer Clock Fall Time	t_{SCKf}	$\overline{SCK}$	(Note 2)	–	–	100	ns	1, 2
Serial Output Data Delay Time	t_{DSO}	SO	(Note 2)	–	–	300	ns	1, 2
Serial Input Data Set-up Time	t_{SSI}	SI		500	–	–	ns	1
Serial Input Data Hold Time	t_{HSI}	SI		150	–	–	ns	1

- At Transfer Clock Input

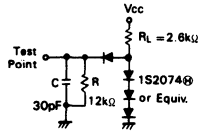
Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Transfer Clock Cycle Time	t_{Scyc}	$\overline{SCK}$		1	–	–	t_{cyc}	1
Transfer Clock "High" Level Width	t_{SCKH}	$\overline{SCK}$		0.5	–	–	t_{Scyc}	1
Transfer Clock "Low" Level Width	t_{SCKL}	$\overline{SCK}$		0.5	–	–	t_{Scyc}	1
Transfer Clock Rise Time	t_{SCKr}	$\overline{SCK}$		–	–	100	ns	1
Transfer Clock Fall Time	t_{SCKf}	$\overline{SCK}$		–	–	100	ns	1
Serial Output Data Delay Time	t_{DSO}	SO	(Note 2)	–	–	300	ns	1, 2
Serial Input Data Set-up Time	t_{SSI}	SI		500	–	–	ns	1
Serial Input Data Hold Time	t_{HSI}	SI		150	–	–	ns	1

(Note 1) Timing Diagram of Serial Interface



* V_{CC} – 2.0V and 0.8V are the threshold voltage for transfer clock output.
0.8 V_{CC} and 0.2 V_{CC} are the threshold voltage for transfer clock input.

(Note 2) Timing Load Circuit



5.9 HMCS408CL Electrical Characteristics

(1) DC characteristics

(V_{CC}=2.5V to 6V, GND=0V, V_{disp}=V_{CC}-40V to V_{CC}, Ta=-20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions	Value min	Value typ	Value max	Unit	Note
Input "High" Voltage	V_{IH}	RESET, SCK, INT_0, INT_1		0.8 V_{CC}	–	V_{CC}+0.3	V	
		SI		0.7V_{CC}	–	V_{CC}+0.3	V	
		OSC_1		V_{CC}–0.5	–	V_{CC}+0.3	V	
Input "Low" Voltage	V_{IL}	RESET, SCK, INT_0, INT_1		–0.3	–	0.2 V_{CC}	V	
		SI		–0.3	–	0.3 V_{CC}	V	
		OSC_1		–0.3	–	0.5	V	
Output "High" Voltage	V_{OH}	SCK, SO	$-I_{OH}$ = 0.3 mA	V_{CC}–0.5	–	–	V	
Output "Low" Voltage	V_{OL}	SCK, SO	I_{OL} = mA	–	–	0.4	V	
Input/Output Leakage Current	$\|I_{IL}\|$	RESET, SCK, INT_0, INT_1, SI, SO, OSC_1	V_{in} = 0V to V_{CC}	–	–	1	μA	1
Current Dissipation in Active Mode	I_{CC}	V_{CC}	V_{CC} = 3V, fosc = 4MHz Divided by 16	–	–	1.1	mA	2.5
			V_{CC} = 3V, fosc = 2MHz Divided by 8	–	–	1.1	mA	2.5
Current Dissipation in Standby Mode	I_{SBY}	V_{CC}	V_{CC} = 3V, fosc = 4MHz Divided by 16	–	–	0.5	mA	3.5
			V_{CC} = 3V, fosc = 2MHz Divided by 8	–	–	0.5	mA	3.5
Current Dissipation in Stop Mode	I_{stop}	V_{CC}	V_{in} (TEST) = V_{CC}–0.3V to V_{CC} V_{in} (RESET) = 0V to 0.3V	–	–	10	μA	4
Stop Mode Retain Voltage	V_{stop}	V_{CC}		2	–	–	V	

(Note 1) **Pull-up MOS current and output buffer current are excluded.**

(Note 2) **The MCU is in the reset state. The input/output current does not flow.**

Test Conditions: MCU state;
- **Reset state in Operation Mode**

Pin state;
- **RESET, TEST ... V_{CC} voltage**
- **D_0–D_3, R3 – R9 ... V_{CC} voltage**
- **D_4–D_{15}, R0 – R2, R_{A0}, R_{A1} ... V_{disp} voltage**

(Note 3) The timer/counter operate and input/output current does not flow.

Test Conditions: MCU state;
- **Standby Mode**
- **Input/Output; Reset state**
- **SERIAL Interface ; Stop**

Pin state;
- **RESET ...GND voltage**
- **TEST ... V_{CC} voltage**
- **D_0–D_3, R3 – R9 ... V_{CC} voltage**
- **D_4–D_{15}, R0 – R2, R_{A0}, R_{A1} ... V_{disp} voltage**

(Note 4) **Pull-down MOS current is excluded.**

(Note 5) **When f_{osc}=x [MHz], the Current Dissipation in Operation mode and Standby mode are estimated as follows:**

$$\text{max. value } (f_{osc}=x\,[\text{MHz}]) = \frac{x}{4} \times \text{max. value } (f_{osc}=4\,[\text{MHz}])$$

(2) Input/output characteristics for standard pin

(V_{CC}=2.5V to 6V, GND=0V, V_{disp}=V_{CC}-40V to V_{CC}, Ta=-20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Input "High" Voltage	V_{IH}	$D_0 - D_3$, R3 – R5, R9		0.7V_{CC}	–	V_{CC}+0.3	V	
Input "Low" Voltage	V_{IL}	$D_0 - D_3$, R3 – R5, R9		–0.3	–	0.3 V_{CC}	V	
Output "High" Voltage	V_{OH}	$D_0 - D_3$, R3 – R8	$-I_{OH}$ = 1.0 mA	V_{CC}–1.0	–	–	V	1
		$D_0 - D_3$, R3 – R8	$-I_{OH}$ = 0.5 mA	V_{CC}–0.5	–	–	V	1
Output "Low" Voltage	V_{OL}	$D_0 - D_3$, R3 – R8	I_{OL} = 1.6 mA	–	–	0.4	V	
Input/Output Leakage Current	$\|I_{IL}\|$	$D_0 - D_3$, R3 – R9	V_{in} = 0V to V_{CC}	–	–	1	μA	2
Pull-Up MOS Current	$-I_p$	$D_0 - D_3$, R3 – R9	V_{CC} = 3V, V_{in} = 0V	3	15	50	μA	3
			V_{CC} = 5V, V_{in} = 0V	30	60	150		

(Note 1) Applied to I/O pins specified as "CMOS Output".
(Note 2) Pull-up MOS current and output buffer current are excluded.
(Note 3) Applied to I/O pins specified as "with Pull-up MOS".

(3) Input/output characteristics for high voltage pin

(V_{CC}=2.5V to 6V, GND=0V, V_{disp}=V_{CC}-40V to V_{CC}, Ta=-20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Input "High" Voltage	V_{IH}	$D_4 - D_{15}$, R1, R2, R_{A0}, R_{A1}		0.7V_{CC}	–	V_{CC}+0.3	V	
Input "Low" Voltage	V_{IL}	$D_4 - D_{15}$, R1, R2, R_{A0}, R_{A1}		V_{CC}–40	–	0.3 V_{CC}	V	
Output "High" Voltage	V_{OH}	$D_4 - D_{15}$	$-I_{OH}$ = 15mA, V_{CC} = 5V ± 20% $-I_{OH}$ = 10mA, V_{CC} = 5V ± 20% $-I_{OH}$ = 4mA	V_{CC} –3.0 V_{CC}–2.0 V_{CC} –1.0	–	–	V	
		R0 – R2	$-I_{OH}$ = 3mA, V_{CC} = 5V $-I_{OH}$ = 2mA, V_{CC} = 5V ± 20% $-I_{OH}$ = 0.8 mA	V_{CC} –3.0 V_{CC} –2.0 V_{CC} –1.0	–	–	V	
Output "Low" Voltage	V_{OL}	$D_4 - D_{15}$ R0 – R2	V_{disp} = V_{CC}–40V	–	–	V_{CC}–37	V	1
		$D_4 - D_{15}$ R0 – R2	150kΩ to V_{CC}–40V	–	–	V_{CC}–37	V	2
Input/Output Leakage Current	$\|I_{IL}\|$	$D_4 - D_{15}$ R0 – R2, R_{A0}, R_{A1}	V_{in} = V_{CC}–40V to V_{CC}	–	–	20	μA	3
Pull Down MOS Current	I_d	$D_4 - D_{15}$ R0 – R2, R_{A0}	V_{disp} = V_{CC}–35V V_{in} = V_{CC}	125	250	600	μA	4

(Note 1) Applied to I/O pins specified as "with Pull-down MOS".
(Note 2) Applied to I/O pins specified as "without Pull-down MOS (PMOS Open Drain)".
(Note 3) Pull-down MOS current and output buffer current are excluded.
(Note 4) Applied to I/O pins specified as "with Pull-down MOS".

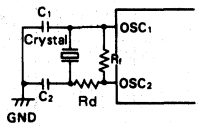
(4) AC characteristics

(V_{CC}=2.5V to 6V, GND=0V, V_{disp}=V_{CC}-40V to V_{CC}, Ta=20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Clock Oscillation Frequency	f_{osc}	OSC_1, OSC_2	divide-by-16	0.8	4	4.5	MHz	
			divide-by-8	0.4	2	2.25	MHz	
Instruction Cycle Time	t_{cyc}			3.55	4	20	μs	
Oscillation Stabilization Time	t_{RC}	OSC_1, OSC_2		–	–	60	ms	1
External Clock "High" Level Width	t_{CPH}	OSC_1	divide-by-16	92	–	–	ns	2
			divide-by-8	203	–	–	ns	2
External Clock "Low" Level Width	t_{CPL}	OSC_1	divide-by-16	92	–	–	ns	2
			divide-by-8	203	–	–	ns	2
External Clock Rise Time	t_{cpr}	OSC_1		–	–	20	ns	2
External Clock Fall Time	t_{cpf}	OSC_1		–	–	20	ns	2
$\overline{INT_0}$ "High" Level Width	t_{I0H}	$\overline{INT_0}$		2	–	–	t_{cyc}	3
$\overline{INT_0}$ "Low" Level Width	t_{I0L}	$\overline{INT_0}$		2	–	–	t_{cyc}	3
$\overline{INT_1}$ "High" Level Width	t_{I1H}	$\overline{INT_1}$		2	–	–	t_{cyc}	3
$\overline{INT_1}$ "Low" Level Width	t_{I1L}	$\overline{INT_1}$		2	–	–	t_{cyc}	3
RESET "High" Level Width	t_{RSTH}	RESET		2	–	–	t_{cyc}	4
Input Capacitance	C_{in}	all pins	f=1MHz V_{in} = 0V	–	–	15	pF	
RESET Fall Time	t_{RSTf}			–	–	15	ms	4

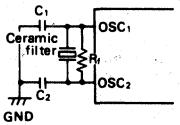
(Note 1) Oscillator stabilization time is the time until the oscillator stabilizes after V_{CC} reaches 2.5V at "Power-on", or after RESET input level goes to "High" by resetting to quit the stop mode by MCU reset on the circuits below. At power ON or recovering from stop mode, apply RESET input more than t_{RC} to obtain the necessary time for oscillator stabilization. When using crystal or ceramic filter oscillator, please ask a crystal oscillator maker's or ceramic filter maker's advice because oscillator stabilization time depends on the circuit constant and stray capacity.

Crystal oscillator



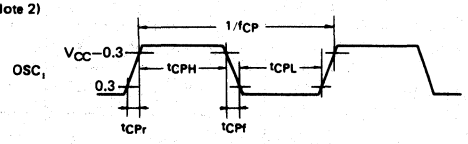
Crystal: 2.097152 MHz 0.5=MGQ308C
Rf : 1MΩ±20%, Rd = 2.2kΩ±20%
C_1 : 10 pF±20%
C_2 : 10 pF±20%

Ceramic filter oscillator

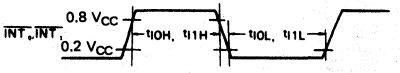


Ceramic filter: CSA2.000MK (Murata)
Rf : 1MHz±20%
C_1 : 30pF±20%
C_2 : 30pF±20%

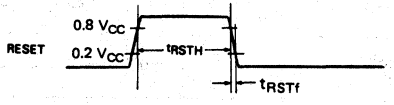
(Note 2)



(Note 3)



(Note 4)



(5) Serial interface timing characteristics

(V_{CC}=2.5V to 6V, GND=0V, V_{disp}=V_{CC}-40V to V_{CC}, Ta=-20 to +75°C, if not specified.)

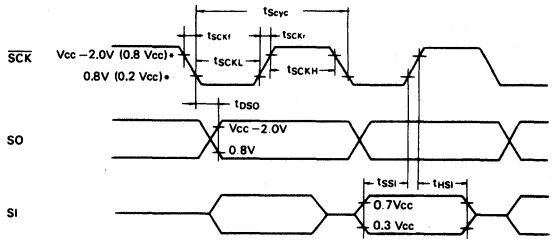
- At Transfer Clock Output

Item	Symbol	Pin Name	Test Conditions	Value min	Value typ	Value max	Unit	Note
Transfer Clock Cycle Time	t_{Scyc}	$\overline{SCK}$	(Note 2)	1	–	–	t_{cyc}	1, 2
Transfer Clock "High" Level Width	t_{SCKH}	$\overline{SCK}$	(Note 2)	0.5	–	–	t_{Scyc}	1, 2
Transfer Clock "Low" Level Width	t_{SCKL}	$\overline{SCK}$	(Note 2)	0.5	–	–	t_{Scyc}	1, 2
Transfer Clock Rise Time	t_{SCKr}	$\overline{SCK}$	(Note 2)	–	–	300	ns	1, 2
Transfer Clock Fall Time	t_{SCKf}	$\overline{SCK}$	(Note 2)	–	–	300	ns	1, 2
Serial Output Data Delay Time	t_{DSO}	SO	(Note 2)	–	–	600	ns	1, 2
Serial Input Data Set-up Time	t_{SSI}	SI		1000	–	–	ns	1
Serial Input Data Hold Time	t_{HSI}	SI		500	–	–	ns	1

- At Transfer Clock Input

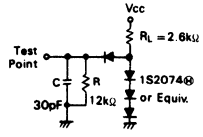
Item	Symbol	Pin Name	Test Conditions	Value min	Value typ	Value max	Unit	Note
Transfer Clock Cycle Time	t_{Scyc}	$\overline{SCK}$		1	–	–	t_{cyc}	1
Transfer Clock "High" Level Width	t_{SCKH}	$\overline{SCK}$		0.5	–	–	t_{Scyc}	1
Transfer Clock "Low" Level Width	t_{SCKL}	$\overline{SCK}$		0.5	–	–	t_{Scyc}	1
Transfer Clock Rise Time	t_{SCKr}	$\overline{SCK}$		–	–	300	ns	1
Transfer Clock Fall Time	t_{SCKf}	$\overline{SCK}$		–	–	300	ns	1
Serial Output Data Delay Time	t_{DSO}	SO	(Note 2)	–	–	600	ns	1, 2
Serial Input Data Set-up Time	t_{SSI}	SI		1000	–	–	ns	1
Serial Input Data Hold Time	t_{HSI}	SI		500	–	–	ns	1

(Note 1) Timing Diagram of Serial Interface



* V_{CC} –2.0V and 0.8V are the threshold voltage for transfer clock output.
0.8 V_{CC} and 0.2 V_{CC} are the threshold voltage for transfer clock input.

(Note 2) Timing Load Circuit



5.10 HMCS408AC Electrical Characteristics

(1) DC characteristics

(V_{CC}=4.5V to 6V, GND=0V, V_{disp}=V_{CC}-40V to V_{CC}, Ta=-20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions	Value min	Value typ	Value max	Unit	Note
Input "High" Voltage	V_{IH}	$\overline{RESET}$, $\overline{SCK}$, $\overline{INT_0}$, $\overline{INT_1}$		0.8V_{CC}	–	V_{CC}+0.3	V	
		SI		0.7V_{CC}	–	V_{CC}+0.3	V	
		OSC_1		V_{CC}– 0.5	–	V_{CC}+0.3	V	
Input "Low" Voltage	V_{IL}	$\overline{RESET}$, $\overline{SCK}$, $\overline{INT_0}$, $\overline{INT_1}$		–0.3	–	0.2 V_{CC}	V	
		SI		–0.3	–	0.3 V_{CC}	V	
		OSC_1		–0.3	–	0.5	V	
Output "High" Voltage	V_{OH}	$\overline{SCK}$, SO	$-I_{OH}$ = 1.0 mA	V_{CC}–1.0	–	–	V	
			$-I_{OH}$ = 0.5 mA	V_{CC}–0.5	–	–	V	
Output "Low" Voltage	V_{OL}	$\overline{SCK}$, SO	I_{OL} = 1.6 mA	–	–	0.4	V	
Input/Output Leakage Current	$\|I_{IL}\|$	$\overline{RESET}$, $\overline{SCK}$, $\overline{INT_0}$, $\overline{INT_1}$, SI, SO, OSC_1	V_{in} = 0V to V_{CC}	–	–	1	μA	1
Current Dissipation in Active Mode	I_{CC}	V_{CC}	V_{CC} = 5V, f_{osc} = 8MHz Divide-by-4	–	–	4.5	mA	2.5
Current Dissipation in Standby Mode	I_{SBY}	V_{CC}	V_{CC} = 5V f_{osc} = 8 MHz Divide-by-4	–	–	1.7	mA	3.5
Current Dissipation in Stop Mode	I_{stop}	V_{CC}	V_{in} (TEST) = V_{CC}–0.3V to V_{CC} V_{in} (RESET) = 0V to 0.3V	–	–	10	μA	4
Stop Mode Retain Voltage	V_{stop}	V_{CC}		2	–	–	V	

(Note 1) Pull-up MOS current and output buffer current are excluded.

(Note 2) The MCU is in the reset state. The input/output current does not flow.

Test Conditions: MCU state;
- Reset state in Operation Mode

Pin state;
- RESET, $\overline{TEST}$ ... V_{CC} voltage
- D_0 – D_3, R3 – R9 ... V_{CC} voltage
- D_4 – D_{15}, R0 – R2, R_{A0}, R_{A1} ... V_{disp} voltage

(Note 3) The timer/counter and input/output current does not flow.

Test Conditions: MCU state;
- Standby Mode
- Input/Output; Reset state
- SERIAL Interface ; Stop

Pin state;
- RESET ... GND voltage
- $\overline{TEST}$ ... V_{CC} voltage
- D_0 – D_3, R3 – R9 ... V_{CC} voltage
- D_4 – D_{15}, R0 – R2, R_{A0}, R_{A1} ... V_{disp} voltage

(Note 4) Pull-down MOS current is excluded.

(Note 5) When f_{osc}=x[MHz], the Current Dissipation in Operation mode and Standby mode are estimated as follows:

$$\text{max. value } (f_{osc}=x\text{[MHz]}) = \frac{x}{4} \times \text{max. value } (f_{osc}=4\text{[MHz]})$$

(2) Input/output characteristics for standard pin

(V_{CC}=4.5V to 6V, GND=0V, V_{disp}=V_{CC}-40V to V_{CC}, Ta=-20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Input "High" Voltage	V_{IH}	$D_0 - D_3$, R3 – R5, R9		0.7V_{CC}	–	V_{CC}+0.3	V	
Input "Low" Voltage	V_{IL}	$D_0 - D_3$, R3 – R5, R9		–0.3	–	0.3 V_{CC}	V	
Output "High" Voltage	V_{OH}	$D_0 - D_3$, R3 – R8	$-I_{OH}$ = 1.0 mA	V_{CC}–1.0	–	–	V	1
		$D_0 - D_3$, R3 – R8	$-I_{OH}$ = 0.5 mA	V_{CC}–0.5	–	–	V	1
Output "Low" Voltage	V_{OL}	$D_0 - D_3$, R3 – R8	I_{OL} = 1.6 mA	–	–	0.4	V	
Input/Output Leakage Current	$\|I_{IL}\|$	$D_0 - D_3$, R3 – R9	V_{in} = 0V to V_{CC}	–	–	1	μA	2
Pull-Up MOS Current	$-I_p$	$D_0 - D_3$, R3 – R9	V_{CC} = 5V, V_{in} = 0V	30	60	150	μA	3

(Note 1) Applied to I/O pins specified as "CMOS Output".
(Note 2) Pull-up MOS current and output buffer current are excluded.
(Note 3) Applied to I/O pins specified as "with Pull-up MOS".

(3) Input/output characteristics for high voltage pin

(V_{CC}=4.5 to 6V, GND=0V, V_{disp}=V_{CC}-40V to V_{CC}, Ta=-20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Input "High" Voltage	V_{IH}	$D_4 - D_{15}$, R1, R2, R_{A0}, R_{A1}		0.7V_{CC}	–	V_{CC}+0.3	V	
Input "Low" Voltage	V_{IL}	$D_4 - D_{15}$, R1, R2, R_{A0}, R_{A1}		V_{CC}–40	–	V_{CC}–0.3	V	
Output "High" Voltage	V_{OH}	$D_4 - D_{15}$	$-I_{OH}$=15mA, V_{CC}=5V±20% $-I_{OH}$=10mA, V_{CC}=5V±20% $-I_{OH}$=4mA	V_{CC} –3.0 V_{CC} –2.0 V_{CC} –1.0	–	–	V	
		R0 – R2	$-I_{OH}$=3mA, V_{CC}=5V±20% $-I_{OH}$=2mA, V_{CC}=5V±20% $-I_{OH}$=0.8mA	V_{CC} –3.0 V_{CC} –2.0 V_{CC} –1.0	–	–	V	
Output "Low" Voltage	V_{OL}	$D_4 - D_{15}$, R0 – R2	V_{disp} = V_{CC}–40V	–	–	V_{CC}–37	V	1
		$D_4 - D_{15}$, R0 – R2	150kΩ to V_{CC}–40V	–	–	V_{CC}–37	V	2
Input/Output Leakage Current	$\|I_{IL}\|$	$D_4 - D_{15}$ R0 – R2, R_{A0}, R_{A1}	V_{in} = V_{CC}–40V to V_{CC}	–	–	20	μA	3
Pull Down MOS Current	I_d	$D_4 - D_{15}$, R0 – R2, R_{A0}	V_{disp} = V_{CC}–35V, V_{in} = V_{CC}	125	250	600	μA	4

(Note 1) Applied to I/O pins specified as "with Pull-down MOS".
(Note 2) Applied to I/O pins specified as "without Pull-down MOS (PMOS Open Drain)".
(Note 3) Pull-down MOS current and output buffer current are excluded.
(Note 4) Applied to I/O pins specified as "with Pull-down MOS".

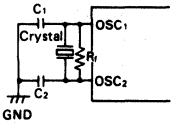
(4) AC characteristics

(V_{CC}=4.5V to 6V, GND=0V, V_{disp}=V_{CC}-40V to V_{CC}, Ta=-20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Clock Oscilation Frequency	f_{osc}	OSC_1, OSC_2	divide-by-4	0.4	4	4.5	MHz	
Instruction Cycle Time	t_{cyc}		divide-by-4	0.89	1	20	μs	
Oscillation Stabilization Time	t_{RC}	OSC_1, OSC_2		–	–	20	ms	1
External Clock "High" Level Width	t_{CPH}	OSC_1		92	–	–	ns	2
External Clock "Low" Level Width	t_{CPL}	OSC_1		92	–	–	ns	2
External Clock Rise Time	t_{cpr}	OSC_1		–	–	20	ns	2
External Clock Fall Time	t_{cpf}	OSC_1		–	–	20	ns	2
$\overline{INT_0}$ "High" Level Width	t_{I0H}	$\overline{INT_0}$		2	–	–	t_{cyc}	3
$\overline{INT_0}$ "Low" Level Width	t_{I0L}	$\overline{INT_0}$		2	–	–	t_{cyc}	3
$\overline{INT_i}$ "High" Level Width	t_{I1H}	$\overline{INT_i}$		2	–	–	t_{cyc}	3
$\overline{INT_i}$ "Low" Level Width	t_{I1L}	$\overline{INT_i}$		2	–	–	t_{cyc}	3
RESET "High" Level Width	t_{RSTH}	RESET		2	–	–	t_{cyc}	4
Input Capacitance	C_{in}	all pins	f=1MHz V_{in} = 0V	–	–	15	pF	
RESET Fall Time	t_{RSTf}			–	–	20	ms	4

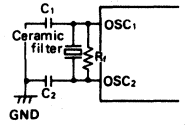
(Note 1) Oscillator stabilization time is the time until the oscillator stabilizes after V_{CC} reaches 4.5V at "Power-on", or after RESET input level goes to "High" by resetting to quit the stop mode by MCU reset on the circuits below. At power ON or recovering from stop mode, apply RESET input more than t_{RC} to obtain the necessary time for oscillator stabilization. When using crystal or ceramic filter oscillator, please ask a crystal oscillator maker's or ceramic filter maker's advice because oscillator stabilization time depends on the circuit constant and stray capacity.

Crystal oscillator



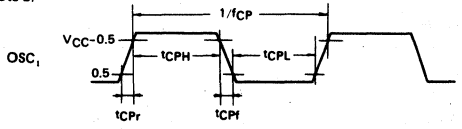
Crystal: 4.194304MHz NC-18C(Nihon Denpa Kogyo)
Rf : 1MΩ ±20%
C_1 : 22pF±20%
C_2 : 22pF±20%

Ceramic filter oscillator

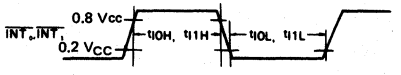


Ceramic filter: CSA4.00MG (Murata)
Rf : 1MHz±20%
C_1 : 30pF±20%
C_2 : 30pF±20%

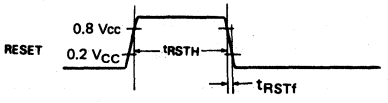
(Note 2)



(Note 3)



(Note 4)



(5) Serial interface timing characteristics

(V_{CC}=4.5V to 6V, GND=0V, V_{disp}=V_{CC}-40V to V_{CC}, Ta=-20 to +75°C, if not specified.)

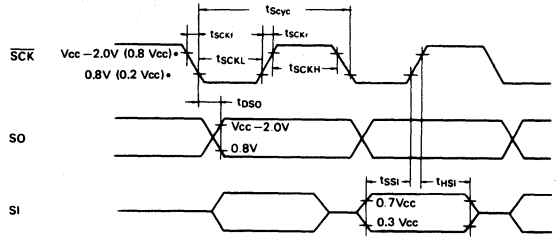
• At Transfer Clock Output

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Transfer Clock Cycle Time	t_{Scyc}	$\overline{SCK}$	(Note 2)	1	–	–	t_{cyc}	1, 2
Transfer Clock "High" Level Width	t_{SCKH}	$\overline{SCK}$	(Note 2)	0.5	–	–	t_{Scyc}	1, 2
Transfer Clock "Low" Level Width	t_{SCKL}	$\overline{SCK}$	(Note 2)	0.5	–	–	t_{Scyc}	1, 2
Transfer Clock Rise Time	t_{SCKr}	$\overline{SCK}$	(Note 2)	–	–	100	ns	1, 2
Transfer Clock Fall Time	t_{SCKf}	$\overline{SCK}$	(Note 2)	–	–	100	ns	1, 2
Serial Output Data Delay Time	t_{DSO}	SO	(Note 2)	–	–	250	ns	1, 2
Serial Input Data Set-up Time	t_{SSI}	SI		300	–	–	ns	1
Serial Input Data Hold Time	t_{HSI}	SI		150	–	–	ns	1

• At Transfer Clock Input

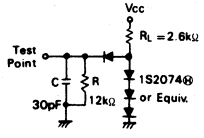
Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Transfer Clock Cycle Time	t_{Scyc}	$\overline{SCK}$		1	–	–	t_{cyc}	1
Transfer Clock "High" Level Width	t_{SCKH}	$\overline{SCK}$		0.5	–	–	t_{Scyc}	1
Transfer Clock "Low" Level Width	t_{SCKL}	$\overline{SCK}$		0.5	–	–	t_{Scyc}	1
Transfer Clock Rise Time	t_{SCKr}	$\overline{SCK}$		–	–	100	ns	1
Transfer Clock Fall Time	t_{SCKf}	$\overline{SCK}$		–	–	100	ns	1
Serial Output Data Delay Time	t_{DSO}	SO	(Note 2)	–	–	250	ns	1, 2
Serial Input Data Set-up Time	t_{SSI}	SI		300	–	–	ns	1
Serial Input Data Hold Time	t_{HSI}	SI		150	–	–	ns	1

(Note 1) Timing Diagram of Serial Interface

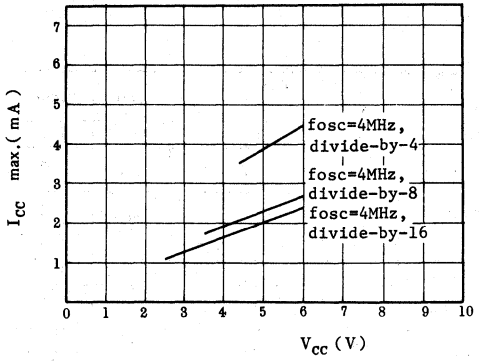


* V_{CC} – 2.0V and 0.8V are the threshold voltage for transfer clock output.
0.8 V_{CC} and 0.2 V_{CC} are the threshold voltage for transfer clock input.

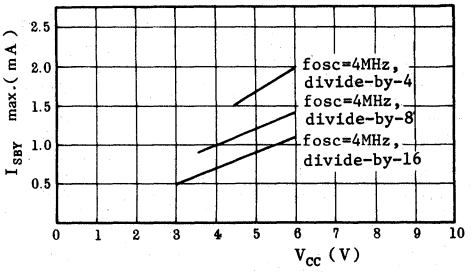
(Note 2) Timing Load Circuit



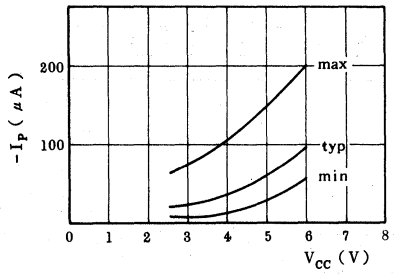
(6) Characteristics Curve (Reference Data)



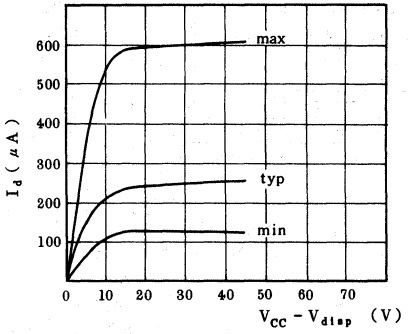
I_{CC} vs V_{CC} Characteristics (Crystal, Ceramic Filter Oscillator Option)



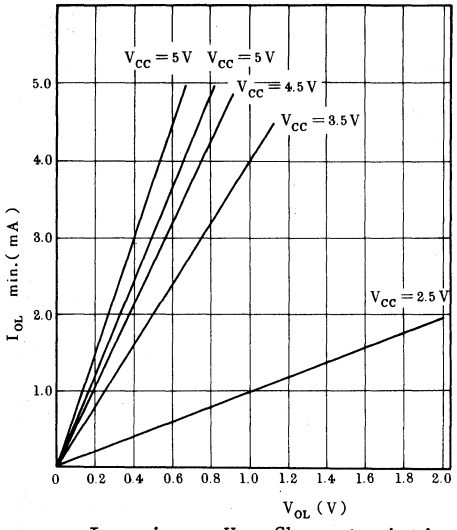
I_{SBY} vs V_{CC} Characteristics (Crystal, Ceramic Filter Oscillator Option)



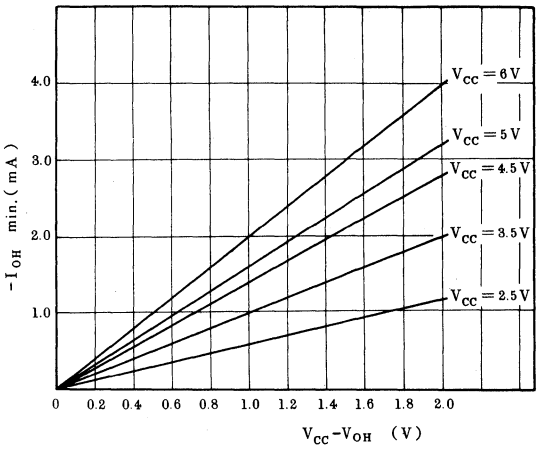
$-I_P$ vs V_{CC} Characteristics



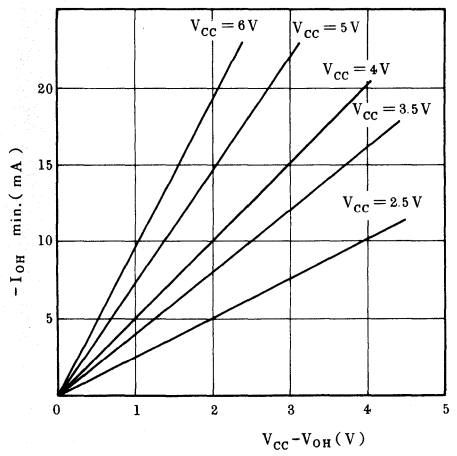
I_d vs (V_{CC} - V_{disp}) Characteristics



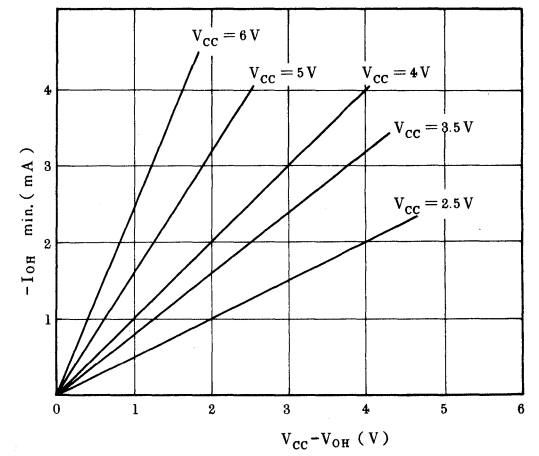
I_{OL} min vs V_{OL} Characteristics
(Standard Pin)



$-I_{OH}$ min vs (V_{CC} - V_{OH}) Characteristics
(Standard Pin)



$-I_{OH}$ min vs (V_{CC} - V_{OH}) Characteristics
(D_4 - D_5 Pins)



$-I_{OH}$ min vs (V_{CC} - V_{OH}) Characteristics
(R_0 - R_2 Pins)

5.11 HMCS412/414 Absolute Maximum Ratings

Item	Symbol	Value		Unit	Note
Supply Voltage	V_{CC}	−0.3 to +7.0		V	
Terminal Voltage	V_T	−0.3 to V_{CC} +0.3		V	3
		V_{CC} −45 to V_{CC} +0.3		V	4
Total Allowance of Input Currents	ΣI_O	25		mA	5
Maximum Input Current	I_O	15		mA	7, 8
Maximum Output Current	$-I_O$	4	2	mA	9, 10, 13
		6	3	mA	9, 11, 13
		30	15	mA	9, 12, 13
Total Allowance of Output Currents	$-\Sigma I_O$	85	100	mA	6, 13
Operating Temperature	T_{opr}	−20 to +75		°C	
Storage Temperature	T_{stg}	−55 to +125		°C	

(Note 1) Permanent damage may occur if "Absolute Maximum Ratings" are exceeded. Normal operation should be under the conditions of "Electrical Characteristics". If these conditions are exceeded, it may cause the malfunction and affect the reliability of LSI.
(Note 2) All voltage are with respect to GND.
(Note 3) Applied to standard pins.
(Note 4) Applied to high voltage pins.
(Note 5) Total allowance of input current is the total sum of input current which flow in from all I/O pins to GND simultaneously.
(Note 6) Total allowance of output current is the total sum of the output current which flow out from V_{CC} to all I/O pins simultaneously.
(Note 7) Maximum input current is the maximum amount of input current from each I/O pin to GND.
(Note 8) Applied to $D_0 - D_3$ and R3 − R4.
(Note 9) Maximum output current is the maximum amount of output current from V_{CC} to each I/O pin.
(Note 10) Applied to $D_0 - D_3$ and R3 − R4.
(Note 11) Applied to R0 − R2.
(Note 12) Applied to $D_4 - D_{15}$.
(Note 13) $-\Sigma I_O$ = 100mA if $-I_O$ is equal to or less than 2mA, 3mA, or 15mA.

5.12 HMCS412C Electrical Characteristics

(1) DC characteristics

(V_{CC} = 3.5Vto 6V, GND = 0V, V_{disp} = V_{CC}−40V to V_{CC}, Ta = −20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Input "High" Voltage	V_{IH}	RESET, $R_{32}/\overline{INT_0}$, $R_{33}/\overline{INT_1}$		0.8V_{CC}	–	V_{CC}+0.3	V	
		OSC_1		V_{CC}−0.5	–	V_{CC}+0.3	V	
Input "Low" Voltage	V_{IL}	RESET, $R_{32}/\overline{INT_0}$, $R_{33}/\overline{INT_1}$		−0.3	–	0.2V_{CC}	V	
		OSC_1		−0.3	–	0.5	V	
Input/Output Leakage Current	$\lvert I_{IL} \rvert$	RESET, $R_{32}/\overline{INT_0}$, $R_{33}/\overline{INT_1}$, OSC_1	V_{in} = 0V to V_{CC}	–	–	1	μA	1
Current Dissipation in Active Mode	I_{CC}	V_{CC}	V_{CC} = 5V, f_{osc} = 4MHz divide-by-8	–	–	1.8	mA	2.5
			V_{CC} = 5V, f_{osc} = 2MHz divide-by-4	–	–	1.8	mA	2.5
Current Dissipation in Standby Mode	I_{SBY}	V_{CC}	V_{CC} = 5V, f_{osc} = 4MHz divide-by-8	–	–	1.0	mA	3.5
			V_{CC} = 5V, f_{osc} = 2MHz divide-by-4	–	–	1.0	mA	3.5
Current Dissipation in Stop Mode	I_{stop}	V_{CC}	V_{in} ($\overline{TEST}$) = V_{CC}−0.3V to V_{CC} V_{CC}, V_{in} (RESET) = 0 to 0.3V	–	–	10	μA	4
Stop Mode Retain Voltage	V_{stop}	V_{CC}		2	–	–	V	

(Note 1) Pull-up MOS current and output buffer current are excluded.

(Note 2) The MCU is in the reset state. The input/output current does not flow.

Test Conditions: MCU state; • Reset state in Operation Mode

Pin state; • RESET, $\overline{TEST}$ ... V_{CC} voltage
• $D_0 - D_3$, R3 − R4 ... V_{CC} voltage
• $D_4 - D_{14}$, R0 − R2, R_{A0}, R_{A1} ... V_{disp} voltage

(Note 3) The timer/counter operate with the fastest clock and input/output current does not flow.

Test Conditions: MCU state; • Standby Mode
• Input/Output; Reset state

Pin state; • $\overline{RESET}$ ... GND voltage
• $\overline{TEST}$ ... V_{CC} voltage
• $D_0 - D_3$, R3 − R4 ... V_{CC} voltage
• $D_4 - D_{14}$, R0 − R2, R_{A0}, R_{A1} ... V_{disp} voltage

(Note 4) Pull-down MOS current is excluded.

(Note 5) When $f_{osc} = x$[MHz], the Current Dissipation in Operation mode and Standby mode are estimated as follows:

$$\text{max. value } (f_{osc} = x\text{[MHz]}) = \frac{x}{4} \times \text{max. value } (f_{osc} = 4\text{[MHz]})$$

(2) Input/output characteristics for standard pin

(V_{CC} = 3.5V to 6V, GND = 0V, V_{disp} = V_{CC}–40V to V_{CC}, Ta = –20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Input "High" Voltage	V_{IH}	$D_0 - D_3$, R_{30}, R_{31}, R4		0.7V_{CC}	–	V_{CC}+0.3	V	
Input "Low" Voltage	V_{IL}	$D_0 - D_3$ R_{31}, R4		–0.3	–	0.3V_{CC}	V	
Output "High" Voltage	V_{OH}	$D_0 - D_3$, R_{31}, R4	$-I_{OH}$ = 1.0mA	V_{CC}–1.0	–	–	V	1
		$D_0 - D_3$, R_{31}, R4	$-I_{OH}$ = 0.5mA	V_{CC}–0.5	–	–	V	1
Output "Low" Voltage	V_{OL}	$D_0 - D_3$, R_{31}, R4	I_{OL} = 1.6mA	–	–	0.4	V	
Input/Output Leakage Current	$\|I_{IL}\|$	$D_0 - D_3$, R_{31}, R4	V_{in} = 0V to V_{CC}	–	–	1	μA	2
Pull-Up MOS Current	$-I_p$	$D_0 - D_3$, R_{31}, R4	V_{CC} = 5V V_{in} = 0V	30	60	150	μA	3

(Note 1) Applied to I/O pins with "CMOS" Output selected by mask option.
(Note 2) Pull-up MOS current and output buffer current are excluded.
(Note 3) Applied to I/O pins with "with Pull-up MOS" selected by mask option.

(3) Input/output characteristics for high voltage pin

(V_{CC} = 3.5V to 6V, GND = 0V, V_{disp} = V_{CC}–40V to V_{CC}, Ta = –20 to +75°C, if not specified.)

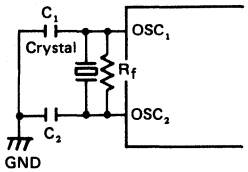
Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Input "High" Voltage	V_{IH}	$D_4 - D_{14}$, R1 R2, R_{A1}		0.7V_{CC}	–	V_{CC}+0.3	V	
Input "Low" Voltage	V_{IL}	$D_4 - D_{14}$, R1 R2, R_{A1}		V_{CC}–40	–	0.3V_{CC}	V	
Output "High" Voltage	V_{OH}	$D_4 - D_{14}$	$-I_{OH}$ = 15mA, V_{CC} = 5V ± 20%	V_{CC}–3.0	–	–	V	
			$-I_{OH}$ = 10mA, V_{CC} = 5V ± 20%	V_{CC}–2.0	–	–	V	
			$-I_{OH}$ = 4mA	V_{CC}–1.0	–	–	V	
		R0 – R2	$-I_{OH}$ = 3mA, V_{CC} = 5V ± 20%	V_{CC}–3.0	–	–	V	
			$-I_{OH}$ = 2mA, V_{CC} = 5V ± 20%	V_{CC}–2.0	–	–	V	
			$-I_{OH}$ = 0.8mA	V_{CC}–1.0	–	–	V	
Output "Low" Voltage	V_{OL}	$D_4 - D_{14}$ R0 – R2	V_{disp} = V_{CC}–40V	–	–	V_{CC}–37	V	1
		$D_4 - D_{14}$ R0 – R2	150kΩ to V_{CC}–40V	–	–	V_{CC}–37	V	2
Input/Output Leakage Current	$\|I_{IL}\|$	$D_4 - D_{14}$ R0 – R2 R_{A1}	V_{in} = V_{CC}–40V to V_{CC}	–	–	20	μA	3
Pull Down MOS Current	I_d	$D_4 - D_{14}$ R0 – R2 R_{A1}	V_{disp} = V_{CC}–35V V_{in} = V_{CC}	125	250	600	μA	4

(Note 1) Applied to I/O pins with "with Pull-down MOS" selected by mask option.
(Note 2) Applied to I/O pins with "without Pull-down MOS (PMOS Open Drain)" selected by mask option.
(Note 3) Pull-down MOS current and output buffer current are excluded.
(Note 4) Applied to I/O pins with "with Pull-down MOS" selected by mask option.

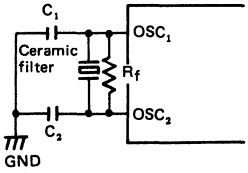
(4) AC characteristics (V_{CC} = 3.5V to 6V, GND = 0V, V_{disp} = V_{CC}–40V to V_{CC}, Ta = –20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Oscillation Frequency	f_{osc}	OSC_1, OSC_2	divide-by-8	0.4	4	4.5	MHz	
			divide-by-4	0.2	2	2.25	MHz	
Instruction Cycle Time	t_{cyc}			1.78	2	20	μs	
Oscillator Stabilization Time	t_{RC}	OSC_1, OSC_2		–	–	20	ms	1
External Clock "High" Level Width	t_{CPH}	OSC_1	divide-by-8	92	–	–	ns	2
			divide-by-4	203	–	–	ns	2
External Clock "Low" Level Width	t_{CPL}	OSC_1	divide-by-8	92	–	–	ns	2
			divide-by-4	203	–	–	ns	2
External Clock Rise Time	t_{CPr}	OSC_1		–	–	20	ns	2
External Clock Fall Time	t_{CPf}	OSC_1		–	–	20	ns	2
$\overline{INT_0}$ "High" Level Width	t_{I0H}	$\overline{INT_0}$		2	–	–	t_{cyc}	3
$\overline{INT_0}$ "Low" Level Width	t_{I0L}	$\overline{INT_0}$		2	–	–	t_{cyc}	3
$\overline{INT_1}$ "High" Level Width	t_{I1H}	$\overline{INT_1}$		2	–	–	t_{cyc}	3
$\overline{INT_1}$ "Low" Level Width	t_{I1L}	$\overline{INT_1}$		2	–	–	t_{cyc}	3
RESET "High" Level Width	t_{RSTH}	RESET		2	–	–	t_{cyc}	4
Input Capacitance	C_{in}	all pins	f = 1MHz V_{in} = 0V	–	–	15	pF	
RESET Fall Time	t_{RSTf}			–	–	20	ms	4

(Note 1) Oscillator stabilization time is the time until the oscillator stabilizes after V_{CC} reaches 3.5V at "Power-on", or after RESET input level goes to "High" by resetting to quit the stop mode by MCU reset on the circuit below. At power-on or stop mode release, equal or more than t_{RC} is required for RESET input to reserve oscillation stabilization time. When using crystal or ceramic filter oscillator, please ask a crystal oscillator maker's or ceramic filter maker's advice because oscillator stabilization time depends on the circuit constant and stray capacity.

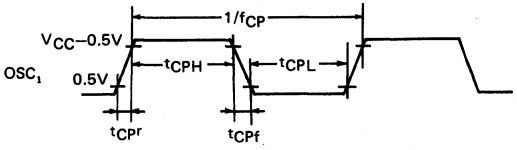


Crystal: 4.194304MHz
NC-18C (Nihon Denpa Kogyo)
R_f = 1 [MΩ] ± 20%,
C_1 = C_2 = 22 [pF] ± 20%

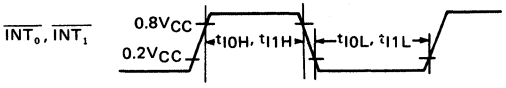


Ceramic filter: CSA 4.00MG (Murata)
R_f = 1 [MΩ] ± 20%,
C_1 = C_2 = 30 [pF] ± 20%

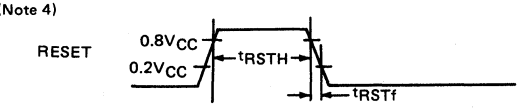
(Note 2)



(Note 3)



(Note 4)



5.13 HMCS412CL Electrical Characteristics

(1) DC characteristics (V_{CC} = 2.5V to 6V, GND = 0V, V_{disp} = V_{CC}–40V to V_{CC}, Ta = –20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Input "High" Voltage	V_{IH}	RESET, $R_{32}/\overline{INT_0}$, $R_{33}/\overline{INT_1}$		0.8V_{CC}	–	V_{CC}+0.3	V	
		OSC_1		V_{CC}–0.3	–	V_{CC}+0.3	V	
Input "Low" Voltage	V_{IL}	RESET, $R_{32}/\overline{INT_0}$, $R_{33}/\overline{INT_1}$		–0.3	–	0.2V_{CC}	V	
		OSC_1		–0.3	–	0.3	V	
Input/Output Leakage Current	$\lvert I_{IL} \rvert$	RESET, $R_{32}/\overline{INT_0}$, $R_{33}/\overline{INT_1}$, OSC_1	V_{in} = 0V to V_{CC}	–	–	1	μA	1
Current Dissipation in Active Mode	I_{CC}	V_{CC}	V_{CC} = 3V, f_{osc} = 4MHz divide-by-16	–	–	0.8	mA	2, 5
			V_{CC} = 3V, f_{osc} = 2MHz divide-by-8	–	–	0.8	mA	2, 5
Current Dissipation in Standby Mode	I_{SBY}	V_{CC}	V_{CC} = 3V, f_{osc} = 4MHz divide-by-16	–	–	0.5	mA	3, 5
			V_{CC} = 3V, f_{osc} = 2MHz divide-by-8	–	–	0.5	mA	3, 5
Current Dissipation in Stop Mode	I_{stop}	V_{CC}	V_{in} ($\overline{TEST}$) = V_{CC}–0.2V to V_{CC} V_{CC}, V_{in} (RESET) = 0 to 0.2V	–	–	10	μA	4
Stop Mode Retain Voltage	V_{stop}	V_{CC}		2	–	–	V	

(Note 1) Pull-up MOS current and output buffer current are excluded.
(Note 2) The MCU is in the reset state. The input/output current does not flow.
Test Conditions: MCU state; • Reset state in Operation Mode
Pin state; • RESET, $\overline{TEST}$ ... V_{CC} voltage
• $D_0 - D_3$, R3 – R4 ... V_{CC} voltage
• $D_4 - D_{14}$, R0 – R2, R_{A1} ... V_{disp} voltage
(Note 3) The timer/counter operate with the fastest clock and input/output current does not flow.
Test Conditions: MCU state; • Standby Mode
• Input/Output; Reset state
Pin state; • RESET ... GND voltage
• $\overline{TEST}$ ... V_{CC} voltage
• $D_0 - D_3$, R3 – R4 ... V_{CC} voltage
• $D_4 - D_{14}$, R0 – R2, R_{A1} ... V_{disp} voltage
(Note 4) Pull-down MOS current is excluded.
(Note 5) When $f_{osc} = x$ [MHz], the Current Dissipation in Operation mode and Standby mode are estimated as follows:

$$\text{max. value } (f_{osc} = x\,[\text{MHz}]) = \frac{x}{2} \times \text{max. value } (f_{osc} = 4\,[\text{MHz}])$$

(2) Input/output characteristics for standard pin
(V_{CC} = 2.5V to 6V, GND = 0V, V_{disp} = V_{CC}–40V to V_{CC}, Ta = –20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Input "High" Voltage	V_{IH}	$D_0 - D_3$, R_{30}, R_{31}, R4		0.7V_{CC}	–	V_{CC}+0.3	V	
Input "Low" Voltage	V_{IL}	$D_0 - D_3$, R_{30}, R_{31}, R4		–0.3	–	0.3V_{CC}	V	
Output "High" Voltage	V_{OH}	$D_0 - D_3$, R_{30}, R_{31}, R4	$-I_{OH}$ = 0.3mA	V_{CC}–0.5	–	–	V	1
Output "Low" Voltage	V_{OL}	$D_0 - D_3$, R_{30}, R_{31}, R4	I_{OL} = 0.4mA	–	–	0.4	V	
Input/Output Leakage Current	$\|I_{IL}\|$	$D_0 - D_3$, R_{30}, R_{31}, R4	V_{in} = 0V to V_{CC}	–	–	1	μA	2
Pull-Up MOS Current	$-I_p$	$D_0 - D_3$, R_{30}, R_{31}, R4	V_{CC} = 3V, V_{in} = 0V	3	15	50	μA	3
			V_{CC} = 5V, V_{in} = 0V	30	60	150	μA	3

(Note 1) Applied to I/O pins with "CMOS" Output selected by mask option.
(Note 2) Pull-up MOS current and output buffer current are excluded.
(Note 3) Applied to I/O pins with "with Pull-up MOS" selected by mask option.

(3) Input/output characteristics for high voltage pin
(V_{CC} = 2.5V to 6V, GND = 0V, V_{disp} = V_{CC}–40V to V_{CC}, Ta = –20 to +75°C, if not specified.)

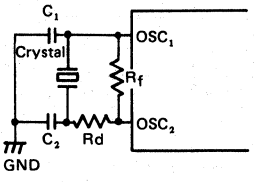
Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Input "High" Voltage	V_{IH}	$D_4 - D_{14}$, R1 R2, R_{A1}		0.7V_{CC}	–	V_{CC}+0.3	V	
Input "Low" Voltage	V_{IL}	$D_4 - D_{14}$, R1 R2, R_{A1}		V_{CC}–40	–	0.3V_{CC}	V	
Output "High" Voltage	V_{OH}	$D_4 - D_{14}$	$-I_{OH}$ = 15mA, V_{CC} = 5V ± 20%	V_{CC}–3.0	–	–	V	
			$-I_{OH}$ = 10mA, V_{CC} = 5V ± 20%	V_{CC}–2.0	–	–	V	
			$-I_{OH}$ = 2.5mA	V_{CC}–1.0	–	–	V	
		R0 – R2	$-I_{OH}$ = 3mA, V_{CC} = 5V ± 20%	V_{CC}–3.0	–	–	V	
			$-I_{OH}$ = 2mA, V_{CC} = 5V ± 20%	V_{CC}–2.0	–	–	V	
			$-I_{OH}$ = 0.5mA	V_{CC}–1.0	–	–	V	
Output "Low" Voltage	V_{OL}	$D_4 - D_{14}$ R0 – R2	V_{disp} = V_{CC}–40V	–	–	V_{CC}–37	V	1
		$D_4 - D_{14}$ R0 – R2	150kΩ to V_{CC}–40V	–	–	V_{CC}–37	V	2
Input/Output Leakage Current	$\|I_{IL}\|$	$D_4 - D_{14}$ R0 – R2 R_{A1}	V_{in} = V_{CC}–40V to V_{CC}	–	–	20	μA	3
Pull Down MOS Current	I_d	$D_4 - D_{14}$ R0 – R2 R_{A1}	V_{disp} = V_{CC}–35V V_{in} = V_{CC}	125	250	600	μA	4

(Note 1) Applied to I/O pins with "with Pull-down MOS" selected by mask option.
(Note 2) Applied to I/O pins with "without Pull-down MOS (PMOS Open Drain)" selected by mask option.
(Note 3) Pull-down MOS current and output buffer current are excluded.
(Note 4) Applied to I/O pins with "with Pull-down MOS" selected by mask option.

(4) AC characteristics (V_{CC} = 2.5V to 6V, GND = 0V, V_{disp} = V_{CC}–40V to V_{CC}, Ta = –20 to +75°C, if not specified.)

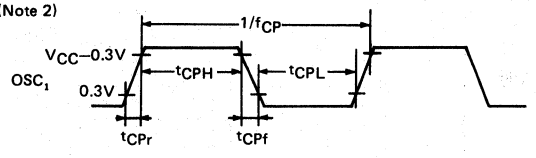
Item	Symbol	Pin Name	Test Conditions	Value min	Value typ	Value max	Unit	Note
Oscillation Frequency	f_{osc}	OSC_1, OSC_2	divide-by-16	0.8	4	4.5	MHz	
			divide-by-8	0.4	2	2.25	MHz	
Instruction Cycle Time	t_{cyc}			3.55	4	20	μs	
Oscillator Stabilization Time	t_{RC}	OSC_1, OSC_2		–	–	60	ms	1
External Clock "High" Level Width	t_{CPH}	OSC_1	divide-by-16	92	–	–	ns	2
			divide-by-8	203	–	–	ns	2
External Clock "Low" Level Width	t_{CPL}	OSC_1	divide-by-16	92	–	–	ns	2
			divide-by-8	203	–	–	ns	2
External Clock Rise Time	t_{CPr}	OSC_1		–	–	20	ns	2
External Clock Fall Time	t_{CPf}	OSC_1		–	–	20	ns	2
$\overline{INT_0}$ "High" Level Width	t_{I0H}	$\overline{INT_0}$		2	–	–	t_{cyc}	3
$\overline{INT_0}$ "Low" Level Width	t_{I0L}	$\overline{INT_0}$		2	–	–	t_{cyc}	3
$\overline{INT_1}$ "High" Level Width	t_{I1H}	$\overline{INT_1}$		2	–	–	t_{cyc}	3
$\overline{INT_1}$ "Low" Level Width	t_{I1L}	$\overline{INT_1}$		2	–	–	t_{cyc}	3
RESET "High" Level Width	t_{RSTH}	RESET		2	–	–	t_{cyc}	4
Input Capacitance	C_{in}	all pins	f = 1MHz V_{in} = 0V	–	–	15	pF	
RESET Fall Time	t_{RSTf}			–	–	15	ms	4

(Note 1) Oscillator stabilization time is the time until the oscillator stabilizes after V_{CC} reaches 2.5V at "Power-on", or after RESET input level goes to "High" by resetting to quit the stop mode by MCU reset on the circuit below. At power-on or stop mode release, equal or more than t_{RC} is required for RESET input to reserve oscillation stabilization time. When using crystal or ceramic filter oscillator, please ask a crystal oscillator maker's or ceramic filter maker's advice because oscillator stabilization time depends on the circuit constant and stray capacity.

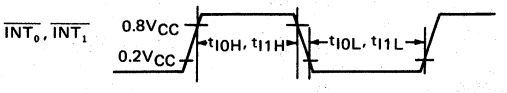


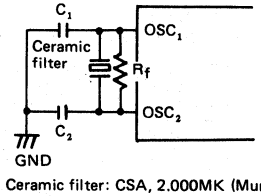
Crystal: 2.097152MHz
DS-MGQ 308 (Seiko)
R_f = 1MΩ ± 20%, R_d = 2.2kΩ ± 20%
C_1 = C_2 = 10pF ± 20%

(Note 2)



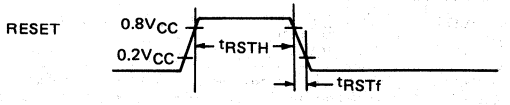
(Note 3)





Ceramic filter: CSA, 2.000MK (Murata)
R_f = 1 [MΩ] ± 20%, C_1 = C_2 = 30 [pF] ± 20%

(Note 4)



5.14 HMCS412AC Electrical Characteristics

(1) DC characteristics (V_{CC} = 4.5V to 6V, GND = 0V, V_{disp} = V_{CC}–40V to V_{CC}, Ta = –20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Input "High" Voltage	V_{IH}	RESET, $R_{32}/\overline{INT_0}$, $R_{33}/\overline{INT_1}$		0.8V_{CC}	–	V_{CC}+0.3	V	
		OSC_1		V_{CC}–0.5	–	V_{CC}+0.3	V	
Input "Low" Voltage	V_{IL}	RESET, $R_{32}/\overline{INT_0}$, $R_{33}/\overline{INT_1}$		–0.3	–	0.2V_{CC}	V	
		OSC_1		–0.3	–	0.5	V	
Input/Output Leakage Current	$\lvert I_{IL} \rvert$	RESET, $R_{32}/\overline{INT_0}$ $R_{33}/\overline{INT_1}$, OSC_1	V_{in} = 0V to V_{CC}	–	–	1	μA	1
Current Dissipation in Active Mode	I_{CC}	V_{CC}	V_{CC} = 5V, f_{osc} = 4MHz divide-by-4	–	–	3.0	mA	2, 5
Current Dissipation in Standby Mode	I_{SBY}	V_{CC}	V_{CC} = 5V, f_{osc} = 4MHz divide-by-4	–	–	1.4	mA	3, 5
Current Dissipation in Stop Mode	I_{stop}	V_{CC}	V_{in} ($\overline{TEST}$) = V_{CC}–0.3V to V_{CC} V_{CC}, V_{in} (RESET) = 0 to 0.3V	–	–	10	μA	4
Stop Mode Retain Voltage	V_{stop}	V_{CC}		2	–	–	V	

(Note 1) Pull-up MOS current and output buffer current are excluded.

(Note 2) The MCU is in the reset state. The input/output current does not flow.

Test Conditions: MCU state;
- Reset state in Operation Mode

Pin state;
- RESET, $\overline{TEST}$ ... V_{CC} voltage
- $D_0 - D_3$, R3 – R4 ... V_{CC} voltage
- $D_4 - D_{14}$, R0 – R2, R_{A1} ... V_{disp} voltage

(Note 3) The timer/counter operate with the fastest clock and input/output current does not flow.

Test Conditions: MCU state;
- Standby Mode
- Input/Output; Reset state

Pin state;
- $\overline{RESET}$ ... GND voltage
- $\overline{TEST}$ ... V_{CC} voltage
- $D_0 - D_3$, R3 – R4 ... V_{CC} voltage
- $D_4 - D_{14}$, R0 – R2, R_{A1} ... V_{disp} voltage

(Note 4) Pull-down MOS current is excluded.

(Note 5) When $f_{osc} = x$ [MHz], the Current Dissipation in Operation mode and Standby mode are estimated as follows:

max. value ($f_{osc} = x$ [MHz]) = $\frac{x}{4}$ × max. value (f_{osc} = 4 [MHz])

(2) Input/output characteristics for standard pin
(V_{CC} = 4.5V to 6V, GND = 0V, V_{disp} = V_{CC}–40V to V_{CC}, Ta = –20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Input "High" Voltage	V_{IH}	$D_0 - D_3$, R_{30}, R_{31}, R4		0.7V_{CC}	–	V_{CC}+0.3	V	
Input "Low" Voltage	V_{IL}	$D_0 - D_3$, R_{30}, R_{31}, R4		–0.3	–	0.3V_{CC}	V	
Output "High" Voltage	V_{OH}	$D_0 - D_3$, R_{30}, R_{31}, R4	$-I_{OH}$ = 1.0mA	V_{CC}–1.0	–	–	V	1
		$D_0 - D_3$, R_{30}, R_{31}, R4	$-I_{OH}$ = 0.5mA	V_{CC}–0.5	–	–	V	1
Output "Low" Voltage	V_{OL}	$D_0 - D_3$, R_{30}, R_{31}, R4	I_{OL} = 1.6mA	–	–	0.4	V	
Input/Output Leakage Current	$\|I_{IL}\|$	$D_0 - D_3$, R_{30}, R_{31}, R4	V_{in} = 0V to V_{CC}	–	–	1	μA	2
Pull-Up MOS Current	$-I_p$	$D_0 - D_3$, R_{30}, R_{31}, R4	V_{CC} = 5V V_{in} = 0V	30	60	150	μA	3

(Note 1) Applied to I/O pins with "CMOS" Output selected by mask option.
(Note 2) Pull-up MOS current and output buffer current are excluded.
(Note 3) Applied to I/O pins with "with Pull-up MOS" selected by mask option.

(3) Input/output characteristics for high voltage pin
(V_{CC} = 4.5V to 6V, GND = 0V, V_{disp} = V_{CC}–40V to V_{CC}, Ta = –20 to +75°C, if not specified.)

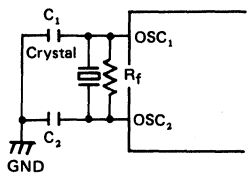
Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Input "High" Voltage	V_{IH}	$D_4 - D_{14}$, R1 R2, R_{A1}		0.7V_{CC}	–	V_{CC}+0.3	V	
Input "Low" Voltage	V_{IL}	$D_4 - D_{14}$, R1 R2, R_{A1}		V_{CC}–40	–	0.3V_{CC}	V	
Output "High" Voltage	V_{OH}	$D_4 - D_{14}$	$-I_{OH}$ = 15mA	V_{CC}–3.0	–	–	V	
			$-I_{OH}$ = 10mA	V_{CC}–2.0	–	–	V	
			$-I_{OH}$ = 4mA	V_{CC}–1.0	–	–	V	
		R0 – R2	$-I_{OH}$ = 3mA	V_{CC}–3.0	–	–	V	
			$-I_{OH}$ = 2mA	V_{CC}–2.0	–	–	V	
			$-I_{OH}$ = 0.8mA	V_{CC}–1.0	–	–	V	
Output "Low" Voltage	V_{OL}	$D_4 - D_{14}$ R0 – R2	V_{disp} = V_{CC}–40V	–	–	V_{CC}–37	V	1
		$D_4 - D_{14}$ R0 – R2	150kΩ to V_{CC}–40V	–	–	V_{CC}–37	V	2
Input/Output Leakage Current	$\|I_{IL}\|$	$D_4 - D_{14}$ R0 – R2 R_{A1}	V_{in} = V_{CC}–40V to V_{CC}	–	–	20	μA	3
Pull Down MOS Current	I_d	$D_4 - D_{14}$ R0 – R2 R_{A1}	V_{disp} = V_{CC}–35V V_{in} = V_{CC}	125	250	600	μA	4

(Note 1) Applied to I/O pins with "with Pull-down MOS" selected by mask option.
(Note 2) Applied to I/O pins with "without Pull-down MOS (PMOS Open Drain)" selected by mask option.
(Note 3) Pull-down MOS current and output buffer current are excluded.
(Note 4) Applied to I/O pins with "with Pull-down MOS" selected by mask option.

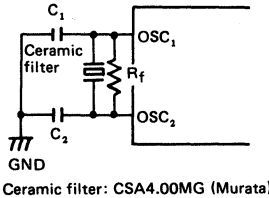
(4) AC characteristics (V_{CC} = 4.5V to 6V, GND = 0V, V_{disp} = V_{CC}–40V to V_{CC}, Ta = –20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Oscillation Frequency	f_{osc}	OSC_1, OSC_2	divide-by-4	0.2	4	4.5	MHz	
Instruction Cycle Time	t_{cyc}			0.89	1	20	μs	
Oscillator Stabilization Time	t_{RC}	OSC_1, OSC_2		–	–	20	ms	1
External Clock "High" Level Width	t_{CPH}	OSC_1	divide-by-4	92	–	–	ns	2
External Clock "Low" Level Width	t_{CPL}	OSC_1	divide-by-4	92	–	–	ns	2
External Clock Rise Time	t_{CPr}	OSC_1		–	–	20	ns	2
External Clock Fall Time	t_{CPf}	OSC_1		–	–	20	ns	2
$\overline{INT_0}$ "High" Level Width	t_{I0H}	$\overline{INT_0}$		2	–	–	t_{cyc}	3
$\overline{INT_0}$ "Low" Level Width	t_{I0L}	$\overline{INT_0}$		2	–	–	t_{cyc}	3
$\overline{INT_1}$ "High" Level Width	t_{I1H}	$\overline{INT_1}$		2	–	–	t_{cyc}	3
$\overline{INT_1}$ "Low" Level Width	t_{I1L}	$\overline{INT_1}$		2	–	–	t_{cyc}	3
RESET "High" Level Width	t_{RSTH}	RESET		2	–	–	t_{cyc}	4
Input Capacitance	C_{in}	all pins	f = 1MHz V_{in} = 0V	–	–	15	pF	
RESET Fall Time	t_{RSTf}			–	–	20	ms	4

(Note 1) Oscillator stabilization time is the time until the oscillator stabilizes after V_{CC} reaches 4.5V at "Power-on", or after RESET input level goes to "High" by resetting to quit the stop mode by MCU reset on the circuit below. At power-on or stop mode release, equal or more than t_{RC} is required for RESET input to reserve oscillation stabilization time. When using crystal or ceramic filter oscillator, please ask a crystal oscillator maker's or ceramic filter maker's advice because oscillator stabilization time depends on the circuit constant and stray capacity.

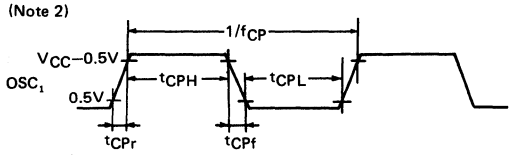


Crystal: 4.194304MHz
NC-18C (Nihon Denpa Kogyo)
R_f = 1 [MΩ] ± 20%,
C_1 = C_2 = 22 [pF] ± 20%

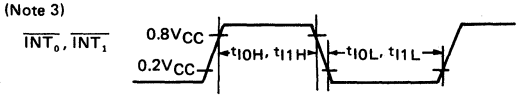


Ceramic filter: CSA4.00MG (Murata)
R_f = 1 [MΩ] ± 20%,
C_1 = C_2 = 30 [pF] ± 20%

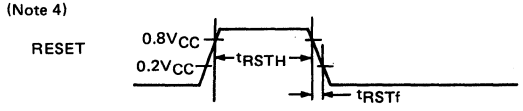
(Note 2)



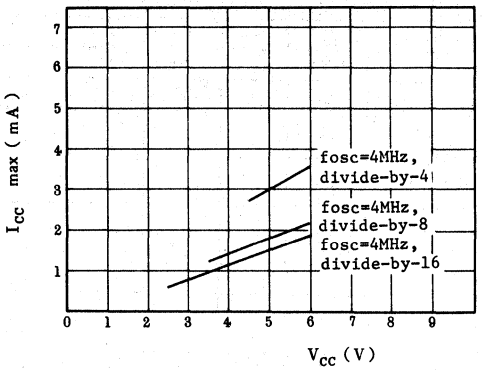
(Note 3)



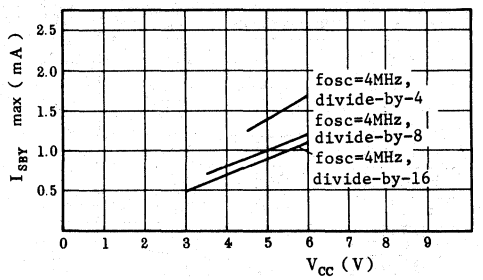
(Note 4)



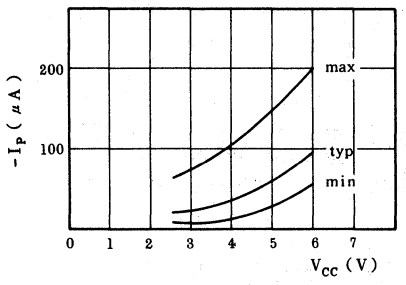
HMCS412 Characteristics Curve (Reference Data)



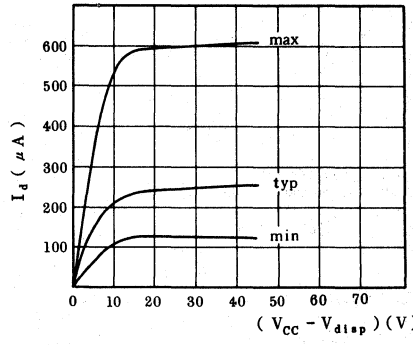
I_{CC} vs V_{CC} Characteristics (Crystal,Ceramic Filter Oscillator)



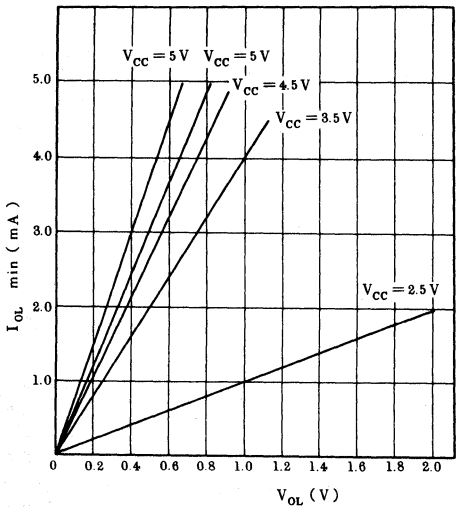
I_{SBY} vs V_{CC} Characteristics (Crystal,Ceramic Filter Oscillator)



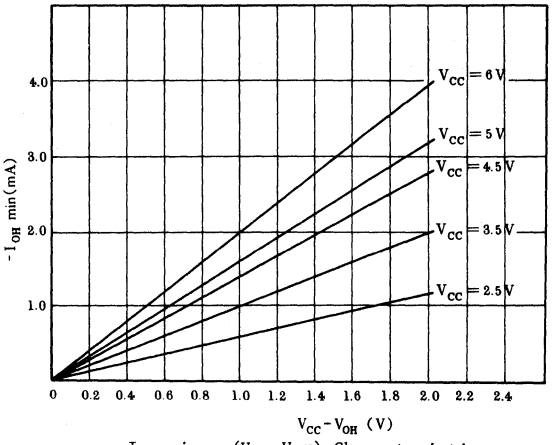
$-I_p$ vs V_{CC} Characteristics



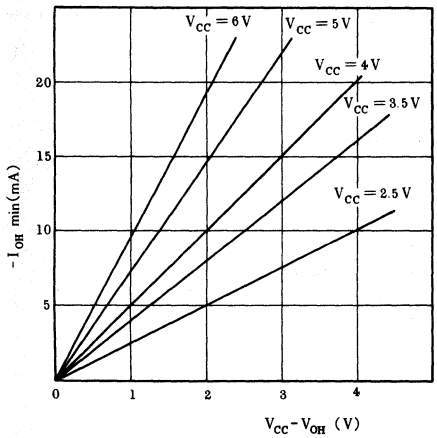
I_d vs (V_{CC}-V_{disp}) Characteristics



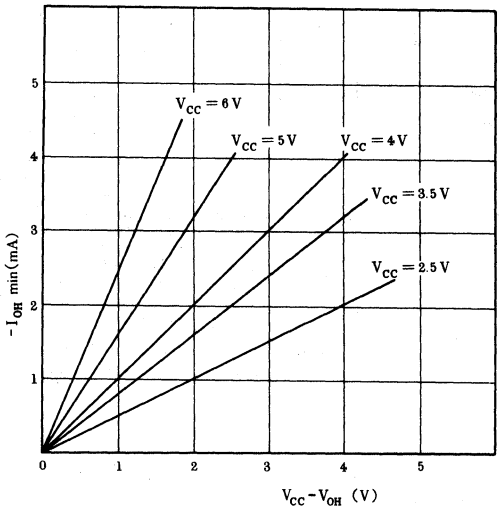
I_{OL} min vs V_{OL} Characteristics
(Standard Pin)



$-I_{OH}$ min vs($V_{CC}-V_{OH}$) Characteristics
(Standard Pin)



$-I_{OH}$ min vs ($V_{CC}-V_{OH}$) Characteristics
(D_4-D_5 Pins)



$-I_{OH}$ min vs ($V_{CC}-V_{OH}$) Characteristics
(R0-R2 Pins)

5.15 HMCS414C Electrical Characteristics

(1) DC characteristics (V_{CC} = 3.5V to 6V, GND = 0V, V_{disp} = V_{CC}–40V to V_{CC}, Ta = –20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Input "High" Voltage	V_{IH}	RESET, $R_{32}/\overline{INT_0}$, $R_{33}/\overline{INT_1}$		0.8V_{CC}	–	V_{CC}+0.3	V	
		OSC_1		V_{CC}–0.5	–	V_{CC}+0.3	V	
Input "Low" Voltage	V_{IL}	RESET, $R_{32}/\overline{INT_0}$, $R_{33}/\overline{INT_1}$		–0.3	–	0.2V_{CC}	V	
		OSC_1		–0.3	–	0.5	V	
Input/Output Leakage Current	$\lvert I_{IL} \rvert$	RESET, $R_{32}/\overline{INT_0}$, $R_{33}/\overline{INT_1}$, OSC_1	V_{in} = 0V to V_{CC}	–	–	1	μA	1
Current Dissipation in Active Mode	I_{CC}	V_{CC}	V_{CC} = 5V, f_{osc} = 4MHz divide-by-8	–	–	1.8	mA	2, 5
			V_{CC} = 5V, f_{osc} = 2MHz divide-by-4	–	–	1.8	mA	2, 5
Current Dissipation in Standby Mode	I_{SBY}	V_{CC}	V_{CC} = 5V, f_{osc} = 4MHz divide-by-8	–	–	1.0	mA	3, 5
			V_{CC} = 5V, f_{osc} = 4MHz divide-by-4	–	–	1.0	mA	3, 5
Current Dissipation in Stop Mode	I_{stop}	V_{CC}	V_{in} ($\overline{TEST}$) = V_{CC}–0.3V to V_{CC} V_{CC}, V_{in} (RESET) = 0 to 0.3V	–	–	10	μA	4
Stop Mode Retain Voltage	V_{stop}	V_{CC}		2	–	–	V	

(Note 1) Pull-up MOS current and output buffer current are excluded.
(Note 2) The MCU is in the reset state. The input/output current does not flow.
Test Conditions: MCU state; • Reset state in Operation Mode
Pin state; • RESET, $\overline{TEST}$ ... V_{CC} voltage
• $D_0 - D_3$, R3 – R4 ... V_{CC} voltage
• $D_4 - D_{14}$, R0 – R2, R_{A1} ... V_{disp} voltage
(Note 3) The timer/counter operate with the fastest clock and input/output current does not flow.
Test Conditions: MCU state; • Standby Mode
• Input/Output; Reset state
Pin state; • $\overline{RESET}$ ... GND voltage
• $\overline{TEST}$ ... V_{CC} voltage
• $D_0 - D_3$, R3 – R4 ... V_{CC} voltage
• $D_4 - D_{14}$, R0 – R2, R_{A1} ... V_{disp} voltage
(Note 4) Pull-down MOS current is excluded.
(Note 5) When $f_{osc} = x$ [MHz], the Current Dissipation in Operation mode and Standby mode are estimated as follows:

$$\text{max. value } (f_{osc} = x\,[\text{MHz}]) = \frac{x}{4} \times \text{max. value } (f_{osc} = 4\,[\text{MHz}])$$

(2) Input/output characteristics for standard pin
(V_{CC} = 3.5V to 6V, GND = 0V, V_{disp} = V_{CC}–40V to V_{CC}, Ta = –20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Input "High" Voltage	V_{IH}	$D_0 - D_3$, R_{30}, R_{31}, R4		0.7V_{CC}	–	V_{CC}+0.3	V	
Input "Low" Voltage	V_{IL}	$D_0 - D_3$, R_{30}, R_{31}, R4		–0.3	–	0.3V_{CC}	V	
Output "High" Voltage	V_{OH}	$D_0 - D_3$, R_{30}, R_{31}, R4	$-I_{OH}$ = 1.0mA	V_{CC}–1.0	–	–	V	1
		$D_0 - D_3$, R_{30}, R_{31}, R4	$-I_{OH}$ = 0.5mA	V_{CC}–0.5	–	–	V	1
Output "Low" Voltage	V_{OL}	$D_0 - D_3$, R_{30}, R_{31}, R4	I_{OL} = 1.6mA	–	–	0.4	V	
Input/Output Leakage Current	$\|I_{IL}\|$	$D_0 - D_3$, R_{30}, R_{31}, R4	V_{in} = 0V to V_{CC}	–	–	1	μA	2
Pull-Up MOS Current	$-I_p$	$D_0 - D_3$, R_{30}, R_{31}, R4	V_{CC} = 5V V_{in} = 0V	30	60	150	μA	3

(Note 1) Applied to I/O pins with "CMOS" Output selected by mask option.
(Note 2) Pull-up MOS current and output buffer current are excluded.
(Note 3) Applied to I/O pins with "with Pull-up MOS" selected by mask option.

(3) Input/output characteristics for high voltage pin
(V_{CC} = 3.5V to 6V, GND = 0V, V_{disp} = V_{CC}–40V to V_{CC}, Ta = –20 to +75°C, if not specified.)

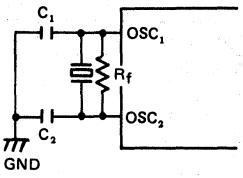
Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Input "High" Voltage	V_{IH}	$D_4 - D_{14}$, R1 R2, R_{A1}		0.7V_{CC}	–	V_{CC}+0.3	V	
Input "Low" Voltage	V_{IL}	$D_4 - D_{14}$, R1 R2, R_{A1}		V_{CC}–40	–	0.3V_{CC}	V	
Output "High" Voltage	V_{OH}	$D_4 - D_{14}$	$-I_{OH}$ = 15mA, V_{CC} = 5V ± 20%	V_{CC}–3.0	–	–	V	
			$-I_{OH}$ = 10mA, V_{CC} = 5V ± 20%	V_{CC}–2.0	–	–	V	
			$-I_{OH}$ = 4mA	V_{CC}–1.0	–	–	V	
		R0 – R2	$-I_{OH}$ = 3mA, V_{CC} = 5V ± 20%	V_{CC}–3.0	–	–	V	
			$-I_{OH}$ = 2mA, V_{CC} = 5V ± 20%	V_{CC}–2.0	–	–	V	
			$-I_{OH}$ = 0.8mA	V_{CC}–1.0	–	–	V	
Output "Low" Voltage	V_{OL}	$D_4 - D_{14}$ R0 – R2	V_{disp} = V_{CC}–40V	–	–	V_{CC}–37	V	1
		$D_4 - D_{14}$ R0 – R2	150kΩ to V_{CC}–40V	–	–	V_{CC}–37	V	2
Input/Output Leakage Current	$\|I_{IL}\|$	$D_4 - D_{14}$ R0 – R2 R_{A1}	V_{in} = V_{CC}–40V to V_{CC}	–	–	20	μA	3
Pull Down MOS Current	I_d	$D_4 - D_{14}$ R0 – R2 R_{A1}	V_{disp} = V_{CC}–35V V_{in} = V_{CC}	125	250	600	μA	4

(Note 1) Applied to I/O pins with "with Pull-down MOS" selected by mask option.
(Note 2) Applied to I/O pins with "without Pull-down MOS (PMOS Open Drain)" selected by mask option.
(Note 3) Pull-down MOS current and output buffer current are excluded.
(Note 4) Applied to I/O pins with "with Pull-down MOS" selected by mask option.

(4) AC characteristics (V_{CC} = 3.5V to 6V, GND = 0V, V_{disp} = V_{CC}–40V to V_{CC}, Ta = –20 to +75°C, if not specified.)

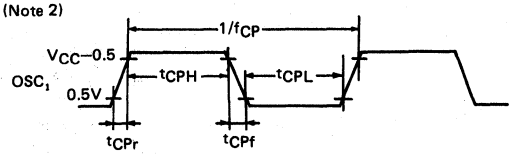
Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Oscillation Frequency	f_{osc}	OSC_1, OSC_2	divide-by-8	0.4	4	4.5	MHz	
			divide-by-4	0.2	2	2.25	MHz	
Instruction Cycle Time	t_{cyc}			1.78	2	20	μs	
Oscillator Stabilization Time	t_{RC}	OSC_1, OSC_2		–	–	20	ms	1
External Clock "High" Level Width	t_{CPH}	OSC_1	divide-by-8	92	–	–	ns	2
			divide-by-4	203	–	–	ns	2
External Clock "Low" Level Width	t_{CPL}	OSC_1	divide-by-8	92	–	–	ns	2
			divide-by-4	203	–	–	ns	2
External Clock Rise Time	t_{CPr}	OSC_1		–	–	20	ns	2
External Clock Fall Time	t_{CPf}	OSC_1		–	–	20	ns	2
$\overline{INT_0}$ "High" Level Width	t_{I0H}	$\overline{INT_0}$		2	–	–	t_{cyc}	3
$\overline{INT_0}$ "Low" Level Width	t_{I0L}	$\overline{INT_0}$		2	–	–	t_{cyc}	3
$\overline{INT_1}$ "High" Level Width	t_{I1H}	$\overline{INT_1}$		2	–	–	t_{cyc}	3
$\overline{INT_1}$ "Low" Level Width	t_{I1L}	$\overline{INT_1}$		2	–	–	t_{cyc}	3
RESET "High" Level Width	t_{RSTH}	RESET		2	–	–	t_{cyc}	4
Input Capacitance	C_{in}	all pins	f = 1MHz V_{in} = 0V	–	–	15	pF	
RESET Fall Time	t_{RSTf}			–	–	20	ms	4

(Note 1) Oscillator stabilization time is the time until the oscillator stabilizes after V_{CC} reaches 3.5V at "Power-on", or after RESET input level goes to "High" by resetting to quit the stop mode by MCU reset on the circuit below. At power-on or stop mode release, equal or more than t_{RC} is required for RESET input to reserve oscillation stabilization time. When using crystal or ceramic filter oscillator, please ask a crystal oscillator maker's or ceramic filter maker's advice because oscillator stabilization time depends on the circuit constant and stray capacity.

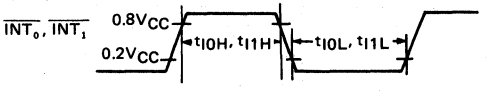


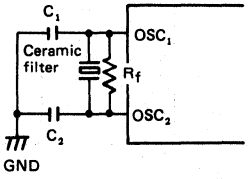
Crystal: 4.194304MHz
NC-18C (Nihon Denpa Kogyo)
R_f = 1 [MΩ] ± 20%, C_1 = C_2 = 22 [pF] ± 20%

(Note 2)



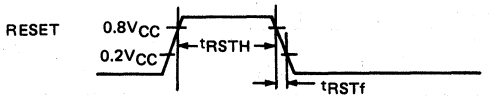
(Note 3)





Ceramic filter: CSA 4.00MG (Murata)
R_f = 1 [MΩ] ± 20%, C_1 = C_2 = 30 [pF] ± 20%

(Note 4)



5.16 HMCS414CL Electrical Characteristics

(1) DC characteristics (V_{CC} = 2.5V to 6V, GND = 0V, V_{disp} = V_{CC}–40V to V_{CC}, Ta = –20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Input "High" Voltage	V_{IH}	RESET, $R_{32}/\overline{INT_0}$, $R_{33}/\overline{INT_1}$		0.8V_{CC}	–	V_{CC}+0.3	V	
		OSC_1		V_{CC}–0.5	–	V_{CC}+0.3	V	
Input "Low" Voltage	V_{IL}	RESET, $R_{32}/\overline{INT_0}$, $R_{33}/\overline{INT_1}$		–0.3	–	0.2V_{CC}	V	
		OSC_1		–0.3	–	0.3	V	
Input/Output Leakage Current	$\lvert I_{IL} \rvert$	RESET, $R_{32}/\overline{INT_0}$, $R_{33}/\overline{INT_1}$, OSC_1	V_{in} = 0V to V_{CC}	–	–	1	μA	1
Current Dissipation in Active Mode	I_{CC}	V_{CC}	V_{CC} = 3V, f_{osc} = 4MHz divide-by-16	–	–	0.8	mA	2, 5
			V_{CC} = 3V, f_{osc} = 2MHz divide-by-8	–	–	0.8	mA	2, 5
Current Dissipation in Standby Mode	I_{SBY}	V_{CC}	V_{CC} = 3V, f_{osc} = 4MHz divide-by-16	–	–	0.5	mA	3, 5
			V_{CC} = 3V, f_{osc} = 2MHz divide-by-8	–	–	0.5	mA	3, 5
Current Dissipation in Stop Mode	I_{stop}	V_{CC}	V_{in} ($\overline{TEST}$) = V_{CC}–0.3V to V_{CC} V_{CC}, V_{in} (RESET) = 0 to 0.3V	–	–	10	μA	4
Stop Mode Retain Voltage	V_{stop}	V_{CC}		2	–	–	V	

(Note 1) Pull-up MOS current and output buffer current are excluded.
(Note 2) The MCU is in the reset state. The input/output current does not flow.
Test Conditions: MCU state; • Reset state in Operation Mode
Pin state; • RESET, $\overline{TEST}$ ... V_{CC} voltage
• $D_0 - D_3$, R3 – R4 ... V_{CC} voltage
• $D_4 - D_{14}$, R0 – R2, R_{A1} ... V_{disp} voltage
(Note 3) The timer/counter operate with the fastest clock and input/output current does not flow.
Test Conditions: MCU state; • Standby Mode
• Input/Output; Reset state
Pin state; • $\overline{RESET}$ ... GND voltage
• $\overline{TEST}$ ... V_{CC} voltage
• $D_0 - D_3$, R3 – R4 ... V_{CC} voltage
• $D_4 - D_{14}$, R0 – R2, R_{A1} ... V_{disp} voltage
(Note 4) Pull-down MOS current is excluded.
(Note 5) When $f_{osc} = x$[MHz], the Current Dissipation in Operation mode and Standby mode are estimated as follows:

$$\text{max. value } (f_{osc} = x\text{[MHz]}) = \frac{x}{2} \times \text{max. value } (f_{osc} = 4\text{[MHz]})$$

(2) Input/output characteristics for standard pin
(V_{CC} = 2.5V to 6V, GND = 0V, V_{disp} = V_{CC}–40V to V_{CC}, Ta = –20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Input "High" Voltage	V_{IH}	$D_0 - D_3$, R_{30}, R_{31}, R4		0.7V_{CC}	–	V_{CC}+0.3	V	
Input "Low" Voltage	V_{IL}	$D_0 - D_3$, R_{30}, R_{31}, R4		–0.3	–	0.3V_{CC}	V	
Output "High" Voltage	V_{OH}	$D_0 - D_3$, R_{30}, R_{31}, R4	$-I_{OH}$ = 0.3mA	V_{CC}–0.5	–	–	V	1
Output "Low" Voltage	V_{OL}	$D_0 - D_3$, R_{30}, R_{31}, R4	I_{OL} = 0.4mA	–	–	0.4	V	
Input/Output Leakage Current	$\|I_{IL}\|$	$D_0 - D_3$, R_{30}, R_{31}, R4	V_{in} = 0V to V_{CC}	–	–	1	μA	2
Pull-Up MOS Current	$-I_p$	$D_0 - D_3$, R_{30}, R_{31}, R4	V_{CC} = 3V, V_{in} = 0V	3	15	50	μA	3
			V_{CC} = 5V, V_{in} = 0V	30	60	150	μA	3

(Note 1) Applied to I/O pins with "CMOS" Output selected by mask option.
(Note 2) Pull-up MOS current and output buffer current are excluded.
(Note 3) Applied to I/O pins with "with Pull-up MOS" selected by mask option.

(3) Input/output characteristics for high voltage pin
(V_{CC} = 2.5V to 6V, GND = 0V, V_{disp} = V_{CC}–40V to V_{CC}, Ta = –20 to +75°C, if not specified.)

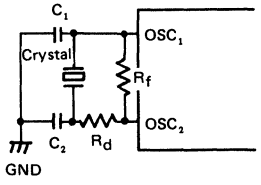
Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Input "High" Voltage	V_{IH}	$D_4 - D_{14}$, R1 R2, R_{A1}		0.7V_{CC}	–	V_{CC}+0.3	V	
Input "Low" Voltage	V_{IL}	$D_4 - D_{14}$, R1 R2, R_{A1}		V_{CC}–40	–	0.3V_{CC}	V	
Output "High" Voltage	V_{OH}	$D_4 - D_{14}$	$-I_{OH}$ = 15mA, V_{CC} = 5V ± 20%	V_{CC}–3.0	–	–	V	
			$-I_{OH}$ = 10mA, V_{CC} = 5V ± 20%	V_{CC}–2.0	–	–	V	
			$-I_{OH}$ = 2.5mA	V_{CC}–1.0	–	–	V	
		R0 – R2	$-I_{OH}$ = 3mA, V_{CC} = 5V ± 20%	V_{CC}–3.0	–	–	V	
			$-I_{OH}$ = 2mA, V_{CC} = 5V ± 20%	V_{CC}–2.0	–	–	V	
			$-I_{OH}$ = 0.5mA	V_{CC}–1.0	–	–	V	
Output "Low" Voltage	V_{OL}	$D_4 - D_{14}$ R0 – R2	V_{disp} = V_{CC}–40V	–	–	V_{CC}–37	V	1
		$D_4 - D_{14}$ R0 – R2	150kΩ to V_{CC}–40V	–	–	V_{CC}–37	V	2
Input/Output Leakage Current	$\|I_{IL}\|$	$D_4 - D_{14}$ R0 – R2 R_{A1}	V_{in} = V_{CC}–40V to V_{CC}	–	–	20	μA	3
Pull Down MOS Current	I_d	$D_4 - D_{14}$ R0 – R2 R_{A1}	V_{disp} = V_{CC}–35V V_{in} = V_{CC}	125	250	600	μA	4

(Note 1) Applied to I/O pins with "with Pull-down MOS" selected by mask option.
(Note 2) Applied to I/O pins with "without Pull-down MOS (PMOS Open Drain)" selected by mask option.
(Note 3) Pull-down MOS current and output buffer current are excluded.
(Note 4) Applied to I/O pins with "with Pull-down MOS" selected by mask option.

(4) AC characteristics (V_{CC} = 2.5V to 6V, GND = 0V, V_{disp} = V_{CC}–40V to V_{CC}, Ta = –20 to +75°C, if not specified.)

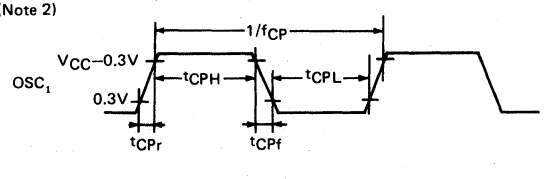
Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Oscillation Frequency	f_{osc}	OSC_1, OSC_2	divide-by-16	0.8	4	4.5	MHz	
			divide-by-8	0.4	2	2.25	MHz	
Instruction Cycle Time	t_{cyc}			3.55	4	20	μs	
Oscillator Stabilization Time	t_{RC}	OSC_1, OSC_2		–	–	60	ms	1
External Clock "High" Level Width	t_{CPH}	OSC_1	divide-by-16	92	–	–	ns	2
			divide-by-8	203	–	–	ns	2
External Clock "Low" Level Width	t_{CPL}	OSC_1	divide-by-16	92	–	–	ns	2
			divide-by-8	203	–	–	ns	2
External Clock Rise Time	t_{CPr}	OSC_1		–	–	20	ns	2
External Clock Fall Time	t_{CPf}	OSC_1		–	–	20	ns	2
$\overline{INT_0}$ "High" Level Width	t_{I0H}	$\overline{INT_0}$		2	–	–	t_{cyc}	3
$\overline{INT_0}$ "Low" Level Width	t_{I0L}	$\overline{INT_0}$		2	–	–	t_{cyc}	3
$\overline{INT_1}$ "High" Level Width	t_{I1H}	$\overline{INT_1}$		2	–	–	t_{cyc}	3
$\overline{INT_1}$ "Low" Level Width	t_{I1L}	$\overline{INT_1}$		2	–	–	t_{cyc}	3
RESET "High" Level Width	t_{RSTH}	RESET		2	–	–	t_{cyc}	4
Input Capacitance	C_{in}	all pins	f = 1MHz V_{in} = 0V	–	–	15	pF	
RESET Fall Time	t_{RSTf}			–	–	15	ms	4

(Note 1) Oscillator stabilization time is the time until the oscillator stabilizes after V_{CC} reaches 2.5V at "Power-on", or after RESET input level goes to "High" by resetting to quit the stop mode by MCU reset on the circuit below. At power-on or stop mode release, equal or more than t_{RC} is required for RESET input to reserve oscillation stabilization time. When using crystal or ceramic filter oscillator, please ask a crystal oscillator maker's or ceramic filter maker's advice because oscillator stabilization time depends on the circuit constant and stray capacity.



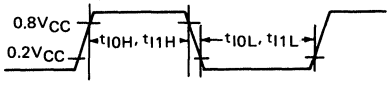
Crystal: 2.097152MHz
DS-MGQ 308 (Seiko)
R_f = 1MΩ ± 20%, Rd = 2.2kΩ ± 20%
C_1 = C_2 = 10pF ± 20%

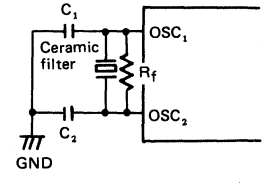
(Note 2)



(Note 3)

$\overline{INT_0}$, $\overline{INT_1}$

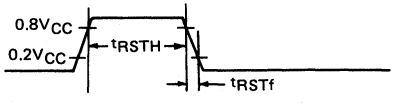




Ceramic filter: CSA 2.000MK (Murata)
R_f = 1 [MΩ] ± 20%,
C_1 = C_2 = 30 [pF] ± 20%

(Note 4)

RESET



5.17 HMCS414AC Electrical Characteristics

(1) DC characteristics (V_{CC} = 4.5V to 6V, GND = 0V, V_{disp} = V_{CC}–40V to V_{CC}, Ta = –20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Input "High" Voltage	V_{IH}	RESET, $R_{32}/\overline{INT_0}$, $R_{33}/\overline{INT_1}$		0.8V_{CC}	–	V_{CC}+0.3	V	
		OSC_1		V_{CC}–0.5	–	V_{CC}+0.3	V	
Input "Low" Voltage	V_{IL}	RESET, $R_{32}/\overline{INT_0}$, $R_{33}/\overline{INT_1}$		–0.3	–	0.2V_{CC}	V	
		OSC_1		–0.3	–	0.5	V	
Input/Output Leakage Current	$\lvert I_{IL} \rvert$	RESET, $R_{32}/\overline{INT_0}$, $R_{33}/\overline{INT_1}$, OSC_1	V_{in} = 0V to V_{CC}	–	–	1	μA	1
Current Dissipation in Active Mode	I_{CC}	V_{CC}	V_{CC} = 5V, f_{osc} = 4MHz divide-by-4	–	–	3.0	mA	2, 5
Current Dissipation in Standby Mode	I_{SBY}	V_{CC}	V_{CC} = 5V, f_{osc} = 4MHz divide-by-4	–	–	1.4	mA	3, 5
Current Dissipation in Stop Mode	I_{stop}	V_{CC}	V_{in} ($\overline{TEST}$) = V_{CC}–0.3V to V_{CC} V_{CC}, V_{in} (RESET) = 0 to 0.3V	–	–	10	μA	4
Stop Mode Retain Voltage	V_{stop}	V_{CC}		2	–	–	V	

(Note 1) Pull-up MOS current and output buffer current are excluded.

(Note 2) The MCU is in the reset state. The input/output current does not flow.

Test Conditions: MCU state; • Reset state in Operation Mode

Pin state; • RESET, $\overline{TEST}$ ... V_{CC} voltage
• $D_0 - D_3$, R3 – R4 ... V_{CC} voltage
• $D_4 - D_{14}$, R0 – R2, R_{A1} ... V_{disp} voltage

(Note 3) The timer/counter operate with the fastest clock and input/output current does not flow.

Test Conditions: MCU state; • Standby Mode
• Input/Output; Reset state

Pin state; • RESET ... GND voltage
• $\overline{TEST}$ ... V_{CC} voltage
• $D_0 - D_3$, R3 – R4 ... V_{CC} voltage
• $D_4 - D_{14}$, R0 – R2, R_{A1} ... V_{disp} voltage

(Note 4) Pull-down MOS current is excluded.

(Note 5) When $f_{osc} = x$[MHz], the Current Dissipation in Operation mode and Standby mode are estimated as follows:

$$\text{max. value } (f_{osc} = x\text{[MHz]}) = \frac{x}{4} \times \text{max. value } (f_{osc} = 4\text{[MHz]})$$

(2) Input/output characteristics for standard pin
(V_{CC} = 4.5V to 6V, GND = 0V, V_{disp} = V_{CC}–40V to V_{CC}, Ta = –20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Input "High" Voltage	V_{IH}	$D_0 - D_3$, R_{30}, R_{31}, R4		0.7V_{CC}	–	V_{CC}+0.3	V	
Input "Low" Voltage	V_{IL}	$D_0 - D_3$, R_{30}, R_{31}, R4		–0.3	–	0.3V_{CC}	V	
Output "High" Voltage	V_{OH}	$D_0 - D_3$, R_{30}, R_{31}, R4	$-I_{OH}$ = 1.0mA	V_{CC}–1.0	–	–	V	1
		$D_0 - D_3$, R_{30}, R_{31}, R4	$-I_{OH}$ = 0.5mA	V_{CC}–0.5	–	–	V	1
Output "Low" Voltage	V_{OL}	$D_0 - D_3$, R_{30}, R_{31}, R4	I_{OL} = 1.6mA	–	–	0.4	V	
Input/Output Leakage Current	$\|I_{IL}\|$	$D_0 - D_3$, R_{30}, R_{31}, R4	V_{in} = 0V to V_{CC}	–	–	1	μA	2
Pull-Up MOS Current	$-I_p$	$D_0 - D_3$, R_{30}, R_{31}, R4	V_{CC} = 5V V_{in} = 0V	30	60	150	μA	3

(Note 1) Applied to I/O pins with "CMOS" Output selected by mask option.
(Note 2) Pull-up MOS current and output buffer current are excluded.
(Note 3) Applied to I/O pins with "with Pull-up MOS" selected by mask option.

(3) Input/output characteristics for high voltage pin
(V_{CC} = 4.5V to 6V, GND = 0V, V_{disp} = V_{CC}–40V to V_{CC}, Ta = –20 to +75°C, if not specified.)

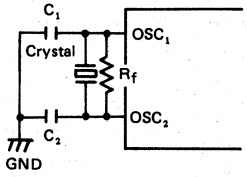
Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Input "High" Voltage	V_{IH}	$D_4 - D_{14}$, R1 R2, R_{A1}		0.7V_{CC}	–	V_{CC}+0.3	V	
Input "Low" Voltage	V_{IL}	$D_4 - D_{14}$, R1 R2, R_{A1}		V_{CC}–40	–	0.3V_{CC}	V	
Output "High" Voltage	V_{OH}	$D_4 - D_{14}$	$-I_{OH}$ = 15mA	V_{CC}–3.0	–	–	V	
			$-I_{OH}$ = 10mA	V_{CC}–2.0	–	–	V	
			$-I_{OH}$ = 4mA	V_{CC}–1.0	–	–	V	
		$R_0 - R_2$	$-I_{OH}$ = 3mA	V_{CC}–3.0	–	–	V	
			$-I_{OH}$ = 2mA	V_{CC}–2.0	–	–	V	
			$-I_{OH}$ = 0.8mA	V_{CC}–1.0	–	–	V	
Output "Low" Voltage	V_{OL}	$D_4 - D_{14}$ R0 – R2	V_{disp} = V_{CC}–40V	–	–	V_{CC}–37	V	1
		$D_4 - D_{14}$ R0 – R2	150kΩ to V_{CC}–40V	–	–	V_{CC}–37	V	2
Input/Output Leakage Current	$\|I_{IL}\|$	$D_4 - D_{14}$ R0 – R2 R_{A1}	V_{in} = V_{CC}–40V to V_{CC}	–	–	20	μA	3
Pull Down MOS Current	I_d	$D_4 - D_{14}$ R0 – R2 R_{A1}	V_{disp} = V_{CC}–35V V_{in} = V_{CC}	125	250	600	μA	4

(Note 1) Applied to I/O pins with "with Pull-down MOS" selected by mask option.
(Note 2) Applied to I/O pins with "without Pull-down MOS (PMOS Open Drain)" selected by mask option.
(Note 3) Pull-down MOS current and output buffer current are excluded.
(Note 4) Applied to I/O pins with "with Pull-down MOS" selected by mask option.

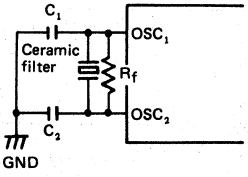
(4) AC characteristics (V_{CC} = 4.5V to 6V, GND = 0V, V_{disp} = V_{CC}–40V to V_{CC}, Ta = –20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Oscillation Frequency	f_{osc}	OSC_1, OSC_2	divide-by-4	0.2	4	4.5	MHz	
Instruction Cycle Time	t_{cyc}			0.89	1	20	μs	
Oscillator Stabilization Time	t_{RC}	OSC_1, OSC_2		–	–	20	ms	1
External Clock "High" Level Width	t_{CPH}	OSC_1		92	–	–	ns	2
External Clock "Low" Level Width	t_{CPL}	OSC_1		92	–	–		2
External Clock Rise Time	t_{CPr}	OSC_1		–	–	20	ns	2
External Clock Fall Time	t_{CPf}	OSC_1		–	–	20	ns	2
$\overline{INT_0}$ "High" Level Width	t_{I0H}	$\overline{INT_0}$		2	–	–	t_{cyc}	3
$\overline{INT_0}$ "Low" Level Width	t_{I0L}	$\overline{INT_0}$		2	–	–	t_{cyc}	3
$\overline{INT_1}$ "High" Level Width	t_{I1H}	$\overline{INT_1}$		2	–	–	t_{cyc}	3
$\overline{INT_1}$ "Low" Level Width	t_{I1L}	$\overline{INT_1}$		2	–	–	t_{cyc}	3
RESET "High" Level Width	t_{RSTH}	RESET		2	–	–	t_{cyc}	4
Input Capacitance	C_{in}	all pins	f = 1MHz V_{in} = 0V	–	–	15	pF	
RESET Fall Time	t_{RSTf}			–	–	15	ms	4

(Note 1) Oscillator stabilization time is the time until the oscillator stabilizes after V_{CC} reaches 4.5V at "Power-on", or after RESET input level goes to "High" by resetting to quit the stop mode by MCU reset on the circuit below. At power-on or stop mode release, equal or more than t_{RC} is required for RESET input to reserve oscillation stabilization time. When using crystal or ceramic filter oscillator, please ask a crystal oscillator maker's or ceramic filter maker's advice because oscillator stabilization time depends on the circuit constant and stray capacity.

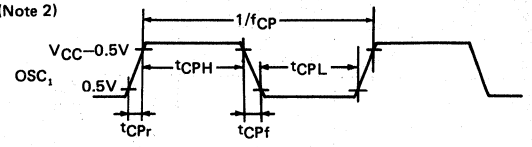


Crystal: 4.194304MHz
NC-18C (Nihon Denpa Kogyo)
R_f = 1 [MΩ] ± 20%,
C_1 = C_2 = 22 [pF] ± 20%

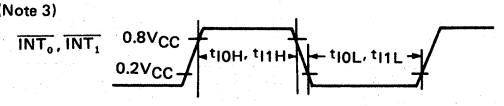


Ceramic filter: CSA4.00 MG (Murata)
R_f = 1 [MΩ] ± 20%,
C_1 = C_2 = 30 [pF] ± 20%

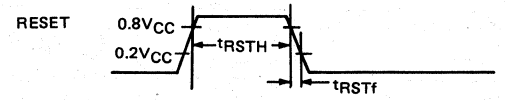
(Note 2)



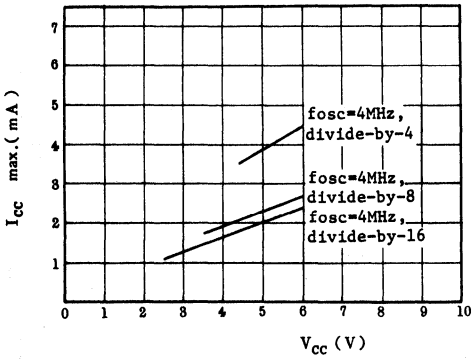
(Note 3)



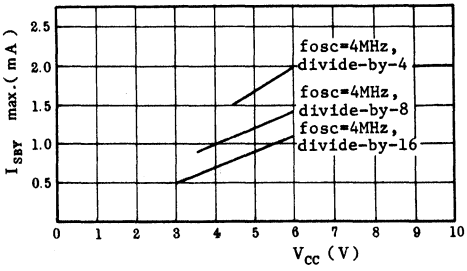
(Note 4)



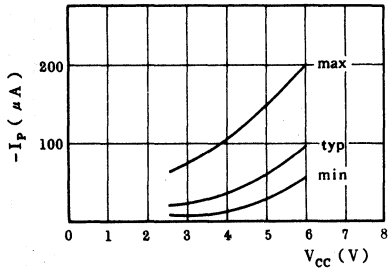
HMCS414 Characteristics Curve (Reference Data)



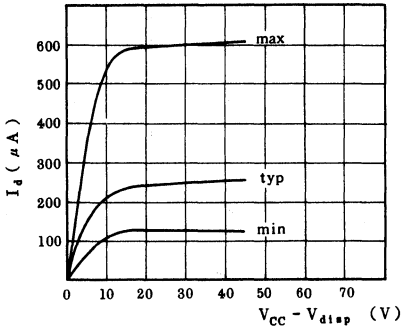
I_{CC} vs V_{CC} Characteristics (Crystal, Ceramic Filter Oscillation)



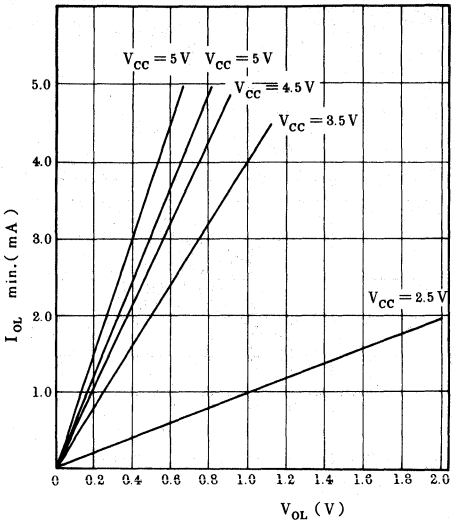
I_{SBY} vs V_{CC} Characteristics (Crystal, Ceramic Filter Oscillator)



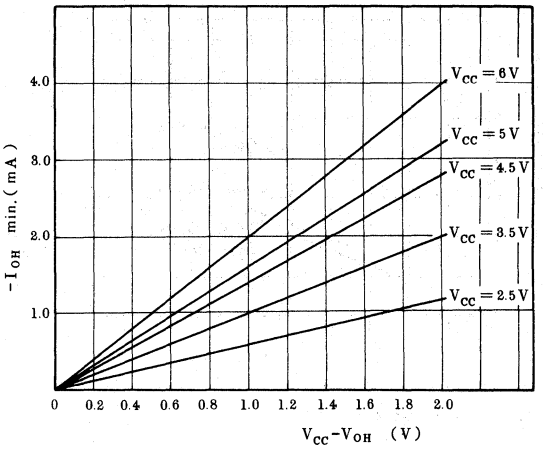
$-I_P$ vs V_{CC} Characteristics



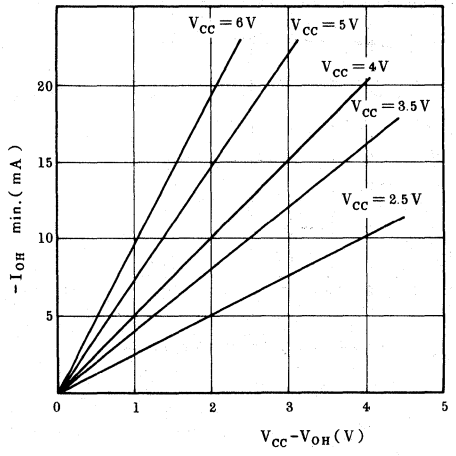
I_d vs (V_{CC}-V_{disp}) Characteristics



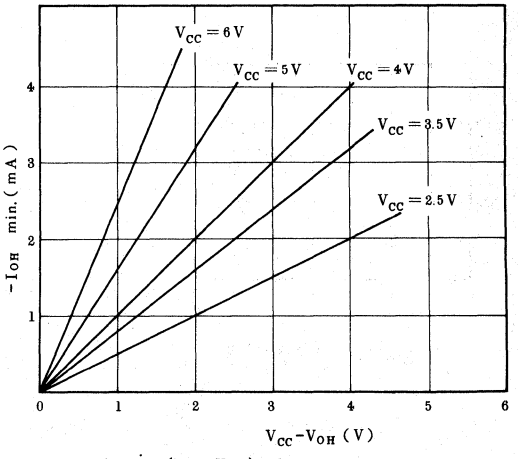
I_{OL} min vs V_{OL} Characteristics
(Standard Pin)



$-I_{OH}$ min vs ($V_{CC}-V_{OH}$) Characteristics
(Standard Pin)



$-I_{OH}$ min vs ($V_{CC}-V_{OH}$) Characteristics
(D_4 - D_5 Pins)



$-I_{OH}$ min vs ($V_{CC}-V_{OH}$) Characteristics
(R0-R2 Pins)

6. ASSEMBLY LANGUAGE

6.1 Symbols and Abbreviations

a → b	Tranfer from "a" to "b"
a ↔ b	Exchange between "a" and "b"
$\overline{X}$	Logical negation (NOT)
"1"	"High" level
"0"	"Low" level
LSB	Least Significant Bit
MSB	Most Significant Bit
NZ	Not Zero
NB	No Borrow
OVF	Overflow
	} Status is set with NZ, NB, or OVF
∩	AND
∪	OR
⊕	Exclusive OR
≠	Not Equal
≦	Less or Equal
DIRECT	Addressing by the Operand in the ROM Code
REGISTER	Addressing by the Content of Address Register

6.2 Instruction Formats

There are eight different formats for the instructions. (See "3.4 Instruction Table".)

No.	Format	Operand	Contents
1	I	-	-
2	II	n	0 - 3
3	III	i or m or p	$0 - $F
4	IV	a	$0 - $3F
5	V	b	$0 - $FF
6	VI	d	$0 - $3FF
7	VII	n, d	0 - 3, $0 - $3FF
8	VIII	p, d	$0 - $F, $0 - $3FF
9	VIII	i, d	$0 - $F, $0 - $3FF

The symbols used in the operand have the following meanings:

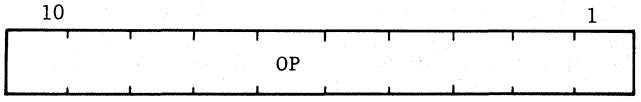
Symbol	Contents	Label	Decimal	Hexa decimal	Binary
n	RAM digit selection with 2 bits	o	o	o	o
m	RAM (MR) digit and port selection with 4 bits	o	o	o	o
p	Replacement of contents of PC with 4 bits	o	o	o	o
a	Replacement of contents of PC with 6 bits	o	×	o	o
b	Replacement of contents of PC with 8 bits	o	×	o	o
d	Replacement of PC or RAM direct address with 10 bits	o	×	o	o
i	Immediate data with 4 bits (Note)	o	o	o	o
u	u = p + d (14 bits)	o	×	o	o

o .... Can be used × .... Cannot be used

(Note) 2-bit data for LWI instruction

(1) Format I

Format I applies to the instructions having no operand.



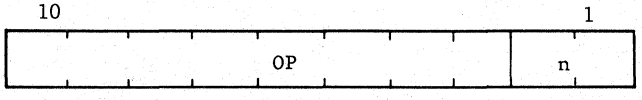
In a format-I instruction, the operation field is directly followed by the comment field.

(Example)

Label	Operation	Operand	Comment
WHERE	AMC		COMMENT
	ROTR		
	NOP		

(2) Format II

Format II applies to instructions having a 2-bit operand field.



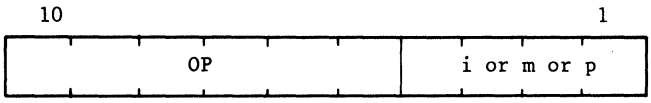
The statement using this type of instruction requires an operand (a binary, decimal or hexadecimal number, or symbol name). The value must be 0 to 3 in terms of decimal number.

(Example)

Label	Operation	Operand	Comment
ONE	EQU	1	
TWO	EQU	2	
	SEM	1	
	TM	TWO	
	REM	ONE	

(3) Format III

Format III applies to instructions having a 4-bit operand field.



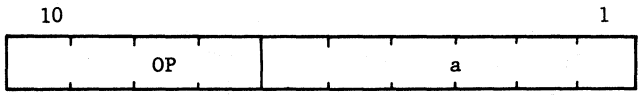
This type of instruction requires an operand (a binary, decimal or hexadecimal number, or symbol name). The value must be 0 to 15 in terms of decimal number. Note that the value must be $0 to $F in terms of hexadecimal number for the LAR, LBR, LRA and LRB instructions.

(Example)

Label	Operation	Operand	Comment
DATA1	EQU	3	
DATA2	EQU	$B	
DATA3	EQU	%1011	
	LYI	14	
	LXI	$A	
	LAI	$0010	
	LBI	DATA1	
	AI	DATA2	
	LRB	DATA3	
	LRA	DATA1	
	P	DATA3	

(4) Format IV

Format IV applies to instructions having a 6-bit operand field.



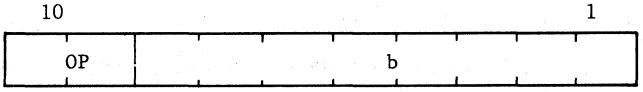
This type of instruction requires an operand (a binary or hexadecimal number, or symbol name). The value must be $0 to $3F in terms of hexadecimal number. This format is applied only to CAL instruction.

(Example)

Label	Operation	Operand	Comment
SUBAD	EQU	60	
	CAL	SUBAD	
	CAL	$3C	

(5) Format V

Format V applies to instructions having an 8-bit operand field.



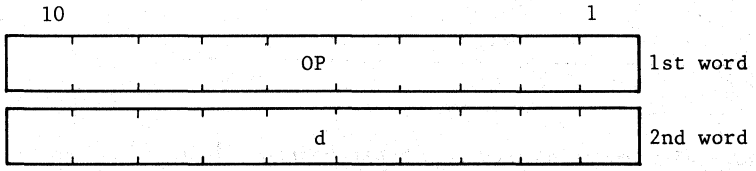
This type of instruction requires an operand (a binary or hexadecimal number, symbol name, or *±n (n: a decimal number) (relative address). This format is applied to BR instruction.

(Example)

Label	Operation	Operand	Comment
	BR	LABEL1	SYMBOL
	BR	*+3	RELATIVE
	BR	$FF	ABSOLUTE
LABEL1	BR	%0000001	

(6) Format VI

Format VI applies to 2-word instructions having a 10-bit operand field.



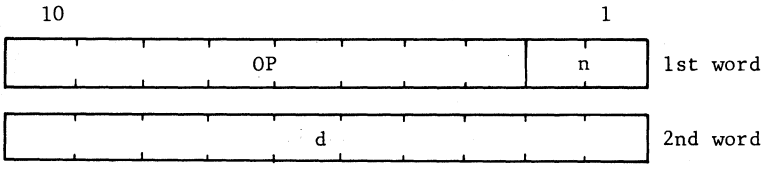
This type of instruction requires an operand (a binary or hexadecimal number, or symbol name). The value must be $0 to $3FF in terms of hexadecimal number.

(Example)

Label	Operation	Operand	Comment
RAMAD	EQU	$3FF	
	AMD	RAMAD	LABEL
	AMD	$3FF	ABSOLUTE
	AMD	%1111111111	

(7) Format VII

Format VII applies to 2-word instructions having a 12-bit operand field.



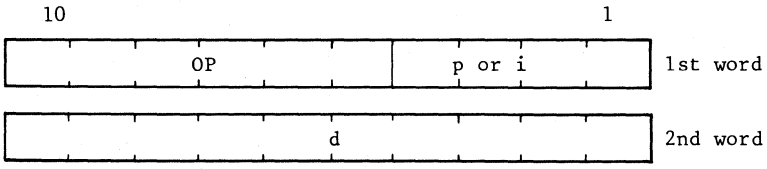
This type of instruction requires one operand or two operands. For a 1-operand instruction, only a symbol name is effective. For a 2-operand instruction, the first operand may be a binary, decimal or hexadecimal number, or symbol name (the value of which must be 0 to 3 in terms of decimal number) and the second operand may be a binary or hexadecimal number, or symbol name (the value of which must be $0 to $3FF in terms of hexadecimal number).

(Example)

Label	Operation	Operand	Comment
AAA	EQU	3, $3FF	
	REMD	AAA	1OPERAND
	SEMD	3, $3FF	2OPERAND
	TMD	%10, $3FF	

(8) Format VIII

Format VIII applies to 2-word instructions having a 14-bit operand field.



This type of instruction requires one operand or two operands. BRL, JMPL and CALL are one-operand instructions, while INEMD, ILEMD and LMID are two-operand instructions. For a 1-operand instruction, a binary or hexadecimal number, or symbol name is effective and its value must be in the range $0 to $3FF. For a 2-operand instruction, the first operand may be a binary, decimal or hexadecimal number, or symbol name (the value of which must be $0 to $F) and the second operand may be a binary or hexadecimal number, or symbol name (the value of which must be $0 to $3FF).

(Example)

Label	Operation	Operand	Comment
LNGBR1	EQU	$1FFF	
	BRL	LNGBR1	P:ABSOLUTE
	CALL	$1FFF	P:ABSOLUTE
BBB	EQU	$3FF	
	LMID	$F, BBB	1:ABSOLUTE, SYMBOL
	ILEMD	1, $3FF	1:ABSOLUTE, ABSOLUTE

6.3 Execution Instructions

No.	MNEMONIC		No.	MNEMONIC	
1	AI	i	51	LMID	i,d
2	ALEI	i	52	LMIIY	i
3	ALEM		53	LRA	m
4	ALEMD	d	54	LRB	m
5	AM		55	LWI	i
6	AMD	d	56	LXA	
7	AMC		57	LXI	i
8	AMCD	d	58	LYA	
9	ANEM		59	LYI	i
10	ANEMD	d	60	NEGA	
11	ANM		61	NOP	
12	ANMD	d	62	OR	
13	AYY		63	ORM	
14	BLEM		64	ORMD	d
15	BNEM		65	P	p
16	BR	b	66	REC	
17	BRL	u	67	RED	
18	CAL	a	68	REDD	m
19	CALL	u	69	REM	n
20	COMB		70	REMD	n,d
21	DAA		71	ROTL	
22	DAS		72	ROTR	
23	DB		73	RTN	
24	DY		74	RTNI	
25	EORM		75	SBY	
26	EORMD	d	76	SEC	
27	IB		77	SED	
28	ILEM	i	78	SEDD	m
29	ILEMD	i,d	79	SEM	n
30	INEM	i	80	SEMD	n,d
31	INEMD	i,d	81	SMC	
32	IY		82	SMCD	d
33	JMPL	u	83	STOP	
34	LAB		84	STS	
35	LAI	i	85	SYY	
36	LAM(XY)		86	TBR	p
37	LAMD	d	87	TC	
38	LAMR	m	88	TD	
39	LAR	m	89	TDD	m
40	LASPX		90	TM	n
41	LASPY		91	TMD	n,d
42	LAY		92	XMA(XY)	
43	LBA		93	XMAD	d
44	LBI	i	94	XMB(XY)	
45	LBM(XY)		95	XMRA	m
46	LBR	m	96	XSPX	
47	LMA(XY)		97	XSPXY	
48	LMAD	d	98	XSPY	
49	LMADY(X)		99	YNEI	i
50	LMAIY(X)				

In the description of Instruction code, "i", "m" and "p" means one digit in hexadecimal number ($0 to $F), and "d" means three-digit hexadecimal number ($000 to $FFF). The Instruction codes including "n", "a" or "b" are described as binary number.

No.	Arithmetic Instruction
1	AI

AI (Add Immediate to A)

Format

AI i

Operation

A + i → A

Status

OVF

A + i ≦ $F ST=0

A + i > $F ST=1

Description

Adds the contents of Accumulator to 4 bit Immediate data (i_{3-0})
and stores the result in Accumulator.
Judges OVF simultaneously.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
	$AI	i	$28i		1	1

Example

```
01830                    *
01831    190  038  0620  SUBC1   LAMD    $038
01832    28F       0622          AI      $F
01833    194  036  0623          LMAD    $036
01834                    *
```

No.	Compare Instruction
2	ALEI

ALEI (A Less or Equal to Immediate)

Format

ALEI i

Status

NB

A > i ST=0

A ≦ i ST=1

Operation

A ≦ i

Description

Compares the contents of Accumulator to 4-bit Immediate data ($i_{3 \sim 0}$).

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
	ALEI	i	$2Bi		1	1

Example

```
01407                    *
01408    2F7    0476     XMRA    $7
01409    281    0477     AI      $1
01410    2B7    0478     ALEI    $7
01411    2F7    0479     XMRA    $7
```

No.	Compare Instruction
3	ALEM

ALEM (A Less or Equal to Memory)

Format

ALEM

Operation

A ≦ M

Status

NB

A > M ST=0

A ≦ M ST=1

Description

Compares RAM addressed by W, X, and Y registers to the contents of Accumulator.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
REGISTER	ALEM		$014		1	1

Example

```
01216    014    03D7          ALEM
01217    1CE    03D8          CAL      OSERROR
```

No.	Compare Instruction
4	ALEMD

ALEMD(A Less or Equal to Memory)

Format

ALEMD d

Status

NB

A > M ST=0

A ≦ M ST=1

Operation

A ≦ M (d)

Description

Compares RAM addressed by 10-bit direct address d_{9-0} to the contents of Accumulator.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
DIRECT	ALEMD	d	$114	d	2	2

Example

```
01218    114 074    03D9          ALEMD    $074
01219    1CF        03DB          CAL      OSERROR
```

No.	Arithmetic Instruction
5	AM

AM (Add A to Memory)

Format

AM

Status

OVF

M + A ≦ $F ST=0

M + A > $F ST=1

Operation

M + A → A

Description

Adds RAM addressed by W, X, Y registers to the contents of Accumulator and stores the result in Accumulator.

Judges OVF.

(Note) When executing subtraction, execute AM after taking complement of Accumulator with NEGA. (M-A→A)

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
REGISTER	AM		$008		1	1

Example

```
00949                       *
00950     093       02FE    KIARTHM6   LAMXY
00951     008       02FF               AM
00952     097       0300               LMAXY
```

No.	Arithmetic Instruction
6	AMD

AMD (Add A to Memory)

Format

AMD d

Status

OVF

M + A ≦ $F ST=0

M + A > $F ST=1

Operation

M(d) + A → A

Description

Adds RAM addressed by 10-bit direct address d_{9-0} to the contents of Accumulator and stores the result in Accumulator.

Judges OVF.

(note) Subtraction (M-A→A) is the same as AM.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
DIRECT	AMD	d	$108	d	2	2

Example

```
00935                          *
00936    190 06C    02F0    KIARTHM4   LAMD       $06C
00937    001        02F2               XSPY
00938    108 04C    02F3               AMD        $04C
00939    051        02F5               LMAIYX
00940                          *
```

No.	Arithmetic Instruction
7	AMC

AMC (Add A to Memory with Carry)

Format

AMC

Status

OVF

M + A + CA ≦ $F ST=0

M + A + CA > $F ST=1

Operation

M + A + CA→A

OVF → CA

Description

Adds RAM addressed by W, X, Y registers, the contents of Accumulator, and those of Carry Flag, and stores the result in Accumulator.
Latches OVF into CA and judges it.

(Example)

	M	A	CA	S
(Initial contents)	8	9	0	0
after executing AMC	8	1	1	1
after re-executing AMC	8	10	0	0

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
REGISTER	AMC		$018		1	1

Example

```
01165                    *
01166   235      03A9          LAI     $5        ;A=5
01167   0EF      03AA          SEC               ;CA=1
01168   018      03AB          AMC               ;A=M(WXY)+A+CA
01169   1CE      03AC          CAL     OSERROR
```

AMCD (Add A to Memory with Carry)

Format

AMCD d

Status

OVF

M + A + CA ≦ $F ST=0

M + A + CA > $F ST=1

Operation

M(d) + A + CA →A

OVF→CA

Description

Adds RAM addressed by 10-bit direct address d_{9-0}, the contents of Accumulator, and those of CA, and stores the result in Accumulator.

Latches OVF into CA and judges it simultaneously.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
DIRECT	AMCD	d	$118	d	2	2

Example

```
00905                          *
00906    091          02D2              LAMX
00907    118 043      02D3              AMCD      $043
00908    0A6          02D5              DAA
```

No.	Compare Instruction
9	ANEM

ANEM (A Not Equal to Memory)

Format

ANEM

Status

NZ

A = M ST=0

A ≠ M ST=1

Operation

A ≠ M

Description

Compares RAM addressed by W, X, Y registers to the contents of Accumulator.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
REGISTER	ANEM		$004		1	1

Example

```
01195                        *
01196    004      03C6               ANEM                ;IF M(WXY)/=ACC
01197    1CE      03C7               CAL        OSERROR
```

No.	Compare Instruction
10	ANEMD

ANEMD (A Not Equal to Memory)

Format

ANEMD d

Status

NZ

A = M ST=0

A ≠ M ST=1

Operation

A ≠ M(d)

Description

Compares RAM addressed by 10-bit direct d_{9-0} to Accumulator.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
DIRECT	ANEMD	d	$104	d	2	2

Example

```
01729                          *
01730    190 036    05A4               LAMD    $036
01731    104 038    05A6               ANEMD   $038
01732    171 235    05A8               BRL     SUBD
```

No.	Arithmetic Instruction
11	ANM

ANM (AND Memory with A)

Format

ANM

Status

NZ

A∩M=0 ST=0

A∩M≠0 ST=1

Operation

A∩M → A

Description

ANDs the contents of Accumulator and RAM addressed by W, X, Y registers, and stores the result in Accumulator.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
REGISTER	ANM		$09C		1	1

Example

```
00925    091      02E5            LAMX
00926    09C      02E6            ANM
00927    051      02E7            LMAIYX
00928                     *
```

No.	Arithmetic Instruction
12	ANMD

ANMD (AND Memory with A)

Format

ANMD d

Status

NZ

A∩M=0 ST=0

A∩M≠0 ST=1

Operation

A∩M(d) → A

Description

ANDs the contents of Accumulator and RAM addressed by 10-bit direct address, and stores the result in Accumulator.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
DIRECT	ANMD	d	$19C	d	2	2

Example

```
01009    0AF       0330          LAY
01010    19C 02C   0331          ANMD     $02C
01011    326       0333          BR       KIINPUT1
```

No.	RAM Address Instruction
13	AYY

AYY (Add A to Y)

Format

AYY

Status

OVF

Y + A ≦ $F ST=0

Y + A > $F ST=1

Operation

Y + A→Y

Description

Adds the contents of Y register to those of Accumulator and stores the result in Y register.

Judges OVF.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
	AYY		$054		1	1

Example

```
01160    0DF      03A5            DY                    ;Y=Y-1
01161    054      03A6            AYY                   ;Y=Y+A
01162    1CE      03A7            CAL       OSERROR
01163                      *
```

No.	Compare Instruction
14	BLEM

BLEM (B Less or Equal to Memory)

Format

BLEM

Status

NB

B > M ST=0

B ≦ M ST=1

Operation

B ≦ M

Description

Compares RAM addressed by W, X, Y registers to the contents of B register.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
REGISTER	BLEM		$0C4		1	1

Example

```
01220                    *
01221   0C4      03DC            BLEM
01222   1CE      03DD            CAL     OSERROR
```

No.	Compare Instruction
15	BNEM

BNEM (B Not Equal to Memory)

Format

BNEM

Status

NZ

B=M ST=0

B≠M ST=1

Operation

B≠M

Description

Compares RAM addressed by W, X, Y registers to the contents of B register.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
REGISTER	BNEM		$044		1	1

Example

```
01201    044      03CB            BNEM
01202    1CE      03CC            CAL         OSERROR     ;IF M(WXY)/=B
```

No.	ROM Address Instruction
16	BR

BR (Branch on Status 1)

Format

BR b

Status

1

Set to 1 irrespectively of whether BR is executed or skipped. (ST=1)

Operation

Conditional jump to an address in the current page (256 words).

Description

Branches to the specified address if ST=1.

If ST=0, this instruction is skipped (takes one cycle time).

(Note) When BR is used at the last address in the page, this instruction is executed in the next page because PC is incremented automatically.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
DIRECT (8 bits)	BR	b	%11-$b_7b_6b_5b_4$ $b_3b_2b_1b_0$		1	1

Example

```
01445                    *
01446   2FE      0496            XMRA    $E
01447   0AF      0497            LAY
01448   2B3      0498            ALEI    $3
01449   3A0      0499            BR      KITIME0    ;IF A=0,1,2,3
01450   2B7      049A            ALEI    $7
01451   3A4      049B            BR      KITIME1    ;IF A=4,5,6,7
01452   2BB      049C            ALEI    $B
01453   3A8      049D            BR      KITIME2    ;IF A=8,9,$A,$B
01454   000      049E            NOP
01455   3AC      049F            BR      KITIME3    ;IF A=$C,$D,$E,$F
01456                    *
```

No.	ROM Address Instruction
17	BRL

BRL (Long Branch on Status 1)

Format

BRL u

Status

1

Set to 1 irrespectively of whether BRL is executed or skipped. (ST=1)

Operation

Conditional jump to any ROM address space.

Description

If ST=1, jumps to the address specified by $P_{3 \sim 0}$, d_{9-0}.

If ST=0, BRL is skipped.

Takes 2-cycle time irrespective of execution and skip.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
DIRECT	BRL	u	$17p	d	2	2

Example

```
01883    18C 002    0650        TMD     $0,$002   IF ST-1
01884    171 256    0652        BRL     TBINTR    THEN JUMP TO TBINTR
01885    171 24F    0654        BRL     TLOOP1    OTHERWISE JUMP TO TLOOP1
```

No.	ROM Address Instruction
18	CAL

CAL (Subroutine Jump on Status 1)

Format

CAL a

Status

1

Set to 1 irrespectively of whether CAL is executed or skipped.

(ST=1)

Operation

Conditional subroutine jump to the address specified by $a_{5 \sim 0}$ in subroutine space.

Description

If ST=1, performs subroutine jump to the specified address.

If ST=0, this instruction is skipped.

Takes 1-cycle time to the skipped.

Subroutine space means 0 to 64 pages.

All bits of PC is saved on RAM.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
DIRECT (6 bits)	CAL	a	%01-11$a_5a_4a_3$ $a_2a_1a_0$		1	2/1 (Skip)

Example

```
00510    18E 000    01A3          TMD     2,$000     ;IF0
00511    1CE        01A5          CAL     OSERROR    ;IF ST=1
```

No.	ROM Address Instruction
19	CALL

CALL (Long Subroutine Jump on Status 1)

Format

CALL u

Status

1

Set to 1 irrespectively of whether CALL is executed or skipped. (ST=1)

Operation

Conditional subroutine jump to any ROM address space.

Description

If ST=1, performs subroutine jump to the specified address (P_{3-0}, d_{9-0}).
If ST=0, this instruction is skipped.
Takes 2-cycle time to be skipped.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
DIRECT	CALL	u	$16p	d	2	2

Example

```
00781                    *
00782    28F       0276  KIRAM5   AI       $F        ;A=A+F
00783    0E8       0277            LXA                ;X=A
00784    160 28E   0278            CALL     KIRAMS
00785                    *
```

No.	Arithmetic Instruction
20	COMB

COMB (Complement B)

Format

COMB

Status

No effect

Operation

$\overline{B} \rightarrow B$

Description

Stores 1'S complement of the contents of B register in B register.

(Example)

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
	COMB		$140		1	1

Example

```
00815                    *
00816   048      0294    KIRAMC   LAB
00817   0D8      0295             LYA                ;Y=B
00818   076      0296             YNEI     $6        ;Y/=6
00819   140      0297             COMB               ;B̄⇒B
00820   010      0298             RTN
```

No.	Arithmetic Instruction
21	DAA

DAA (Decimal Adjust for Addition)

Format

DAA

Status

No effect

Operation

Decimal adjust for addition

Description

- If A ≧ 10 or CA=1, A+6→A and 1→CA.
- If A < 10 and CA=0, the contents of A and CA are unchanged.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
	DAA		$0A6		1	1

Example

```
00898                    *
00899    091     02CC    KIARTHM1   LAMX
00900    018     02CD               AMC
00901    0A6     02CE               DAA
```

No.	Arithmetic Instruction
22	DAS

DAS (Decimal Adjust for Subtraction)

Format

DAS

Status

No effect

Operation

Decimal adjust for subtraction

Description

- If A ≧ 10 or CA=0, A+10→A and 0→CA.
- If A < 10 and CA=1, the contents of A and CA are unchanged.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
	DAS		$0AA		1	1

Example

```
00914                       *
00915    091        02DC    KIARTHM2   LAMX
00916    098        02DD               SMC
00917    0AA        02DE               DAS
00918    051        02DF               LMAIYX
```

No.	Arithmetic Instruction
23	DB

DB (Decrement B)

Format

DB

Status

NB

B - 1 < 0 ST=0

B - 1 ≧ 0 ST=1

Operation

B - 1→B

Description

Decrements the contents of B register.

Judges NB.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
	DB		$0CF		1	1

Example

```
01145    0CF      0399          DB                        ;B=B-1
01146    1CE      039A          CAL       OSERROR
01147                  *
```

No.	RAM Address Instruction
24	DY

DY (Decrement Y)

Format

DY

Status

NB

Y - 1 < 0 ST=0

Y - 1 ≧ 0 ST=1

Operation

Y - 1 → Y

Description

Decrements the contents of Y register.

Judges NB.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
	DY		$0DF		1	1

Example

```
00744                    *
00745    0DF     025E            DY
00746    064     025F            RED
00747    05C     0260            IY
00748                    *
```

No.	Arithmetic Instruction
25	EORM

EORM (EOR Memory with A)

Format

EORM

Status

NZ

A=0 ST=0

A≠0 ST=1

Operation

$A \oplus M \rightarrow A$

Description

Performs the logical exclusive "OR" operation between the contents of Accumulator and those of RAM addressed by W, X, Y registers.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
REGISTER	EORM		$01C		1	1

Example

```
00921                       *
00922    091        02E2    KIARTHM3   LAMX
00923    01C        02E3               EORM
00924    051        02E4               LMAIYX
```

No.	Arithmetic Instruction
26	EORMD

EORMD (EOR Memory with A)

Format

EORMD d

Status

NZ

A=0 ST=0

A≠0 ST=1

Operation

$A \oplus M(d) \rightarrow A$

Description

Performs the logical exclusive "OR" operation between the contents of Accumulator and those of RAM addressed by 10-bit direct address d_{9-0}.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
DIRECT	EORMD	d	$11C	d	2	2

Example

```
00932    091        02EC          LAMX
00933    11C 04B    02ED          EORMD     $04B
00934    051        02EF          LMAIYX
```

No.	Arithmetic Instruction
27	IB

IB (Increment B)

Format

IB

Status

NZ

B + 1 = 0 ST=0

B + 1 ≠ 0 ST=1

Operation

B + 1 → B

Description

Increments the contents of B register.

Judges NZ.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
	IB		$04C		1	1

Example

```
01142                    *
01143    04C    0397             IB                        ;B=B+1
01144    1CE    0398             CAL       OSERROR
```

No.	Compare Instruction
28	ILEM

ILEM (Immediate Less or Equal to Memory)

Format

ILEM i

Status

NB

i > M ST=0

i ≦ M ST=1

Operation

i ≦ M

Description

Compares the contents of RAM addressed by W, X, Y registers to 4-bit immediate data i_{3-0}.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
REGISTER	ILEM	i	$03i		1	1

Example

```
01210                          *
01211    034       03D2              ILEM      $4
01212    1CE       03D3              CAL       OSERROR
```

No.	Compare Instruction
29	ILEMD

ILEMD (Immediate Less or Equal to Memory)

Format

ILEMD i, d

Status

NB

i > M ST=0

i ≦ M ST=1

Operation

i ≦ M(d)

Description

Compares the contents of RAM addressed by 10-bit direct address d_{9-0} to 4-bit immediate data $i_{3\sim0}$.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
DIRECT	ILEMD	i,d	$13i	d	2	2

Example

```
01820                          *
01821    133 030    060E               ILEMD    $3,$030
01822    171 160    0610               BRL      INITD       ;IF M(030)≧3
01823    132 030    0612               ILEMD    $2,$030
01824    171 152    0614               BRL      PROGCX      ;IF M(030)=2
01825    131 030    0616               ILEMD    $1,$030
01826    171 140    0618               BRL      PROGB       ;IF M(030)=1
01827    130 030    061A               ILEMD    $0,$030
01828    171 12E    061C               BRL      PROGA       ;IF M(030)=0
01829    151 100    061E               JMPL     MAIN
```

No.	Compare Instruction
30	INEM

INEM (Immediate Not Equal to Memory)

Format

INEM i

Status

NZ

i = M ST=0

i ≠ M ST=1

Operation

i ≠ M

Description

Compares the contents of RAM addressed by W, X, Y registers to 4-bit immediate data.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
REGISTER	INEM	i	$02i		1	1

Example

```
00700                       *
00701    201      023E      KIZMN     LBI      $1
00702    02A      023F                INEM     $A
00703    342      0240                BR       KIZM1
```

No.	Compare Instruction
31	INEMD

INEMD (Immediate Not Equal to Memory)

Format

INEMD i,d

Status

NZ

i = M ST=0

i ≠ M ST=1

Operation

i ≠ M(d)

Description

Compares the contents of RAM addressed by 10-bit direct address d_{9-0} to 4-bit immediate data i_{3-0}.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
DIRECT	INEMD	i,d	$12i	d	2	2

Example

```
01893    124 030    0656    TBINTR    INEMD    4,$030
01894    171 25C    0658              BRL      INEXT1
01895    171 104    065A              BRL      MBACK0
```

No.	RAM Address Instruction
32	IY

IY (Increment Y)

Format

IY

Status

NZ

Y + 1 = 0 ST=0

Y + 1 ≠ 0 ST=1

Operation

Y + 1 → Y

Description

Increments the contents of Y register.

Judges NZ.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
	IY		$05C		1	1

Example

```
00848    002      02AB              XSFY
00849    05C      02AC              IY
00850    3A3      02AD              BR        KINAME0
00851                       *
```

No.	ROM Address Instruction
33	JMPL

JMPL (Long Jump Unconditionally)

Format

JMPL u

Status

No effect

Operation

Unconditional jump to any ROM address space.

Description

All bits of PC are replaced with 14-bit direct address P_{3-0}, d_{9-0}.

OP-code is as follows according to ROM capacity.

ROM 2k	$P_3=P_2=P_1=0$	ROM 8k	$P_3=0$
ROM 4k	$P_3=P_2=0$		

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
DIRECT	JMPL	u	$15p	d	2	2

Example

```
01611    1A0 030    052E    PROGA    LMID    0,$030
01612    1A3 032    0530             LMID    $3,$032
01613    1A0 035    0532             LMID    $0,$035
01614    1AD 036    0534             LMID    $D,$036
01615    1A3 037    0536             LMID    $3,$037
01616    1AE 038    0538             LMID    $E,$038
01617    1AF 039    053A             LMID    $F,$039
01618    1A8 03A    053C             LMID    $8,$03A
01619    151 18B    053E             JMPL    PROG
```

LAB (Load A from B)

Format

LAB

Status

No effect

Operation

B → A

Description

Transfers the contents of B register to Accumulator.

The contents of B register are unchanged.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
	LAB		$048		1	1

Example

```
01839                    *
01840     048      0629          LAB
01841     194 035  062A          LMAD      $035
01842                    *
```

No.	Immediate Instruction
35	LAI

LAI (Load A from Immediate)

Format

LAI i

Status

No effect

Operation

i → A

Description

Transfers 4-bit immediate data i_{3-0} to Accumulator.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
	LAI	i	$23i		1	1

Example

```
01152                    *
01153    230    039F             LAI     $0          ;A=0
01154    210    03A0             LYI     $0          ;Y=0
01155                    *
```

No.	RAM Register Instruction
36	LAM

LAM (Load A from Memory)

Format

LAM (XY)

Status

No effect

Operation

M→A,(X↔SPX, Y↔SPY)

Description

Transfers the contents of RAM addressed by W, X, Y registers to Accumulator.
The contents of RAM is unchanged.
(When executing M A, exchanges x register for SPX register, and Y register for SPY register according to the contents of x, y in OP-code.)

MNEMONIC	y	x	FUNCTION
LAM	0	0	—
LAMX	0	1	X↔SPX
LAMY	1	0	Y↔SPY
LAMXY	1	1	X↔SPX, Y↔SPY

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
REGISTER	LAM(XY)		%01-1001 -00yx		1	1

Example

```
00941    091        02F6    KIARTHM5    LAMX
00942    198 04D    02F7                SMCD      $04D
00943    094        02F9                LMA
```

No.	RAM Register Instruction
37	LAMD

LAMD (Load A from Memory)

Format

LAMD d

Status

No effect

Operation

M(d) → A

Description

Transfers the contents of RAM addressed by 10-bit direct address to Accumulator.

The contents of RAM is unchanged.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
DIRECT	LAMD	d	$190	d	2	2

Example

```
01716    190 032    0592    PNEXT1    LAMD    $032
01717    281        0594              A1      1
01718    194 032    0595              LMAD    $032
```

No.	Register-to-Register Instruction
38	LAMR

LAMR (Load A from MR)

Format

LAMR m

Status

No effect

Operation

MR(m) → A

Description

Transfers the contents of Memory Register (MR) addressed by 4-bit direct address m_{3-0} to Accumulator.
(MR, second file of RAM, can be selected (16-digits) by $m_{3 \sim 0}$.)

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
DIRECT (4 bits)	LAMR	m	$27m		1	1

Example

```
01532                        *
01533   276     04E2         LAMR    $6
01534   281     04E3         AI      $1
01535   203     04E4         LBI     $3
01536   1B1     04E5         P       $1
```

No.	Input/Output Instruction
39	LAR

LAR (Load A from R-Port Register)

Format

LAR m

Status

No effect

Operation

R(m) → A

Description

Transfers the contents of R-Port addressed by 4-bit direct address to Accumulator.
The contents of Port Register are unchanged.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
DIRECT (4 bits)	LAR	m	$25m		1	1

Example

```
00980                          *
00981    253          0314     KIINPUTO  LAR       $3
00982    180 02D      0315               XMAD      $02D
```

No.	Register-to-Register Instruction
40	LASPX

LASPX (Load A from SPX)

Format

LASPX

Status

No effect

Operation

SPX → A

Description

Transfers the contents of SPX register to Accumulator.
The contents of SPX register are unchanged.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
	LASPX		$068		1	1

Example

```
00835   068     029F        LASPX               ;A=6
00836   281     02A0        AI      $1          ;A=A+1 A=7
00837   0E8     02A1        LXA                 ;X=A   X=7
```

No.	Register-to-Register Instruction
41	LASPY

LASPY (Load A from SPY)

Format

LASPY

Status

No effect

Operation

SPY → A

Description

Transfers the contents of SPY register to Accumulator.
The contents of SPY register are unchanged.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
	LASPY		$058		1	1

Example

```
01015    002     0336          XSPY
01016    058     0337          LASPY
01017    2B3     0338          ALEI      $3
```

No.	Register-to-Register Instruction
42	LAY

LAY (Load A from Y)

Format

LAY

Status

No effect

Operation

Y → A

Description

Transfers the contents of Y register to Accumulator.

The contents of Y register is unchanged.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
	LAY		$0AF		1	1

Example

```
00585    0AF        01E5          LAY                    ;A=Y
00586    281        01E6          AI        $1           ;A=A+1
00587    108 02F    01E7          AMD       $02F         ;A=A+M(02F)
```

No.	Register-to-Register Instruction
43	LBA

LBA (Load B from A)

Format

LBA

Status

No effect

Operation

A → B

Description

Transfers the contents of Accumulator to B register.

The contents of Accumulator is unchanged.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
	LBA		$0C8		1	1

Example

```
00839                          *
00840    234       02A3    KINAME0   LA1       $4          ;A=4
00841    0C8       02A4              LBA                   ;B=A B=4
```

No.	Immediate Instruction
44	LBI

LBI (Load B from Immediate)

Format

LBI i

Status

No effect

Operation

i → B

Description

Transfers 4-bit immediate data i_{3-0} to B register.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
	LBI	i	$20i		1	1

Example

```
00888                    *
00889    0AF     02C4            LAY
00890    206     02C5            LBI     $6
```

No.	RAM Register Instruction
45	LBM

LBM (Load B from Memory)

Format

LBM (XY)

Status

No effect

Operation

M → B, (X ↔ SPX, Y ↔ SPY)

Description

Transfers the contents of RAM addressed by W, X, Y registers to B register.
The contents of RAM is unchanged.
During executing the above, executes the followings according to the value of x, y in OP-code.

MNEMONIC	y	x	FUNCTION
LBM	0	0	——
LBMX	0	1	X ↔ SPX
LBMY	1	0	Y ↔ SPY
LBMXY	1	1	X ↔ SPX, Y ↔ SPY

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
REGISTER	LBM(XY)		%00-0100 -00yx		1	1

Example

```
00809   0C0     028E    KIRAMS  XMB             ;SAVE B TO M(WXY)
00810   200     028F            LBI     $0      ;B=0
00811   040     0290            LBM             ;B=M(WXY)
00812   0DF     0291            DY              ;Y=Y-1
00813   38E     0292            BR      KIRAMS  ;IF Y>=0 THEN BR
00814   010     0293            RTN
```

No.	Input/Output Instruction
46	LBR

LBR (Load B from R-Port Register)

Format

LBR m

Status

No effect

Operation

R(m) → B

Description

Transfers the contents of R-Port addressed by 4-bit direct address to B register.
The contents of Port Register is unchanged.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
DIRECT (4 bits)	LBR	m	$24m		1	1

Example

```
LBR     $3          ;B←R(3)
LMB                 ;M(WXY)← B
```

No.	RAM Register Instruction
47	LMA

LMA (Load Memory from A)

Format

LMA (XY)

Status

No effect

Operation

A → M (X ↔ SPX , Y ↔ SPY)

Description

Transfers the contents of Accumulator to RAM addressed by W, X, Y registers.

The contents of Accumulator is unchanged.

MNEMONIC	y	x	FUNCTION
LMA	0	0	—
LMAX	0	1	X ↔ SPX
LMAY	1	0	Y ↔ SPY
LMAXY	1	1	X ↔ SPX, Y ↔ SPY

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
REGISTER	LMA(XY)		%00-1001 -01yx		1	1

Example

```
        LWI     $0
        LXI     $4
        XSPX
        LXI     $3
        LYI     $0
        LAI     $5
        LMAX                    ;M(030) ← $5
        LAI     $A
        LMAX                    ;M(040) ← $A
```

No.	RAM Register Instruction
48	LMAD

LMAD (Load Memory from A)

Format

LMAD d

Status

No effect

Operation

A→M(d)

Description

Transfers the contents of Accumulator to RAM addressed to 10-bit direct address.

The contents of Accumulator is unchanged.

Address format and the number of cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
DIRECT	LMAD	d	$194	d	2	2

Example

```
01901    230        0663          LAI      0
01902    118 034    0664          AMCD     $034
01903    194 034    0666          LMAD     $034
```

No.	RAM Register Instruction
49	LMADY

LMADY (Load Memory from A, Decrement Y)

Format

LMADY (X)

Status

NB

Y - 1 < 0 ST=0

Y - 1 ≧ 0 ST=1

Operation

Y - 1→Y (X↔SPX), A→M

Description

Transfers the contents of Accumulator to RAM addressed by W, X, and Y registers.
The contents of Accumulator is unchanged.
Decrements the contents of Y register and judges NB.
(During executing the above, executes the following operation according to the value of x in OP-code.

MNEMONIC	x	FUNCTION
LMADY	0	—
LMADYX	1	X ←→ SPX

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
REGISTER	LMADY(X)		%00-1101 -000x		1	1

Example

```
01048                        *
01049    230       0354              LAI       $0
01050    0D0       0355    LPE1      LMADY
01051    355       0356              BR        LPE1
```

No.	RAM Register Instruction
50	LMAIY

LMAIY (Load Memory from A, Increment Y)

Format

LMAIY (X)

Status

NZ

Y + 1 = 0 ST=0

Y + 1 ≠ 0 ST=1

Operation

Y + 1→Y (X↔SPX), A→M

Description

Transfers the contents of Accumulator to RAM addressed by W, X, and Y registers.
The contents of Accumulator is unchanged.
Increments the contents of Y and judges NZ.
(During executing the above, executes the following operation according according to the value of x in OP-code.

MNEMONIC	x	FUNCTION
LMAIY	0	—
LMAIYX	1	X ↔ SPX

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
REGISTER	LMAIY(X)		%00-0101 -000x		1	1

Example

```
00909    051        02D6          LMAIYX
00910    091        02D7          LAMX
00911    118 044    02D8          AMCD      $044
00912    0A6        02DA          DAA
00913    051        02DB          LMAIYX
```

No.	Immediate Instruction
51	LMID

LMID (Load Memory from Immediate)

Format

LMID i,d

Status

No effect

Operation

$i \rightarrow M(d)$

Description

Transfers 4-bit immediate data i_{3-0} to RAM addressed by 10-bit direct address d_{9-0}.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
DIRECT	LMID	i,d	$1Ai	d	2	2

Example

```
01492    1A8 004    04B9          LMID     $8,$004
01493    1AF 009    04BB          LMID     $F,$009     ;TMB EXT INPUT
01494    1AE 00A    04BD          LMID     $E,$00A     ;TCBL=E
01495    1AF 00B    04BF          LMID     $F,$00B     ;TLRU=F
01496                    *
```

No.	Immediate Instruction
52	LMIIY

LMIIY (Load Memory from Immediate, Increment Y)

Format

LMIIY i

Status

NZ

Y + 1 = 0 ST=0

Y + 1 ≠ 0 ST=1

Operation

i → M

Y + 1 → Y

Description

Transfers 4-bit immediate data i_{3-0} to RAM addressed by W, X, and Y registers.

Increments the contents of Y register and judges NZ.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
REGISTER	LMIIY	i	$29i		1	1

Example

```
00961                   *
00962   290     0309    KIARTHM8    LMIIY       $0
00963   07A     030A                YNEI        $A
00964   309     030B                BRS         KIARTHM8
00965                   *
```

No.	Input/Output Instruction
53	LRA

LRA (Load R-Port Register from A)

Format

LRA m

Status

No effect

Operation

A → R(m)

Description

Transfers the contents of Accumulator to R-Port register addressed by 4-bit direct address m_{3-0}.
The contents of Accumulator is unchanged.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
DIRECT (4 bits)	LRA	m	$2Dm		1	1

Example

```
00675                          *
00676     000          022D    KIOUTC4    NOP
00677     160 255      022E               CALL       KIZMC
00678     2D1          0230               LRA        $1
00679     2C2          0231               LRB        $2
00680     338          0232               BR         KIOUTC9
00681                          *
```

No.	Input/Output Instruction
54	LRB

LRB (Load R-Port Register from B)

Format

LRB m

Status

No effect

Operation

B → R(m)

Description

Transfers the contents of B register to R-Port register addressed by 4-bit direct address m_{3-0}.
The contents of B register is unchanged.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
DIRECT (4 bits)	LRB	m	$2Cm		1	1

Example

```
00664                      *
00665     000        0222  KIOUTC0   NOP
00666     160 255    0223            CALL      KIZMC
00667     2D2        0225            LRA       $2
00668     2C4        0226            LRB       $4
00669     338        0227            BR        KIOUTC9
00670                      *
```

No.	RAM Address Instruction
55	LWI

LWI (Load W from Immediate)

Format

LWI i

Status

No effect

Operation

$i \rightarrow W$

Description

Transfers 2-bit immediate data i_{1-0} to W register.
Specifies RAM file No. with X register.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
	LWI	i	%00-1111 -00$i_1 i_0$		1	1

Example

```
01690    222     057E   RAMINIT   LXI     2
01691    0F0     057F              LWI     0
01692    210     0580              LYI     0
01693    290     0581   RLOOP      LMIIY   0
```

No.	RAM Address Instruction
56	LXA

LXA (Load X from A)

Format

LXA

Status

No effect

Operation

A → X

Description

Transfers the contents of Accumulator to X register.
The contents of Accumulator is unchanged.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
	LXA		$0E8		1	1

Example

```
01696    068    0584          LASPX
01697    281    0585          AI      1
01698    0E8    0586          LXA
```

No.	RAM Address Instruction
57	LXI

LXI (Load X from Immediate)

Format

LXI i

Status

No effect

Operation

$i \rightarrow X$

Description

Transfers 4-bit immediate data i_{3-0} to X register.
Specifies RAM file No.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
	LXI	i	$22i		1	1

Example

```
00492    0F0    0192        LWI     $0      ;W=0
00493    224    0193        LXI     $4      ;X=4
00494    21F    0194        LYI     $F      ;Y=F
00495    23F    0195        LAI     $F      ;A=F
00496    281    0196        AI      $1      ;A=A+1
00497    0D0    0197        LMADY           ;M(WXY)=A Y=Y-1
```

No.	RAM Address Instruction
58	LYA

LYA (Load Y from A)

Format

LYA

Status

No effect

Operation

A → Y

Description

Transfers the contents of Accumulator to Y register.
The contents of Accumulator is unchanged.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
	LYA		$0D8		1	1

Example

```
01874   190 032   0645   TIMER   LAMD   $032
01875   0D8       0647           LYA
01876   0E4       0648           SED
```

No.	RAM Address Instruction
59	LYI

LYI (Load Y from Immediate)

Format

LYI i

Status

No effect

Operation

i → Y

Description

Transfers 4-bit immediate data i_{3-0} to Y register.
Specifies RAM digit No. and address of discrete I/O.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
	LYI	i	$21i		1	1

Example

```
00218    0F0     00C4      LWI     $0
00219    220     00C5      LXI     $0
00220    210     00C6      LYI     $0
00221    002     00C7      XSPY
00222    218     00C8      LYI     $8
```

No.	Arithmetic Instruction
60	NEGA

NEGA (Negate A)

Format

NEGA

Status

No effect

Operation

$\overline{A} + 1 \rightarrow A$

Description

Takes 2's complement of the contents of Accumulator and stores the result in Accumulator.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
	NEGA		$060		1	1

Example

```
00795                    *
00796    04C     0282    KIRAM7   IB          ;B=B+1
00797    048     0283             LAB         ;A=B
00798    060     0284             NEGA        ;A=-A
```

No.	Control Instruction
61	NOP

NOP (No Operation)

Format

NOP

Status

No effect

Operation

Updates the program counter only and has no effect on the other registers.

Description

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
	NOP		$000		1	1

Example

```
01549   200   04EA   KIDLYON   LBI    $0
01550   04C   04EB             IB
01551   000   04EC             NOP
01552   000   04ED             NOP
01553   000   04EE             NOP
01554   000   04EF             NOP
01555   000   04F0             NOP
01556   3EB   04F1             BR     *-6
01557   010   04F2             RTN
```

No.	Arithmetic Instruction
62	OR

OR (OR A and B)

Format

OR

Status

No effect

Operation

AUB → A

Description

Performs logical OR between the contents of Accumulator and those of B register, and stores the result in Accumulator.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
	OR		$144		1	1

Example

```
01136                        *
01137    20A       0392      LBI     $A      ;B=1010
01138    235       0393      LAI     $5      ;A=0101
01139    144       0394      OR              ;A=AUB A=1111
01140    0C8       0395      LBA             ;B=F
```

No.	Arithmetic Instruction
63	ORM

ORM (OR Memory with A)

Format

ORM

Status

NZ

AUM = 0 ST=0

AUM ≠ 0 ST=1

Operation

AUM → A

Description

Performs logical OR between the contents of Accumulator and those of RAM addressed by W, X, and Y registers, and stores the result in Accumulator.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
REGISTER	ORM		$00C		1	1

Example

```
01253   21F     03F6        LYI     $F
01254   00C     03F7        ORM
01255   1CE     03F8        CAL     OSERROR
```

ORMD (OR Memory with A)

Format

ORMD d

Status

NZ

AUM = 0 ST=0

AUM ≠ 0 ST=1

Operation

AUM(d) → A

Description

Performs logical OR between the contents of Accumulator and those of 10-bit direct address d_{9-0}, and stores the result in Accumulator.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
DIRECT	ORMD	d	$10C	d	2	2

Example

```
01256    10C 07F    03F9          ORMD     $07F
01257    1CE        03FB          CAL      OSERROR
01258                  *
```

No.	Input/Output Instruction
65	P

P (Pattern Generation)

Format

P p

Status

No effect

Operation

R_9,R_8	$R_7 \sim R_0$	Destination
01	ROM pattern	to A,B registers
10	ROM pattern	to R_1,R_2 registers
11	ROM pattern	to A,B and R_1,R_2
00	ROM pattern	—

Description

Loads ROM bit pattern addressed by PC of which the contents are replaced with Accumulator, B register, and 4-bit register into Port register R_1, R_2, or Accumulator, B register.

Replaced PC

PC_{13}													PC_0
0	0	P_3	P_2	P_1	P_0	B_3	B_2	B_1	B_0	A_3	A_2	A_1	A_0

R_9	R_8	R_7	R_6	R_5	R_4	R_3	R_2	R_1	R_0	ROM pattern
*	1	B_3	B_2	B_1	B_0	A_3	A_2	A_1	A_0	Loaded into Accumulator, B register
1	*	R_{33}	R_{32}	R_{31}	R_{30}	R_{23}	R_{22}	R_{21}	R_{20}	Loaded into R_1,R_2 registers

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
REGISTER + DIRECT(4 bits)	P	p	$1Bp		1	2

Example

```
01782                    *
01783   236      05E2          LAI     6
01784   20E      05E3          LBI     $E
01785   1B7      05E4          P       7
```

No.	Arithmetic Instruction
66	REC

REC (Reset Carry)

Format

REC

Status

No effect

Operation

0 → CA

Description

Resets Carry.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
	REC		$0EC		1	1

Example

```
01896    000          065C    INEXT1    NOP
01897    231          065D              LAI       1
01898    0EC          065E              REC
01899    118 033      065F              AMCD      $033
01900    194 033      0661              LMAD      $033
```

No.	Input/Output Instruction
67	RED

RED (Reset Discrete I/O Latch)

Format

RED

Status

No effect

Operation

$0 \rightarrow D(Y)$

Description

Resets discrete I/O latch addressed by Y register.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
REGISTER	RED		$064		1	1

Example

```
00223                    *
00224  064   00C9  RMLOOP  RED
00225  002   00CA         XSPY
00226  05C   00CB         IY
00227  002   00CC         XSPY
00228  05C   00CD         IY
00229  3D0   00CE         BR      *+2
00230  002   00CF         XSPY
00231  042   00D0         LBMY
00232  092   00D1         LAMY
00233                    *
00234  2D1   00D2         LRA     $1
00235  2C2   00D3         LRB     $2
00236  0E4   00D4         SED
00237                    *
```

REDD (Reset Discrete I/O Latch Direct)

Format

REDD m

Status

No effect

Operation

$0 \rightarrow D(m)$

Description

Resets discrete I/O latch addressed by 4-bit direct address m_{3-0}.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
DIRECT (4 bits)	REDD	m	$26m		1	1

Example

```
01240    1CE    03EB          CAL     OSERROR
01241    262    03EC          REDD    $2
```

No.	RAM Bit Manipulation Instruction
69	REM

REM (Reset Memory Bit)

Format

REM n

Status

No effect

Operation

$0 \rightarrow M(n)$

Description

Resets the bit specified by n_{1-0} of RAM addressed by W, X, and Y registers.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
REGISTER	REM	n	%00-1000 -10 n_1 n_0		1	1

Example

```
01034    094    0347        LMA
01035    08B    0348        REM     3
01036    08A    0349        REM     2
```

No.	RAM Bit Manipulation Instruction
70	REMD

REMD (Reset Memory Bit)

Format

REMD n,d

Status

No effect

Operation

$0 \rightarrow M(d,n)$

Description

Resets the bit specified by n_{1-0} of RAM addressed by 10-bit direct address d_{9-0}.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
DIRECT	REMD	n,d	%01-1000 -10 n_1 n_0	d	2	2

Example

```
01497    185 001    04C1          SEMD    1,$001    ;SET IM1
01498    188 002    04C3          REMD    0,$002    ;RESET IFTB
01499    189 002    04C5          REMD    1,$002    ;RESET IMTB
01500    184 000    04C7          SEMD    0,$000    ;SET I/E
01501                      *
```

No.	Arithmetic Instruction
71	ROTL

ROTL (Rotate Left A with Carry)

Format

ROTL

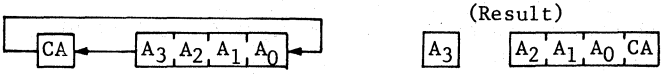
Status

No effect

Operation

Rotates the contents of Accumulator with Carry (CA) to the left by 1 bit.

Description



Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
	ROTL		$0A1		1	1

Example

```
00588    0EC    01E9          REC
00589    0A1    01EA          ROTL
00590    200    01EB          LBI     $0
00591    0B1    01EC          TBR     $1
00592                *
```

ROTR (Rotate Right A with Carry)

Format

ROTR

Status

No effect

Operation

Rotates the contents of Accumulator with Carry (CA) to the right by 1 bit.

Description

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
	ROTR		$0A0		1	1

Example

```
01431                        *
01432   058      0489    KISCANT   LASPY
01433   0A0      048A              ROTR
01434   06F      048B              TC
01435   390      048C              BR        KISCANR
```

No.	ROM Address Instruction
73	RTN

RTN (Return from Subroutine)

Format

RTN

Status

No effect

Operation

Retures from subroutine.

Description

Returnes the contents of PC saved in RAM (stack area) when occuring subroutine instruction or interrupt to PC.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
	RTN		$010		1	3

Example

```
00593                          *
00594     20C       01ED       KIDLYKEY   LBI       $R
00595     04C       01EE                  IB
00596     3EE       01EF                  BR        *-1
00597     010       01F0                  RTN
```

No.	ROM Address Instruction
74	RTNI

RTNI (Return from Interrupt)

Format

RTNI

Status

Restores the contents of Status saved before.

Operation

1 → I/E return from subroutine

Description

This is the return instruction (RTN) accompanied by I/E set instruction.
Restores Carry and Status simultaneously.

Address format and the number of cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
	RTNI		$011		1	3

Example

```
01478    2FF       04B1   KITMRTN   XMRA     $F
01479    18A 001   04B2             REMD     2,$001
01480    011       04B4             RTNI
```

No.	Control Instruction
75	SBY

SBY (Stand-by Mode)

Format

SBY

Status

No effect

Operation

Brings to Stand-by mode.

Description

The SBY instruction puts the MCU into the Standby mode. In the Standby mode, the oscillator circuit is active and timer/counter and serial interface continue working. On the other hand, the CPU stops since the clock related to the instruction execution stops. Registers, RAM and Input/Output pins retain the state they had just before going into the Standby mode. The Stand-by mode is released by RESET input or CPU interrupt. If I/E=1, enters into interrupt sequence and if I/E=0, executes the instruction next to SBY without executing interrupt process.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
	SBY		$14C		1	1

Example

```
REM     0
IY
SBY

LAR     $1
LMAIY
LAR     $1
LMAIY
```

No.	Arithmetic Instruction
76	SEC

SEC (Set Carry)

Format

SEC

Status

No effect

Operation

1 → CA

Description

Sets Carry.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
	SEC		$0EF		1	1

Example

```
SEC
SMCD                    ;M-A-CA(0)→A
```

No.	Input/Output Instruction
77	SED

SED (Set Discrete I/O Latch)

Format

SED

Status

No effect

Operation

1 → D(Y)

Description

Sets discrete I/O addressed by Y register.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
REGISTER	SED		$0E4		1	1

Example

```
        LYI     $0
        SED                 ;D(0)=1
        IY
        SED                 ;D(1)=1
        IY
        SED                 ;D(2)=1
        IY
        SED                 ;D(3)=1
```

No.	Input/Output Instruction
78	SEDD

SEDD (Set Discrete I/O Latch Direct)

Format

SEDD m

Status

No effect

Operation

1 → D(m)

Description

Sets discrete I/O addressed by 4-bit direct address m_{3-0}.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
DIRECT (4 bits)	SEDD	m	$2Em		1	1

Example

```
01101                        *
01102    2E0      037A   KIDINCLR  SEDD    $0        ;D(0)=1
01103    2E1      037B             SEDD    $1        ;D(1)=1
01104    2E2      037C             SEDD    $2        ;D(2)=1
01105    2E3      037D             SEDD    $3        ;D(3)=1
01106                        *
```

No.	RAM Bit Manipulation Instruction
79	SEM

SEM (Set Memory Bit)

Format

SEM n

Status

No effect

Operation

$1 \rightarrow M(n)$

Description

Sets the bit specified by n_{1-0} of RAM addressed by W, X, and Y registers.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
REGISTER	SEM	n	%00-1000 -01 n_1 n_0		1	1

Example

```
01319                    *
01320   220      0428          LXI     $0
01321   08A      0429          REM     2
01322   08B      042A          REM     3
01323   084      042B          BEM     0
01324                    *
```

No.	RAM Bit Manipulation Instruction
80	SEMD

SEMD (Set Memory Bit)

Format

SEMD n,d

Status

No effect

Operation

$1 \rightarrow M(d,n)$

Description

Sets the bit specified by n_{1-0} of RAM addressed by 10-bit direct address d_{9-0}.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
DIRECT	SEMD	n,d	%01-1000 -01 n_1 n_0	d	2	2

Example

```
01062    187 043    035E              SEMD     $3,$043
01063    363        0360              BR       LPE3
01064    18B 043    0361              REMD     $3,$043
01065    000        0363    LPE3      NOP
01066                       *
```

No.	Arithmetic Instruction
81	SMC

SMC (Subtract A from Memory with Carry)

Format

SMC

Status

NB of (M-A-$\overline{CA}$)

M - A - $\overline{CA}$ < 0 ST=0

M - A - $\overline{CA}$ ≧ 0 ST=1

Operation

M - A - $\overline{CA}$ → A, NB → CA

Description

Subtracts the contents of Accumulator and $\overline{CA}$ from the contents of RAM addressed by W, X, and Y registers and stores the result in Accumulator.

Latches NB into CA and judges it.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
REGISTER	SMC		$098		1	1

Example

```
01170   05C     03AD    IY                      ;Y=Y+1
01171   098     03AE    SMC                     ;A=M(WXY)-A-CA
01172   1CE     03AF    CAL     OSERROR
01173   05C     03B0    IY                      ;Y=Y+1
01174   008     03B1    AM                      ;A=M(WXY)+A
```

No.	Arithmetic Instruction
82	SMCD

SMCD (Subtract A from Memory with Carry)

Format

SMCD d

Operation

M(d) - A - $\overline{CA}$→A, NB→CA

Status

NB of (M-A-$\overline{CA}$)

M - A - $\overline{CA}$ < 0 ST=0

M - A - $\overline{CA}$ ≧ 0 ST=1

Description

Subtracts the contents of Accumulator and $\overline{CA}$ from the contents of RAM addressed by 10-bit direct address d_{9-0}, and stores the result in Accumulator.
Latches NB into CA and judges it.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
DIRECT	SMCD	d	$198	d	2	2

Example

```
IY                  ;Y=Y+1
SMCD    $035        ;A=M(035)-A-CA
CAL     OSERROR
IY                  ;Y=Y+1
AM                  ;A=M(WXY)+A
```

No.	Control Instruction
83	STOP

STOP (Stop Mode)

Format

STOP

Status

No effect

Operation

Brings to Stop mode.

Description

Brings all operations to halt by stopping oscillation.
The contents of RAM are held.
Stop mode is released with reset and the next operation starts from the reset state.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
	STOP		$14D		1	1

Example

```
01593    134 030    0518    ELSE1    ILEMD    4,$030
01594    171 18B    051A             BRL      PROG
01595    133 030    051C             ILEMD    3,$030
01596    171 161    051E             BRL      PROGD
01597    132 030    0520             ILEMD    2,$030
01598    171 1B6    0522             BRL      PROGC
01599    131 030    0524             ILEMD    1,$030
01600    171 18B    0526             BRL      PROG
                                     STOP
```

(Note) The STS instruction is not available in the HMCS412 and HMCS414.

No.	Control Instruction
84	STS

STS (Start Serial)

Format

STS

Status

No effect

Operation

Serial Start

Description

Starts serial operation.
Outputs serial internal clock.
Enables itself to reset serial counter and input serial clock to serial counter and serial shift register.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
	STS		$148		1	1

Example

```
LMID    $3, $       ;PMR
LMID    $5, $       ;SRL
LMID    $A, $       ;SRU
LMID    $3, #       ;SMR
SEM     $0, $000    ;SET I/E
REM     $0, $003    ;RESET IFS
REM     $1, $003    ;RESET IMS
STS                 ;START SCI
```

No.	RAM Address Instruction
85	SYY

SYY (Subtract A from Y)

Format

SYY

Status

NB of (Y-A)

Y - A < 0 ST=0

Y - A ≧ 0 ST=1

Operation

Y - A → Y

Description

Adds two's complement of the contents of Accumulator to the contents of Y register. (Y+$\overline{A}$+1).

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
	SYY		$0D4		1	1

Example

```
01158    0D4     03A3        SYY                    ;Y=Y-A
01159    1CE     03A4        CAL      OSERROR
```

No.	ROM Address Instruction
86	TBR

TBR (Table Branch)

Format

TBR p

Status

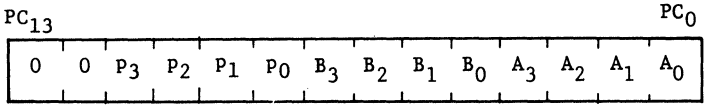
No effect

Operation

Unconditional branch with Table.

Description

PC_{11-0} are modified by Accumulator, B register, and 4-bit direct address P_{3-0}. That is, performs unconditional branch with the data of Accumulator, B register, and 4-bit direct address. PC_{13-12} are 0.



Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
REGISTER + DIRECT (4 bits)	TBR	p	$0B_p$		1	1

Example

```
LAI     $5
LBI     $7
TBR     $3      ;JUMP TO Address ($0375)
```

No.	Arithmetic Instruction
87	TC

TC (Test Carry)

Format

TC

Status

CA	CA=0	ST=0
	CA=1	ST=1

Operation

Tests the contents of CA (Carry).

Description

The contents of CA remains unchanged.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
	TC		$06F		1	1

Example

```
00244                        *
00245    25A      00DB       LAR      $A
00246    0A0      00DC       ROTR
00247    0A0      00DD       ROTR
00248    06F      00DE       TC
00249    0F3      00DF       LWI      $3
00250    3E2      00E0       BR       *+2
```

No.	Input/Output Instruction
88	TD

TD (Test Discrete I/O Latch)

Format

TD

Status

D (Y) D(Y)=0 ST=0
D(Y)=1 ST=1

Operation

Tests D(Y).

Description

Trsts discrete I/O addressed by Y register.
The contents of discrete I/O latch.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
REGISTER	TD		$0E0		1	1

Example

```
LYI     $12
TD                      ;TEST D(12)
BRL     OSERROR         ;IF ST=1
```

No.	Input/Output Instruction
89	TDD

TDD (Test Discrete I/O Latch Direct)

Format

TDD m

Status

D(m) D(m)=0 ST=0
D(m)=1 ST=1

Operation

Tests D(m).

Description

Tests discrete I/O addressed by 4-bit direct address m_{3-0}.
The contents of discrete I/O latch.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
DIRECT (4 bits)	TDD	m	$2Am		1	1

Example

```
01581   2A3         0504   MBACK0   TDD    3
01582   171 109     0505            BRL    NEXT1
01583   171 12E     0507            BRL    PROGA
01584   2A2         0509   NEXT1    TDD    2
01585   171 10E     050A            BRL    NEXT2
01586   171 140     050C            BRL    PROGB
01587   2A1         050E   NEXT2    TDD    1
01588   171 113     050F            BRL    NEXT3
01589   171 152     0511            BRL    PROSCX
01590   2A0         0513   NEXT3    TDD    0
01591   171 118     0514            BRL    ELSE1
01592   171 160     0516            BRL    INITD
```

No.	RAM Bit Manipulation Instruction
90	TM

TM (Test Memory Bit)

Format

TM n

Status

M(n)	M(n)=0	ST=0
	M(n)=1	ST=1

Operation

Tests M(n).

Description

Tests the bit specified by n_{1-0} of RAM addressed by W, X, and Y registers.

The contents of RAM remains unchaned.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
REGISTER	TM	n	%00-1000 -11n_1n_0		1	1

Example

```
01226                    *
01227    21E     03E0                LYI     $E
01228                    *
01229    08F     03E1                TM      3
01230    1CE     03E2                CAL     OBERROR
```

No.	RAM Bit Manipulation Instruction
91	TMD

TMD (Test Memory Bit)

Format

TMD n,d

Status

M(n)	M(n)=0	ST=0
	M(n)=1	ST=1

Operation

Tests M(d,n)

Description

Tests the state of bit specified by n_{1-0} of RAM addressed by W, X, and Y registers.

The contents of RAM remains unchanged.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
DIRECT	TMD	n,d	%01-1000 -11 n_1 n_0	d	2	2

Example

```
01512                          *
01513    253          04D2     LAR      $3
01514    2FC          04D3     XMRA     $C
01515    18E 02C      04D4     TMD      2,$02C
01516    3D8          04D6     BR       KICNTNXT
01517    3CA          04D7     BR       KICNTDSP
```

XMA (Exchange Memory and A)

Format

XMA (XY)

Status

No effect

Operation

(X ↔ SPX, Y ↔ SPY), M ↔ A

Description

Exchanges the contents of RAM addressed by W, X, and Y registers with those of Accumulator.
(During executing the above, executes the followings according to the value of x and y in OP-code.

MNEMONIC	y	x	FUNCTION
XMA	0	0	——
XMAX	0	1	X ↔ SPX
XMAY	1	0	Y ↔ SPY
XMAXY	1	1	X ↔ SPX, Y ↔ SPY

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
REGISTER	XMA(XY)		%00-1000 -00yx		1	1

Example

```
LWI     $0
LXI     $4
LYI     $5
XSPY
LYI     $0
LAI     $5
XMAY                ;M(040) ↔ A, Y=5
XMA                 ;M(045) ↔ A
```

No.	RAM·Register Instruction
93	XMAD

XMAD (Exchange Memory and A)

Format

XMAD d

Status

No effect

Operation

M(d) → A

Description

Exchanges the contents of RAM addressed by 10-bit direct address d_{9-0} with those of Accumulator.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
DIRECT	XMAD	d	$180	d	2	2

Example

```
01067    253        0364    KIINPUT2  LAR     $3
01068    180 02D    0365              XMAD    $02D
01069    23C        0367              LAI     $C
01070    19C 02D    0368              ANMD    $02D
01071    364        036A              BRS     KIINPUT2
01072                       *
01073    010        036B              RTN
```

XMB (Exchange Memory and B)

Format

XMB (XY)

Status

No effect

Operation

(X ↔ SPX, Y ↔ SPY), M ↔ B

Description

Exchanges the contents of RAM addressed by W, X, and Y registers with those of B register.
(During executing the above, executes the followings according to the value of x and y in OP-code.

MNEMONIC	y	x	FUNCTION
XMB	0	0	——
XMBX	0	1	X ↔ SPX
XMBY	1	0	Y ↔ SPY
XMBXY	1	1	X ↔ SPX, Y ↔ SPY

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
REGISTER	XMB(XY)		%00-1100 -00yx		1	1

Example

```
01129                          *
01130    0AF       038C        LAY             ;A=0
01131    207       038D        LBI     $7      ;B=7
01132    1B1       038E        P       $1      ;B,A=ROM(17Y)
01133    0C1       038F        XMBX            ;M(WXY)=B X↔SPX
01134    051       0390        LMAIYX          ;M(WXY)=A Y=Y+1 X↔SPX
01135    38C       0391        BR      *-5
```

No.	Register-to-Register Instruction
95	XMRA

XMRA (Exchange MR and A)

Format

XMRA m

Status

No effect

Operation

MR(m) ↔ A

Description

Exchanges the contents of Memory Register (MR) in RAM with those of Accumulator.
16-digits (2 files) are MR, and 4-bit direct address m_{3-0} can select any digit.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
DIRECT (4 bits)	XMRA	m	$2Fm		1	1

Example

```
01057                    *
01058    230     033B            LAI     $0
01059    2F3     035C            XMRA    $3
01060                    *
```

No.	RAM Address Instruction
96	XSPX

XSPX (Exchange Y and SPX)

Format

XSPX

Status

No effect

Operation

X ↔ SPX

Description

Exchanges the contents of X register with those of SPX register.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
	XSPX		$001		1	1

Example

```
01123                       *
01124   226     0387    KISTATUS    LXI     $6      ;X=6
01125   001     0388                XSPX            ;SPX
01126   0F0     0389                LWI     $0      ;W=0
01127   227     038A                LXI     $7      ;SPX=7
01128   210     038B                LYI     $0      ;Y=0
```

No.	RAM Address Instruction
97	XSPXY

XSPXY (Exchange X and SPX, Y and SPY)

Format

XSPXY

Status

No effect

Operation

X ↔ SPX

Y ↔ SPY

Description

Exchanges the contents of X register with those of SPX register, and the contents of Y register with those of SPY register simultaneously.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
	XSPXY		$003		1	1

Example

```
        LXI     $5
        XSPX
        LXI     $0
        LYI     $5
        XSPY
        LYI     $0
        XSPXY                   ;X=5, Y=5
        XSPXY                   ;X=0, Y=0
```

No.	RAM Address Instruction
98	XSPY

XSPY (Exchange Y and SPY)

Format

XSPY

Status

No effect

Operation

Y ↔ SPY

Description

Exchanges the contents of Y register with those of SPY register.

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
	XSPY		$002		1	1

Example

```
00987                    *
00988    225    031D     LXl     $5
00989    001    031E     XSPX            ;X=5
00990    0F0    031F     LWI     $0      ;W=0
00991    224    0320     LXI     $4      ;SPX=4
00992    210    0321     LYI     $0      ;Y=0
00993    002    0322     XSPY
00994    210    0323     LYI     $0      ;SPY=0
```

No.	Compare Instruction
99	YNEI

YNEI (Y Not Equal to Immediate)

Format

YNEI i

Status

NZ of (Y - i)

Y = 1 ST=0

Y ≠ i ST=1

Operation

Y ≠ i

Description

Compares the contents of Y register to 4-bit immediate data (i_{3-0}).

Address format and the number of execution cycles

Address format	Mnemonic	Operand format	Instruction word		Number of words	Number of execution cycles
			First	Second		
	YNEI	i	$07i		1	1

Example

```
01203                    *
01204    07A     03CD            YNEI    $A
01205    1CE     03CE            CAL     OSERROR
01206                    *
```

7. APPLICATIONS

Note that the circuits and programs shown in this item are example. Please examine them on your application carefully.

7.1 Example of Subroutine Program

Example of subroutine program much used in the HMCS400 series is shown in this item. Subroutine call is effective only when ST is "1". As for subroutine 1 to 6, the following preconditions are applied.

(1) The locations from $030 to $06F of RAM are used as data area.
(2) The digits 4 to 15 of data area hold data.
(3) The locations from $020 to $02F of RAM (Memory Register MR0 to MR15 are used to save the register contents during interrupt service.
(4) A', B', SPX', SPY', X', Y', and W' show save area of A, B, SPX, SPY, X, Y, and W during interrupt service. Carry (CA) and Status (ST) save and return automatically.
(5) On the program to be interrupted, if writing a value into W, the same value is to be written into the location $020 (W''=MR0).

Address RAM memory map

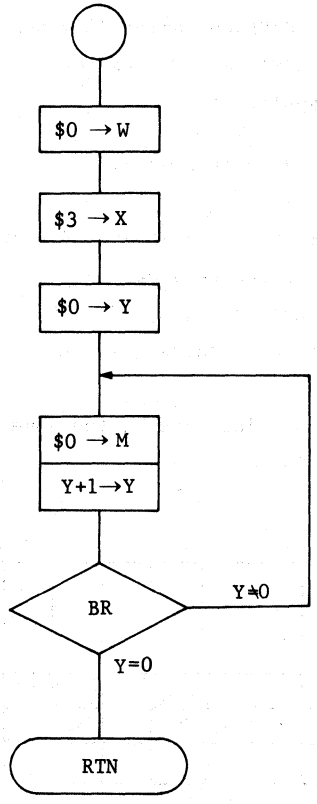
Upper 6 bits \ Lower 4 bits	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
$00																
$01						Unusable										-
$02	A'	B'	SPX'	SPY'	X'	Y'	W'									W''
$03	MSD											LSD				
$04					Data area											
$05																
$06	MSD											LSD				
$07																

Fig. 7-1 RAM Memory Map

7.1.1 RAM Clear

RAM is not initialized by reset function. Therefore user initializes RAM with program when initializing RAM.

Subroutine which clears RAM address from $030 to $03F is shown in Fig. 7-2.



(Label)	(OP)	(operand)
	LWI	$0
SB0	LXI	$3
SF0	LXI	$0
CL	LWIIY	$0
	BR	CL
	RTN	

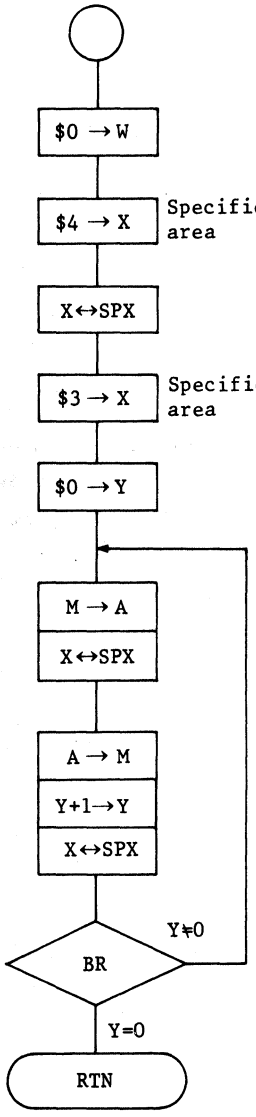
(Note) When clearing $PQ0 to $PQF, loads $P into W register and $Q into X register, and performs subroutine call for SF0 at ST=1.

(OP)	(Operand)
LWI	$P
LXI	$Q
CALL	SF0

Fig. 7-2 RAM Clear Subroutine

7.1.2 RAM Data Transfer

The subroutine shown in Fig. 7-3 transfers the data of $030 - $03F to $040 - $04F




```
(Label)   (OP)     (Operand)
  CB      LWI        $0
  VD      LXI        $4
          XSPX
          LXI        $3
  CF      LYI        $0
  TR      LAMX
          LMAIYX
          BR         TR
          RTN
```

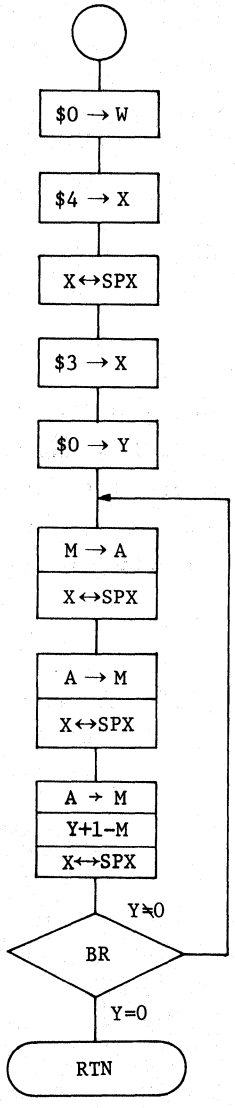
(Note) When transfering $RP0 - $RPF to $RQ0 - $RQF, loads R into W register, $Q into SPX register and $P into X register, and performs subroutine call for CF at ST=1.

```
(OP)      (Operand)
LWI          $R
LXI          $Q
XSPX
LXI          $P
CALL         CF
```

Fig. 7-3 RAM Data Transfer Subroutine

7.1.3 RAM Data Exchange

The subroutine shown in Fig. 7-4 exchanges the data of $03D to $03F with that of $040 - $04F.




```
(Label)     (OP)      (Operand)
CHB         LWI       $0
CHF         LXI       $5
            XSPX
            LXI       $3
CHD         LYI       $0
EX          LAMX
            XMAX
            LMAIY
            BR        EX
            RTN
```

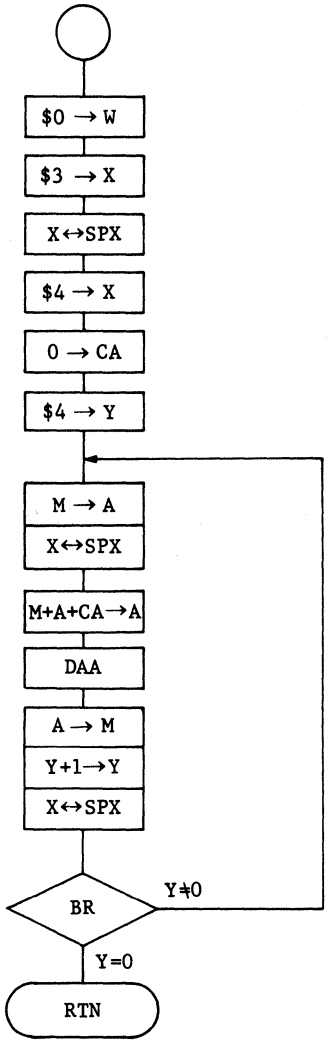
(Note) When exchanging the data of $RP0 - $RPF with that of $RQ0 - $RQF, loads $P into X register and $Q into SPX register, and performs subroutine call for CHD at ST=1.

```
(OP)      (Operand)
LWI       $R
LXI       $Q
XSPX
LXI       $P
CALL      CHD
```

Fig. 7-4 RAM Data Exchange Subroutine

7.1.4 Decimal Addition

The subroutine shown in Fig. 7-5 performs decimal addition between decimal 12-digit data $034 - $03F and $044 - $04F, and stores the result in $034 - $04F.



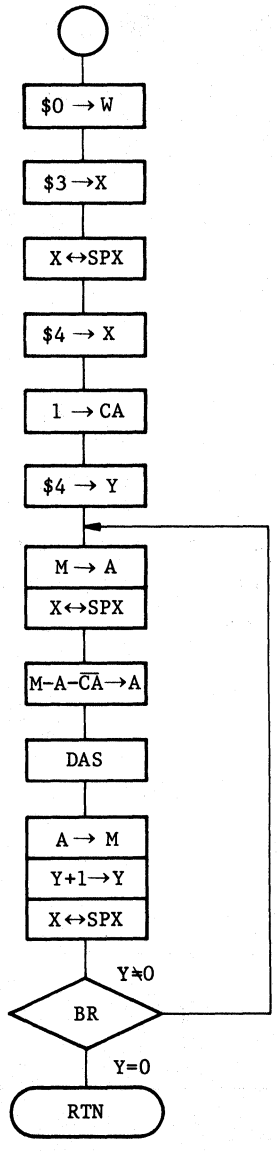

```
(Label)     (OP)      (Operand)
            LWI       $0
            LXI       $3
            XSPX
            LXI       $4
            REC
            LYI       $4
ADD         LAMX
            AMC
            DAA
            LMAIYX
            BR        ADD
            RTN
```

(Note) Carry data of the most significant digit remaines in Carry (CA) in this example.

Fig. 7-5 Decimal Addition Subroutine

7.1.5 Decimal Subtraction

The subroutine shown in Fig. 7-6 subtracts the decimal 12-digit data of $044 - $04F from that of $034 - $03F, and stores the result in $034 - $03F.




```
(Label)     (OP)      (Operand)
            LWI       $0
            LXI       $3
            XSPX
            LXI       $4
            SEC
            LYI       $4
SUB         LAMX
            SMC
            DAS
            LMAIYX
            BR        SUB
            RTN
```

(Note) Borrow data of the most significant digit remaines in Carry (CA) in this example.

Fig. 7-6 Decimal Subtraction Subroutine

7.1.6 Interrupt Service

Carry (CA) and Status (ST) are saved and restored automatically. The A, B, SPX, SPY, X, Y, and W registers should be saved by software. However, W must be saved by the following process since it cannot be directly saved in memory or a register. If a value is written into W with the LWI instruction by a program requiring an interrupt service, the same value as W must be written to RAM area MR0; MR0 is then saved in W' through the accumulator. The value of MR0 should be the same as that of W during program execution. In addition, due to RAM retriction, the value of W must be 0 or 3. Refer to Fig. 7-7.

```
                (Label)     (OP)        (Operand)      (Comment)
Register save   SAVE        XMRA        $F             A    SAVE
                            LAB
                            XMRA        $E             B    SAVE
                            LASPX
                            XMRA        $D             SPX  SAVE
                            LASPY
                            XMRA        $C             SPY  SAVE
                            XSPX
                            LASPX
                            XMRA        $B             X    SAVE
                            LAY
                            XMRA        $A             Y    SAVE
                            LAMR        $0
                            XMRA        $9             W    SAVE
                            RTN

Register return
                LOAD        INEMD       $3, $029
                            BR           LOOP1
                            LWI         $3
                            LMID        $3, $020
                            BR           LOOP2
                LOOP1       LWI         $0
                            LMID        $0, $020
                LOOP2       XMRA        $A
                            LYA                        Y    LOAD
                            XMRA        $B
                            LXA                        X    LOAD
                            XSPXY
                            XMRA        $C
                            LYA                        SPY  LOAD
                            XMRA        $D
                            LXA                        SPX  LOAD
                            XSPXY
                            XMRA        $E
                            LBA                        B    LOAD
                            XMRA        $F             A    LOAD
                            RTN
```

Fig. 7-7 Interrupt Service Program

7.1.7 Display Tube Dynamic Drive

The program performs dynamic display of 9-digit BCD data, $037-$03F in decimal. The display tube has 7 display segments (excluding decimal point) and is composed of 9 digits in all (refer to Fig. 7-8).

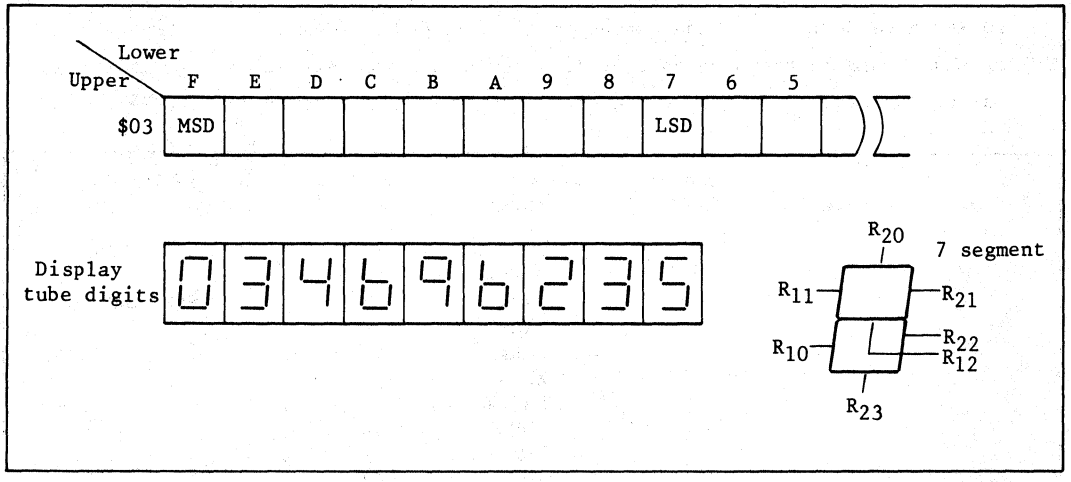


Fig. 7-8 Configuration of Display Tube

Discrete I/O pins D_7-D_{15} are used as digit control signals and data I/O ports R_1, R_2 are used as segment signals. The Y register addresses D port.

Segment display data are held in ROM pattern area and accessed by pattern instructions. This data is then loaded into registers and output from R_1, R_2 ports.

For example, data "5" is stored at address $037 as a binary number "(0101)2". If this data is displayed in decimal, each segment signal becomes R_{10}=0, R_{11}=1, R_{12}=1, R_{20}=1, R_{21}=0, R_{22}=1, and R_{23}=1.

The above data is stored at address $115 of ROM pattern area as shown in Table 7-1.

Table 7-1 Contents of ROM Address $115

Control part		Data for segment display							
r9	r8	r7	r6	r5	r4	r3	r2	r1	r0
1	0	1	1	0	1	0	1	1	0

(10)2 of the control part specifies that the contents of r7-r0 are output to R_1 and R_2 ports.

At this time, the pattern reference address should specify $115 as shown in Table 7-2. $5 is written into the accumulator, $1 into B register and $1 is specified as direct data of the pattern instruction. The upper 2 bits are fixed. Refer to pattern generation for additional information.

Fig. 7-9 shows allocation of ROM pattern area. The segment data represents "0" in $110, and "1" in $111. In the same manner, segment data from 0 to 9 are represented from $110 to $119, respectively.

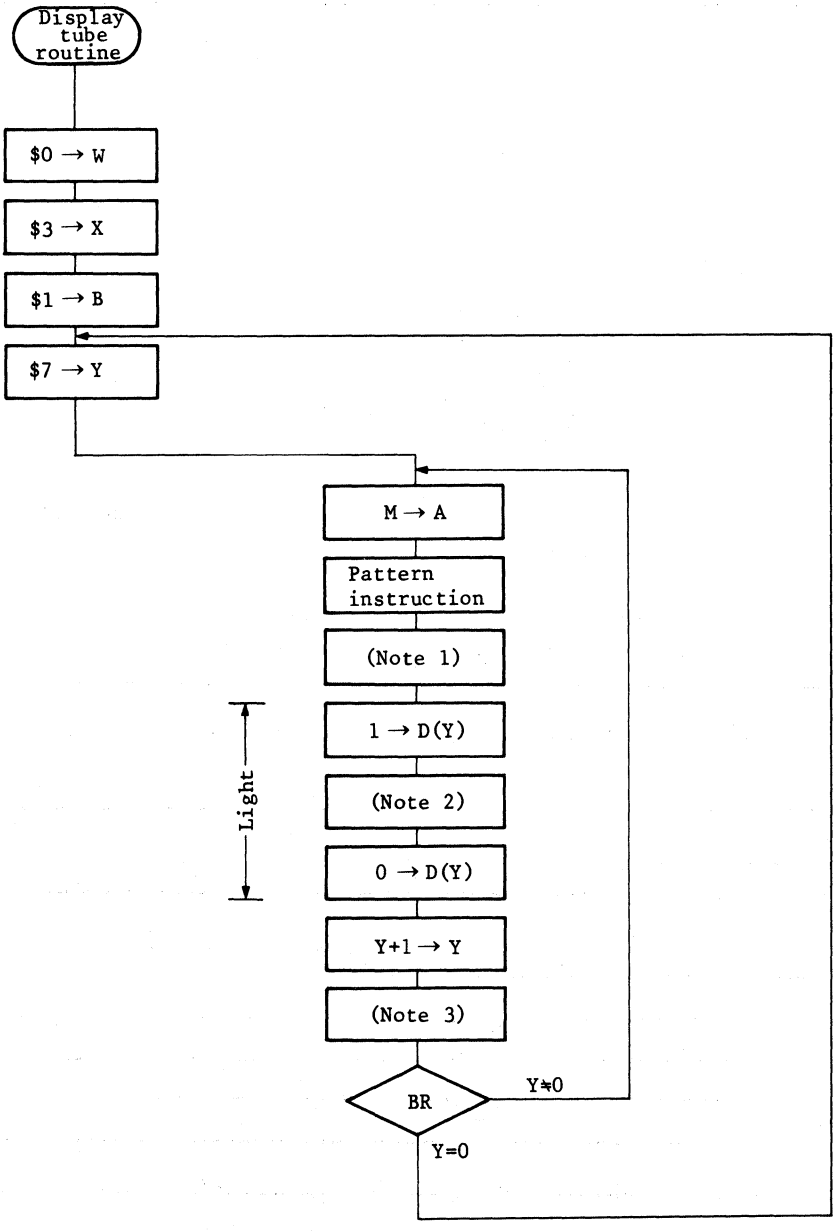
The flowchart for the display tube dynamic drive routine is shown in Fig. 7-10. Fig. 7-11 and Fig. 7-12 show the program listing and timing chart, respectively.

Table 7-2 Pattern Reference Address

Fixed		Direct				B register				Accumulator			
						B_3	B_2	B_1	B_0	A_3	A_2	A_1	A_0
0	0	0	0	0	1	0	0	0	1	0	1	0	1

Address	Control part address		Data for segment display								Display character
	r_9	r_8	r_7	r_6	r_5	r_4	r_3	r_2	r_1	r_0	
$110	1	0	1	1	1	1	0	0	1	1	0
$111	1	0	0	1	1	0	0	0	0	0	1
	1	0	1	0	1	1	0	1	0	1	2
	1	0	1	1	1	1	0	1	0	0	3
	1	0	0	1	1	0	0	1	1	0	4
	1	0	1	1	0	1	0	1	1	0	5
	1	0	1	1	0	0	0	1	1	1	6
	1	0	0	1	1	1	0	0	1	0	7
	1	0	1	1	1	1	0	1	1	1	8
$119	1	0	0	1	1	1	0	1	1	0	9

Fig. 7-9 Allocation of ROM Pattern Area



(Note 1) The latter half of blanking time.

(Note 2) Lighting time.

(Note 3) The first half of blanking time.

* Adjust the time by inserting appropreate process in (Note 1 ∿ 3).
Retain W, X, Y registers and status.

Fig. 7-10 Flowchart of Display Tube Dynamic Drive

```
(Label)      (OP)      (Operand)      (Comment)
             LWI       $0
             LXI       $3
             LBI       $1
START        LYI       $7
TOP          LAM
             P         $1
             Note 1
             SED                      Light
             Note 2
             RED
             IY                       Light out
             Note 3
             BR        TOP
             BR        START
```

Fig. 7-11 Program Listing of Display Tube Dynamic Drive

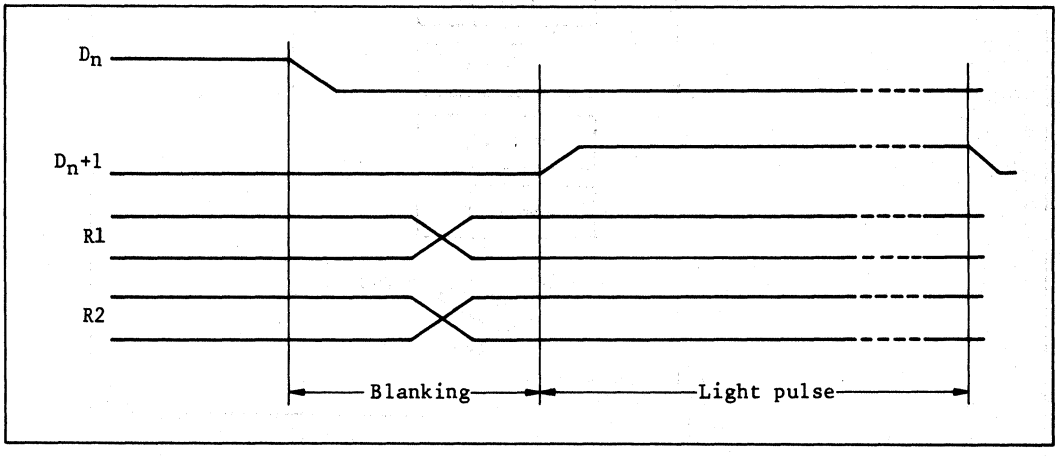
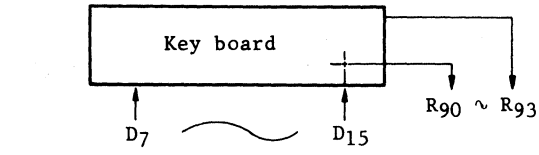


Fig. 7-12 Timing Chart

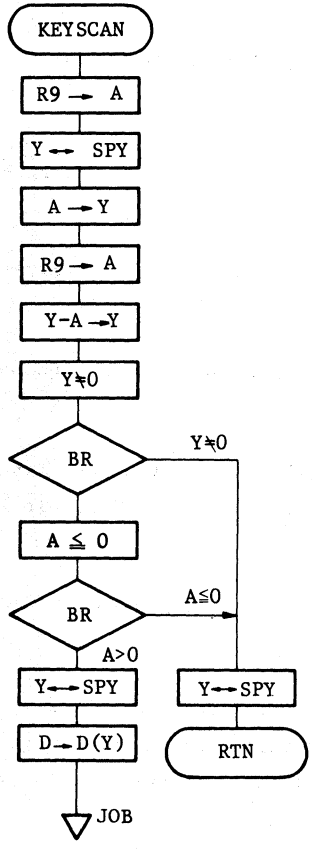
7.1.8 Keyboad Scan

This program is used by calling from Note 2 of display tube dynamic drive.

Fig. 7-13 is a keyboad scan routine (9×4) composed of key timing signal of $D_7 \sim D_{15}$ pins and input of R0 port.



The timing of $D_7 \sim D_{15}$ has already specified in the display tube dynamic routine. (Y register)



(Label)	(OP)	(Operand)	(Comment)
KEYSCAN	LAR	$9	The first INPUT
	XSPY		
	LYA		
	LAR	$9	The second INPUT
	SYY		
	YNEI	$0	Compares
	BR	KEYRET	Not the same key
	ALEI	$0	
	BR	KEYRET	No input key
	XSPY		
	RED		Lights out the display
	BRL	JOB	Goes processing routine (JOB) having key in Y and A registers
KEYRET	XSPY		
	RTN		Returnes to dynamic display

(Note) This program inputs data twice successively and compares then to keep from keyboard chattering. If two inputs agree, key is loaded and the program jumps to processing routine (JOB). Return to display tube routine by RTN instruction at the end of JOB because return address to display tube routine remains loaded in the stack if as it is. Retain W, X, Y registers and Status in the JOB routine.

Fig. 7-13 Keyboard Scan Routine

7.1.9 Timer A Application Example

Timer A is used as follows. Its logical flowchart is shown in Fig. 7-17.

(1) Value is set in Timer Mode Register A (TMA).
Timer Mode Register A is a 3-bit write-only register located at address $008 of RAM.
Prescaler divide ratio is determined by contents of the register and generates Timer A clock pulse. Timer Mode Register A is 0 during MCU reset.

(2) Interrupt is initiated when an interrupt request occurs.
First, reset Timer A Interrupt Request Flag (IFTA) located in bit 2 of $001. Then set or reset Timer A Interrupt Mask (IMTA) located in bit 3 of $001. Set or reset Interrupt Enable Flag (I/E).
IMTA is set to "1" and I/E to "0" during MCU reset.
These operations determine if the system will be interrupted or not after Timer A interrupt request.

(3) Timer Counter A (TCA) starts to count clock pulses generated at or after TMA setting. TCA is $00 during MCU reset.

(4) The counter generates an overflow at the 256th pulse, and then starts to count from $00.

(5) Overflow is latched in Timer A Interrupt Flag (IFTA) and the process is then performed according to the initial value. IFTA is located in bit 2 of RAM $001. This flag is reset by software.

(Note) The number of pulse counts immediately after setting Timer Mode Register A to the first overflow is indefinite.

The followings are example of a clock using Timer A. The clock shows minutes and seconds. Data of $030-$033 shows lower digits and upper digits of seconds and minutes. Refer to Fig. 7-14. The clock is incremented every second and figures are counted upward as in a normal clock.

The clock returns "0" after 59' 59", and the continues to count upwards. The flowchart is shown in Fig. 7-15.

Address

Upper 6 bits \ Lower 4 bits	3	2	1	0
$03	minute (upper digits)	minute (lower digits)	second (upper digits)	second (lower digits)
$04				

Fig. 7-14 RAM Map of Clock using Timer A

When using a 4.19MHz crystal resonator as the oscillator, the system clock becomes 525kHz. Clock pulse frequency is precisely set at 256Hz by setting the prescaler divide ratio to 1/2048. Consequently, the Timer A Interrupt Request Flag is set every second in this state.

An example of the program is shown in Fig. 7-16.

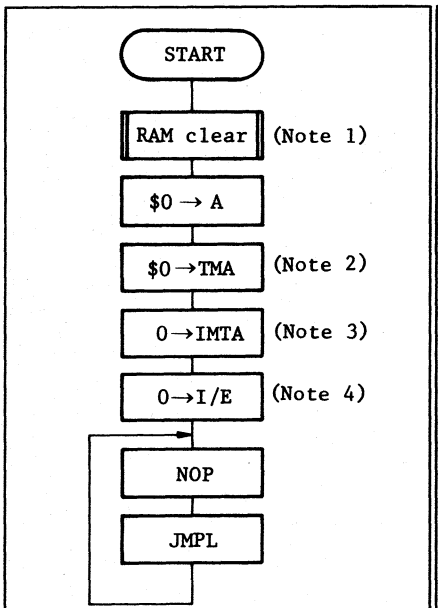


Fig. 7-15 Flowchart of Main Routine

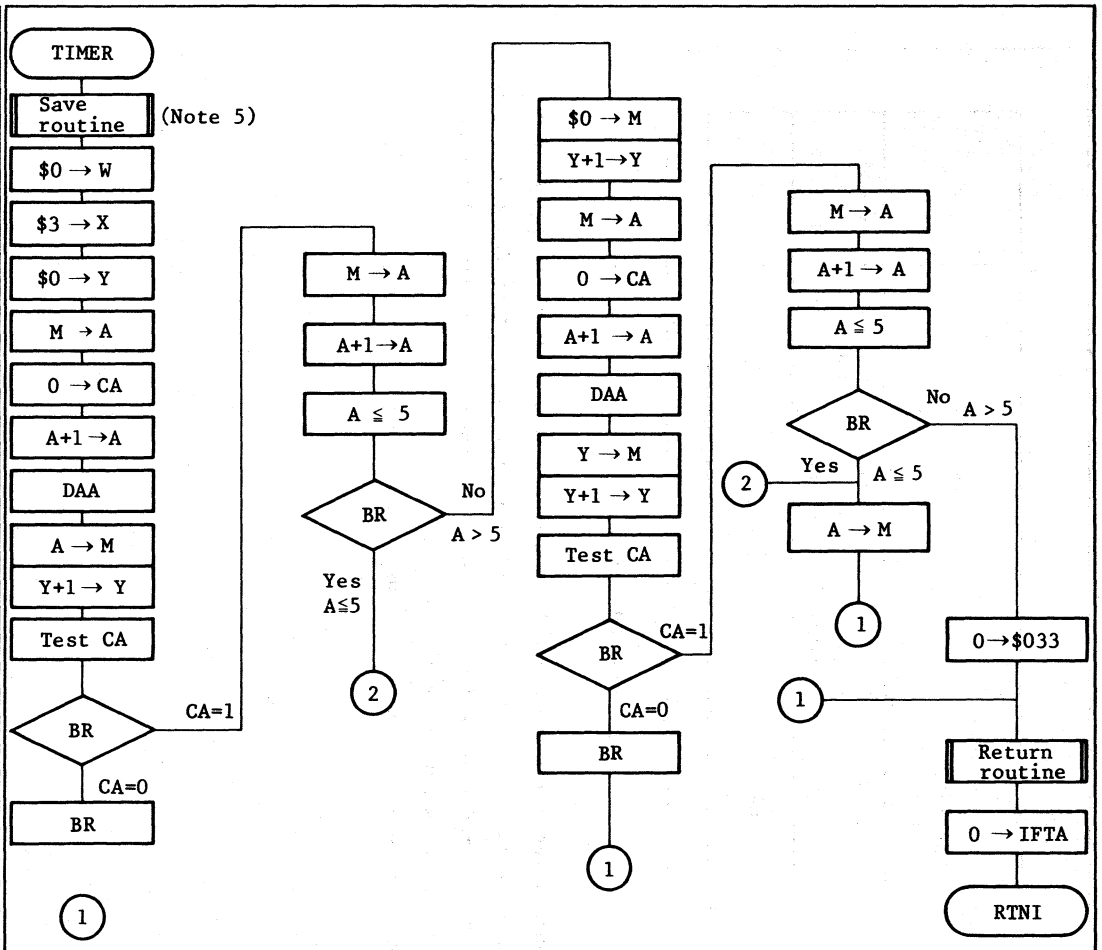


Fig. 7-16 Flowchart of Timer Subroutine

```
                (Label)     (OP)        (Operand)          (Comment)
Main routine                CALL        RAMCLR      Calls RAM clear subroutine.
                            LAI         $0
                            LMAD        $008        Sets prescaler divide ratio.
                            REMD        $3, $0001   Resets timer interrupt mask.
                            SEMD        $0, $000    Sets interrupt enable flag.
                TIMER       NOP
                            JMPL        TIMER


Timer subroutine            CALL        SAVE        Calls save routine.
                            LWI         $0
                            LXI         $3
                            LYI         $0
                            LAM
                            REC
                            AI          $1
                            DAA
                            LMAIY
                            TC
                            BR          SECH
                            BR          RET

                SECH        LAM
                            AI          $1
                            ALEI        $5
                            BR          DISP
                            LMIIY       $0
                            LAM
                            REC
                            AI          $1
                            DAA
                            LMAIY
                            TC
                            BR          MINH
                            BR          RET

                MINH        LAM
                            AI          $1
                            ALEI        $5
                            BR          DISP
                            LMID        $0, $033
                            BR          RET

                DISP        LMA
                RET         CALL        LOAD        Calls return routine.
                            REMD        $2, $001    Resets Timer A interrupt
                                                    flag.
                            RTNI
```

Fig. 7-17 Program Listing of Timer A Application

7.1.10 Timer B Application Example

Timer B is used as follows when using clock as an input. Its flowchart and program listing are shown in Fig. 7-18 and 7-19, respectively.

(1) Timer B Interrupt Mask (IMTB) is set to inhibit interrupts during setting of each mode register. IMTB exists is located in 1 of RAM $002. It is set to "1", and Interrupt Enable Flag to "0" during MCU reset.

(2) Setting of Timer Mode Register B
4-bit Timer Mode register B (TMB) is located at $009 of RAM. First, it is determined whether or not the auto reload function is used by setting or resetting bit 3.
Then the prescaler divide ratio and clock input source are specified. In the case of an external event input, specify bit 0, 1 and 2 according to the prescaler divide ratio. The Timer Mode Register B is not set bit by bit, but the set value is loaded into RAM at one time. This register is set to "0000" during MCU reset.

(3) Initial value is set in lower bits (TLR0-3) of reload register located in $00A of RAM.
Initial value is set in upper bits (TLR4-7) of reload register located in $00B of RAM. The value of reload register is $00 during MCU reset.

(4) Timer B Interrupt Request Flag (IFTB) is reset. IFTB is located in bit 0 of RAM $002.
Timer B Interrupt Mask (IMTB) is reset.
Interrupt Enable Flag (I/E) is set.

(5) Overflow generated from Timer Counter B is latched into Timer B Interrupt Request Flag (IFTB) and the process is then performed according to initialization of interrupt. IFTB is located in bit 0 of RAM $002 and is reset only by software.
The following sentences describes an example of setting the interrupt cycle. Suppose a 50.176 (msec) interrupt cycle is generated by using a 4MHz crystal. 8 frequency divider generates a 500kHz system clock, and a 1.024 (msec) prescaler output cycle is obtained by using prescaler divide ratio 1/512. If the value 207 is loaded into the reload register,

$$1.024 \times (256-207) = 50.176 \text{ (msec)}.$$

Because $(207)_{10}=(11001111)_2$, set $(1111)_2=\$F$ in lower digits of the reload register and $(1100)_2=\$C$ in upper digits.

The flowchart and program listing are shown in Fig. 7-18 and 7-19, respectively. Note that the process time of this subroutine does not exceed 50 msec.

(6) The contents of Timer Counter are read in twice; the lower 4 bits are read immediately after reading the upper 4 bits because the lower 4 bits are latched simultaneously when the upper 4 bits are read.

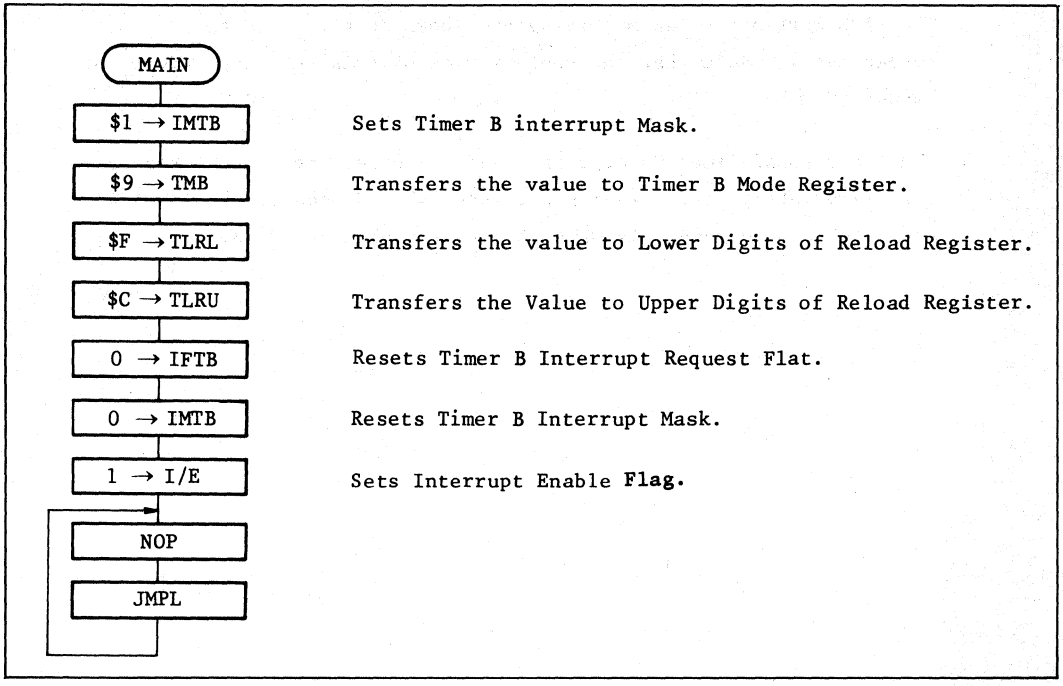


Fig. 7-18 Flowchart of Timer B Application Example

```
                (Label)   (OP)      (Operand)        (Comment)
Main routine              SEMD      $1, $002         Sets Timer B Interrupt Mask.
                          LMID      $9, $009         Transfers the value to Timer B
                                                     Mode register.
                          LMID      $F, $010         Transfers the value to lower
                                                     digits of reload register.
                          LMID      $C, $011         Transfers the value to upper
                                                     digits of reload register.
                          REMD      $0, $002         Resets Timer B Interrupt Request
                                                     Flag.
                          REMD      $1, $002         Resets Timer B Interrupt Mask.
                          SEMD      $0, $000         Sets Interrupt Enable Flag.
                TMB       NOP
                          JMPL      TMB

Timer B interrupt routine
                          CALL      SAVE             Calls save routine.

                        [Processing]

                          CALL      LOAD             Calls return routine.
                          REMD      $0, $002         Resets interrupt request flag.
                          RTNI
```

Fig. 7-19 Program Listing of Timer B Application

7.1.11 Serial Interface Application Example

Serial interface is used as follows.

(1) Data transfer (prescaler divide ratio: 1/128)

Fig. 7-20 shows the data transfer routine. The program listing is shown in Fig. 7-21.

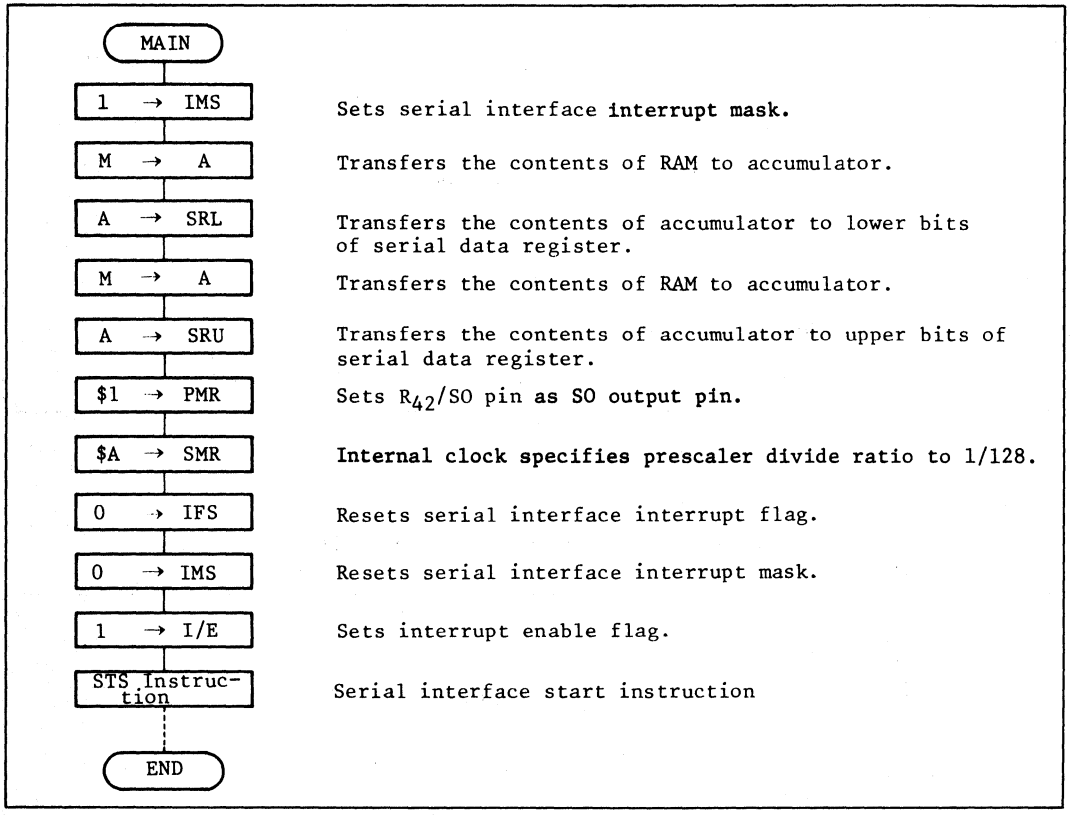


Fig. 7-20 Data Transfer Routine

After STS instruction, 8-bit data is transferred from R_{42}/SO pin synchronously with pulse output from $R_{40}/\overline{SCK}$ pin. IFS is then set and interrupt vector occurs. If the program does not need to jump to an interrupt routine, set IMS or reset I/E before executing STS instruction.

Main routine	(OP)	(Operand)	(Comment)
	SEMD	$1, $003	Sets serial interface interrupt mask.
	LAMD	$030	Transfers the contents of RAM location $030 to accumulator.
	LMAD	$006	Transfers the contents of accumulator to SRL.
	LAMD	$031	Transfers the contents of RAM location $031 to accumulator.
	LMAD	$007	Transfers the contents of accumulator to SRU.
	LMID	$1, $004	Loads the value of $1 into PMR (SO).
	LMID	$A, $005	Loads the value of $A into SMR.
	REMD	$0, $003	Resets IFS.
	REMD	$1, $003	Resets IMS.
	SEMD	$0, $000	Sets I/E.
	STS		Starts serial interface operation.

Fig. 7-21 Program Listing of Data Transfer

(2) Data reception (external clock)

Fig. 7-22 and Fig. 7-23 are flowchart and program listing of data reception respectively.

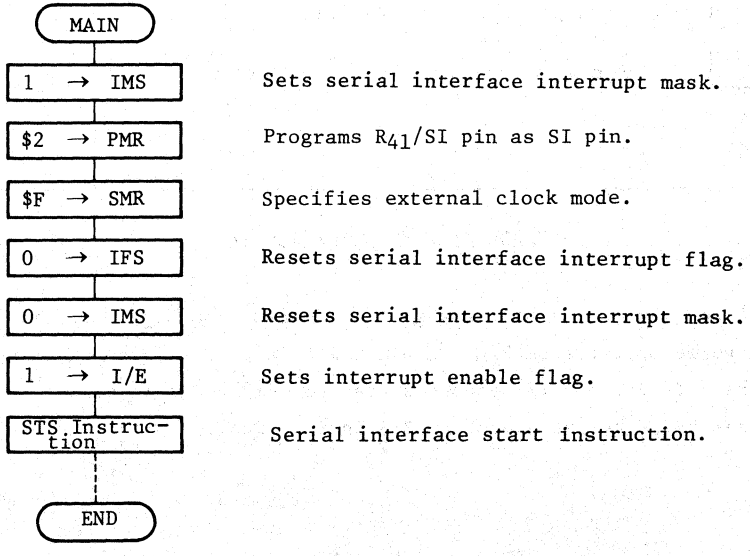


Fig. 7-22 Flowchart of Data Reception

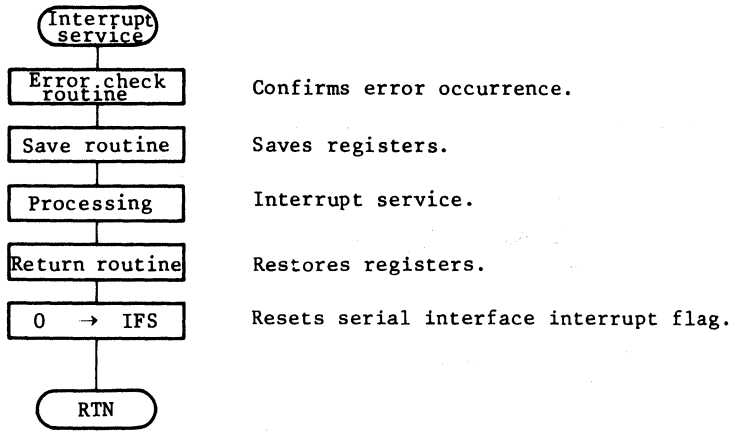


Fig. 7-22 Flowchart of Data Reception

```
Main routine    (OP)     (Operand)          (Comment)
    REPEAT      SEMD     $1, $003     Sets IMS.
                LMID     $2, $004     Loads $2 into PMR. (SI)
                LMID     $F, $005     Loads $F into SMR. (external clock)
                REMD     $0, $003     Resets IFS.
                REMD     $1, $003     Resets IMS.
                SEMD     $0, $000     Sets I/E.
                STS                   Starts serial interface operation.


Interrupt routine

                SEMD     $1, $003     Sets IMS.
                REMD     $0, $003     Resets IFS.
                LMID     $F, $005     Loads $F into SMR.     Error check
                TMD      $0, $003     Tests IFS.
                BRL      REPEAT
                CALL     SAVE         Calls save routine.

                [Processing]          Interrupt service.

                CALL     LOAD         Calls return routine.
                RTNI
```

Fig. 7-23 Program Listing of Data Reception

7.2 ALU (Arithmetic Logic Unit) and Decimal Adjust Instruction

ALU **basically** consists of a binary adder. The block diagram of ALU is shown in Fig. 7-24.

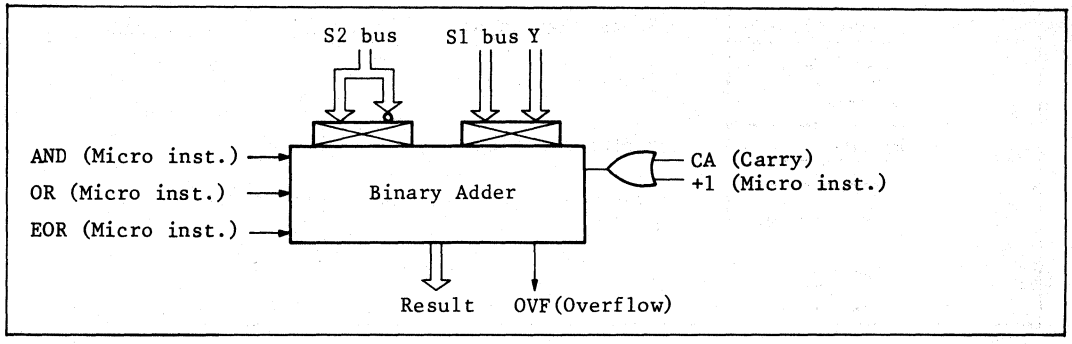


Fig. 7-24 ALU Block Diagram

When the addition of two or more digits is performed, OVF (Carry) is latched in CA and is put into the ALU at calculation of the next digit.

Subtraction is performed by using the inverse input on the bus S2. That is, calculation S1 - S2 is performed by the operation: S1 + $\overline{S2}$ + 1. But in this case, OVF means no borrow.

When the subtraction of two or more digits is performed, OVF (that is, no borrow) should be latched in CA. In this case, the second calculation is performed by the operation: S1 + $\overline{S2}$ + CA.

The following example shows the addition of 12-bit data.

$(159)_H + (26A)_H = (3C3)_H$

	0001	0101	1001	$(159)_H$
+)	0010	0110	1010	$(26A)_H$
	0011	01011	10011	
+)	CA	CA	CA ← 0	
	0011	⓪1100	①0011	$(3C3)_H$

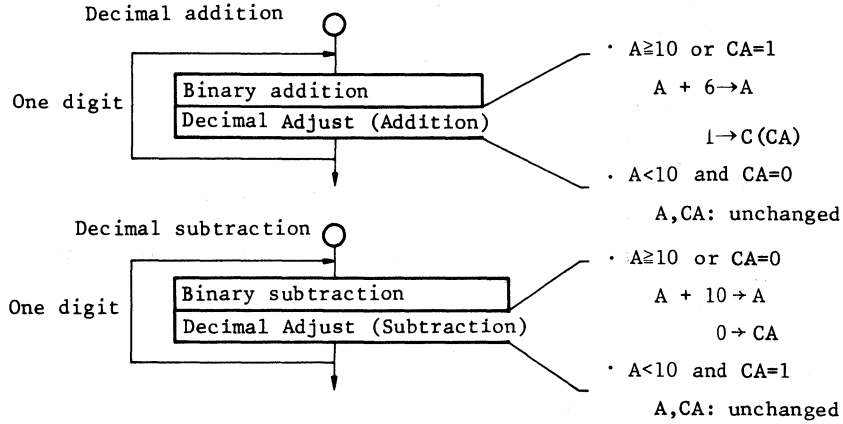
$(3C3)_H - (159)_H = (26A)_H$

	0011	1100	0011	$(3C3)_H$
+)	1110	1010	0110	$(159)_H$
	10001	10110	01001	
+)	CA	CA	CA ← 1	
	0010	①0110	⓪1010	$(26A)_H$

(NOTE) H means hexa-decimal number.

Decimal Calculation

Decimal addition/subtraction using binary-coded decimal (BCD) is performed by the combination of binary addition/subtraction and decimal adjust instruction. The decimal adjust instruction performs the adjustment of Accumulator and CA according to their contents.



For example,

169+245=417

	0001	0110	1001	169	Binary addition
+)	0010	0100	1000	248	
	00011	01010	10001		
+)	CA	CA	CA ← 0		
	00100	01011	10001		Decimal Adjust (Addition)
+)	-	0110	0110		
	0100	(1)0001	(1)0111	417	

417-159=243

	0100	0001	0111	417	Binary subtraction
+)	1110	1001	0110	169	
	10010	01010	01101		
+)	CA	CA	CA ← 1		
	10010	01010	(0)1101		Decimal Adjust (Substraction)
+)	-	1010	1010		
	10010	(0)0100	1000	248	

7.3 Application of Logical Operation

Examples of bit manipulation of the data using logical operation instruction are as follows.

(1) Resets the least significant bit of the data latched in R1 port.

Fig. 7-25 is the flowchart.

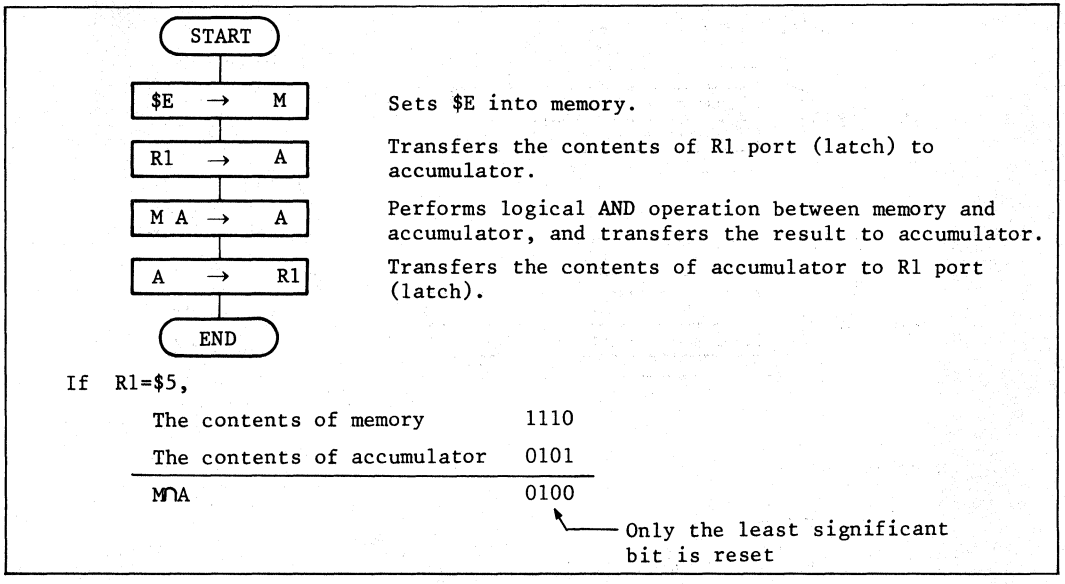


Fig. 7-25 Flowchart Using Logical and Operation

(2) Sets the least significant bit of the data latched in R1 port.

Fig. 7-26 is the flowchart.

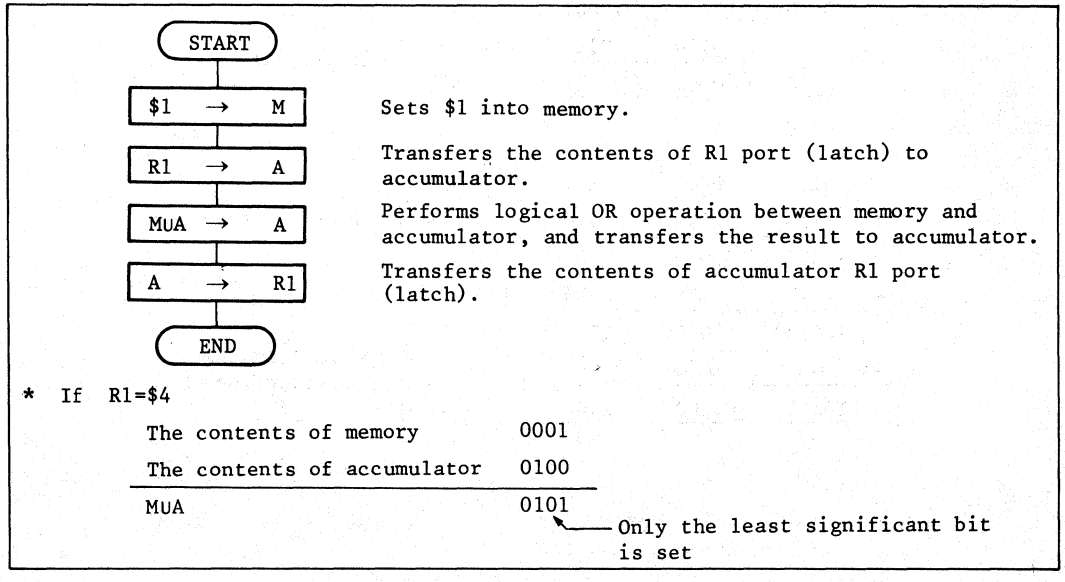
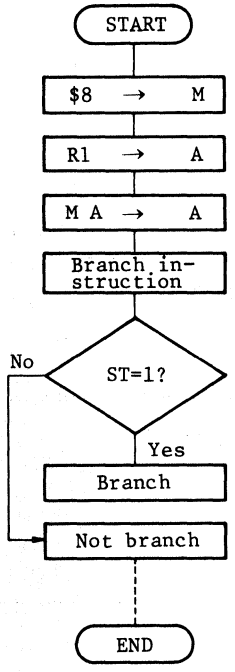


Fig. 7-26 Flowchart Using Logical OR Operation

(3) Tests the most significant bit of the data latched in R1 port.

Fig. 7-27 is the flowchart.



Loads $8 into memory.

Transfers the contents of R1 port (latch) to accumulator.

Performs logical AND operation between memory and accumulator.

If R1=$5,

The contents of memory	1000	
The contents of accumulator	0101	
M∩A	0000	ST (status) → 0

Consequently program does not branch because ST=0.

↓

The value of the most significant bit of R1 port is 0.

Fig. 7-27 Flowchart Using Logical and Operation

7.4 Checking Operation Frequency

Connecting $\overline{TEST}$ pin to GND causes D_6 pin to output pulse. The cycle of the D_6 pin is eight times as that of OSC_1 pin. Connect external pull-down resistor if I/O circuit type of D_6 pin is not "no pull-down MOS" S shown in Fig. 7-28.

The method in Fig. 7-28 may be inconvenient after the MCU has been built into a system since it operates in test mode.

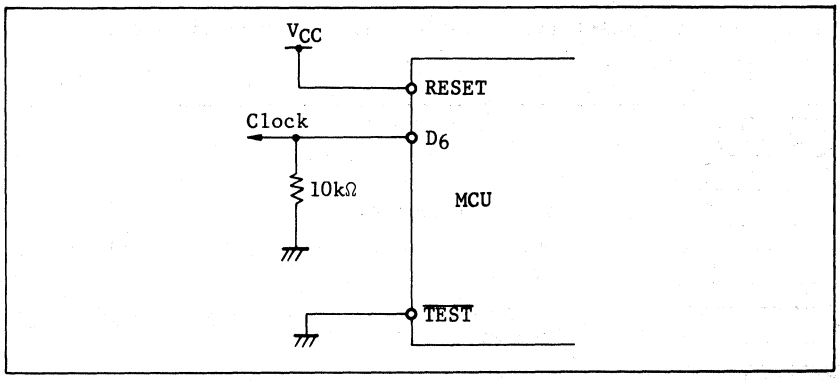


Fig. 7-28 Recommended Wiring to Check Operation Frequency

7.5 Watchdog Timer - System burst preventing circuit-

A simple method of implementing the watchdog timer which is generally known as a means of recovery from a system problem is described below.

(1) Basic circuit

Fig. 7-29 shows the basic circuit of the watchdog timer.

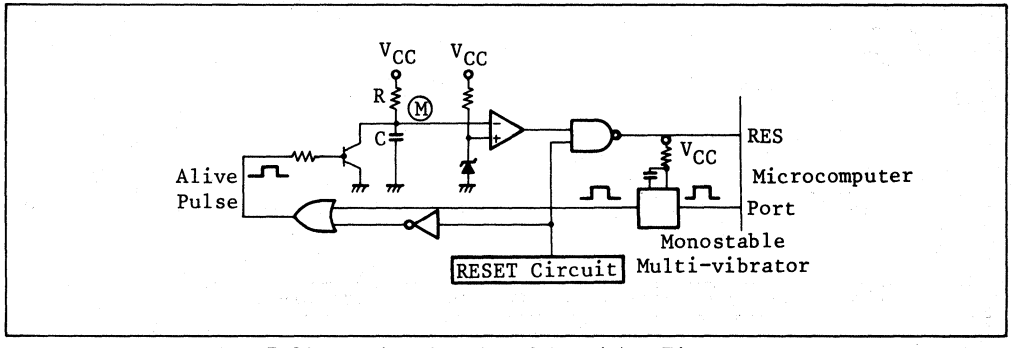


Fig. 7-29 Basic Circuit of Watchdog Timer

(2) Operation outline

(a) System software should be designed to generate a pulse through the port within a predetermined period (T<RC) once the program is normally executed.

(b) When the system is deadlocked or operates outside its normal routes, the system does not present port output change (i.e. it does not generate an active pulse), so that the potential of the Ⓜ point increases and the microcomputer is reset.

(c) It is necessary to preset the RC at a value above the maximum value among all the routes that can exist in normal operation. RC is set to greater than 10 ms in Fig. 7-30,

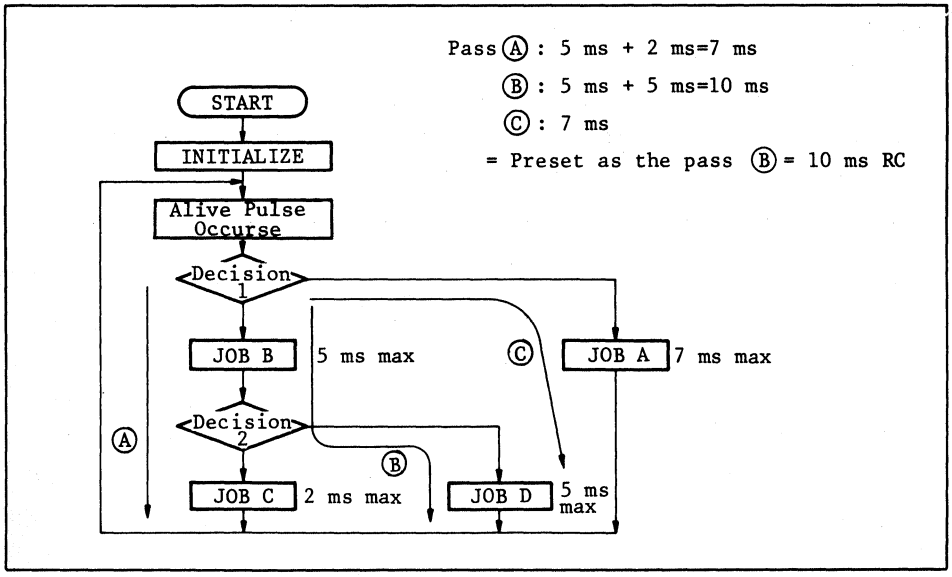


Fig. 7-30 Flowchart of Watchdog Timer

7.6 Auto Reset Circuit

Reset circuit which can perform power-on reset and auto reset at $V_{CC}<V_{LM}$ will be explained below. (V_{LM}; Threshold level)

(1) Basic circuit

Fig. 7-31 shows the basic auto reset circuit.

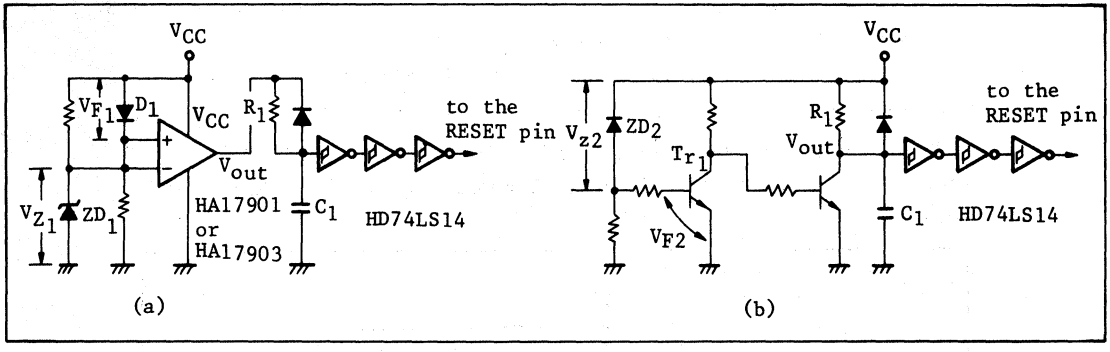


Fig. 7-31 Basic Auto Reset Circuit

(2) Operation

(a) Power-on Reset

Since Vout in the circuits of Fig. 7-32 is pulled up by R_1, C_1 is charged. The Vout rising time is determined by the charge time ($C_1 \times R_1$). Thus the power on reset is applied.

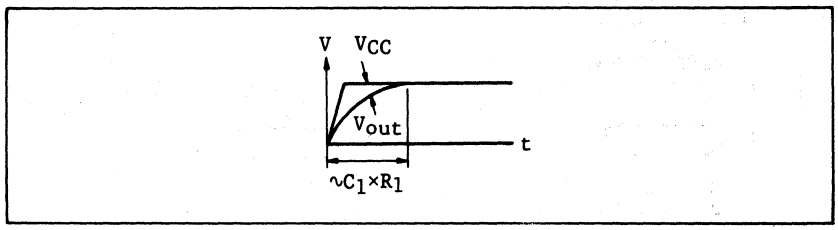


Fig. 7-32 Vout in Power-On Reset Circuit

(b) Auto Reset

The auto reset operation of the basic circuits (a, b) in Fig. 7-31 is explained below.

(i) Since the voltage V_{Z1} and V_{F1} applied to ZD_1 and D_1 are constant even if V_{CC} changes, the Vout changes, the Vout level is as follows:

$V_{out}=V_{CC}$ if $V_{CC}>V_{Z1}+V_{F1}$

$V_{out}=GND$ if $V_{CC}<V_{Z1}+V_{F1}$

(ii) Since the voltage V_{Z2} applied to ZD_2 and the ON level V_{F2} of Tr_1 are constant even if the V_{CC} changes, the Vout level is as follows:

$V_{out}=V_{CC}$ if $V_{CC}>V_{Z2}+V_{F2}$

$V_{out}=GND$ if $V_{CC}<V_{Z2}+V_{F2}$

When the V_{CC} changes, threshold level can be changed by executing the Zener diodes ZD_1 with ZD_2.

(c) By feeding back the output signal to the RESET pin and fine tuning the reference voltage, and by providing hysteresis to the V_{LM1} during the shifting from operation to resetting and the V_{LM2} at the time of recovery, the auto reset function becomes more stable. Refer to Fig. 7-33.

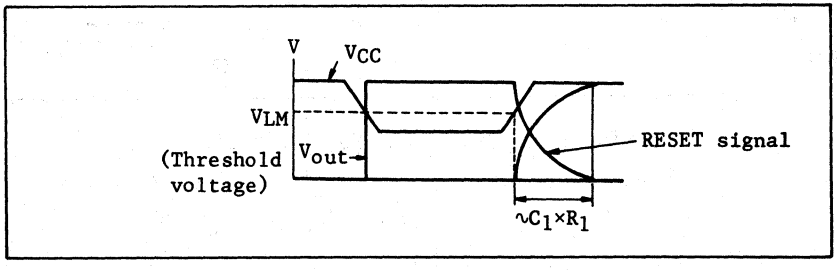


Fig. 7-33 Vout in Auto Reset Operation

7.7 Manual Reset Circuit

When the MCU is reset by an external switch, steps must be taken to prevent chattering. The reset circuit shown in Fig. 7-34 can prevent chattering and provide power on reset.

(1) Basic circuit

The basic circuit for manual reset is shown in Fig. 7-34.

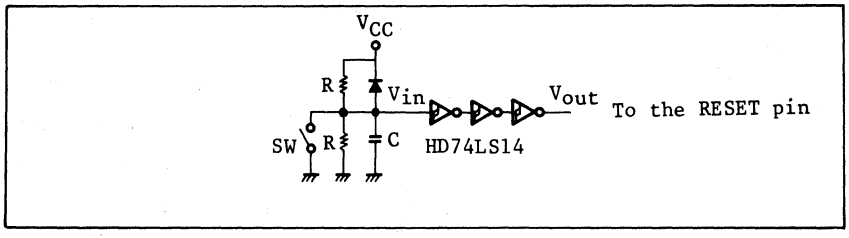


Fig. 7-34 Basic Manual Reset Circuit

(2) Operation

(a) Power on reset

During power-up, capacitor C will be charged and Vin rising time is determined by the CR charge time as shown in Fig. 7-35. The MCU can thus be reset normally.

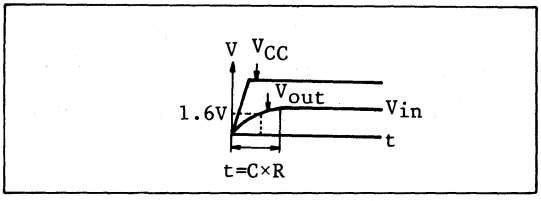
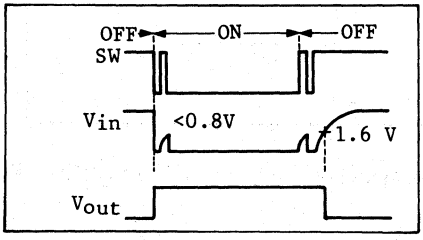


Fig. 7-35 Vin of Power-on Reset Circuit

(b) Manual reset

When SW is on, Vout is equal to V_{CC} and the MCU is reset. When SW is off, capacitor C is charged, and the rising time of Vin is determined by the CR charge time. Chattering is prevented by capacitor C and the HD74LS14 Schmitt trigger. Refer to Fig. 7-36.



① When SW goes from OFF to ON, Vout=V_{CC} if Vin<0.8V

② When SW goes from ON to OFF, Vout=V_{CC} if Vin>1.6V

Fig. 7-36 Manual Reset Timing

7.8 Serial Data Transfer between HMCS402/404/408 and Other MPUs

The routine shown in Fig. 7-37 describes **serial data transmission/** reception between the HMCS402/404/408 and other MPUs and storing the data in RAM. Hardware configuration available for this routine is shown in Fig. 7-38.

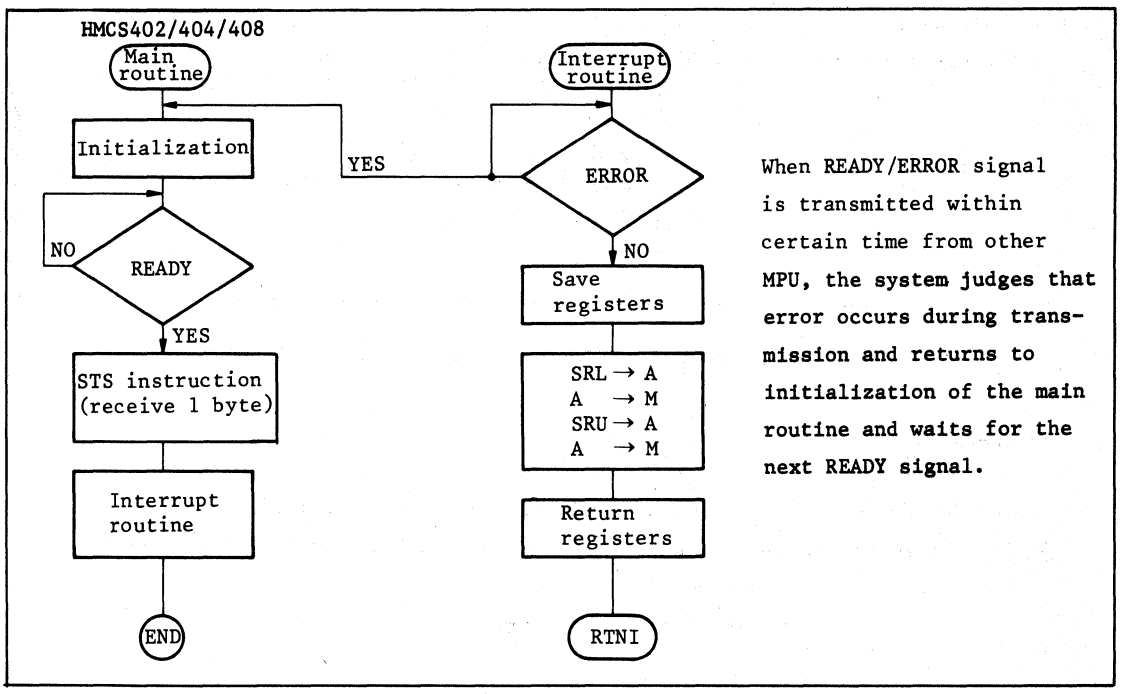


Fig. 7-37 Serial Data Transfer Routine

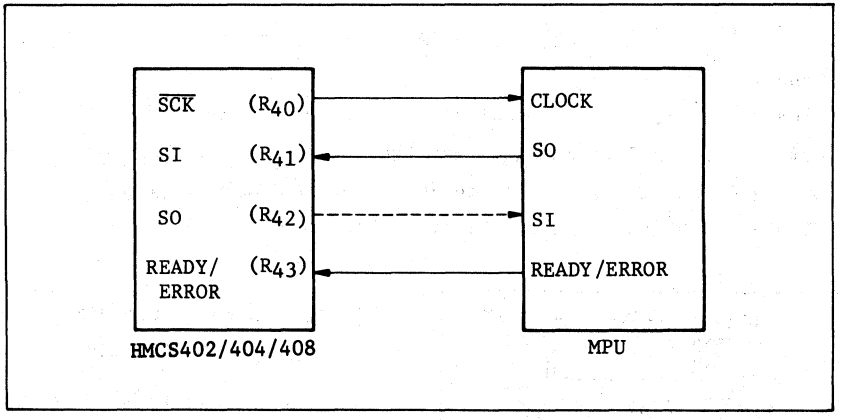


Fig. 7-38 Example of Connection Between HMCS402/404/408 and Other MPU

7.9 Reversing a String of Transmit/Receive Data in a Serial Interface (LSB-MSB)

Serial interface for the HMCS402/404/408 is shown in Fig. 7-39. Data is input to the serial data register LSB first through the SI pin synchronously with the rising edge of the transfer clock, while data is output from the serial data register LSB first synchronously with the falling edge of the transfer clock.

When serial data transfer is performed between the HMCS402/404/408 and other MPUs, the string of data in the serial data register must be reversed (LSB - MSB) if their data formats are different. Programs for data string conversion of the serial data register are shown in Fig. 7-40 and described as follows.

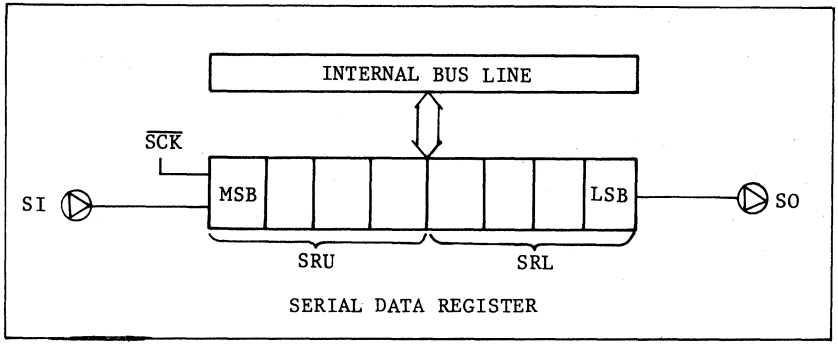


Fig. 7-39 Serial Interface for the HMCS402/404/408

When MSB is transmitted first from another MPU to the HMCS402/404/408, the data string input into the serial data register can be reversed using the pattern instruction. Pattern data is provided by the DC control instruction. First, the order of bits in the serial data register are reversed by conversion program (a), storing the data in memory (MD). Next, by executing program (b), this rearranged data is loaded from memory to the SDR in the correct bit significance order. The pattern data provided for ROM in the above case is shown in Fig. 7-41.

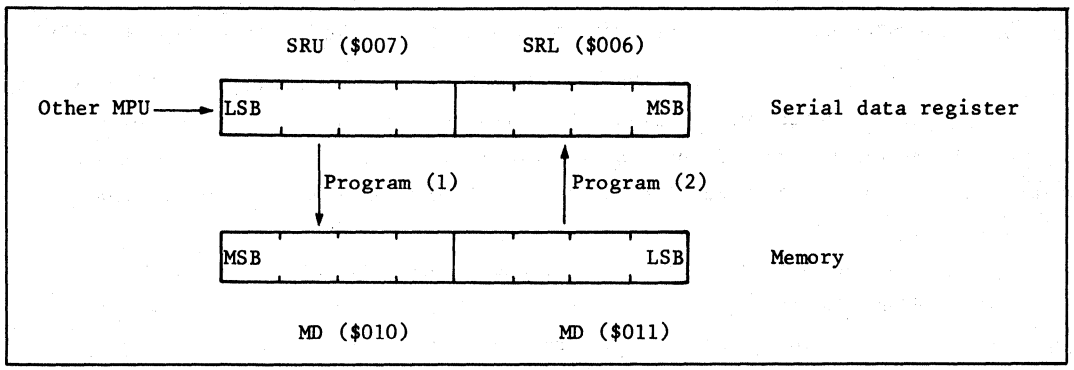


Fig. 7-40 LSB - MSB Conversion

(a) Serial data → program for conversion to memory

```
LBI     1
LAMD    $006 (SRL)  ③
P       0           ④
LMAD    $011        ⑤
LAMD    $007 (SRU)
P       0
LMAD    $010
```

(b) Memory → program for conversion to serial data register

```
LBI     1
LAMD    $011
P       0
LMAD    $006 (SRL)
LAMD    $010
P       0
LMAD    $007 (SRU)
```

Program example (a) is explained as follows.

① Another MPU transmits serial data $E5 MSB first to the HMCS402/404/408.
② $E5 is input to the HMCS402/404/408 serial data register in reverse bit order.
③ The real SRL is loaded into A (Accumulator) from $7 because of this reverse byte order.
④ After executing the pattern instruction (p), $E and $0 are loaded into A and B, respectively, since the specified address is $0017 at this time.
⑤ The contents of A are stored in memory.
⑥ Processes the real SRU in the same way as ① to ⑤.
⑦ Consequently the bit order is reversed and stored in memory correctly.

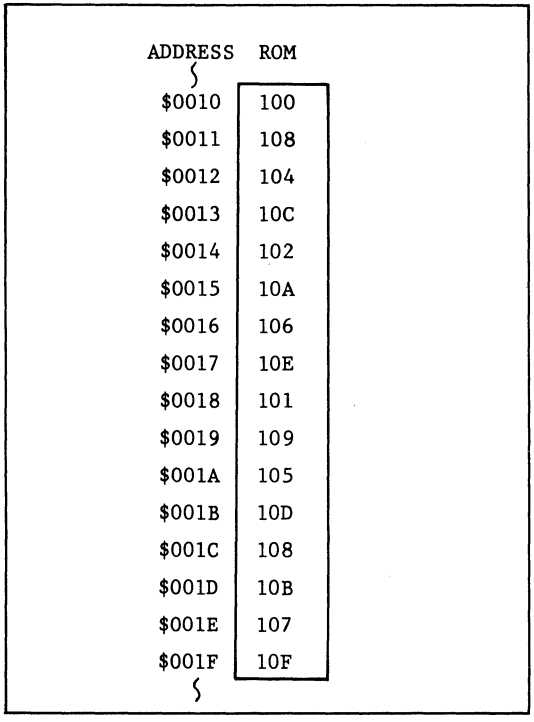


Fig. 7-41 Pattern Data

7.10 Expansion of Input Ports

The number of MCU input ports can be expanded using the HD14021B (8-bit Static Shift Register) as shown in Fig. 7-42.

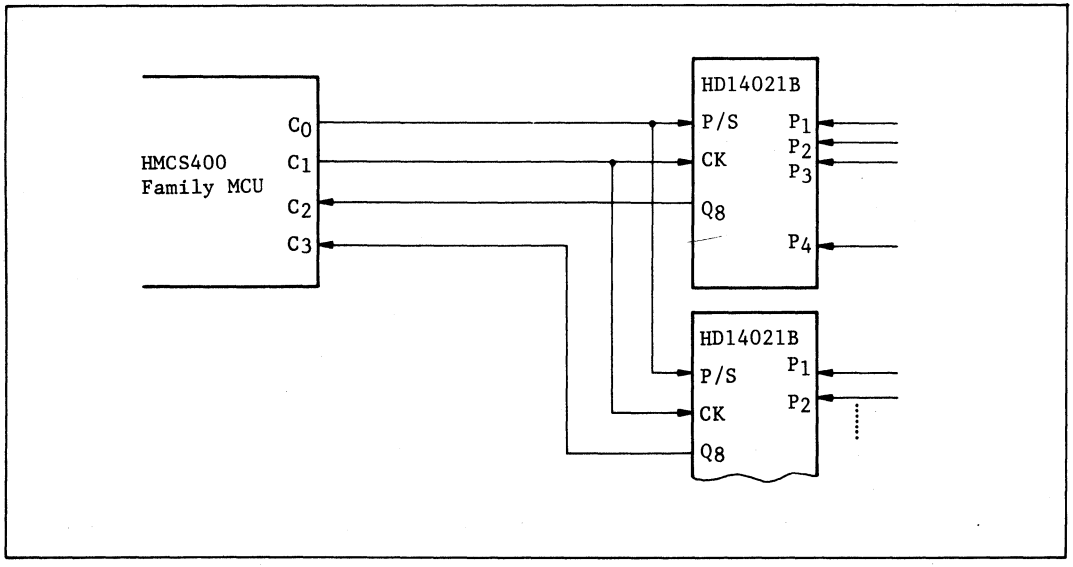


Fig. 7-42 Expansion of Input Ports

The number of total input pins depends on the number of HD14021Bs. Thus, the number of I/O lines in a system can be multipled as shown in Table 7-3.

Table 7-3 Expansion of I/O Pin

The number of the HD14021Bs	The number of I/O lines in a system
0	58
1	63
2	70
3	77
⋮	⋮

7.11 A/D Conversion Circuit (I) - High speed version

Implementing a simple A/D conversion (analog-digital conversion) using a resistance ladder (R-2R system) is described in this section.

(1) Basic Circuit

The A/D conversion basic circuit and timing chart are shown in Fig. 7-43 and 7-44, respectively.

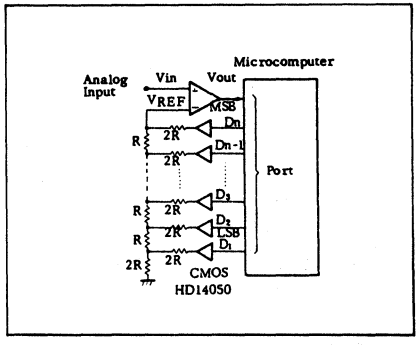


Fig. 7-43 A/D Conversion Basic Circuit

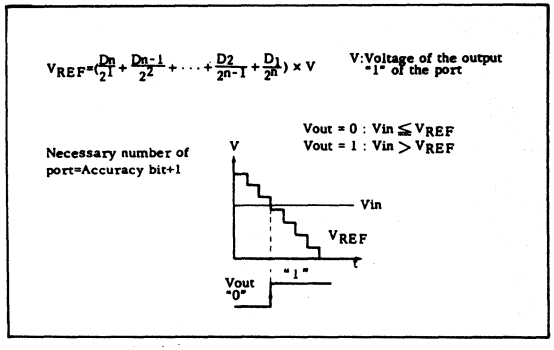


Fig. 7-44 Timing Chart

(2) Operation

(a) to (c) describes a circuit for 3-bit accuracy.

(a) Digital outputs (D_3, D_2, D_1) range from (1, 1, 1) to (0, 0, 0) at the ports. V_{REF} is reduced from 7/8 V to 0 in 1/8 V decrements as shown in Table 7-4.

(b) Output Vout, which is the result of comparison between the analog input Vin and V_{REF}, is reversed from 0 to 1 when Vin is greater than V_{REF} as shown in Fig. 7-44.

(c) The value of (D_3, D_2, D_1) when Vout is reversed from 0 to 1 is a digital value corresponding to the analog input Vin as shown in Fig. 7-45.

Table 7-4 Value of V_{REF}

D_3	D_2	D_1	V_{REF}
0	0	0	0
0	0	1	1/8×V
0	1	0	2/8×V
0	1	1	3/8×V
1	0	0	4/8×V
1	0	1	5/8×V
1	1	0	6/8×V
1	1	1	7/8×V

$$V_{REF} = \left(\frac{D3}{2^1} + \frac{D2}{2^2} + \frac{D1}{2^3}\right) \times V$$

D_1 to D_3 are denoted by "1" or "0"

V: Voltage of the port output "1"

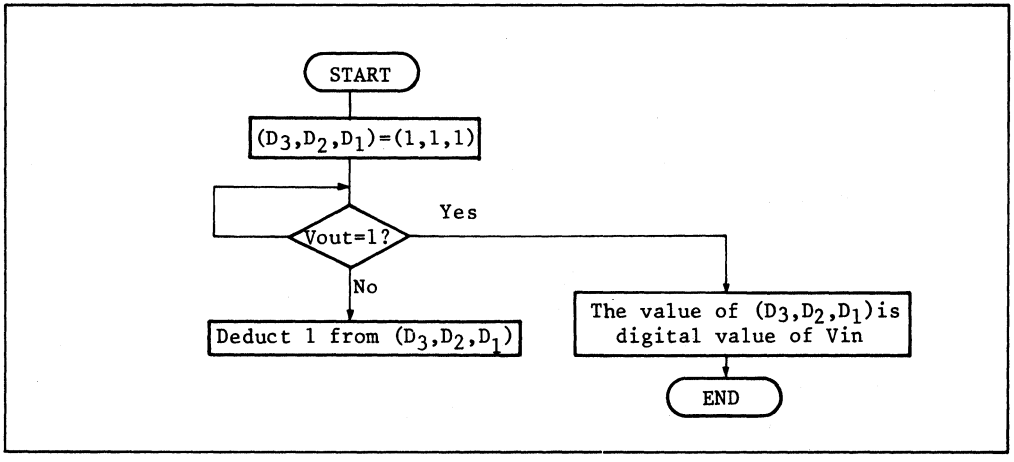


Fig. 7-45 Flowchart to Obtain Digital Value of Vin

7.12 A/D Conversion Circuit (II) - Low speed version

An A/D conversion system utilizing the time for charge/discharge of a CR circuit is explained in this section. This system is applied when high speed A/D conversion is not needed.

(1) Basic Circuit

An A/D conversion basic circuit of low speed version and the timing chart are shown in Fig. 7-46 and 7-47, respectively.

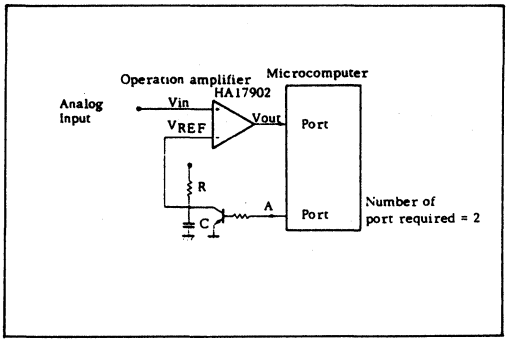


Fig. 7-46 A/D Conversion Basic Circuit

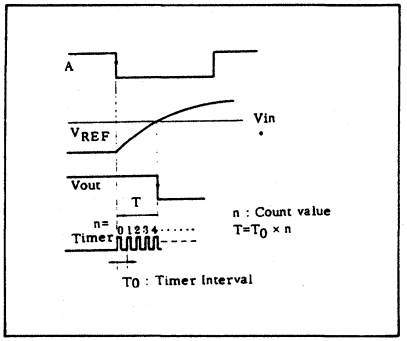


Fig. 7-47 Timing Chart

(2) Operation

(a) Set the output at A of the port to 1, and discharge external capacitor C sufficiently. Then switch A from 1 to 0 synchronously with the timer interrupt, and charge capacitor C. Refer to Fig. 7-46.

(b) Check Vout every timer interrupt to observe the time T when Vout switches from 1 to 0. Vout becomes 0 when Vin is less than V_{REF}. T is obtained as T=TO × n, where n represents the number of timer interrupts and TO represents the timer interval as shown in Fig. 7-47.

(c) The obtained time T which is voltage converted is a digital value of the analog input Vin as shown in Fig. 7-48.

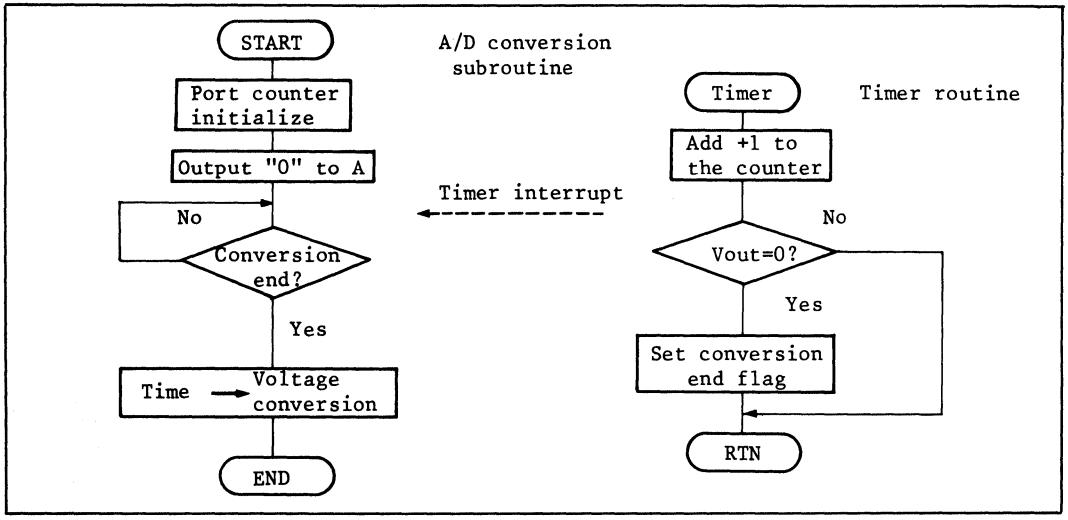


Fig. 7-48 Flowchart to Obtain Digital Value of Vin

7.13 Fluorescent Display Tube Drive Application (I) HMCS402/HMCS404/HMCS408

In the HMCS400 series, a fluorescent display tube can be driven by software using a display controller. This section explains how to drive a fluorescent display tube by timer B interrupt service routine.

Fluorescent display is performed by outputting grid signals (1G-7G) from pins D_9-D_{15} and outputting segment signals (a-h) from pins R_{00}-R_{03}, R_{10}-R_{13}.

Front view of the FL display tube is shown in Fig. 7-49, its specifications in Table 7-5, and an application of it is in Fig. 7-51. RAM map for FL display is shown in Fig. 7-50.

Grid and segment are selected by utilizing RAM area (for display) and the P instruction twice. D port is addressed by the value of GRID NO ($0C0) and outputs the grid signal. Segment signal is output from ports R_0 and R_1 through A register and B register.

(1) CODE data display

The value set in the display area is loaded into A register. Then segment data is loaded into A register and B register by executing the second P instruction. The segment signal is obtained by outputting the register value from R_0 and R_1 ports.

(2) Bit data display

The value set in the display area is loaded in A register and B register. Segment signal is output from R_0 and R_1 ports in the same way as CODE data display.

Intensity of the FL display tube is adjusted by resetting D port during (i) Timer B interrupt cycle, (ii) Timer B interrupt service routine.

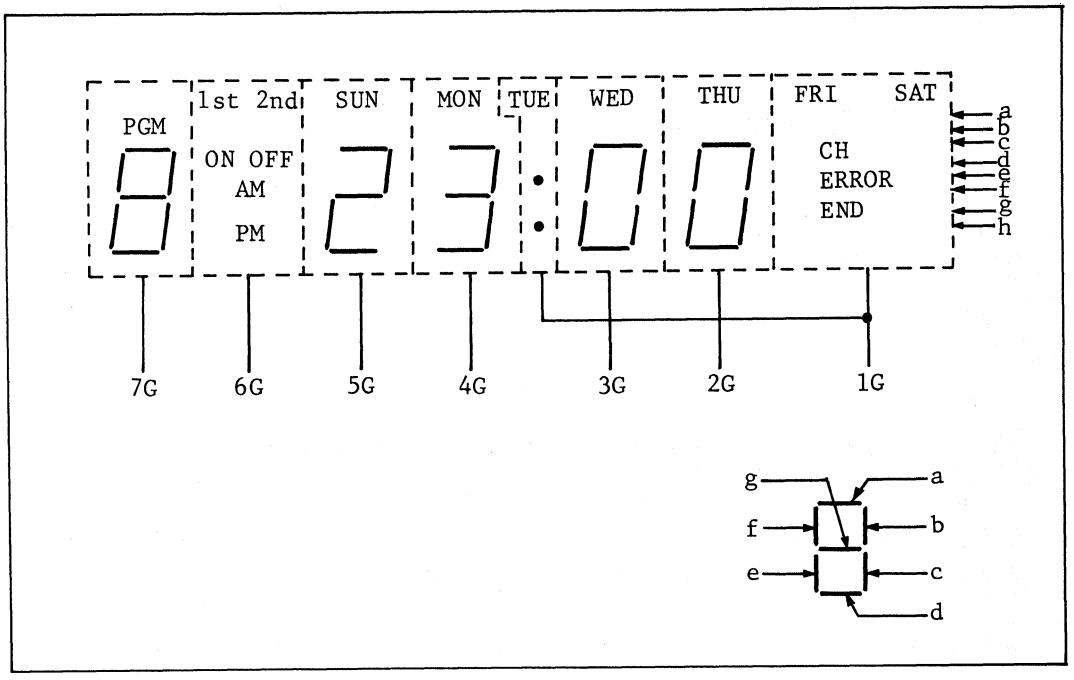
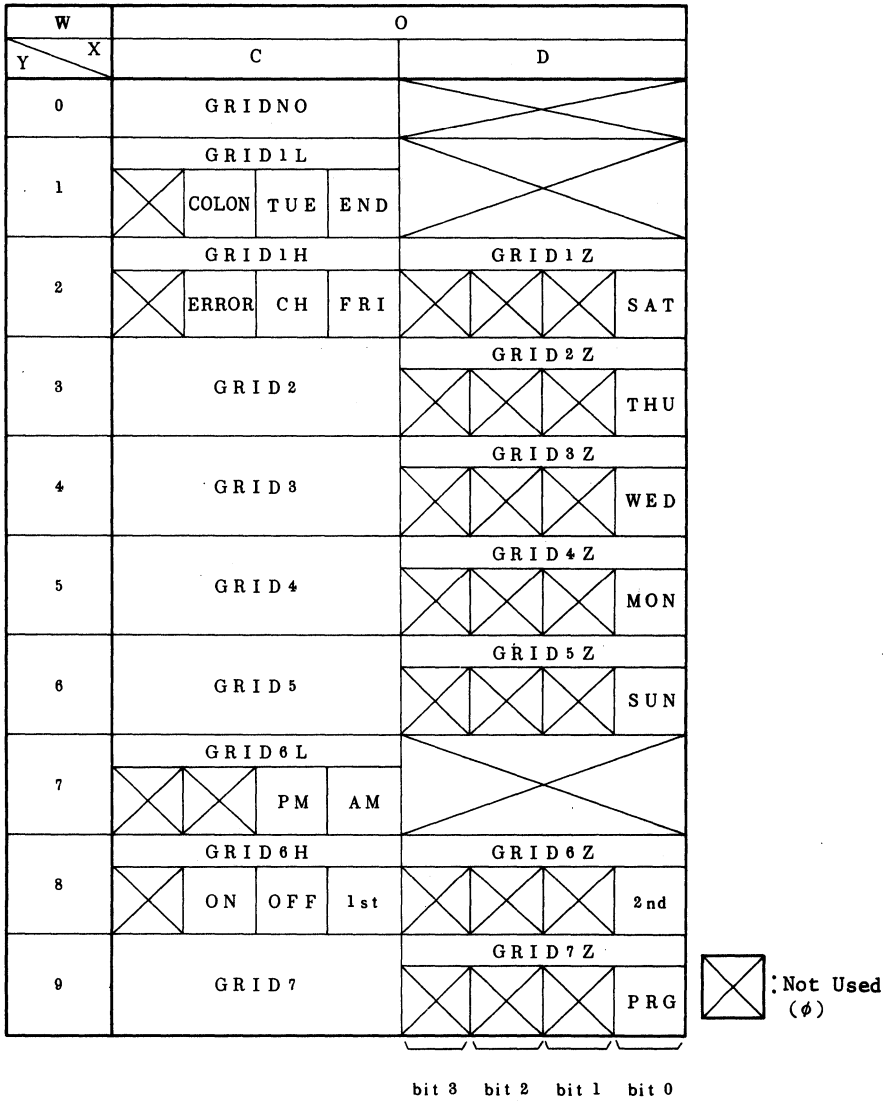


Fig. 7-49 FL Display Tube Front View

Table 7-5 FL Display Tube Specification

Microcomputer's Ports	R_{13}	R_{12}	R_{11}	R_{10}	R_{03}	R_{02}	R_{01}	R_{00}
FL	a	b	c	d	e	f	g	h
D9	1G	—	COLON	TUE	END	ERROR	CH	FRI	SAT
D10	2G	a	b	c	d	e	f	g	THU
D11	3G	a	b	c	d	e	f	g	WED
D12	4G	a	b	c	d	e	f	g	MON
D13	5G	a	b	c	d	e	f	g	SUN
D14	6G	—	—	PM	AM	ON	OFF	1st	2nd
D15	7G	a	b	c	d	e	f	g	PRG



GRIDNO: Area for storing pin No. of D port outputting High level to light the FL display tube.

Decimal number to be displayed is set in GRID2, GRID3, GRID4, GRID5, and GRID7 in terms of BCD. "A" will be set in them to light out. As for other area, to light, set the corresponding bit, and to light off, reset it.

Fig. 7-50 RAM Map for FL Display

```
*****************************************
*                                       *
*     TIMER-B INTERRUPT ROUTIN          *
*                                       *
*****************************************

+-------------------------------------+
|          REGISTER SAVE              |
+-------------------------------------+

          REMD     IFTB.$002
*****************************************
*          LAST GRID RESET              *
*****************************************
          LAMD     GRIDNO
          LYA
          RED
*****************************************
*          FL DRIVE                     *
*****************************************
          AI       $1
          BR       FLDRIVE1
          BR       FLDRIVE2
FLDRIVE1  LAI      $9
FLDRIVE2  LMAD     GRIDNO
          LBI      $F
          P        $F          A:FL DATA AREA X REG
*                               B 0:CODE 1:BIT
          LWI      $0
          LXI      $C
          LYA
          DB
          LBI      $E
          LAM
          BR       FLDRIVE3
          P        $F
          BR       FLDRIVE4
```

Program Listing (2)

```
FLDRIVE2 IY
         LBA
         LAM
FLDRIVE3 LXI     $D
         TM      $0
         SEC
         BR      FLDRIVE4
         REC
FLDRIVE4 ROTL
         LRA     $0
         LRB     $1
****************************************
*        GRID ON                       *
****************************************
         LAMD    GRIDNO
         LYA
         SED
```

REGISTER RESTORE

```
         RTNI
```

Program Listing (3) Pattern area

```
**************************************************
*            SEGMENT PATTERN DATA                *
**************************************************
        DRG     $FE0
        DC      $1F6    0
        DC      $160    1
        DC      $1D5    2
        DC      $1F1    3
        DC      $168    4
        DC      $1B8    5
        DC      $187    6
        DC      $1E0    7
        DC      $1F7    8
        DC      $1E8    9
        DC      $100    BLANK
        DC      $000    UNDEFINDED
        DC      $000    UNDEFINDED
        DC      $000    UNDEFINDED
        DC      $000    UNDEFINDED
        DC      $000    UNDEFINDED
**************************************************
*       GRID DATA ADDRESS PATTERN AREA           *
**************************************************
        ORG     $FF9
        DC      $109    GRID7
        DC      $117    GRID6
        DC      $106    GRID5
        DC      $105    GRID4
        DC      $104    GRID3
        DC      $108    GRID2
        DC      $111    GRID1
```

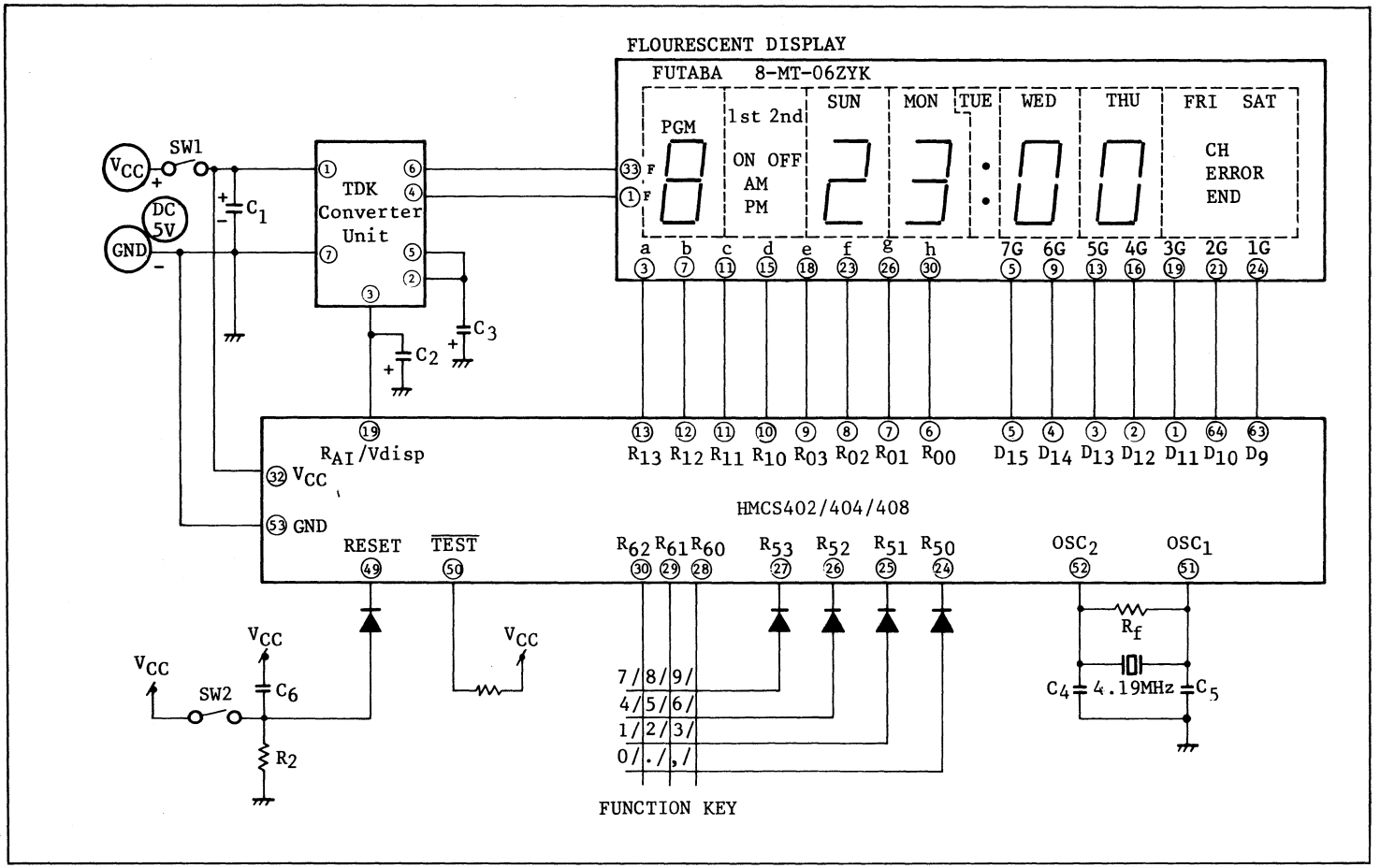


Fig. 7-51 Application of FL Display Tube Drive

7.14 Fluorescent Display Tube Drive Application (II) HMCS412/HMCS414

In the HMCS400 series, a fluorescent display tube can be driven by software using a display controller. This section explains how to drive a fluorescent display tube by timer B interrupt service routine.

Fluorescent display is performed by outputting grid signals (1G-7G) from pins D_8-D_{14} and outputting segment signals (a-h) from pins R_{00}-R_{03}, R_{10}-R_{13}.

Front view of the FL display tube is shown in Fig. 7-52, its specifications in Table 7-6, and an application of it is in Fig. 7-54. RAM map for FL display is shown in Fig. 7-53.

Grid and segment are selected by utilizing RAM area (for display) and the P instruction twice. D port is addressed by the value of GRID NO ($0C0) and outputs the grid signal. Segment signal is output from ports R_0 and R_1 through A register and B register.

(1) CODE data display

The value set in the display area is loaded into A register. Then segment data is loaded into A register and B register by executing the second P instruction. The segment signal is obtained by outputting the register value from R_0 and R_1 ports.

(2) Bit data display

The value set in the display area is loaded in A register and B register. Segment signal is output from R_0 and R_1 ports in the same way as CODE data display.

Intensity of the FL display tube is adjusted by resetting D port during (i) Timer B interrupt cycle, (ii) Timer B interrupt service routine.

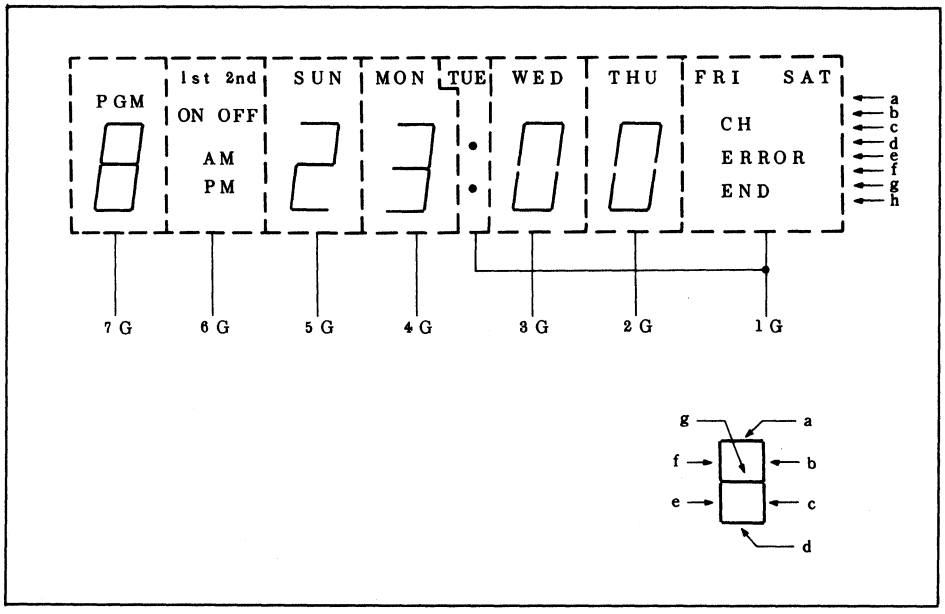
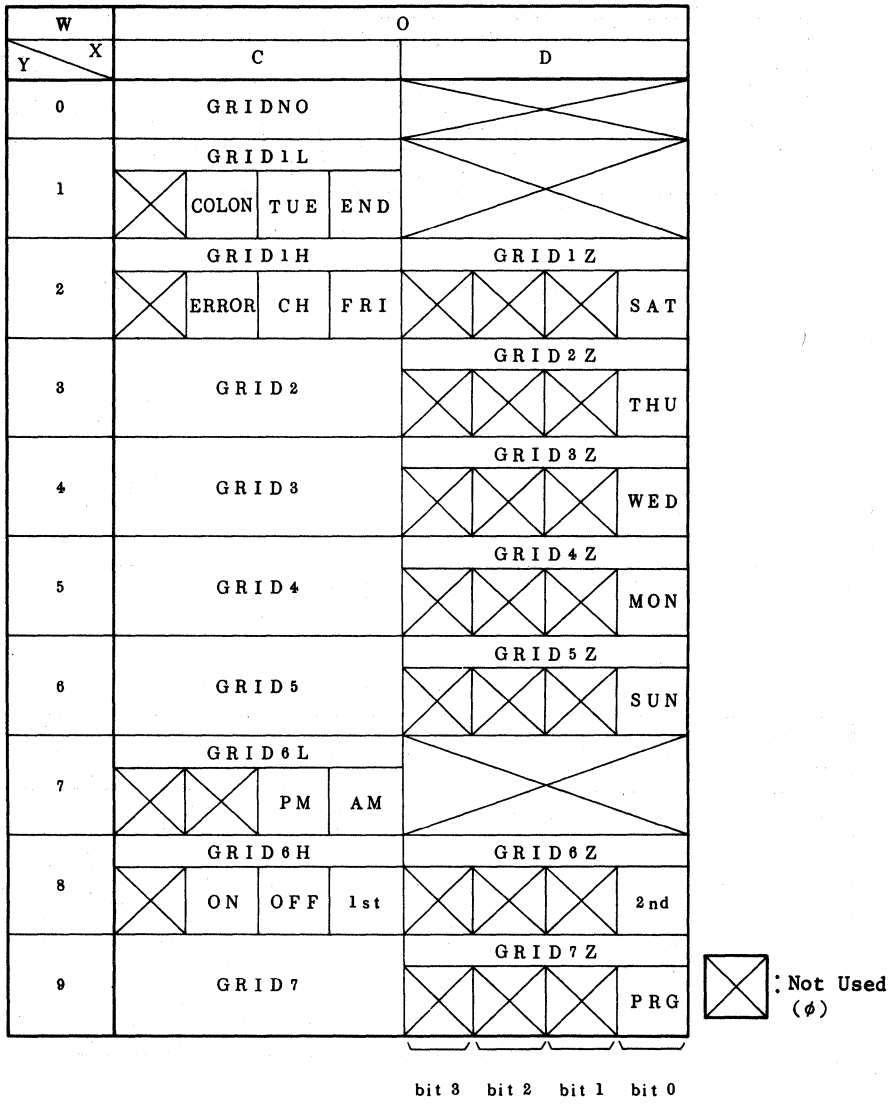


Fig. 7-52 FL Display Tube Front View

Table 7-6 FL Display Tube Specification

Microcomputer's Port	F L	R_{13}	R_{12}	R_{11}	R_{10}	R_{03}	R_{02}	R_{01}	R_{00}
		a	b	c	d	e	f	g	h
D 8	1G	—	COLON	TUE	END	ERROR	CH	FRI	SAT
D 9	2G	a	b	c	d	e	f	g	THU
D10	3G	a	b	c	d	e	f	g	WED
D11	4G	a	b	c	d	e	f	g	MON
D12	5G	a	b	c	d	e	f	g	SUN
D13	6G	—	—	PM	AM	ON	OFF	1st	2nd
D14	7G	a	b	c	d	e	f	g	PRG



GRIDNO: Area for storing pin No. of D port outputting High level to light the FL display tube.

Decimal number to be displayed is set in GRID2, GRID3, GRID4, GRID5, and GRID7 in terms of BCD. "A" will be set in them to light out. As for other area, to light, set the corresponding bit, and to light off, reset it.

Fig. 7-53 RAM Map for FL Display

Program Listing (1)

```
*****************************************
*                                       *
*     TIMER-B INTERRUPT ROUTIN          *
*                                       *
*****************************************

          REGISTER SAVE

          REMD    IFTB,$002
*****************************************
*          LAST GRID RESET              *
*****************************************
          LAMD    GRIDNO
          LYA
          RED
*****************************************
*          FL DRIVE                     *
*****************************************
          AI      $1
          ALEI    $E
          BR      FLDRIVE1
          LAI     $8
FLDRIVE1  LMAD    GRIDNO
          LBI     $F
          P       $F        A:FL DATA AREA X REG
*                           B 0:CODE 1:BIT
          LWI     $0
          LXI     $C
          LYA
          DB
          LBI     $E
          LAM
          BR      FLDRIVE3
          P       $F
          BR      FLDRIVE4
```

Program Listing (2)

```
FLDRIVE3  IY
          LBA
          LAM
FLDRIVE4  LXI     $0
          TM      $0
          SEC
          BR      FLDRIVE5
          REC
FLDRIVE5  ROTL
          LRA     $0
          LRB     $1
****************************************
*         GRID ON                      *
****************************************
          LAMD    GRIDNO
          LYA
          SED
```

REGISTER RESTORE

```
          RTNI
```

Program Listing (3) ... ... Pattern Area

```
*************************************************
*       SEGMENT PATTERN DATA                    *
*************************************************
        ORG     $FE0
        DC      $1F6    0
        DC      $160    1
        DC      $1D5    2
        DC      $1F1    3
        DC      $168    4
        DC      $1B8    5
        DC      $137    6
        DC      $1E0    7
        DC      $1F7    8
        DC      $1E8    9
        DC      $100    BLANK
        DC      $000    UNDEFINDED
        DC      $000    UNDEFINDED
        DC      $000    UNDEFINDED
        DC      $000    UNDEFINDED
        DC      $000    UNDEFINDED
*************************************************
*       GRID DATA ADDRESS PATTERN AREA          *
*************************************************
        ORG     $FF9
        DC      $109    GRID7
        DC      $117    GRID6
        DC      $106    GRID5
        DC      $105    GRID4
        DC      $104    GRID3
        DC      $108    GRID2
        DC      $111    GRID1
```

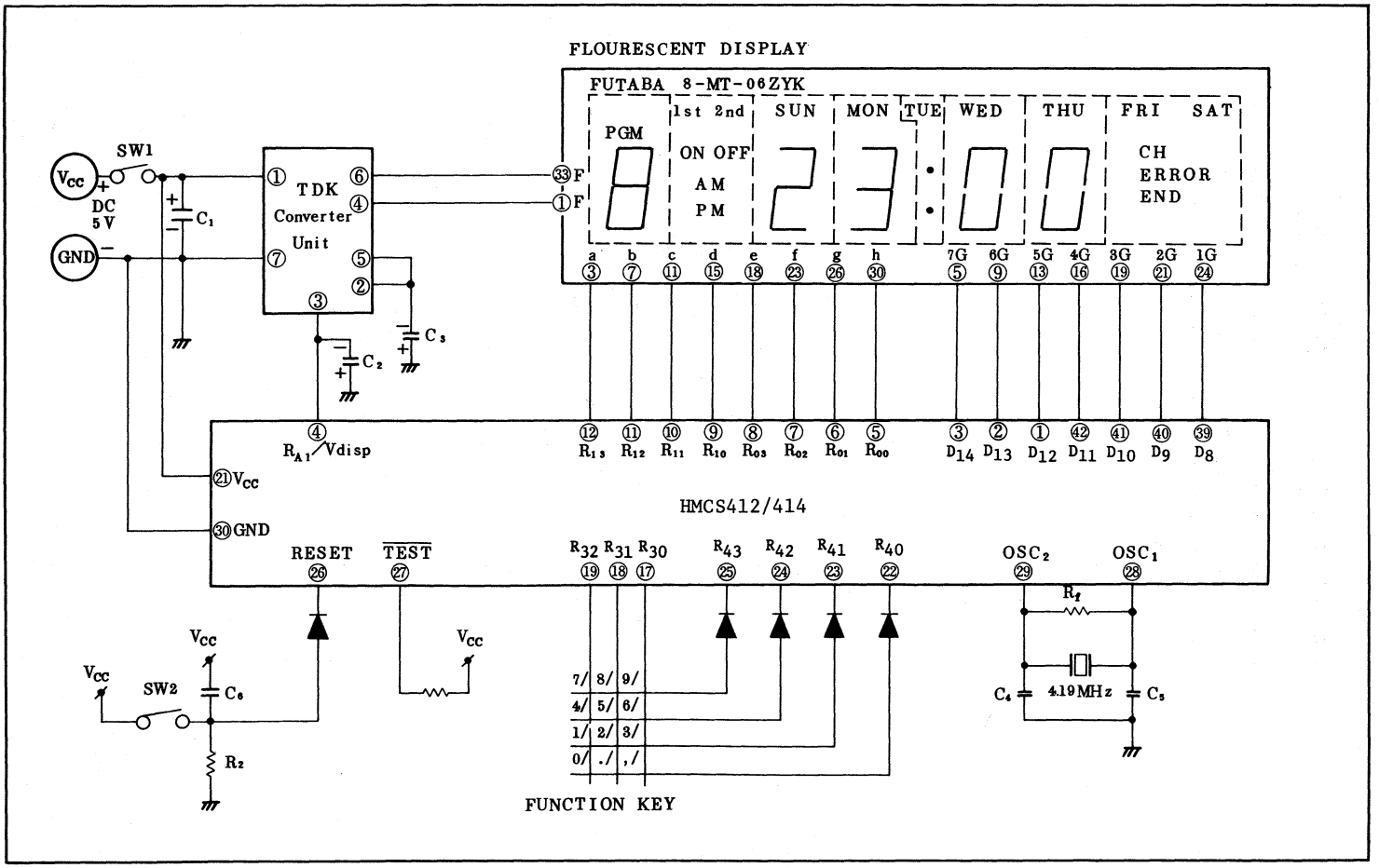


Fig. 7-54 Application of FL Display Drive

8. USER NOTES

8.1 Precautions on Using W Register

(1) In register indirect addressing (RAM addressing), the contents of W register are used for specifying RAM addresses. However, saving the contents of W register in a calling routine or an interrupt service routine is not simple because the W register is a write only register.

The contents of W register should be saved before an interrupt routine is invoked and restored just before its return. However, as the contents of W register cannot be read and saved, the identical contents of W register must be stored in RAM. The contents of W register can then be restored with the LWI instruction at the end of the interrupt service routine. This procedure is shown in Fig. 8-1.

However, this procedure is not efficient because of (a) the increase in the number of steps and processing time, and (b) possible problems in program debugging. (Program error may not be found depending on the address interrupt is serviced. For example, if interrupt is not inhibited when changing the contents of W register, error cannot be found without interrupt servicing at this address.)

(2) Consequently, the following procedure is recommended for not saving the contents of W register in a calling routine and interrupt service routine, and to prevent a situation which the contents of W register are unrecoverable.

(a) Set the contents of W register to $0.
Write $0 into W register at MCU reset and leave it unchanged from then on. In this case, register indirect addressing instruction (W,X,Y registers) can access the range from $000 to $0FF in RAM. When accessing the range from $100 to $3FF in RAM, use direct addressing instructions.

(b) Use only direct addressing in an interrupt service routine.
In the interrupt enable state with W register = $0, leave the value of W register unchanged during an interrupt service routine, and do not use register indirect addressing instructions.

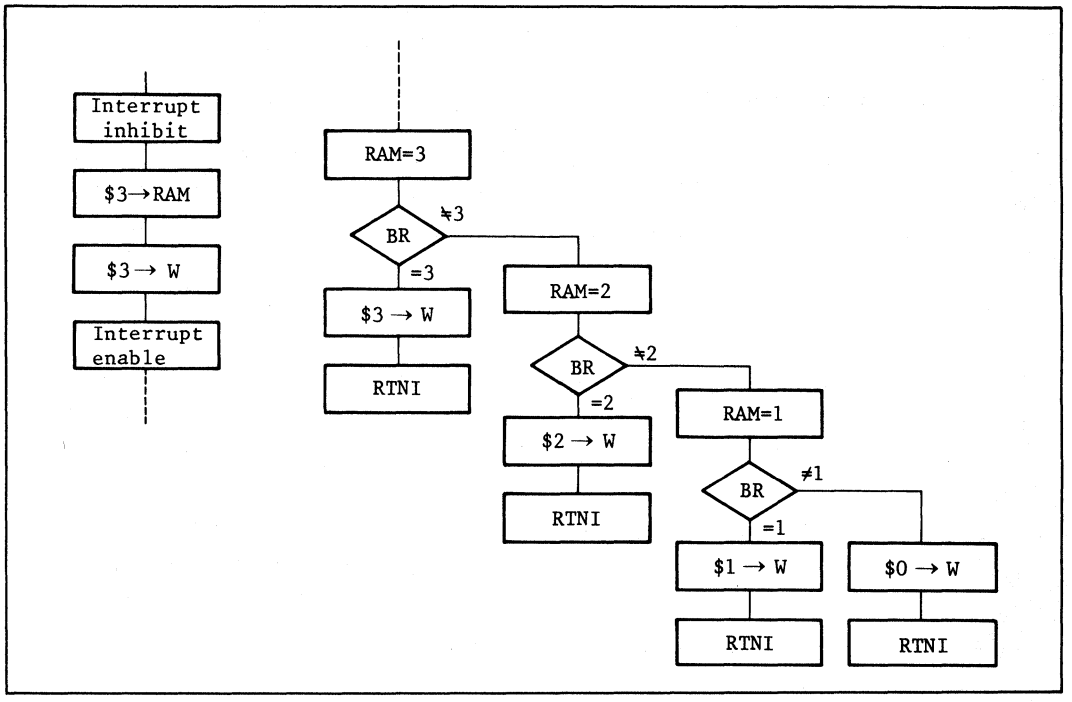


Fig. 8-1 Change of the Contents of W Register (Change to $3)

8.2 Precautions regarding the Contents of RAM and Registers after Reset

The MCU is reset by setting RESET pin to 1. At power ON in recovering from stop mode, apply RESET input longer than t_{RC} to obtain the necessary time for oscillator stabilization. In all other cases, MCU reset requires at least two-instruction cycles of RESET input.

Table 8-1 shows the items to be initialized by MCU reset and their initial values.

Table 8-1 Initial Value by MCU Reset

Items			Initial value by MCU reset	Contents
Program counter (PC)			$0000	Execute program from the top of ROM address.
Status (ST)			"1"	Enable to branch with conditional branch instructions.
Stack pointer (SP)			$3FF	Stack level is 0.
I/O pin output register	Standard pin	(A) Without pull-up MOS	"1"	Enable to input.
		(B) With pull-up MOS	"1"	Enable to input
		(C) CMOS	"1"	–
	High voltage pin	(D) Without pull-down MOS	"0"	Enable to input.
		(E) With pull-down MOS	"0"	Enable to input.
Interrupt flag	Interrupt Enable Flag (I/E)		"0"	Inhibit all interrupts.
	Interrupt Request Flag (IF)		"0"	No interrupt request.
	Interrupt Mask (IM)		"1"	Mask interrupt request.
Mode register	Port Mode Register (PMR)		"0000"	See Item "Port Mode Register".
	Serial Mode Register (SMR)		"0000"	See Item "Serial Mode Register".
	Timer Mode Register A (TMA)		"000"	See Item "Timer Mode Register A".
	Timer Mode Register B (TMB)		"0000"	See Item "Timer Mode Register B".
Timer/Counter, Serial interface	Prescaler		$000	–
	Timer/Counter A (TCA)		$00	–
	Timer/Event Counter B (TCB)		$00	–
	Timer Load Register (TLR)		$00	–
	Octal Counter		"000"	–

(Note) MCU reset affects to the rest of registers as follows:

Item		After recovering from STOP mode by MCU reset	After MCU reset except for the left condition
Carry	(CA)	The contents of the items before MCU reset are not retained. It is necessary to intialize them by software again.	The contents of the items before MCU reset are not retained. It is necessary to initialize them by software again.
Accumulator	(A)		
B Register	(B)		
W Register	(W)		
X/SPX Registers	(X/SPX)		
Y/SPY Registers	(Y/SPY)		
Serial Data Register	(SR)	Same as above	Same as above
RAM		The contents of RAM before MCU reset (just before STOP instruction) are retained.	Same as above

8.3 Notes on Unused Pins

When input/output pins not used in a user system are in a floating state, electric potential of the I/O pins should be fixed because the LSI may malfunction by noise. The following describes some examples.

High break-down voltage pin: Select "without pull-down MOS (PMOS open drain)" with mask option list, and connect the pin to V_{CC} on the print board of the user system.

Standard pins: Select "without pull-up MOS (NMOS open drain)", and connect the pin to GND on the print board of the user system. $R_{40}/\overline{SCK}$ pin and R_{42}/SO pin are programmed as R_{40} pin and R_{42} pin, respectively by the serial mode register and port mode register.

8.4 Notes on Board Design of an Oscillation Circuit

When connecting crystal and ceramic resonators with the OSC_1 and OSC_2 pins for oscillation, observe the following design precautions.

(1) Locate crystal, ceramic resonator, and load capacity C_1 and C_2 as close to the LSI as possible as shown in Fig. 8-2 (induction of outside noise on the OSC_1 and OSC_2 pins may cause problems in oscillation).

(2) Wire the signal lines adjacent to OSC_1 and OSC_2 pins as far apart as possible and not in parallel in order to prevent crosstalk.

(3) If the signal line or power supply line is laid out near the oscillator circuit as shown in Fig. 8-3, normal oscillation may not operate as specified. The resistor between OSC_1 or OSC_2 and nearby pins should be 10MΩ or more.

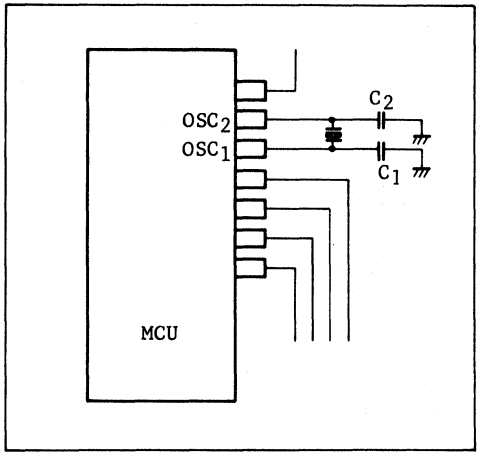


Fig. 8-2 Design of Oscillation Circuit Board (I)

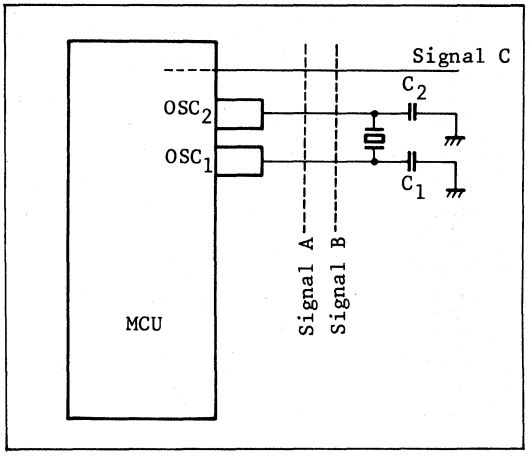


Fig. 8-3 Example of Circuit Causing Trouble in Oscillation

(4) When operating MCU using an external clock, be careful of a long drop from the external clock to the OSC_1 pin. Induction of noise on the OSC_1 pin from signal lines or power supply lines can lead to malfunction of the MCU. Refer to Fig. 8-4.

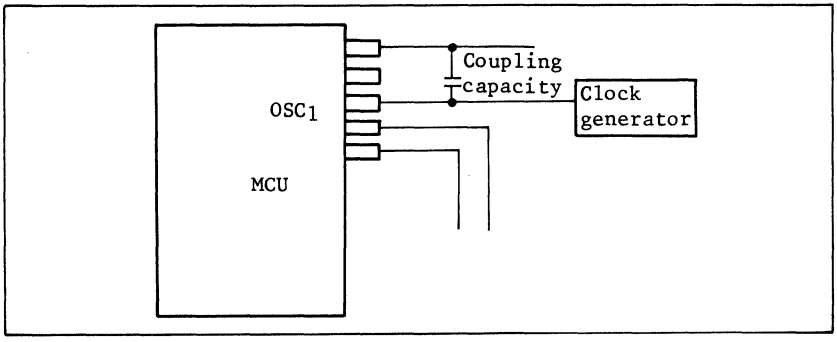


Fig. 8-4 Design of Oscillation Circuit Board (II)

8.5 Automatic Paging Facility of Cross Assembler for the HMCS400 Series

The following cross assemblers for the HMCS400 series have an automatic paging facility.

(1) The S400XAS2F cross assembler for the H68SD operates under FDOS-III or FDOS-II with its host computer Hitachi development system H68SD5A, H68SD5 or H68SD20.

(2) The S400XAS6F cross assembler for the H680SD200 operates under CP/M-68K* with its host computer Hitachi development system H68SD200.

(3) The S400MDS1F and S400MDS2F cross assemblers for MDS operate under ISIS-II or CP/M* with their host computer Intel development system MDS-220/230. Owing to the automatic paging facility, programmers can use conditional branch instructions for any size ROM without being concerned with page (256 words/page) boundaries. Two conditional branch instructions are available for the HMCS400 series : BR and BRL. The above cross assemblers have a unique instruction which has no object code, i.e., BRS. With the BRS instruction, the object code is automatically converted to BR if the destination address is within the current page and to BRL if it is not within the page. With such object code generated, the mnemonic of the instruction in the source statement remains "BRS".

* CP/M-68K and CP/M are registered trademarks of Digital Research.

Destination / Instruction	Within the page (1-word instruction)	Out of the page (2-word instruction)
BRS	BR	BRL

(4) When a BRS instruction appears, object code for BR is generated if the destination address is within the current page and that of BRL if it is not within the page.

(5) The BRS operand is a symbol name having a ROM address.

```
ST-NO   OBJECT      ADRS    SOURCE      STATEMENTS
00001                       LBL1        EQU     $1
00002                       LBL2        EQU     $100
00003   301         0000                BRS     LBL1 ..... (1)
00004   170  100    0001                BRS     LBL2 ..... (2)
00005                       *
00006                                   ORG     $FF
00007   300         00FF                BRS     LBL2 ..... (3)
00008                       *
00009                                   ORG     $1FF
00010   170  100    01FF                BRS     LBL2 ..... (4)
00011                       *
00012   302         0201                BRS     LBL3
00013                                               ..... (5)
00014   231         0202    LBL3        LAI     $1
00015                       *
00016                                   END
```

Fig. 8-5 Example of Automatic Paging

(1) Object code for the BR instruction is generated if LBL1 is in the current page.

(2) Object code for the BRL instruction is generated if LBL2 is not in the current page.

(3) Object code for the BR instruction is generated because BRS is on the page boundary and LBL2 is in the following page.

(4) Object code for the BRL instruction is generated because BRS is on the page boundary and LBL2 is not in the following page.

(5) If the number of automatic paging operations exceeds 255, the assembler outputs an error message and terminates the process. In this case the destination address of the BRS instruction is undefined.

8.6 Precautions for Port Mode Register (PMR) Setting

IF0 and IF1 (External Interrupt Request Flags) are set if $R_{32}/\overline{INT_0}$ and $R_{33}/\overline{INT_1}$ programmed as $\overline{INT_0}$ input pin, $\overline{INT_1}$ input pin, respectively by setting the bit 2 and 3 of PMR to 1 during $R_{32}/\overline{INT_0}$ and $R_{33}/\overline{INT_1}$ input level LOW. Thus, PMR should be specified as the external interrupt input pin ($\overline{INT_0}$ or $\overline{INT_1}$) only in external interrupt disable state. Then IF0 or IF1 should be reset.

Although PMR is set as external interrupt input during $R_{32}/\overline{INT_0}$ and $R_{33}/\overline{INT_1}$ input level high, IF0 and IF1 cannot be set.

External interrupt	Notes for PMR setting
$\overline{INT_0}$	IF0 is set if $R_{32}/\overline{INT_0}$ pin is programmed as $\overline{INT_0}$ input pin by setting bit 2 of PMR to 1 during $R_{32}/\overline{INT_0}$ input level low.
$\overline{INT_1}$	IF1 is set if $R_{33}/\overline{INT_1}$ pin is programmed as $\overline{INT_1}$ input pin by setting bit 3 of PMR to 1 during $R_{33}/\overline{INT_1}$ input level low.

The following examples shows the programming example of PMR for $\overline{INT_0}$.

(Example 1) PMR is set in main routine

```
        .
        .

SEMD IM0, $000          INT0 MASK
LMID %0100, PMR         R32=INT0
REMD IF0, $000          IF0 RESET
REMD IM0, $000          INT0 ENABLE
        .
        .
```

(Example 2) PMR is set in $\overline{INT_0}$ interrupt routine

```
        .
        .
LMID %0000, PMR         R32=R32
        .
        .
LMID %0100, PMR         R32=INT0
REMD IF0, $0000         IF0 RESET
        .
        .
```

9. Difference between EPROM in-package type, EPROM on-package type and Mask ROM type

Items \ Type Name		EPROM in package			EPROM on package	
		HD4074008			HD614P080S HD614P0160S	HD614P180 HD40P4181
Typical instruction exection time		1 µs			1.33 µs	1.33 µs/1 µs
Power supply voltage [V]		4.5 - 5.5			4.5 - 5.5	4.5 - 5.5
ROM		8192 words×10 bits PROM			4096 words×10 bits 8192 words×10 bits	(EPROM 2764) (EPROM 27128,27256)
RAM		512 digits×4 bits			576 digits×4 bits	576 digits×4 bits 992 digits×4 bits
I/O pin circuit	Standard pins	NMOS Open drain*1			NMOS Open drain	NMOS Open drain
	High voltage pins	PMOS Open drain*2			PMOS Open drain	PMOS Open drain
Clock generation	Crystal	o			o	o
	Ceramic	o			o	o
	Resistance	-			-	-
Package		Shrink type 64-pin dual-in-line plastic package. 64-pin flat plastic package. Shrink type 64-pin dual-in-line ceramic package.			Shrink type 64-pin EPROM on package.	42 pin EPROM on package.
	Type	DC-64S	DP-64S	FP-64	DC-64SP	DC-42P
	Occupied area	18.8×57.3	17×58	19.6×25.6	23×57.3	19×52.8
	High from stand-off (mm)	5.1 (max)	5.1 (max)	2.9 (max)	7.5 (max) EPROM on-package	

*1, *2 Typical 5V use

Mask ROM				
HMCS402AC HMCS402C HMCS402CL	HMCS404AC HMCS404C HMCS404CL	HMCS408AC HMCS408C HMCS408CL	HMCS412AC HMCS412C HMCS412CL	HMCS414AC HMCS414C HMCS414CL
1.33 µs 2 µs 4 µs	1.33 µs 2 µs 4 µs	1 µs 2 µs 4 µs	1 µs 2 µs 4 µs	1 µs 2 µs 4 µs
4.5 - 6 4 - 6 2.7 - 6	4.5 - 6 4 - 6 2.7 - 6	4.5 - 6 3.5 - 6 2.5 - 6	4.5 - 6 3.5 - 6 2.5 - 6	4.5 - 6 3.5 - 6 2.5 - 6
2048 words×10 bits	4096 words×10 bits	8192 words×10 bits	2048 words×10 bits	4096 words×10 bits
160 digits×4 bits	256 digits×4 bits	512 digits×4 bits	160 digits×4 bits	160 digits×4 bits
Each pin selects "without pull-up MOS (NMOS open drain)", "with pull-up MOS", or "CMOS".				
Each pin selects "without pull-down MOS (PMOS open drain)" or "with pull-down MOS".				
o	o	o	o	o
o	o	o	o	o
HMCS402C	HMCS404C	-	-	-
Shrink type 64 pin dual-in-line plastic package. 64 pin flat plastic package.			42 pin dual-in-line plastic package. Shrink type 42 pin dual-in-line plastic package.	
DP-64S, FP-64			DP-42, DP-42S	
17×58, 19.4×25.6			13.4×52.8, 14×37.34	
5.1 (max), 2.9 (max)			5.08 (max)	

10. EPROM IN PACKAGE TYPE SINGLE CHIP MICROCOMPUTER HD4074008

Under Development

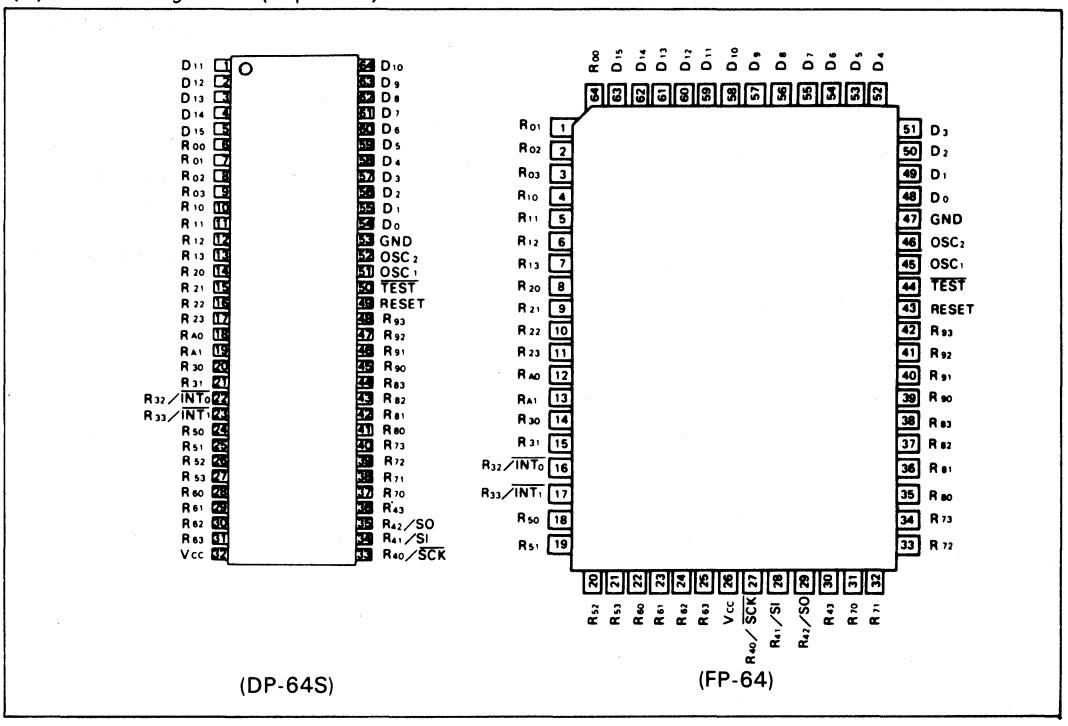
10.1 Overview

The HD4074008 is a mass storage ZTAT microcomputer incorporating 8k words of programmable ROM and 512 digits of RAM. It is a CMOS 4-bit single-chip microcomputer which is a member of the HMCS400 series microcomputers providing the characteristics of high program productivity, high speed operation, and low power dissipation.

(1) Features

- Instruction Set Compatible with the HMCS402/404/408
- 8,192 words × 10 bits programmable ROM (Program spec. is compatible with the 27256 type)
- 512 digits × 4 bits RAM
- 58 I/O Lines Including 12 Large Current Pins (15mA), I/O Pin Circuit Type; Open Drain with 5 Voltage use.
- Two On-chip Timer/Counters
- Clock Synchronous 8-bit Serial Interface
- Five Interrupt Sources
  - External 2
  - Internal 3
- Subroutine Stack
  - Up to 16 levels including Interrupts
- Two Low Power Dissipation Mode
  - Standby Mode
  - Stop Mode
- On-chip Oscillator
  - Crystal or Ceramic Filter (Externally drivable)
- Minimum Instruction Cycle Time 0.89μs
- Operation Mode
  - MCU Mode
  - PROM Mode
- Package
  - 64-Pin Shrink Type Plastic DIP
  - 64-Pin Shrink Type Ceramic DIP with Window
  - 64-Pin Flat Plastic Package

(2) Pin Arrangement (Top View)



(3) Block Diagram

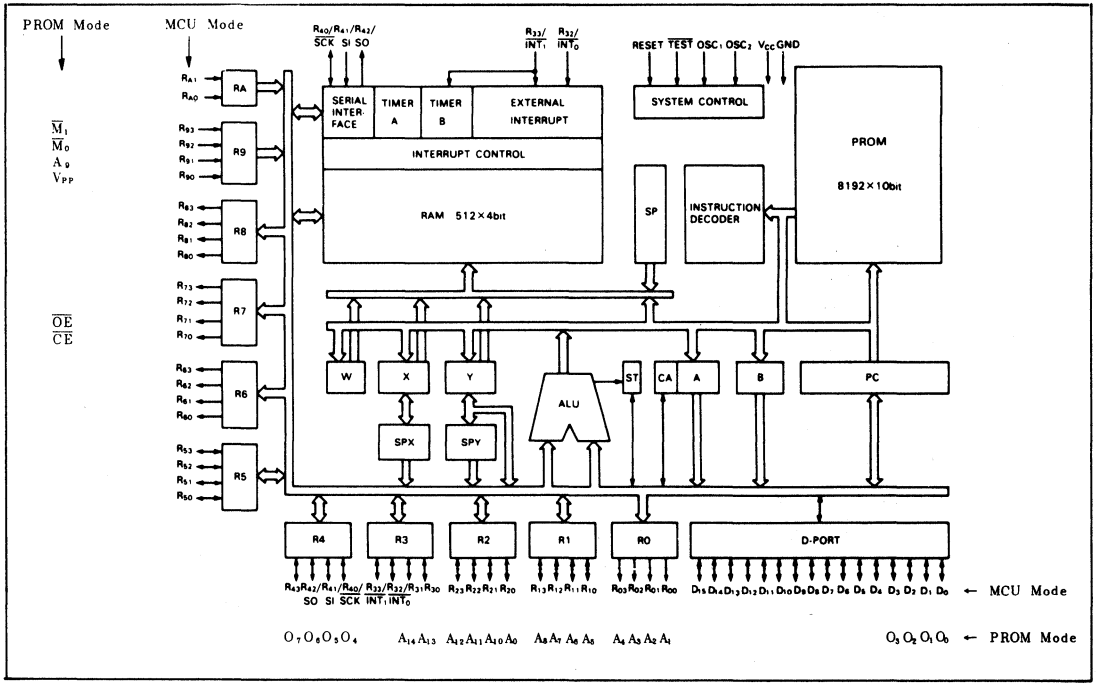


Fig. 10-1 Block Diagram

(4) Pin Function

Pin arrangement depending on the mode

Pin No.		MCU Mode		PROM Mode		Pin No.		MCU Mode		PROM Mode	
DC-64S DP-64S	FP-64	Symbol	I/O	Symbol	I/O	DC-64S DP-64S	FP-64	Symbol	I/O	Symbol	I/O
1	59	D_{11}	I/O	V_{CC}		33	27	$R_{40}/\overline{SCK}$	I/O	O_4	I/O
2	60	D_{12}	I/O			34	28	R_{41}/SI	I/O	O_5	I/O
3	61	D_{13}	I/O			35	29	R_{42}/SO	I/O	O_6	I/O
4	62	D_{14}	I/O			36	30	R_{43}	I/O	O_7	I/O
5	63	D_{15}	I/O			37	31	R_{70}	O	$\overline{CE}$	I
6	64	R_{00}	O	A_1	I	38	32	R_{71}	O	$\overline{OE}$	I
7	1	R_{01}	O	A_2	I	39	33	R_{72}	O		
8	2	R_{02}	O	A_3	I	40	34	R_{73}	O		
9	3	R_{03}	O	A_4	I	41	35	R_{80}	O		
10	4	R_{10}	I/O	A_5	I	42	36	R_{81}	O		
11	5	R_{11}	I/O	A_6	I	43	37	R_{82}	O		
12	6	R_{12}	I/O	A_7	I	44	38	R_{83}	O		
13	7	R_{13}	I/O	A_8	I	45	39	R_{90}	I	V_{PP}	
14	8	R_{20}	I/O	A_0	I	46	40	R_{91}	I	A_9	I
15	9	R_{21}	I/O	A_{10}	I	47	41	R_{92}	I	$\overline{M_0}$	I
16	10	R_{22}	I/O	A_{11}	I	48	42	R_{93}	I	$\overline{M_1}$	I
17	11	R_{23}	I/O	A_{12}	I	49	43	RESET	I	RESET	I
18	12	R_{A0}	I			50	44	$\overline{TEST}$	I	$\overline{TEST}$	I
19	13	R_{A1}	I			51	45	OSC_1	I		
20	14	R_{30}	I/O	A_{13}	I	52	46	OSC_2	O		
21	15	R_{31}	I/O	A_{14}	I	53	47	GND	I/O	GND	
22	16	$R_{32}/\overline{INT_0}$	I/O			54	48	D_0	I/O	O_0	I/O
23	17	$R_{33}/\overline{INT_1}$	I/O			55	49	D_1	I/O	O_1	I/O
24	18	R_{50}	I/O			56	50	D_2	I/O	O_2	I/O
25	19	R_{51}	I/O			57	51	D_3	I/O	O_3	I/O
26	20	R_{52}	I/O			58	52	D_4	I/O		
27	21	R_{53}	I/O			59	53	D_5	I/O		
28	22	R_{60}	O			60	54	D_6	I/O		
29	23	R_{61}	O			61	55	D_7	I/O		
30	24	R_{62}	O			62	56	D_8	I/O		
31	25	R_{63}	O			63	57	D_9	I/O		
32	26	V_{CC}		V_{CC}		64	58	D_{10}	I/O	V_{CC}	

(Note) I/O : Input/Output Pins
I : Input Pins
O : Output Pins

(5) Pin Description

The MCU input and output signals are described below.

- GND, V_{CC}

  These are the power supply pins for the MCU. Connect the GND to the ground (0V) and apply the V_{CC} power supply voltage to the V_{CC} pin.

- $\overline{TEST}$

  This pin is not for use by users. It should be connected to V_{CC} pin.

- RESET

  This pin is used to reset the MCU.

- OSC_1, OSC_2

  These are input pins for the internal oscillator circuit. They can be connected to the crystal resonator, ceramic filter resonator, or external oscillator circuits. The internal oscillator should be selected using a mask option.

- D-port

  The D-port is input/output port addressed by one bit. All pins (D_0-D_{15}) are I/O pins. The pins D_0 to D_3 are standard pins and their circuit type is NMOS open drain. The pins D_4 to D_{15} are large current standard pins, and their circuit type is PMOS open drain.

- R-ports (R0 to RA)

  These are 4-bit I/O ports. (RA however, is 2-bit construction.) R0, R6, R7 and R8 are output ports, R9 and RA are input ports, and R1 to R5 are I/O ports. All pins of port R0-RA are standard pins.
  The circuit type of D_4-D_{15} and R0-R2 is PMOS open drain, and that of D_0-D_3 and R3-R8 is NMOS open drain. The pins R_{32}, R_{33}, R_{40}, R_{41}, and R_{42} are multiplexed with $\overline{INT_0}$, $\overline{INT_1}$, $\overline{SCK}$, SI, and SO respectively.

- $\overline{INT_0}$, $\overline{INT_1}$

  These are input pins with which MCU operations can be interrupted externally. $\overline{INT_1}$ can be used as and external event input pin for Timer B. $\overline{INT_0}$ and $\overline{INT_1}$ are multiplexed with R_{32}, R_{33} respectively.

- $\overline{SCK}$, $\overline{SI}$, SO

  The Transfer Clock I/O pin ($\overline{SCK}$), Serial Data Input pin (SI), and Serial Data Output pin (SO) are used for serial interface. $\overline{SCK}$, SI, and SO are multiplexed with R_{40}, R_{41}, and R_{42} respectively.

PROM Mode Pins

- V_{PP}

  This pin is used for applying program voltage (12.5V ±0.3V) to internal PROM.

- $\overline{CE}$

  This pin is input for programming and verifying internal PROM.

- $\overline{OE}$

  This pin is input of data output control signal for verify.

- A_0-A_{14}

  These pins are address input pins for internal PROM.

- O_0-O_7

  These are data buses for internal PROM.

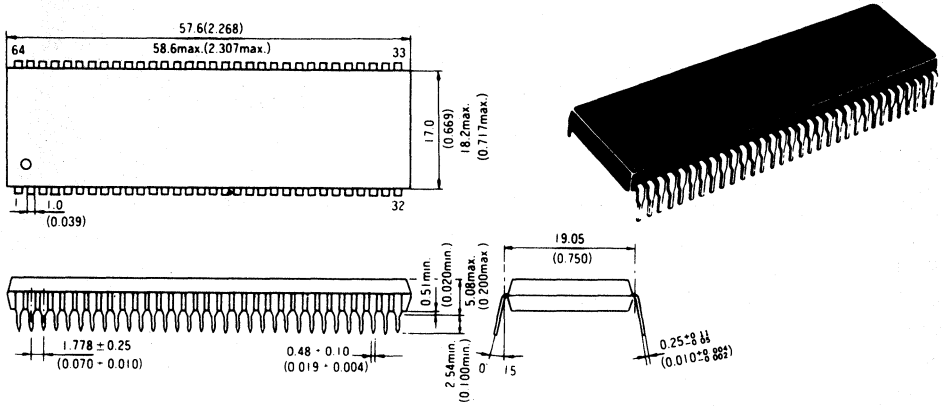
- $\overline{M_0}$, $\overline{M_1}$

  These pins are used for setting EPROM mode. EPROM mode is set when $\overline{M_0}$, $\overline{M_1}$, and $\overline{TEST}$ pins are Low level and RESET pin is High level.

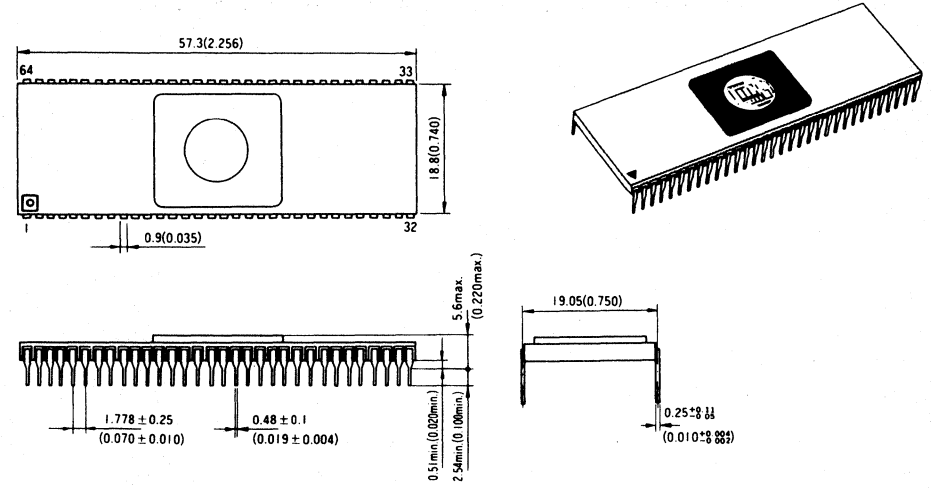
(6) Package Dimensions

Unit: mm(inch)

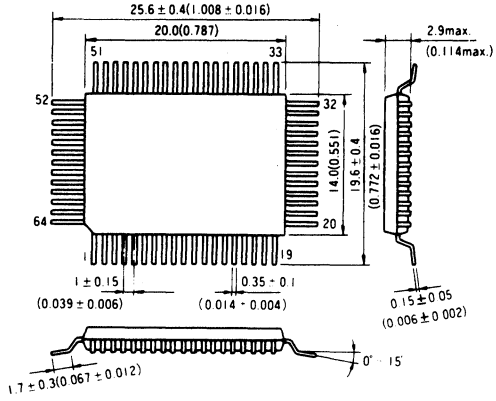
DP-64S

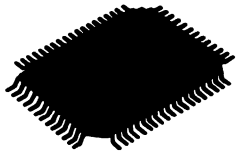


DC-64S



FP-64





10.2 ROM Memory Map

The MCU includes 8,192 words × 10 bits PROM. PROM is described in the following paragraphs and PROM Memory Map is illustrated in Fig. 10-2.

- Vector Address Area --- $0000 to $000F
  Locations $0000 through $000F are reserved for JMPL instructions to branch to the starting address of the initialization program and of the interrupt service programs. After reset of interrupt routine is serviced, the program is executed from the vector address.

- Zero-Page Subroutine Area --- $0000 to $003F
  Locations $0000 through $003F are reserved for subroutines. CAL instruction allows to branch to the subroutine.

- Pattern Area --- $0000 to $0FFF
  Locations $0000 through $0FFF are reserved for ROM data. P instruction allows referring to the ROM data as a pattern.

- Program Area --- $0000 to $1FFF

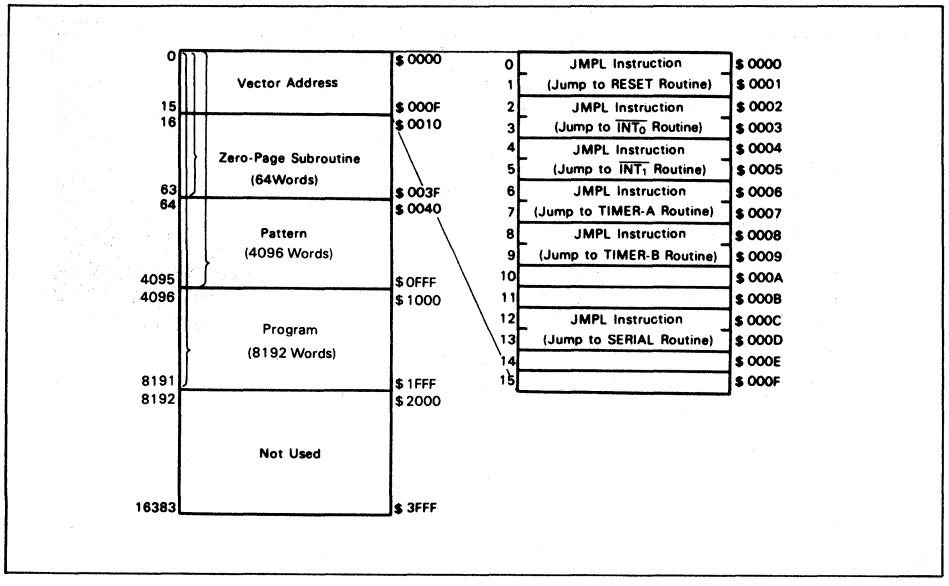


Fig. 10-2 PROM Memory Map

10.3 RAM Memory Map

The MCU includes 512 digits × 4 bits RAM as the data area and stack area. In addition to these areas, interrupt control bits and special function registers are also mapped on the RAM memory space. RAM memory map is illustrated in Fig. 10-3 and described in the following paragraphs.

- o Interrupt Control Bit Area --- $000 to $003
  This area is used for interrupt controls, and is illustrated in Fig. 10-4. It is accessable only by RAM bit manipulation instruction. However, the interrupt request flag cannot be set by software. RSP bit is used to reset the stack pointer.

- o Special Function Registers Area --- $004 to $00B
  The Special Function Registers are the mode or data registers for the external interrupt, the serial interface, and the timer/counter. These registers are classified into three types: Write-only, Read-only, and Read/Write as shown in Fig. 10-3. These registers cannot be accessed by RAM bit manipulation instruction.

o Data Area --- $020 to $1DF

16 digits of $020 through $02F are called memory register (MR) and accessable by LAMR and XMRA instructions. The configuration is shown in Fig. 10-5.

o Stack Area --- $3C0 to $3FF

Locations $3C0 through $3FF are reserved for LIFO stacks to save the contents of the program counter (PC), status (ST) and carry (CA) when interruption is serviced. This area can be used as 16 nesting level stack which one level requires 4 digits. A save condition is shown in Fig. 10-5. The program counter is restored by RTN and RTNI instructions. Status and Carry are restored only by RTNI instruction. The area, not used for stacking, is available as a data area.

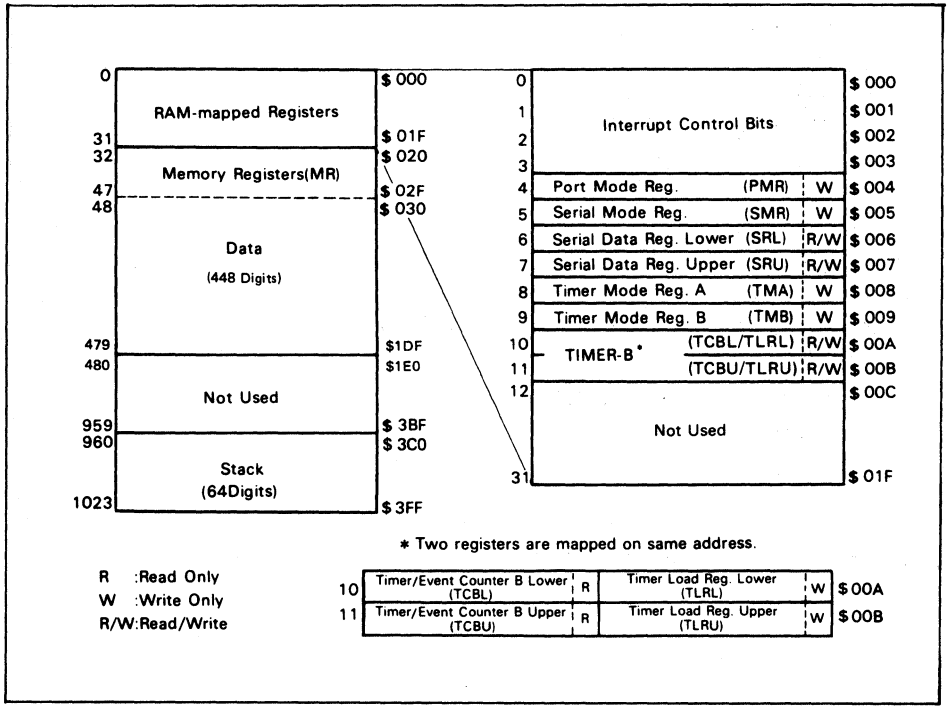


Fig. 10-3 RAM Memory Map

	bit 3	bit 2	bit 1	bit 0	
0	IM0 (IM of $\overline{INT_0}$)	IF0 (IF of $\overline{INT_0}$)	RSP (Reset SP Bit)	I/E (Interrupt Enable Flag)	$000
1	IMTA (IM of TIMER-A)	IFTA (IF of TIMER-A)	IM1 (IM of $\overline{INT_1}$)	IF1 (IF of $\overline{INT_1}$)	$001
2	Not Used	Not Used	IMTB (IM of TIMER-B)	IFTB (IF of TIMER-B)	$002
3	Not Used	Not Used	IMS (IM of SERIAL)	IFS (IF of SERIAL)	$003

IF : Interrupt Request Flag
IM : Interrupt Mask
I/E : Interrupt Enable Flag
SP : Stack Pointer
(Note) Each bit in Interrupt Control Bits Area is set by SEM/SEMD instruction, is reset by REM/REMD instruction and is tested by TM/TMD instruction. It is not affected by other instructions. Furthermore, Interrupt Request Flag is not affected by SEM/SEMD instruction. The content of Status becomes invarid when "Not Used" bit is tested.

Fig. 10-4 Configuration of Interrupt Control Bit Area

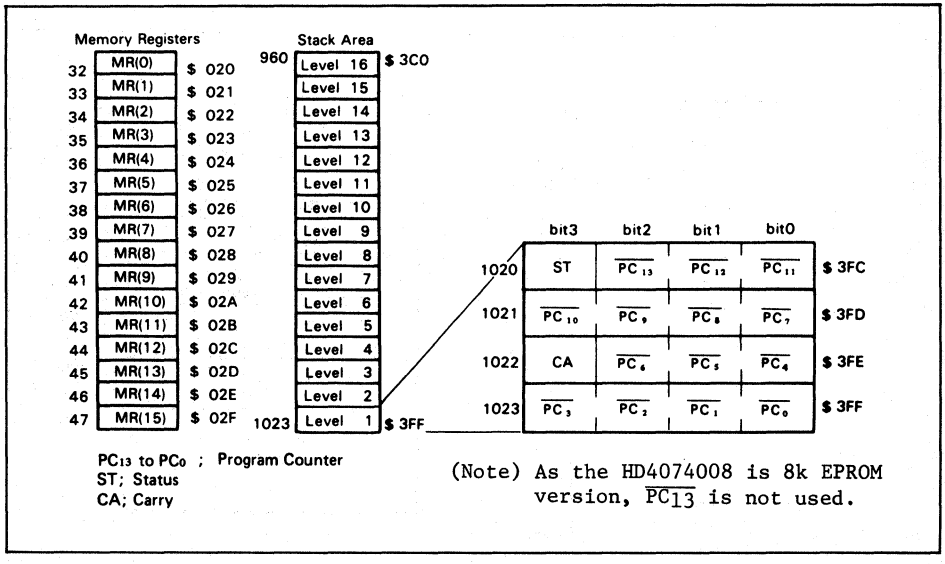


Fig. 10-5 Configuration of Memory Register, Stack Area and Stack Position

10.4 Absolute Maximum Ratings

Item	Symbol	Value	Unit	Note
Supply Voltage	V_{CC}	−0.3 to +7.0	V	
Pin Voltage	V_T	−0.3 to V_{CC} +0.3	V	3
Total Allowance of Input Currents	ΣI_O	50	mA	4
Total Allowance of Output Currents	$-\Sigma I_O$	150	mA	5
Maximum Input Current	I_O	15	mA	6, 7
Maximum Output Current	$-I_O$	4	mA	8, 9
		6	mA	8, 10
		30	mA	8, 11
Operating Temperature	T_{opr}	−20 to +75	°C	
Storage Temperature	T_{stg}	−55 to +125	°C	

(Note 1) Permanent damage may occur if "Absolute Maximum Ratings"of the LSI or the EPROM are exceeded. Normal operation should be under the conditions of "Electrical Characteristics". If these conditions are exceeded, it may cause the malfunction and affect the reliability of LSI.
(Note 2) All voltages are with respect to GND.
(Note 3) Applied to standard pins.
(Note 4) Total allowance of input current is the total sum of input current which flow in from all I/O pins to GND simultaneously.
(Note 5) Total allowance of output current is the total sum of the output current which flow out from V_{CC} to all I/O pins simultaneously.
(Note 6) Maximum input current is the maximum amount of input current from each I/O pin to GND.
(Note 7) Applied to $D_0 \sim D_3$ and R3 ~ R8.
(Note 8) Maximum output current is the maximum amount of output current from V_{CC} to each I/O pin.
(Note 9) Applied to $D_0 \sim D_3$ and R3 ~ R8.
(Note 10) Applied to R0 ~ R2.
(Note 11) Applied to $D_4 \sim D_{15}$.

10.5 HD4074008 Electrical Characteristics

(1) DC Characteristics (V_{CC} = 5V ± 10%, GND = 0V, Ta = –20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions	Value min	Value typ	Value max	Unit	Note
Input "High" Voltage	V_{IH}	$\overline{RESET}$, $\overline{SCK}$, $\overline{INT_0}$, $\overline{INT_1}$		0.8V_{CC}	–	V_{CC}+0.3	V	
		SI		0.7V_{CC}	–	V_{CC}+0.3	V	
		OSC_1		V_{CC}– 0.5	–	V_{CC}+0.3	V	
Input "Low" Voltage	V_{IL}	$\overline{RESET}$, $\overline{SCK}$, $\overline{INT_0}$, $\overline{INT_1}$		–0.3	–	0.2V_{CC}	V	
		SI		–0.3	–	0.2V_{CC}	V	
		OSC_1		–0.3	–	0.5	V	
Output "High" Voltage	V_{OH}	$\overline{SCK}$, SO	$-I_{OH}$ = 1.0 mA	V_{CC}–1.0	–	–	V	
			$-I_{OH}$ = 0.01 mA	V_{CC}–0.5	–	–	V	
Output "Low" Voltage	V_{OL}	$\overline{SCK}$, SO	I_{OL} = 1.6 mA	–	–	0.4	V	
Input/Output Leakage Current	$\|I_{IL}\|$	$\overline{RESET}$, $\overline{SCK}$, $\overline{INT_0}$, $\overline{INT_1}$, SI, SO, OSC_1	V_{in} = 0V to V_{CC}	–	–	1	μA	1
Current Dissipation in Active Mode	I_{CC}	V_{CC}	V_{CC}=5V	–	–	4.5	mA	2, 4
Current Dissipation in Standby Mode	I_{SBY}	V_{CC}	Maximum Logic Operation V_{CC} = 5V	–	–	1.7	mA	3, 4
Current Dissipation in Stop Mode	I_{stop}	V_{CC}	$V_{in}(\overline{TEST})$ = V_{CC}-0.3V to V_{CC} V_{in}(RESET) = 0V to 0.3V	–	–	10	μA	
Stop Mode Retain Voltage	V_{stop}	V_{CC}		2	–	–	V	

(Note 1) Pull-up MOS current and output buffer current are excluded.

(Note 2) The MCU is in the reset state. The input/output current does not flow.
Test Conditions: MCU state; • Reset state in Operation Mode
Pin state; • RESET, $\overline{TEST}$ ··· V_{CC} voltage
• D_0~D_3, R3~R9 ··· V_{CC} voltage
• D_4~D_{15}, R0~R2, R_{A0}, R_{A1} ··· GND voltage

(Note 3) The timer/counter operate and input/output current does not flow.
Test Conditions: MCU state; • Standby Mode
• Input/Output; Reset state
• SERIAL Interface ; Stop
Pin state; • RESET ···GND voltage
• $\overline{TEST}$ ··· V_{CC} voltage
• D_0~D_3, R3~R9 ··· V_{CC} voltage
• D_4~D_{15}, R0~R2, R_{A0}, R_{A1} ··· GND voltage

(Note 4) When f_{osc}=x[MHz], the Current Dissipation in Operation mode and Standby mode are estimated as follows:

$$\text{max. value } (f_{osc}=x\text{[MHz]}) = \frac{x}{8} \times \text{max. value } (f_{osc}=8\text{[MHz]})$$

(2) Input/Output Characteristics for standard pin – 1
(V_{CC} = 5V ± 10%, GND = 0V, Ta = –20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Input "High" Voltage	V_{IH}	$D_0 \sim D_3$, R3 ~ R5, R9		0.7V_{CC}	–	V_{CC}+0.3	V	
Input "Low" Voltage	V_{IL}	$D_0 \sim D_3$, R3 ~ R5, R9		–0.3	–	0.3V_{CC}	V	
Output "Low" Voltage	V_{OL}	$D_0 \sim D_3$, R3 ~ R8	I_{OL} = 1.6 mA	–	–	0.4	V	
Input/Output Leakage Current	$\|I_{IL}\|$	$D_0 \sim D_3$, R3 ~ R9	V_{in} = 0V to V_{CC}	–	–	1	μA	1

(Note 1) Output buffer current are excluded.

(3) Input/output characteristics for standard pin – 2
(V_{CC} = 5V ± 10%, GND = 0V, Ta = –20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Input "High" Voltage	V_{IH}	$D_4 - D_{15}$, R1 R2, R_{A0}, R_{A1}		0.7V_{CC}	–	V_{CC}+0.3	V	
Input "Low" Voltage	V_{IL}	$D_4 - D_{15}$, R1 R2, R_{A0}, R_{A1}		–0.3	–	0.3V_{CC}	V	
Output "High" Voltage	V_{OH}	$D_4 - D_{15}$	$-I_{OH}$ = 15mA	V_{CC}–3.0	–	–	V	
			$-I_{OH}$ = 10mA	V_{CC}–2.0	–	–	V	
			$-I_{OH}$ = 4mA	V_{CC}–1.0	–	–	V	
		R0 – R2	$-I_{OH}$ = 3mA	V_{CC}–3.0	–	–	V	
			$-I_{OH}$ = 2mA	V_{CC}–2.0	–	–	V	
			$-I_{OH}$ = 0.8mA	V_{CC}–1.0	–	–	V	
Input/Output Leakage Current	$\|I_{IL}\|$	$D_4 - D_{15}$ R0 – R2 R_{A0}, R_{A1}	V_{in} = 0 to V_{CC}	–	–	1	μA	1

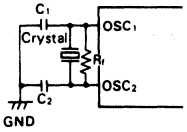
(Note 1) Output buffer current are excluded.

(4) AC Characteristics (V_{CC} = 5V ± 10%, GND = 0V, Ta = –20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Oscillation Frequency	f_{osc}	OSC_1, OSC_2	divide-by-8	0.4	8	9	MHz	
Instruction Cycle Time	t_{cyc}		divide-by-8	0.89	1	20	μs	
Oscillator Stabilization Time	t_{RC}	OSC_1, OSC_2		–	–	20	ms	1
External Clock "High" Level Width	t_{CPH}	OSC_1	divide-by-8	41	–	–	ns	2
External Clock "Low" Level Width	t_{CPL}	OSC_1	divide-by-8	41	–	–	ns	2
External Clock Rise Time	t_{CPr}	OSC_1		–	–	15	ns	2
External Clock Fall Time	t_{CPf}	OSC_1		–	–	15	ns	2
$\overline{INT_0}$ "High" Level Width	t_{I0H}	$\overline{INT_0}$		2	–	–	t_{cyc}	3
$\overline{INT_0}$ "Low" Level Width	t_{I0L}	$\overline{INT_0}$		2	–	–	t_{cyc}	3
$\overline{INT_1}$ "High" Level Width	t_{I1H}	$\overline{INT_1}$		2	–	–	t_{cyc}	3
$\overline{INT_1}$ "Low" Level Width	t_{I1L}	$\overline{INT_1}$		2	–	–	t_{cyc}	3
RESET "High" Level Width	t_{RSTH}	RESET		2	–	–	t_{cyc}	4
Input Capacitance	C_{in}	all pins	f=1MHz V_{in} = 0V	–	–	15	pF	
RESET Fall Time	t_{RSTf}			–	–	20	ms	4

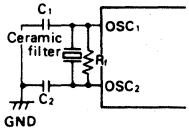
(Note 1) Oscillator stabilization time is the time until the oscillator stabilizes after V_{CC} reaches 4.5V at "Power-on", or after RESET input level goes to "High" by resetting to quit the stop mode by MCU reset on the circuits below. At power ON or recovering from stop mode, apply RESET input more than t_{RC} to obtain the necessary time for oscillator stabilization. When using crystal or ceramic filter oscillator, please ask a crystal oscillator maker's or ceramic filter maker's advice because oscillator stabilization time depends on the circuit constant and stray capacity.

Crystal oscillator



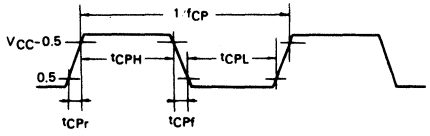
Crystal: 8.388608MHz NC-18 (Nihon Denpa Kogyo)
Rf = 1MΩ ±20%
$C_1 = C_2$ = 10pF ± 20%

Ceramic filter oscillator

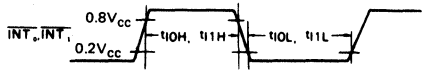


Ceramic filter: CSA8.00MT (Murata)
Rf = 1MΩ±20%
$C_1 = C_2$ = 30pF ± 20%

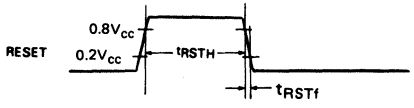
(Note 2)



(Note 3)



(Note 4)



(5) Serial Interface Timing Characteristics
(V_{CC} = 5V ± 10%, GND = 0V, Ta = −20 to +75°C, if not specified.)

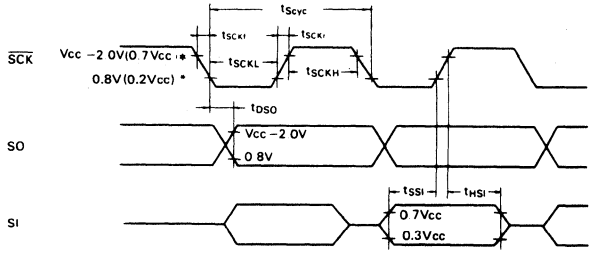
• At Transfer Clock Output

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Transfer Clock Cycle Time	t_{Scyc}	$\overline{SCK}$	(Note 2)	1	–	–	t_{cyc}	1, 2
Transfer Clock "High" Level Width	t_{SCKH}	$\overline{SCK}$	(Note 2)	0.5	–	–	t_{Scyc}	1, 2
Transfer Clock "Low" Level Width	t_{SCKL}	$\overline{SCK}$	(Note 2)	0.5	–	–	t_{Scyc}	1, 2
Transfer Clock Rise Time	t_{SCKr}	$\overline{SCK}$	(Note 2)	–	–	100	ns	1, 2
Transfer Clock Fall Time	t_{SCKf}	$\overline{SCK}$	(Note 2)	–	–	100	ns	1, 2
Serial Output Data Delay Time	t_{DSO}	SO	(Note 2)	–	–	250	ns	1, 2
Serial Input Data Set-up Time	t_{SSI}	SI		300	–	–	ns	1
Serial Input Data Hold Time	t_{HSI}	SI		150	–	–	ns	1

• At Transfer Clock Input

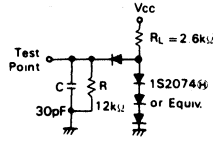
Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Transfer Clock Cycle Time	t_{Scyc}	$\overline{SCK}$		1	–	–	t_{cyc}	1
Transfer Clock "High" Level Width	t_{SCKH}	$\overline{SCK}$		0.5	–	–	t_{Scyc}	1
Transfer Clock "Low" Level Width	t_{SCKL}	$\overline{SCK}$		0.5	–	–	t_{Scyc}	1
Transfer Clock Rise Time	t_{SCKr}	$\overline{SCK}$		–	–	100	ns	1
Transfer Clock Fall Time	t_{SCKf}	$\overline{SCK}$		–	–	100	ns	1
Serial Output Data Delay Time	t_{DSO}	SO	(Note 2)	–	–	250	ns	1, 2
Serial Input Data Set-up Time	t_{SSI}	SI		300	–	–	ns	1
Serial Input Data Hold Time	t_{HSI}	SI		150	–	–	ns	1

(Note 1) Timing Diagram of Serial Interface



* V_{CC} − 2.0V and 0.8V are the threshold voltage for transfer clock output.
$0.8V_{CC}$ and $0.2\ V_{CC}$ are the threshold voltage for transfer clock input.

(Note 2) Timing Load Circuit



10.6 Programming the On-Chip Programmable ROM

The HD4074008's on-chip PROM is programmed in PROM mode. PROM mode is set by bringing $\overline{TEST}$, $\overline{M_0}$, and $\overline{M_1}$ low, and RESET high as shown in Fig. 10-7. In PROM mode the MCU does not operate. It can be programmed like a standard 27256 EPROM using a standard PROM programmer and a 64-to-28-pin socket adapter. Table 10-2 lists recommeded PROM programmers and socket adapters.

Since an instruction of the HMCS400 series consists of 10 bits, the HMCS400 series microcomputer incorporate conversion circuit to use general perpose PROM programmer. By this circuit, an instruction is read or programmed using 2 addresses, lower 5 bits and upper 5 bits as shown in Fig. 10-8. For example, if 8k words of on-chip PROM is programmed by general purpose PROM programmer, 16k bytes of addresses ($0000-$3FFF) should be specified.

Precautions

1. Addresses $0000 to $3FFF should be specified if the PROM is programmed by the PROM programmer. If addresses of $4000 or higher is accessed, the PROM may not be programmed or verified. Note that the plastic package type cannot be erased and reprogrammed. Data in unused address should be set to $FF. (Ceramic window packages can be erased and reprogrammed by ultraviolet light.)

2. Be careful that the PROM programmer, socket adapter and LSI match. Using the wrong programmer of socket adapter may cause an overvoltage and damage the LSI. Make sure that the LSI is firmly fixed in the socket adapter, and that the socket adapter is firmly fixed in the programmer.

3. The PROM should be programmed with V_{PP}=12.5V. Other PROMs use 21V. If 21V is applied to the HD4074008, the LSI may be permanently damaged. 12.5V is Intel's 27256 V_{PP}.

(1) Programming and Verification

The HD4074008 can be high-speed programmed without causing voltage stress or affecting data reliability.

Fig.10-9 is a programming flowchart, and Fig. 10-10 is a timing chart. For precautions on PROM programming, refer to "ZTAT MCU On-Chip PROM Characteristics and Precautions for Applications".

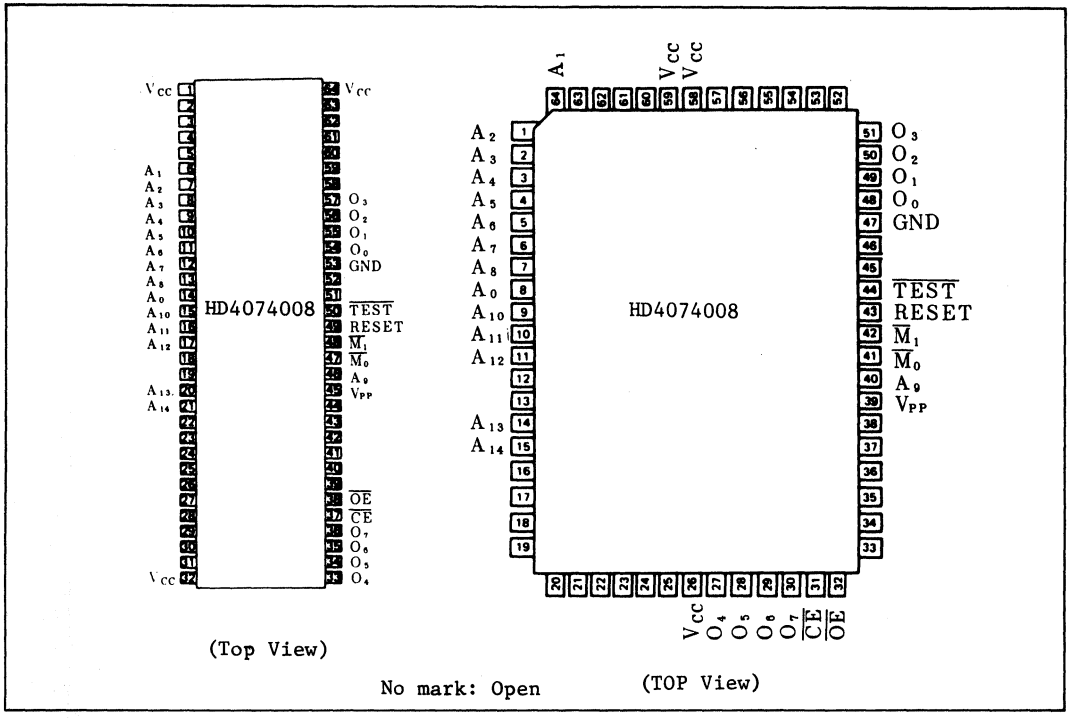


Fig. 10-6 PROM Mode Pin Arrangement

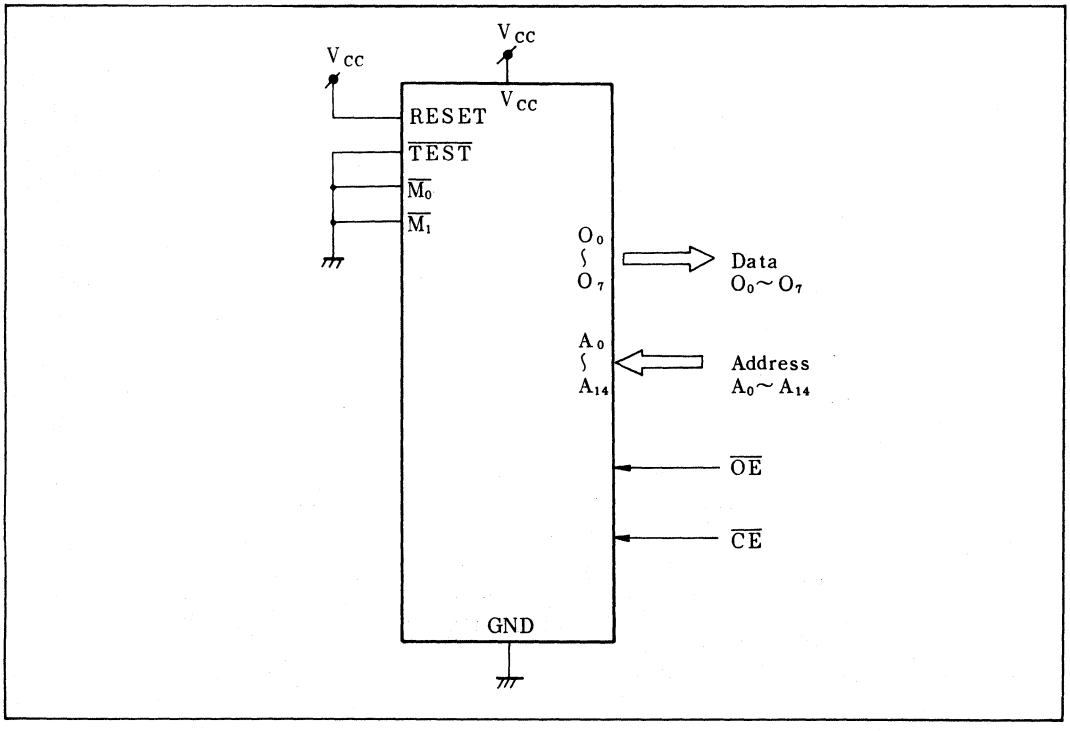


Fig. 10-7 PROM Mode Function Diagram

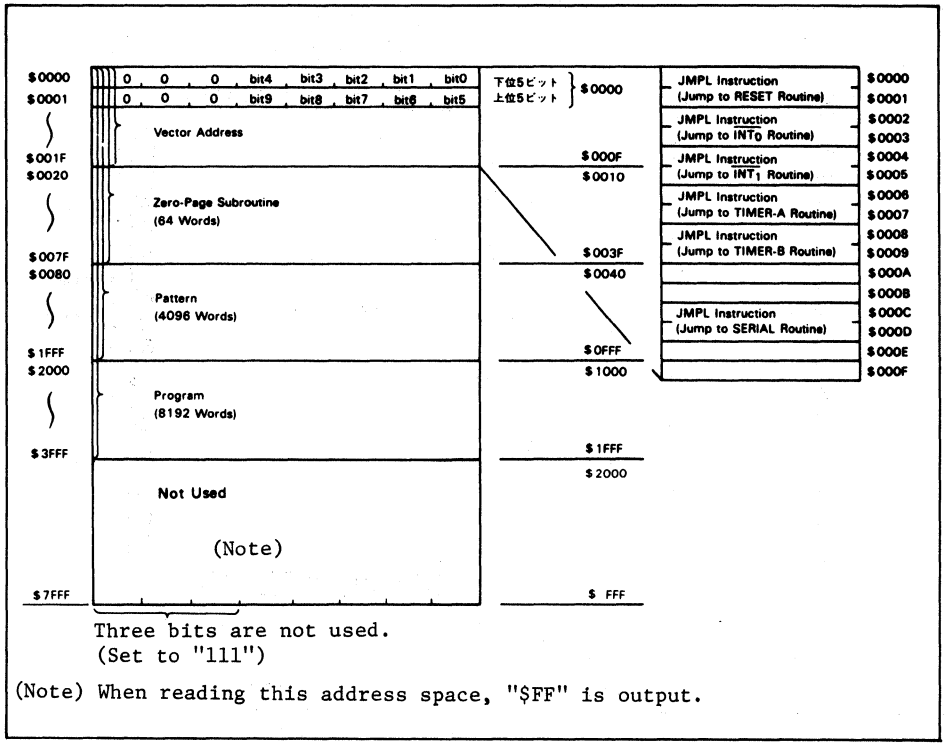


Fig. 10-8 PROM Mode Memory Map

(2) Erasing (Window package type)

The PROMs on HD4074008's in ceramic "window" packages can be erased by ultra violet light. All erased bits become ones.

Erasing conditions are: ultraviolet (UV) light with wavelength 2537A with a minimum irradiation of 15W S/cm. These conditions are satisfied by exposing the LSI to a 12,000 μW/cm UV source for 15-20 minutes, at a distance of 1 inch.

For window-type packages, refer to "Window-Type Package Precautions".

Table 10-1 Mode Selection

Mode \ Pin	$\overline{CE}$	$\overline{OE}$	V_{PP}	
Programming	Low	High	V_{PP}	Data input
Verify	High	Low	V_{PP}	Data output
Programming inhibited	High	High	V_{PP}	High impedance

Table 10-2 PROM Programmer and Socket Adapter

PROM Programmer		Socket Adapter	
Maker	Type name	Maker	Type name
DATA I/O	29A 29B	Hitachi	HS408ESS11H
AVAL Corp	PKW-1000 PKW-7000	Hitachi	HS408ESS21H

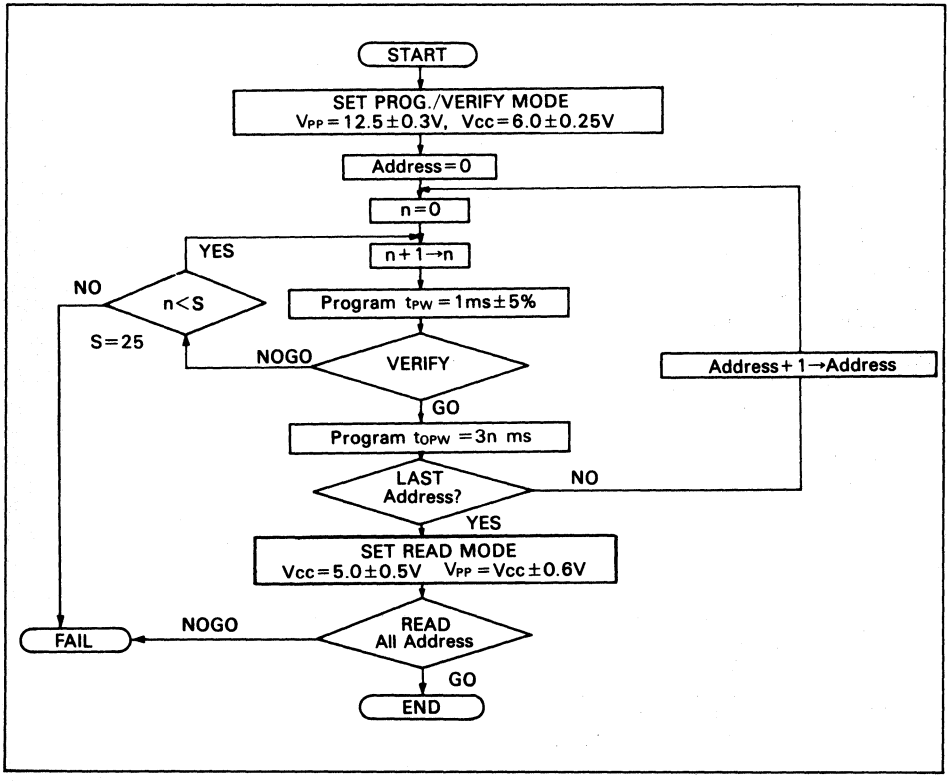


Fig. 10-9 High Speed Programming Flowchart

(3) Programming electrical characteristics

DC Characteristics (V_{CC}=6V ±0.25V, V_{PP}=12.5V ±0.3V, V_{SS}=0V, Ta=25°C ±5°C, unless otherwise noted.)

Item		Symbol	Test Condition	min	typ	max	Unit
Input High Voltage	O_0-O_7, A_0-A_{14} $\overline{OE}$, $\overline{CE}$	V_{IH}		2.2	-	V_{CC}+0.3	V
Input Low Voltage	O_0-O_7, A_0-A_{14} $\overline{OE}$, $\overline{CE}$	V_{IL}		-0.3	-	0.8	V
Output High Voltage	O_0-O_7	V_{OH}	I_{OH}=-200μA	2.4	-	-	V
Output Low Voltage	O_0-O_7	V_{OL}	I_{OL}=1.6mA	-	-	0.45	V
Input Leakage Current	O_0-O_7, A_0-A_{14} $\overline{OE}$, $\overline{CE}$	$\|I_{LI}\|$	V_{in} =5.25V/0.5V	-	-	2	μA
V_{CC} Current		I_{CC}		-	-	30	mA
V_{PP} Current		I_{PP}		-	-	40	mA

AC Characteristics (V_{CC}=6V ±0.25V, V_{PP}=12.5V ±0.3V, Ta=25°C ±5°C, unless otherwise noted.)

Item	Symbol	Test Condition	min	typ	max	Unit
Address Set-up Time	t_{AS}	Fig.10-10	2	-	-	μs
$\overline{OE}$ Set-up Time	t_{OES}		2	-	-	μs
Data Set-up Time	t_{DS}		2	-	-	μs
Address Hold Time	t_{AH}		0	-	-	μs
Data Hold Time	t_{DH}		2	-	-	μs
Output Disable Delay Time	t_{DF}		-	-	130	ns
V_{PP} Set-up Time	t_{VPS}		2	-	-	μs
Program Pulse Width	t_{PW}		0.95	1.0	1.05	ms
$\overline{CE}$ Pulse Width when Overprogramming	t_{OPW}		2.85	-	78.75	ms
V_{CC} Set-up Time	t_{VCS}		2	-	-	μs
Data Output Delay Time	t_{OE}		0	-	500	ns

Input pulse level 0.8 - 2.2V

Input rising/falling time ≦ 20ns

Timing reference level { input: 1.0V, 2.0V
output: 0.8V, 2.0V

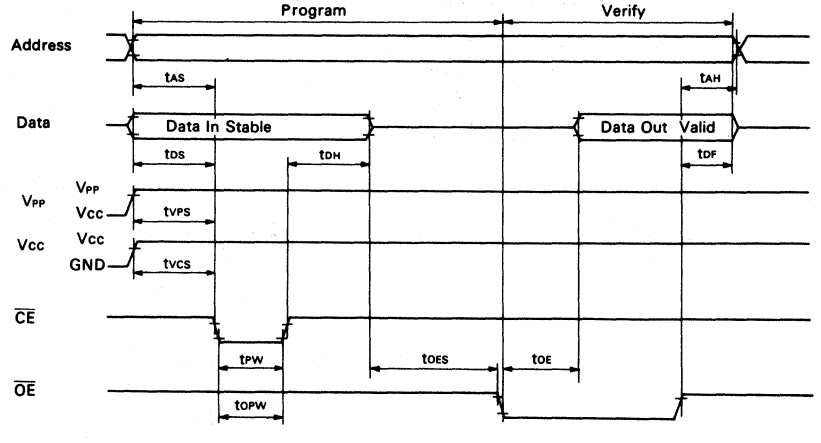


Fig. 10-10 PROM Programming/Verify Timing

10.7 ZTAT MCU On-Chip PROM Characteristics and Precautions for Applications

(1) Principles of Programming/Erasing

The HD4074008's memory cells are the same as an EPROM's. Therefore they are programmed by applying high voltage to control gates and drains, which injects hot electrons into the floating gate (Fig. 10-11). The condensed electrons in the floating gate are stable, surrounded by an energy barrier of SiO_2 film. Such a cell becomes a 0 bit due to the memory threshold voltage change. A cell with no condensed electrons at its floating gate appears as a 1 bit.

The electron charge in memory cells may decrease as time goes by. This can be caused by:

① Ultraviolet light, discharged by photo-emitting electrons (erasure principle)
② Heat, discharged by thermal emitting electrons
③ High voltage, discharged by a high electric field at the control gate or drain

If the oxide film covering a floating gate is defective, the erasure rate is great. Normally, electron erasure does not occur, because such defective devices are found and removed during testing.

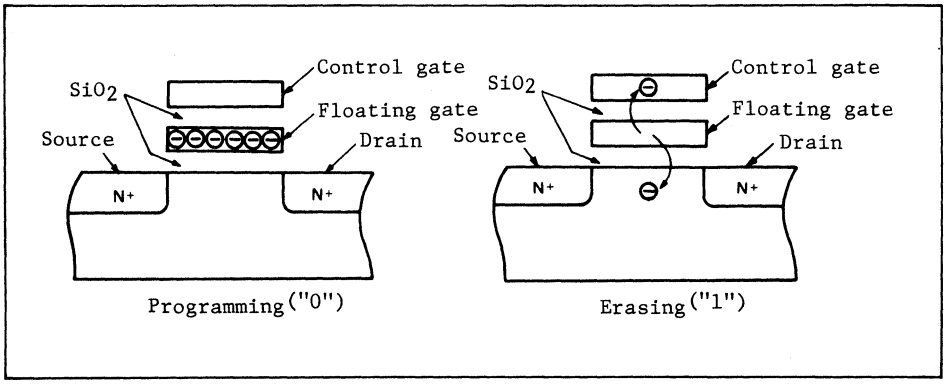


Fig. 10-11 Cross-Section of EPROM Memory Cell

(2) Programming Precautions

The PROM memory cells should be programmed under specific voltage and timing conditions. The higher the program voltage and the longer the program pulse is applied, the more electrons will be injected into the floating gate. However, if an overvoltage is applied to V_{PP}, the p-n junction may be permanently damaged. Pay particular attention to PROM programmer overshot. Negative voltage noise will cause a parasitic transistor effect, which may reduce breakdown voltage.

The HD4074008 is connected electrically to the PROM programmer through a socket adapter. Therefore, pay attention to the following:

1. Confirm that the socket adapter is firmly fixed on the PROM programmer.
2. Do not touch the socket adapter or the LSI during programming. Mis-programming can be caused by poor contacts.

(3) HD4074008 Reliability After Programming

Generally, semiconductors are reliable except for initial failures. To avoid failures, screening can be performed. Screening at high temperature removes PROM memory cells with data hold failures in a short time. This is done to the ZTAT's in the wafer stage, so ZTAT data hold characteristics are high. Exposing the LSI to 150°C after user programming can effectively up grade these characteristics. Fig. 10-12 shows the recommended screening flow.

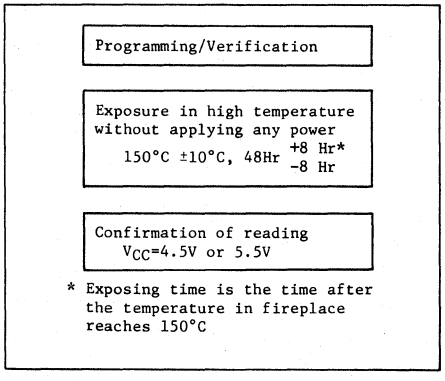


Fig. 10-12 Recommended Screening Flow

Note: If programming errors occur continuously during programming with one PROM programmer, stop programming and check the PROM programmer or socket adapter. If trouble occurs in verification after programming, or after exposure to high temperatures, please inform a Hitachi engineer.

(4) Window-Type Package Precautions

① Glass Erasure Window

If the glass window comes in contact with plastic or anything with a static charge, the LSI may malfunction due to the electrostatic charge on the surface of the window. If this occurs, exposing the LSI to ultraviolet light for a few minutes neutralizes the charge, and restores the LSI to normal operation. However, charge stored in the floating gate decreases at the same time, so reprogramming is recommended.

Electrostatic charge buildup on the window is a fundamental cause of malfunctions. Measures for its prevention are the same as those for preventing electrostatic breakdown:

Operators should be grounded when handling equipment.
Do not rub the glass window with plastics.
Be careful of coolant sprays, which may contain a few ions.
The ultraviolet shading label (which includes conductive material) effectively neutralizes charge.

② Ultraviolet Shading Label

If the LSI is exposed to fluorescent light or sunlight, its memory contents may be erased by the small quantity of ultraviolet light in these sources. In strong light, the MCU may fail under the influence of photocurrent. To prevent these problems, it is recommended that the device be used with an ultraviolet shading label covering the erasure window after programming.

Special labels are sold for this purpose. They contain metal to absorb ultraviolet light. When choosing a label, note the following:

Adhesion (mechanical intensity) - Re-use and dust reduce adhesion. Peeling off a label may cause static electricity. Therefore, erasing and rewriting is recommended after peeling. Sticking a new label over the old one is better than replacing a label.

Allowable temperature range - The allowable environmental temperature range of the label should be noted. If it is used under conditions outside this range, the paste may stiffen or adhere to the label, causing paste to remain on the window when the label is removed.

Moisture resistance - The allowable moisture range and environmental conditions of the label should be noted. It is difficult to find a shade label applicable to all conditions. The proper label should be selected depending on the intended use of the MCU.

11. EPROM ON PACKAGE TYPE SINGLE CHIP MICROCOMPUTER HD614P080S/HD614P0160S

11.1 Overview

The HD614P080S is a 4-bit single chip microcomputer which can mount a standard EPROM 2764/27128 as program memory, and a standard EPROM 27256 for the HD614P0160S.

The HD614P080S/HD614P0160S is pin-compatible with the mask ROM type HMCS402C/AC/CL, HMCS404C/AC/CL, HMCS408C/AC/CL and has the same functions except for the range of power-supply voltage, ROM/RAM capacity, mask option, and package. By modifying the program in the EPROM, they can be used for the evaluation or small scale production of the HMCS402C/AC/CL, HMCS404C/AC/CL, HMCS408C/AC/CL.

(1) Hardware Features

- 4-bit Architecture
- Capacity of Program Memory(ROM) and EPROM

  4096 words × 10 bits.....HN482764, HN27C64 } HD614P080S
  8192 words × 10 bits.....HN4827128 }
  16384 words × 10 bits.....HN27256 } HD614P0160S

- Data Memory (RAM) Capacity.....576 digits × 4 bits
- 58 I/O Pins.....26 I/O pins are high voltage (max. 40V)
- 2 Timer/Counters

  11-bit Prescaler

  8-bit Free Running Counter

  8-bit Auto-reload Timer/Event Counter

- Clocked Synchronous 8-bit Serial Interface
- 5 Interrupt sources

External	2
Timer/Counter	2
Serial Interface	1

- Subroutine Stack

  Up to 16 levels including interrupts

- Minimum Instruction Cycle Time; 1.29 µs
- 2 Low Power Dissipation Modes

  Standby - Stops instruction execution while keeping clock generator and interrupt functions.

  Stop - Stops instruction execution and clock generation while retaining RAM data

- Clock Generator

  External Connection of Crystal Resonator or Ceramic Filter Resonator (externally drivable)

- Power Voltage Range; 5V ± 10%

- I/O Pin Circuit Type

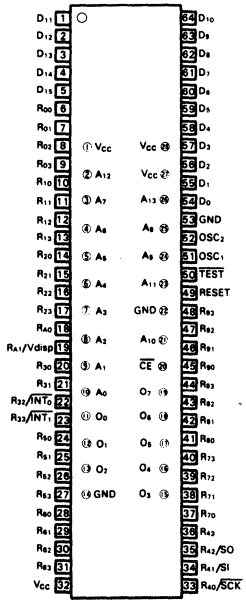
  All standard pins are "without pull-up MOS".

  All high voltage pins are "without pull-down MOS".
- Shrink Type 64 Pin EPROM On-package

(2) Software Features

- Software Compatible with HMCS402C/AC/CL, HMCS404C/AC/CL, HMCS408C/AC/CL
- Instruction Set Similar to and More Powerful than HMCS40 Series; 99 Instructions
- High Programming Efficiency with 10-bit ROM/Word; 79 instructions are single-word instructions.
- Direct Branch to ROM Area
- Direct Addressing to All RAM Area
- Subroutine Nesting Up to 16 Levels Including Interrupts
- Binary and BCD Arithmetic Operation
- Powerful Logic Arithmetic Operation
- Pattern Generation - Table Look Up Capability -
- Bit Manipulation for Both RAM and I/O

(3) Pin Arrangement



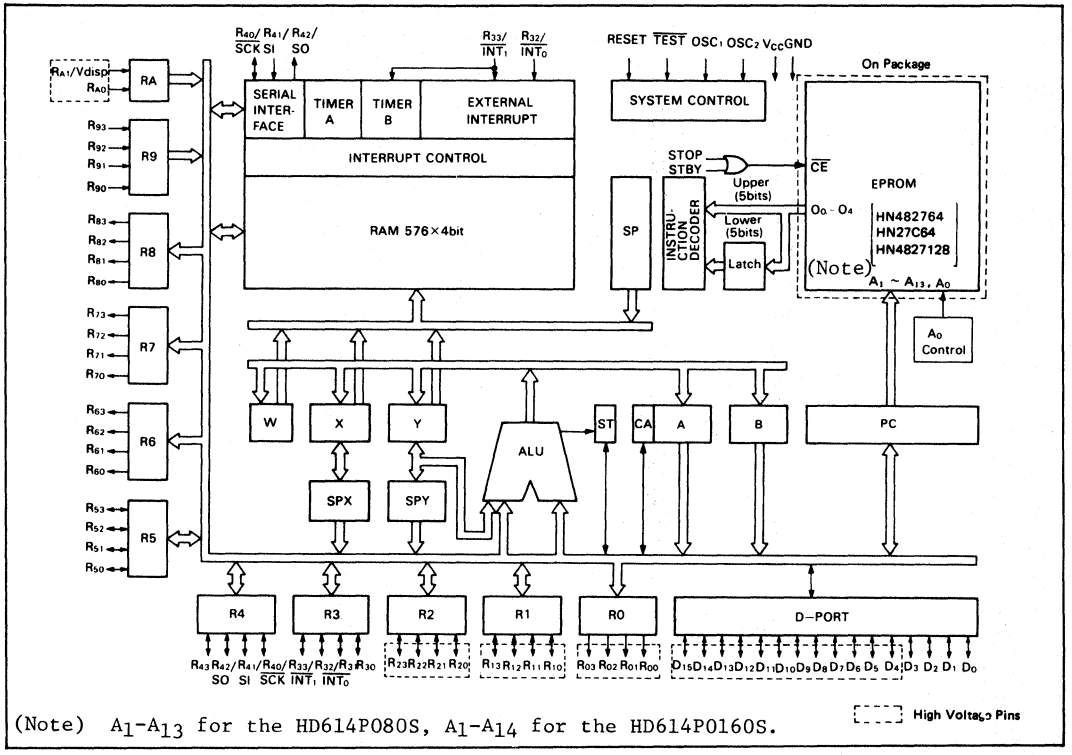
(Top View)
HD614P080S

(Top View)
HD614P0160S

(4) Recommended EPROM

Type No.	Program Memory Capacity	f_{osc} (MHz)	EPROM Type No.
HD614P080S	4096 words	4	HN27C64-30 HN482764-3
		6	HN27C64-25 HN482764
	8192 words	4	HN4827128-45
		6	HN4827128-25
HD614P0160S	16384 words	4	HN27256-30
		6	HN27256-25

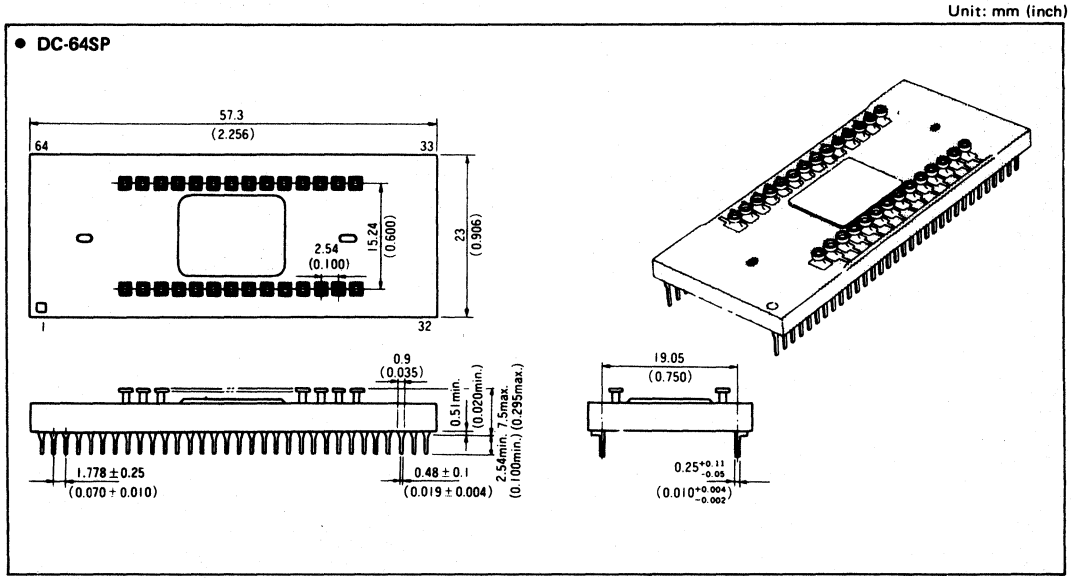
(5) Block Diagram (HD614P080S/HD614P0160S)



(Note) A_1-A_{13} for the HD614P080S, A_1-A_{14} for the HD614P0160S.

(6) Package Dimensions

Unit: mm (inch)



11.2 ROM Memory Map

ROM memory map is illustrated in Fig. 11-1 and ROM is described in the following paragraphs.

(1) Vector Address Area --- $0000 to $000F

Locations $0000 through $000F are reserved for JMPL instructions to branch to the starting address of the initialization program and of the interrupt service programs. After reset of interrupt routine is serviced, the program is executed from the vector address.

(2) Zero-Page Subroutine Area --- $0000 to $003F

Locations $0000 through $003F are reserved for subroutines. CAL instruction allows to branch to the subroutine.

(3) Pattern Area --- $0000 to $0FFF

Locations $0000 through $0FFF are reserved for ROM data. P instruction allows referring to the ROM data as a pattern.

(4) Program Area --- $0000 to $1FFF (HD614P080S)
$0000 to $3FFF (HD614P0160S)

11.3 RAM Memory Map

The MCU includes RAM as the data area and stack area. In addition to these areas, interrupt control bits and special function registers are also mapped on the RAM memory space. RAM memory map is illustrated in Fig. 11-2 and described in the following paragraphs.

(1) Interrupt Control Bit Area --- $000 to $003

This area is used for interrupt controls, and is illustrated in Fig. 11-3. It is accessable only by RAM bit manipulation instruction. However, the interrupt request flag cannot be set by software.

(2) Special Function Registers Area --- $004 to $00B

The Special Function Registers are the mode or data registers for the external interrupt, the serial interface, and the timer/counter. These registers are classified into three types: Write-only, Read-only, and Read/Write as shown in Fig. 11-2. These registers cannot be accessed by RAM bit manipulation instruction.

(3) Data Area --- $020 to $21F

16 digits of $020 through $02F are called memory register (MR) and accessable by LAMR and XMRA instructions. The configuration is shown in Fig. 11-4.

(4) Stack Area --- $3C0 to $3FF

Locations $3C0 through $3FF are reserved for LIFO stacks to save the contents of the program counter (PC), status (ST) and carry (CA) when interruption is serviced. This area can be used as 16 nesting level stack which one level requires 4 digits. A save condition is shown in Fig. 11-4. The program counter is restored by RTN and RTNI instructions. Status and Carry are restored only by RTNI instruction. The area, not used for stacking, is available as a data area.

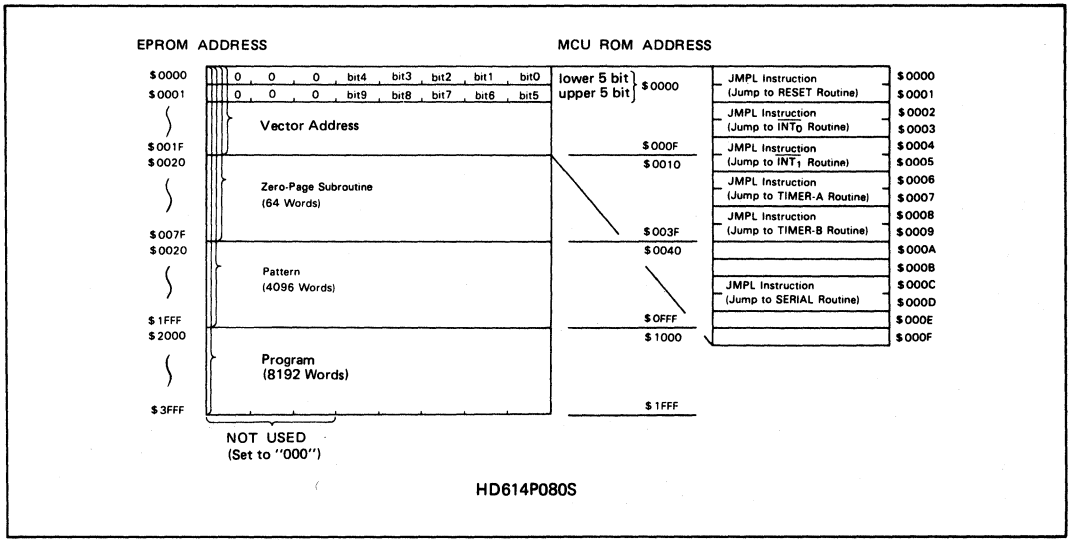
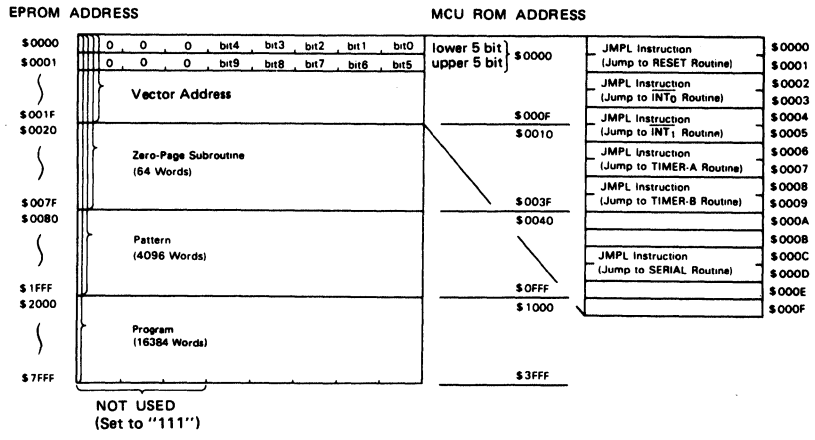
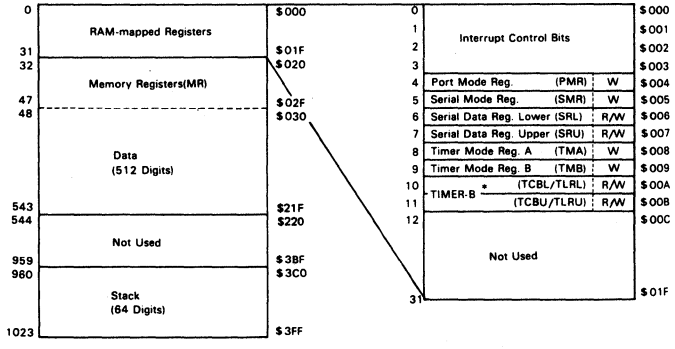


Fig. 11-1 ROM Memory Map



HD614P0160S

Fig. 11-1 ROM Memory Map



* Two registers are mapped on same address.

R : Read Only
W : Write Only
R/W : Read/Write

Timer/Event Counter B Lower (TCBL)	R	Timer Load Reg. Lower (TLRL)	W	$00A
Timer/Event Counter B Upper (TCBU)	R	Timer Load Reg. Upper (TLRU)	W	$00B

Fig. 11-2 RAM Memory Map (HD614P080S/HD614P0160S)

	bit 3	bit 2	bit 1	bit 0	
0	IM0 (IM of $\overline{INT_0}$)	IF0 (IF of $\overline{INT_0}$)	RSP (Reset SP Bit)	I/E (Interrupt Enable Flag)	$000
1	IMTA (IM of TIMER-A)	IFTA (IF of TIMER-A)	IM1 (IM of $\overline{INT_1}$)	IF1 (IF of $\overline{INT_1}$)	$001
2	Not Used	Not Used	IMTB (IM of TIMER-B)	IFTB (IF of TIMER-B)	$002
3	Not Used	Not Used	IMS (IM of SERIAL)	IFS (IF of SERIAL)	$003

IF : Interrupt Request Flag
IM : Interrupt Mask
I/E : Interrupt Enable Flag
SP : Stack Pointer
(Note) Each bit in Interrupt Control Bits Area is set by SEM/SEMD instruction, is reset by REM/REMD instruction and is tested by TM/TMD instruction. It is not affected by other instructions. Furthermore, Interrupt Request Flag is not affected by SEM/SEMD instruction. The content of Status becomes invarid when "RSP" bit and "Not Used" bit is tested.

Fig. 11-3 Configuration of Interrupt Control Bit Area

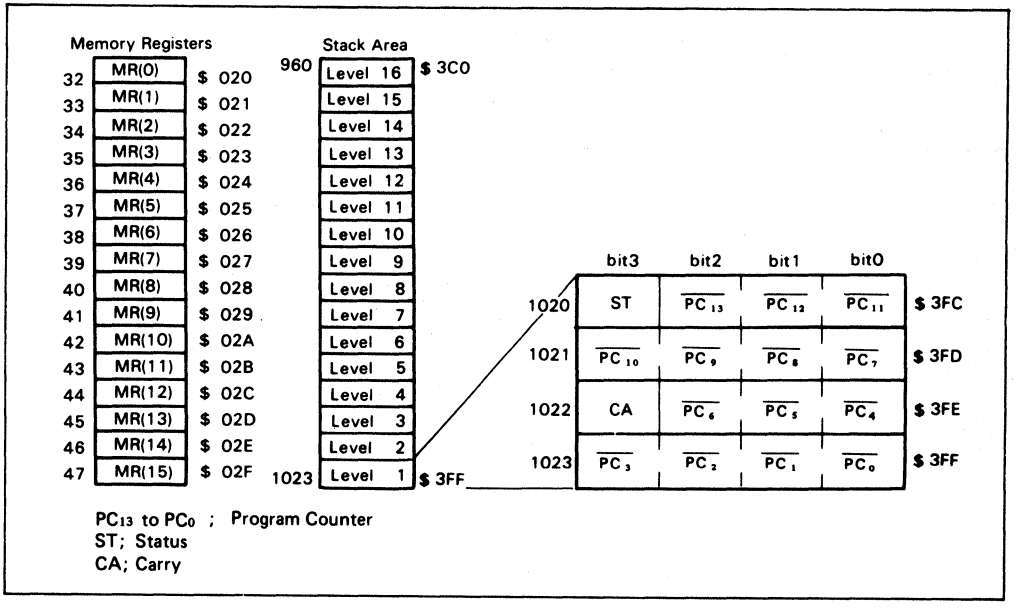


Fig. 11-4 Configuration of Memory Register, Stack Area and Stack Position

11.4 Precautions on using EPROM on Package Type Microcomputer

Since the HD614P080S/HD614P0160S has a special structure with pin sockets installed on the surface of the package, the following should be noted when using it.

(1) Do not apply an electrostatic voltage or surge voltage more than the maximum ratings to the pin socket pins. This may destroy the LSI permanently.

(2) When installing this LSI in system products in the same way as the mask ROM 4-bit single chip microcomputer, observe the following in order to maintain good ohmic contact between EPROM pins and pin sockets.

(a) When soldering the LSI on a printed circuit board, keep pin conditions under 250°C within 10 seconds. If these conditions are exceeded, the solder fixing the pin sockets may melt and the pins may fall out.

(b) Keep out detergent or coater from the pin sockets during flux removal or board coating. Flux or coater may decrease pin socket contactivity.

(c) Avoid permanent use of this LSI in places with excessive vibration.

(d) Since repeated insertion/removal of EPROMs may decrease pin sockets' contactivity, it is recommended to use new ones for your system products.

11.5 Absolute Maximum Ratings

Item	Symbol	Value	Unit	Note
Supply Voltage	V_{CC}	−0.3 to +7.0	V	
Pin Voltage	V_T	−0.3 to V_{CC} +0.3	V	3
		V_{CC} −45 to V_{CC} +0.3	V	4
Total Allowance of Input Currents	ΣI_O	50	mA	5
Total Allowance of Output Currents	$-\Sigma I_O$	150	mA	6
Maximum Input Current	I_O	15	mA	7, 8
Maximum Output Current	$-I_O$	4	mA	9, 10
		6	mA	9, 11
		30	mA	9, 12
Operating Temperature	T_{opr}	−20 to +75	°C	
Storage Temperature	T_{stg}	−55 to +125	°C	

(Note 1) Permanent damage may occur if "Absolute Maximum Ratings" of the LSI or the EPROM are exceeded. Normal operation should be under the conditions of "Electrical Characteristics". If these conditions are exceeded, it may cause the malfunction and affect the reliability of LSI.
(Note 2) All voltages are with respect to GND.
(Note 3) Applied to standard pins.
(Note 4) Applied to high voltage I/O pins.
(Note 5) Total allowance of input current is the total sum of input current which flow in from all I/O pins to GND simultaneously.
(Note 6) Total allowance of output current is the total sum of the output current which flow out from V_{CC} to all I/O pins simultaneously.
(Note 7) Maximum input current is the maximum amount of input current from each I/O pin to GND.
(Note 8) Applied to D_0 − D_3 and R3 − R8.
(Note 9) Maximum output current is the maximum amount of output current from V_{CC} to each I/O pin.
(Note 10) Applied to D_0 − D_3 and R3 − R8.
(Note 11) Applied to R0 − R2.
(Note 12) Applied to D_4 − D_{15}.

11.6 HD614P080S/HD614P0160S Electrical Characteristics

(1) DC Characteristics

(V_{CC}=4.5V to 5.5V, GND=0V, Ta=-20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions		Value min	Value typ	Value max	Unit	Note
Input "High" Voltage	V_{IH}	$\overline{RESET}$, $\overline{SCK}$, $\overline{INT_0}$, $\overline{INT_1}$			0.7V_{CC}	–	V_{CC}+0.3	V	
		SI			0.7V_{CC}	–	V_{CC}+0.3	V	
		OSC_1			V_{CC}-0.5	–	V_{CC}+0.3	V	
Input "Low" Voltage	V_{IL}	$\overline{RESET}$, $\overline{SCK}$, $\overline{INT_0}$, $\overline{INT_1}$			-0.3	–	0.22V_{CC}	V	
		SI			-0.3	–	0.22V_{CC}	V	
		OSC_1			-0.3	–	0.5	V	
Output "High" Voltage	V_{OH}	$\overline{SCK}$, SO	$-I_{OH}$ = 1.0 mA		V_{CC}-1.0	–	–	V	
			$-I_{OH}$ = 0.01 mA		V_{CC}-0.3	–	–	V	
Output "Low" Voltage	V_{OL}	$\overline{SCK}$, SO	I_{OL} = 1.6 mA		–	–	0.4	V	
Input/Output Leakage Current	$\|I_{IL}\|$	$\overline{RESET}$, $\overline{SCK}$, $\overline{INT_0}$, $\overline{INT_1}$, SI, SO, OSC_1	V_{in} = 0V to V_{CC}		–	–	1	μA	1
Current Dissipation in Operation Mode	I_{CC}	V_{CC}	V_{CC} = 5V	Crystal or Ceramic Filter Resonator f_{osc} = 4MHz	–	–	2.0	mA	2, 5
Current Dissipation in Standby Mode	I_{SBY1}	V_{CC}	Maximum Logic Operation V_{CC} = 5V	Crystal or Ceramic Filter Resonator f_{osc} = 4MHz	–	–	1.2	mA	3, 5
	I_{SBY2}	V_{CC}	Minimum Logic Operation V_{CC} = 5V	Crystal or Ceramic Filter Resonator f_{osc} = 4MHz	–	–	0.9	mA	4, 5
Current Dissipation in Stop Mode	I_{stop}	V_{CC}	V_{in} (TEST) = V_{CC} ~ V_{CC}-0.3V V_{in} (RESET) = 0 ~ 0.3V		–	–	10	μA	
Stop Mode Retain Voltage	V_{stop}	V_{CC}			2.0	–	–	V	

(Note 1) Output buffer current are excluded.

(Note 2) The MCU is in the reset state. The input/output current does not flow.

Test Conditions: MCU state;
- Reset state in Operation Mode

Pin state;
- RESET, $\overline{TEST}$ – V_{CC} voltage
- D_0–D_3, R3–R9 – V_{CC} voltage
- D_4–D_{15}, R0–R2, R_{A0}, R_{A1} – V_{CC} to V_{CC}-40V

(Note 3) The timer/counter with the fastest clock and input/output current does not flow.

Test Conditions: MCU state;
- Standby Mode
- Input/Output; Reset state
- TIMER-A; ÷2 prescaler divide ratio
- TIMER-B; ÷2 prescaler divide ratio
- SERIAL; Stop

Pin state;
- $\overline{RESET}$ – GND voltage
- $\overline{TEST}$ – V_{CC} voltage
- D_0–D_3, R3–R9 – V_{CC} voltage
- D_4–D_{15}, R0–R2, R_{A0}, R_{A1} – V_{CC} to V_{CC}-40V

(Note 4) The timer/counter with the slowest clock and input/output current does not flow.

Test Conditions: MCU state;
- Standby Mode
- Input/Output; Reset state
- TIMER-A; ÷2048 prescaler divide ratio
- TIMER-B; ÷2048 prescaler divide ratio
- SERIAL; Stop

Pin state;
- $\overline{RESET}$ – GND voltage
- $\overline{TEST}$ – V_{CC} voltage
- D_0–D_3, R3–R9 – V_{CC} voltage
- D_4–D_{15}, R0–R2, R_{A0}, R_{A1} – V_{CC} to V_{CC}-40V

(Note 5) The consumption of current in operation and standby mode is proportional to f_{osc}. When $f_{osc} = x$ [MHz], the value of each current is calculated as follows.

$$\text{max. value } (f_{osc} = x) = \frac{x}{4} \times \text{max. value } (f_{osc} = 4 \text{ [MHz]}).$$

(2) Input/output characteristics for standard pin
(V_{CC}=4.5V to 5.5V, GND=0V, Ta=-20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Input "High" Voltage	V_{IH}	D_0 – D_3, R3 – R5, R9		0.7V_{CC}	–	V_{CC}+0.3	V	
Input "Low" Voltage	V_{IL}	D_0 – D_3, R3 – R5, R9		–0.3	–	0.22V_{CC}	V	
Output "Low" Voltage	V_{OL}	D_0 – D_3, R3 – R8	I_{OL} = 1.6 mA	–	–	0.4	V	
Input/Output Leakage Current	$\|I_{IL}\|$	D_0 – D_3, R3 – R9	V_{in} = 0V–V_{CC}	–	–	1	μA	1

(Note 1) Output buffer current are excluded.

(3) Input/output characteristics for high voltage pin
(V_{CC}=4.5V to 5.5V, GND=0V, Ta=-20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Input "High" Voltage	V_{IH}	D_4 – D_{15}, R1 R2, R_{A0}, R_{A1}		0.7V_{CC}	–	V_{CC}+0.3	V	
Input "Low" Voltage	V_{IL}	D_4 – D_{15}, R1 R2, R_{A0}, R_{A1}		V_{CC}-40	–	0.22V_{CC}	V	
Output "High" Voltage	V_{OH}	D_4 – D_{15}	$-I_{OH}$ = 15mA	V_{CC}-3.0	–	–	V	
			$-I_{OH}$ = 9mA	V_{CC}-2.0				
		R0 – R2	$-I_{OH}$ = 3mA	V_{CC}-3.0	–	–	V	
			$-I_{OH}$ = 1.8 mA	V_{CC}-2.0				
Output "Low" Voltage	V_{OL}	D_4 – D_{15} R0 – R2	150kΩ to V_{CC}-40V	–	–	V_{CC}-37	V	
Input/Output Leakage Current	$\|I_{IL}\|$	D_4 – D_{15} R0 – R2 R_{A0}, R_{A1}	V_{in} = V_{CC}-40V to V_{CC}	–	–	20	μA	1

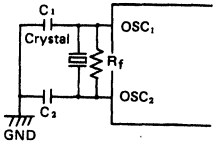
(Note 1) Output buffer current are excluded.

(4) AC characteristics

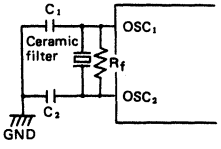
(V_{CC}=4.5V to 5.5V, GND=0V, Ta=-20 to +75°C, if not specified.)

	Item	Symbol	Pin Name	Test Conditions	Value min	Value typ	Value max	Unit	Note
Crystal Resonator	Oscillation Frequency	f_{osc}	OSC_1, OSC_2	divide-by-8	0.4	–	6.2	MHz	
	Instruction Cycle Time	t_{cyc}		divide-by-8	1.29	–	20	μs	
	Oscillator Stabilization Time	t_{RC}	OSC_1, OSC_2		–	–	20	ms	1
Ceramic Filter Resonator	Oscillation Frequency	f_{osc}	OSC_1, OSC_2	divide-by-8	0.4	–	6.2	MHz	
	Instruction Cycle Time	t_{cyc}		divide-by-8	1.29	–	20	μs	
	Oscillator Stabilization Time	t_{RC}	OSC_1, OSC_2		–	–	20	ms	1
External Clock	External Clock Frequency	f_{CP}	OSC_1	divide-by-8	0.4	–	6.2	MHz	2
	External Clock "High" Level Width	t_{CPH}	OSC_1	divide-by-8	70	–	–	ns	2
	External Clock "Low" Level Width	t_{CPL}	OSC_1	divide-by-8	70	–	–	ns	2
	External Clock Rise Time	t_{CPr}	OSC_1		–	–	20	ns	2
	External Clock Fall Time	t_{CPf}	OSC_1		–	–	20	ns	2
	Instruction Cycle Time	t_{cyc}			1.29	–	20	μs	2
$\overline{INT_0}$ "High" Level Width		t_{I0H}	$\overline{INT_0}$		2	–	–	t_{cyc}	3
$\overline{INT_0}$ "Low" Level Width		t_{I0L}	$\overline{INT_0}$		2	–	–	t_{cyc}	3
$\overline{INT_1}$ "High" Level Width		t_{I1H}	$\overline{INT_1}$		2	–	–	t_{cyc}	3
$\overline{INT_1}$ "Low" Level Width		t_{I1L}	$\overline{INT_1}$		2	–	–	t_{cyc}	3
RESET "High" Level Width		t_{RSTH}	RESET		2	–	–	t_{cyc}	4
Input Capacitance		C_{in}	all pins	f=1MHz V_{in} = 0V	–	–	15	pF	
Reset Fall Time		t_{RSTf}			–	–	20	ms	4

(Note 1) Oscillator stabilization time is the time until the oscillator stabilizes after V_{CC} reaches 4.5V at "Power-on", or after RESET input level goes "High" by resetting to quit the stop mode by MCU reset. At power ON or recovering from stop mode, apply RESET input more than t_{RC} to obtain the necessary time for oscillator stabilization. The circuits used to measure the value are described below. When using crystal or ceramic filter oscillator, please ask a crystal oscillator maker's or ceramic filter maker's advice because oscillator stabilization time depends on the circuit constant and stray capacity.

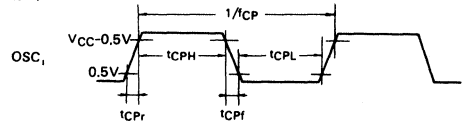


Crystal: 6.0 [MHz]
NC-18C (Nihon Denpa Kogyo)
R_f = 1 [MΩ] ± 2%, C_1 = C_2 = 20 [pF] ± 20%

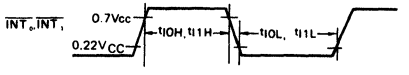


Ceramic filter: CSA6.00 MG (Murata)
R_f = 1 [MΩ] ± 2%, C_1 = C_2 = 30 [pF] ± 20%

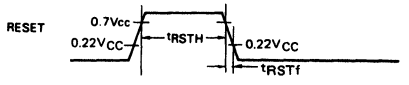
(Note 2)



(Note 3)



(Note 4)



(5) Serial interface timing characteristics
(V_{CC}=4.5V to 5.5V, GND=0V, Ta=-20 to +75°C, if not specified.)

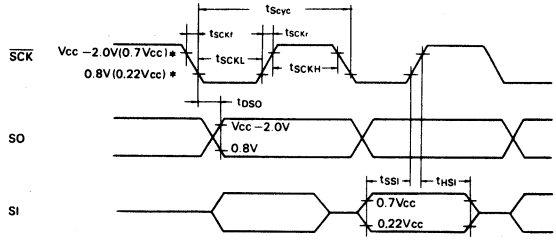
- At Transfer Clock Output

Item	Symbol	Pin Name	Test Conditions	Value min	Value typ	Value max	Unit	Note
Transfer Clock Cycle Time	t_{Scyc}	$\overline{SCK}$	(Note 2)	1	–	–	t_{cyc}	1, 2
Transfer Clock "High" Level Width	t_{SCKH}	$\overline{SCK}$	(Note 2)	0.5	–	–	t_{Scyc}	1, 2
Transfer Clock "Low" Level Width	t_{SCKL}	$\overline{SCK}$	(Note 2)	0.5	–	–	t_{Scyc}	1, 2
Transfer Clock Rise Time	t_{SCKr}	$\overline{SCK}$	(Note 2)	–	–	100	ns	1, 2
Transfer Clock Fall Time	t_{SCKf}	$\overline{SCK}$	(Note 2)	–	–	100	ns	1, 2
Serial Output Data Delay Time	t_{DSO}	SO	(Note 2)	–	–	250	ns	1, 2
Serial Input Data Set-up Time	t_{SSI}	SI		300	–	–	ns	1
Serial Input Data Hold Time	t_{HSI}	SI		150	–	–	ns	1

- At Transfer Clock Input

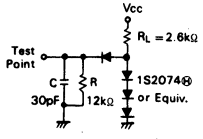
Item	Symbol	Pin Name	Test Conditions	Value min	Value typ	Value max	Unit	Note
Transfer Clock Cycle Time	t_{Scyc}	$\overline{SCK}$		1	–	–	t_{cyc}	1
Transfer Clock "High" Level Width	t_{SCKH}	$\overline{SCK}$		0.5	–	–	t_{Scyc}	1
Transfer Clock "Low" Level Width	t_{SCKL}	$\overline{SCK}$		0.5	–	–	t_{Scyc}	1
Transfer Clock Rise Time	t_{SCKr}	$\overline{SCK}$		–	–	100	ns	1
Transfer Clock Fall Time	t_{SCKf}	$\overline{SCK}$		–	–	100	ns	1
Serial Output Data Delay Time	t_{DSO}	SO	(Note 2)	–	–	250	ns	1, 2
Serial Input Data Set-up Time	t_{SSI}	SI		300	–	–	ns	1
Serial Input Data Hold Time	t_{HSI}	SI		150	–	–	ns	1

(Note 1) Timing Diagram of Serial Interface

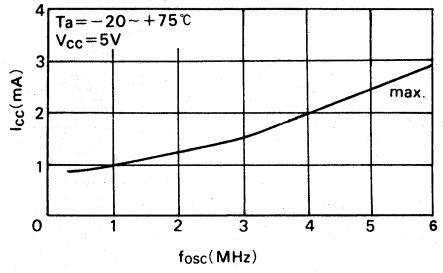


* V_{CC}−2.0V and 0.8V are the threshold voltage for transfer clock output.
0.7V_{CC} and 0.22 V_{CC} are the threshold voltage for transfer clock input.

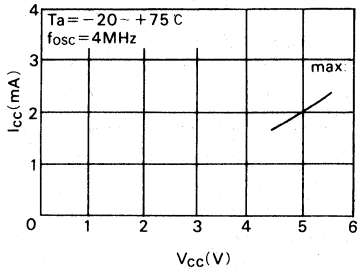
(Note 2) Timing Load Circuit



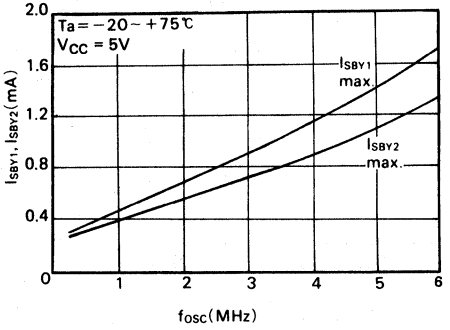
(6) Characteristics Curve (Reference data)



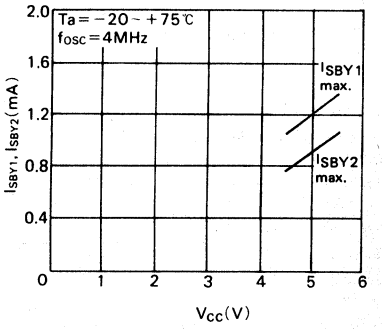
I_{CC} vs. f_{osc} characteristic
(crystal, ceramic resonator)



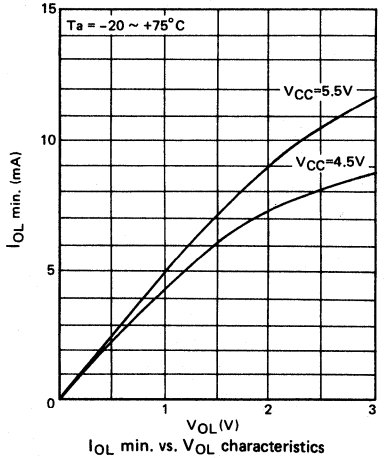
I_{CC} vs. V_{CC} characteristic
(crystal, ceramic resonator)



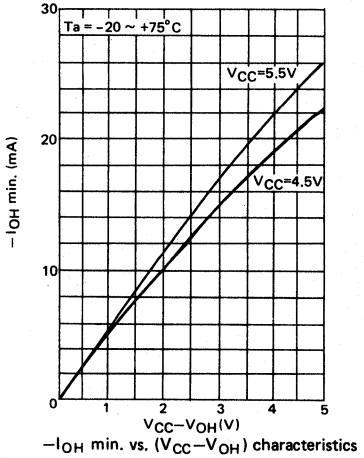
I_{SBY} vs. f_{osc} characteristics
(crystal, ceramic resonator)



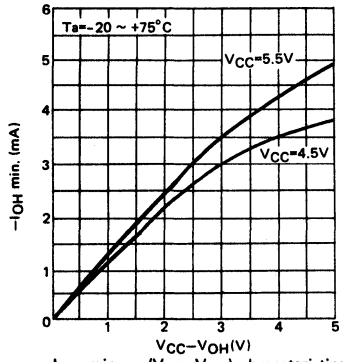
I_{SBY} vs. V_{CC} characteristics
(crystal, ceramic resonator)



I_{OL} min. vs. V_{OL} characteristics
(Standard Pin)



$-I_{OH}$ min. vs. ($V_{CC}-V_{OH}$) characteristics
(D_4 ~ D_{15} pins)



$-I_{OH}$ min. vs. ($V_{CC}-V_{OH}$) characteristics (R0 – R2 pins)

12. EPROM ON PACKAGE TYPE MICROCOMPUTER HD614P180/HD40P4181

12.1 Overview

The HD614P180/HD40P4181 are 4 bit single-chip microcomputer which can mount a standard EPROM 2764/27128 for program memory.

The HD614P180/HD40P4181 are pin compatible with the mask ROM type HMCS412C/CL/AC and HMCS414C/CL/AC, and have the same function as them except power supply voltage range, ROM capacity, RAM capacity, mask option, and package. By modifying the program in the EPROM, the HD614P180/HD40P4181 can be used for the evaluation of the HMCS412C/CL/AC and HMCS414C/CL/AC or for small-scale production.

(1) Hardware features

- o 4-bit Architecture
- o Application to 4k, 8k words × 10 bits of EPROM
  4096 words × 10 bits HN482764, HN27C64
  8192 words × 10 bits HN4827128
- o Data Memory (RAM) Capacity ..... 576 digits × 4 bits (HD614P180)
  992 digits × 4 bits (HD40P4181)
- o 36 I/O Pins - 24 I/O pins are high voltage up to 40V (max).
- o Timer/Counter
  11-bit Prescaler
  8-bit Auto-reload Timer/Event Counter
- o 3 Interrupts
  External 2
  Timer/Counter 1
- o Subroutine Stack
  Up to 16 levels including interrupts
- o Minimum Instruction Execution Time; 1.29 µs (HD614P180), 0.89 µs (HD40P4181)
- o 2 Low Power Modes
  Standby - Stops instruction execution while keeping clock generator and interrupt functions included Timer/Counter and Serial Interface in operation
  Stop - Stops instruction execution and clock generation while retaining RAM data
- o Clock Generator
  External Connection of Crystal Resonator or Ceramic Filter Resonator (externally drivable)
- o Power Voltage Range; 5V ± 10%

- o I/O Pin Circuit Form

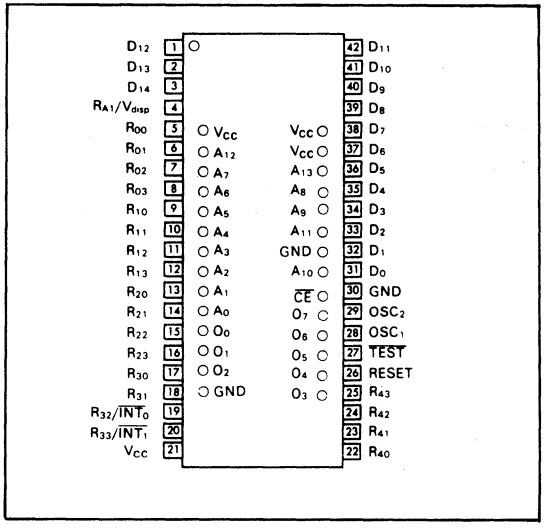
  All standard pins are "without pull-up MOS".

  All high voltage pins are "without pull-down MOS".
- o Shrink Type 42 Pin EPROM On-package

(2) Software features

- o Software Compatibel with HMCS412/414
- o Instruction Set Similar to and More Powerful than HMCS40 Series; 98 Instructions
- o High Programming Efficiency with 10-bit ROM/Word; 78 instructions are single word instructions.
- o Direct Branch to All ROM Area
- o Direct or Indirect Addressing to All RAM Area
- o Subroutine Nesting Up to 16 Levels Including Interrupts
- o Binary and BCD Arithmetic Operation
- o Powerful Logic Arithmetic Operation
- o Pattern Generation - Table Look Up Capability -
- o Bit Manipulation for Both RAM and I/O

(3) Pin arrangement (Top View)



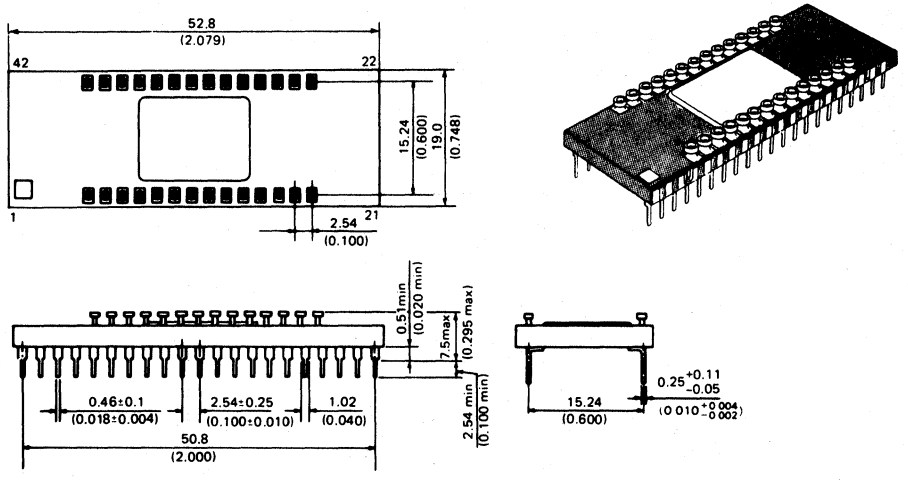
(4) Recommended applicable EPROM

Type No.	Program Memory Capacity	f_{OSC}(MHz)	EPROM Type No.
HD614P180 HD40P4181	4096 words	4	HN27C64-30 HN482764-3
		6	HN27C64G-25 HN482764
	8192 words	4	HN4827128-45
		6	HN4827128-25

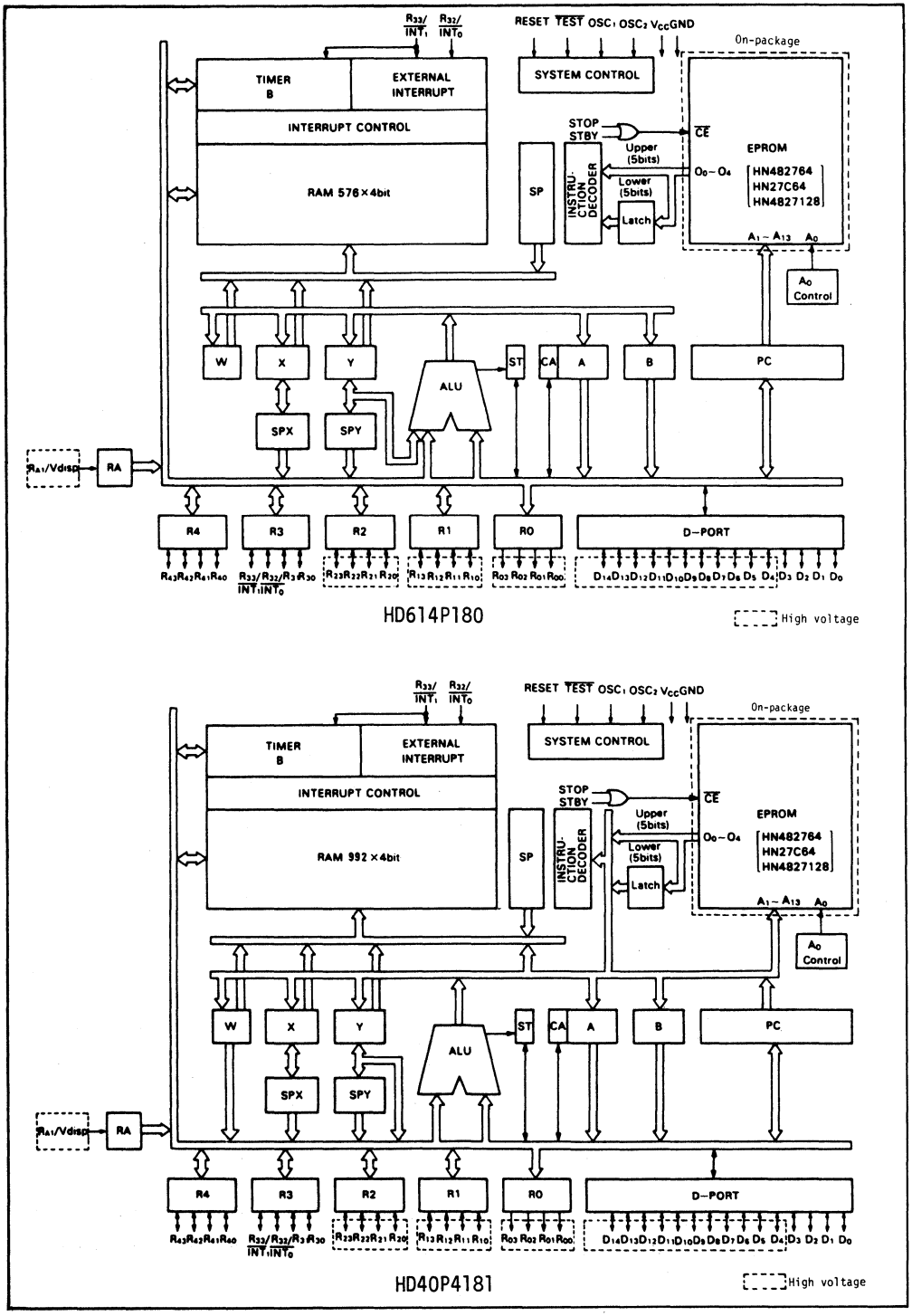
(5) Package dimension

unit: mm (inch)

DC-42P



(6) Block diagrams



12.2 ROM Memory Map

ROM is described in the following paragraphs and ROM Memory Map is illustrated in Fig. 12-1.

(1) Vector Address Area --- $0000 to $000F

Locations $0000 through $000F are reserved for JMPL instructions to branch to the starting address of the initialization program and of the interrupt service programs. After reset of interrupt routine is serviced, the program is executed from the vector address.

(2) Zero-Page Subroutine Area --- $0000 to $003F

Locations $0000 through $003F are reserved for subroutines. CAL instruction allows to branch to the subroutine.

(3) Pattern Area --- $0000 to $0FFF

Locations $0000 through $0FFF are reserved for ROM data. P instruction allows referring to the ROM data as a pattern.

(4) Program Area --- $000 to $1FFF

12.3 RAM Memory Map

The HD614P180 includes 576 digits × 4 bits RAM as the data area and stack area. Also, the HD40P4181 includes 992 digits × 4 bits RAM. In addition to these areas, interrupt control bits and special function registers are also mapped on the RAM memory space. RAM memory map is illustrated in Fig. 12-2 and described in the following paragraphs.

(1) Interrupt Control Bit Area --- $000 to $003

This area is used for interrupt controls, and is illustrated in Fig. 12-3. It is accessable only by RAM bit manipulation instruction. However, the interrupt request flag cannot be set by software. RSP bit is only used to reset the stack Pointer.

(2) Special Function Registers Area --- $004 to $00B

The Special Function Registers are the mode or data registers for the external interrupt, the serial interface, and the timer/counter.

These registers are classified into three types: Write-only, Read-only, and Read/Write as shown in Fig. 12-2. These registers cannot be accessed by RAM bit manipulation instruction.

(3) Data Area --- $020 to $21F [HD614P180]
$020 to $3BF [HD40P4181]

16 digits of $020 through $02F are called memory register (MR) and accessable by LAMR and XMRA instructions. The configuration is shown in Fig. 12-4.

(4) Stack Area --- $3C0 to $3FF

Locations $3C0 through $3FF are reserved for LIFO stacks to save the contents of the program counter (PC), status (ST) and carry (CA) when interruption is serviced. This area can be used as 16 nesting level stack which one level requires 4 digits. A save condition is shown in Fig. 12-4. The program counter is restored by RTN and RTNI instructions. Status and Carry are restored only by RTNI instruction. The area, not used for stacking, is available as a data area.

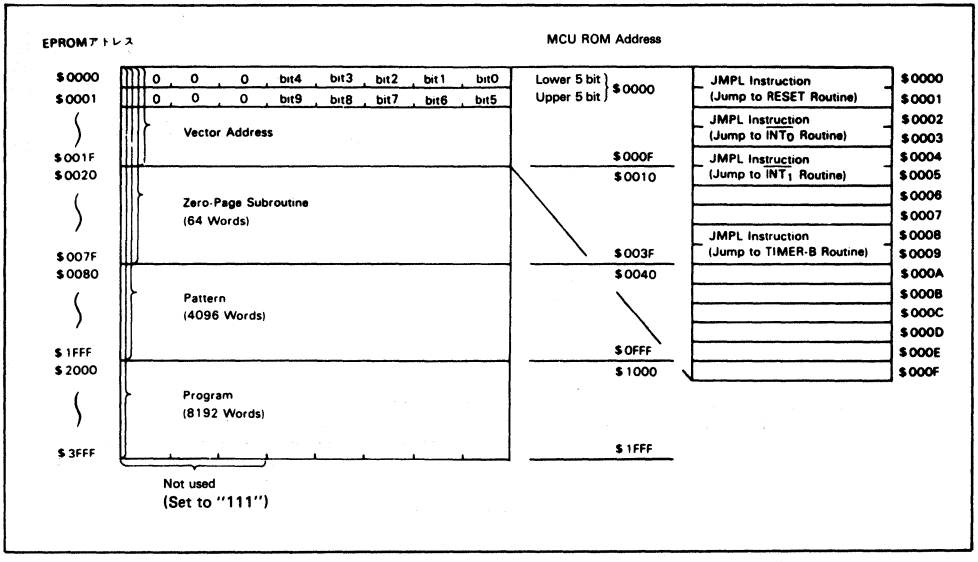
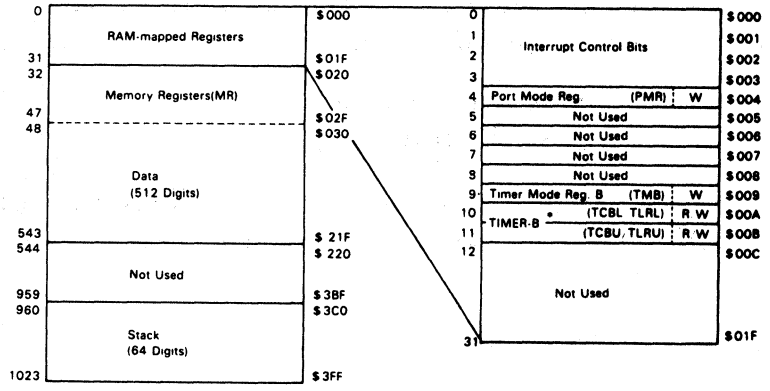


Fig. 12-1 ROM Memory Map

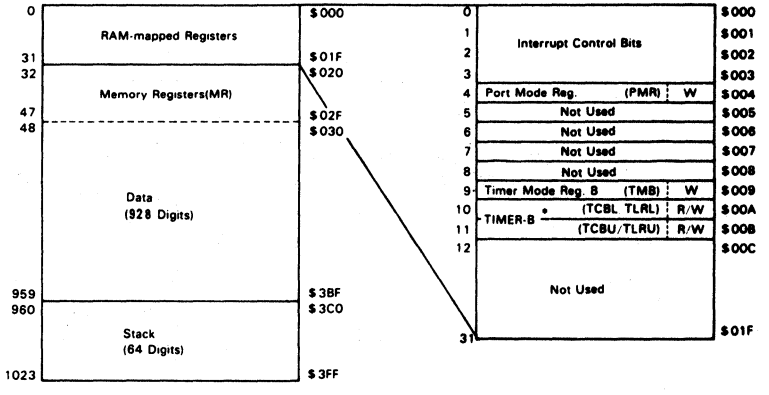


* Two registers are mapped on same address.

R : Read Only
W : Write Only
R/W : Read/Write

Timer/Event Counter B Lower (TCBL)	R	Timer Load Reg. Lower (TLRL)	W	$00A
Timer/Event Counter B Upper (TCBU)	R	Timer Load Reg. Upper (TLRU)	W	$008

HD614P180



* Two registers are mapped on same address.

R : Read Only
W : Write Only
R/W : Read/Write

Timer/Event Counter B Lower (TCBL)	R	Timer Load Reg. Lower (TLRL)	W	$00A
Timer/Event Counter B Upper (TCBU)	R	Timer Load Reg. Upper (TLRU)	W	$008

HD40P4181

Fig. 11-2 RAM Memory Map

	bit 3	bit 2	bit 1	bit 0	
0	IM0 (IM of $\overline{INT_0}$)	IF0 (IF of $\overline{INT_0}$)	RSP (Reset SP Bit)	I/E (Interrupt Enable Flag)	$000
1	Not Used	Not Used	IM1 (IM of $\overline{INT_1}$)	IF1 (IF of $\overline{INT_1}$)	$001
2	Not Used	Not Used	IMTB (IM of TIMER-B)	IFTB (IF of TIMER-B)	$002
3	Not Used	Not Used	Not Used	Not Used	$003

IF : Interrupt Request Flag
IM : Interrupt Mask
I/E : Interrupt Enable Flag
SP : Stack Pointer

(Note) Each bit in Interrupt Control Bits Area is set by SEM/SEMD instruction, is reset by REM/REMD instruction and is tested by TM/TMD instruction. It is not affected by other instructions. Furthermore, Interrupt Request Flag is not affected by SEM/SEMD instruction. The content of Status becomes invaid when "Not Used" bit is tested.

Fig. 12-3 Configuration of Interrupt Control Bit Area

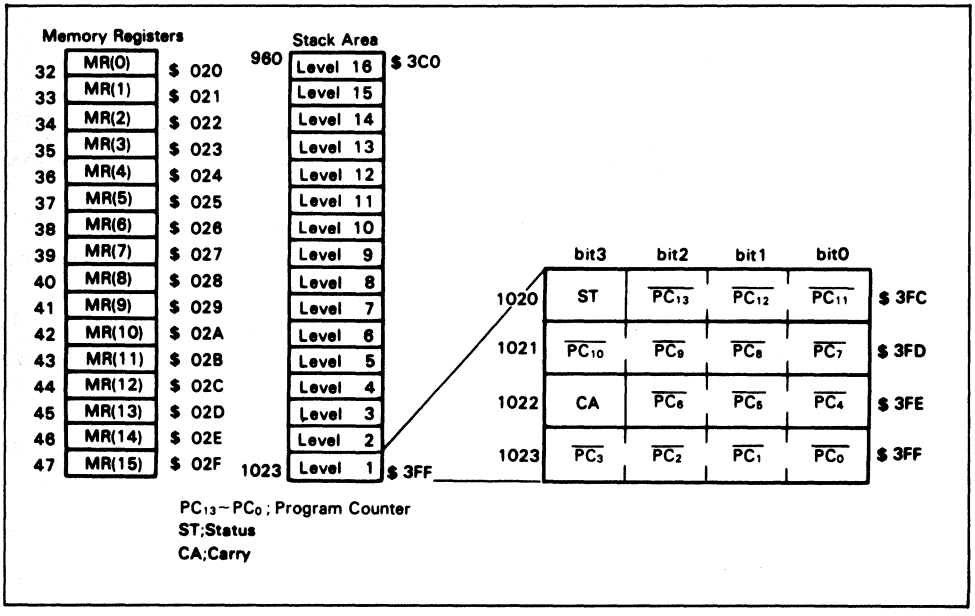


Fig. 12-4 Configuration of Memory Register, Stack Area and Stack Position

12.4 Precautions on using EPROM on Package Type Microcomputer

Since the HD614P180S/HD40P4181 has a special structure with pin sockets installed on the surface of the package, the following should be noted when using it.

(1) Do not apply an electrostatic voltage or surge voltage more than the maximum ratings to the pin socket pins. This may destroy the LSI permanently.

(2) When installing this LSI in system products in the same way as the mask ROM 4-bit single chip microcomputer, observe the following in order to maintain good ohmic contact between EPROM pins and pin sockets.

(a) When soldering the LSI on a printed circuit board, keep pin conditions under 250°C within 10 seconds. If these conditions are exceeded, the solder fixing the pin sockets may melt and the pins may fall out.

(b) Keep out detergent or coater from the pin sockets during flux removal or board coating. Flux or coater may decrease pin socket contactivity.

(c) Avoid permanent use of this LSI in places with excessive vibration.

(d) Since repeated insertion/removal of EPROMs may decrease pin sockets' contactivity, it is recommended to use new ones for your system products.

12.5 Absolute Maximum Ratings

Item	Symbol	Value	Unit	Note
Supply Voltage	V_{CC}	−0.3 to +7.0	V	
Terminal Voltage	V_T	−0.3 to V_{CC} +0.3	V	3
		V_{CC} −45 to V_{CC} +0.3	V	4
Total Allowance of Input Currents	ΣI_O	50	mA	5
Total Allowance of Output Currents	$-\Sigma I_O$	150	mA	6
Maximum Input Current	I_O	15	mA	7, 8
Maximum Output Current	$-I_O$	4	mA	9, 10
		6	mA	9, 11
		30	mA	9, 12
Operating Temperature	T_{opr}	−20 to +75	°C	
Storage Temperature	T_{stg}	−55 to +125	°C	

(Note 1) Permanent damage may occur if "Absolute Maximum Ratings" are exceeded. Normal operation should be under the conditions of "Electrical Characteristics". If these conditions are exceeded, it may cause the malfunction and affect the reliability of LSI.
(Note 2) All voltages are with respect to GND.
(Note 3) Applied to standard pins.
(Note 4) Applied to high voltage pins.
(Note 5) Total allowance of input current is the total sum of input current which flow in from all I/O pins to GND simultaneously.
(Note 6) Total allowance of output current is the total sum of the output current which flow out from V_{CC} to all I/O pins simultaneously.
(Note 7) Maximum input current is the maximum amount of input current from each I/O pin to GND.
(Note 8) Applied to D_0 – D_3 and R3, R4
(Note 9) Maximum output current is the maximum amount of output current from V_{CC} to each I/O pin.
(Note 10) Applied to D_0 –D_3 and R3, R4.
(Note 11) Applied to R0 – R2.
(Note 12) Applied to D_4–D_{14}

12.6 HD614P180 Electrical Characteristics

(1) DC Characteristics (V_{CC} = 4.5V to 5.5V, GND = 0V, Ta = −20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Input "High" Voltage	V_{IH}	RESET, $\overline{INT_0}$, $\overline{INT_1}$		0,7V_{CC}	–	V_{CC}+0.3	V	
		OSC_1		V_{CC}−0.5	–	V_{CC}+0.3	V	
Input "Low" Voltage	V_{IL}	RESET, $\overline{INT_0}$, $\overline{INT_1}$		−0.3	–	0.22V_{CC}	V	
		OSC_1		−0.3	–	0.5	V	
Input/Output Leakage Current	$\|I_{IL}\|$	RESET, $\overline{INT_0}$, $\overline{INT_1}$, OSC_1	V_{in} = 0V to V_{CC}	–	–	1	μA	1
Current Dissipation in Active Mode	I_{CC}	V_{CC}	V_{CC}=5V fosc = 4MHz, divide-by-8	–	–	2.0	mA	2,4
Current Dissipation in Standby Mode	I_{SBY1}	V_{CC}	Maximum Logic Operation V_{CC} = 5V fosc = 4MHz, divide-by-8	–	–	1.2	mA	3,4
	I_{SBY2}	V_{CC}	Minimum Logic Operation V_{CC} = 5V fosc = 4MHz, divide-by-8	–	–	0.9	mA	4,5
Current Dissipation in Stop Mode	I_{stop}	V_{CC}	V_{in} ($\overline{TEST}$) = V_{CC}-0.3V to V_{CC} V_{in} (RESET) = 0V to 0.3V	–	–	10	μA	
Stop Mode Retain Voltage	V_{stop}	V_{CC}		2	–	–	V	

(Note 1) Output buffer current are excluded.

(Note 2) The MCU is in the reset state. The input/output current does not flow.

Test Conditions: MCU state; • Reset state in Operation Mode

Pin state; • RESET, $\overline{TEST}$ ... V_{CC} voltage
• D_0~D_3, R3~R4 ... V_{CC} voltage
• D_4~D_{14}, R0~R2, R_{A1} ... V_{CC} ~V_{CC}-40V

(Note 3) The timer/counter with the fastest clock and input/output current does not flow.

Test Conditions: MCU state; • Input/Output; Reset state

(Note 4) The timer/counter with the slowest clock and input/output current does not flow.

Test Conditions: MCU state; • Standby Mode
• Input/Output; Reset state
• TIMER-B;÷2048 prescaler divide ratio

Pin state; • RESET ... GND voltage
• $\overline{TEST}$ ... V_{CC} voltage
• D_0~D_3, R3~R4 ... V_{CC} voltage
• D_4~D_{14}, R0~R2, R_{A1} ... V_{CC} ~ V_{CC}−40V

(Note 5) When f_{osc}=x[MHz], the Current Dissipation in Operation mode and Standby mode are estimated as follows:

$$\text{max. value } (f_{osc}=x\text{[MHz]}) = \frac{x}{4} \times \text{max. value } (f_{osc}=4\text{[MHz]})$$

(2) Input/output characteristics for standard pin

(V_{CC} = 4.5V to 5.5V, GND = 0V, Ta = −20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Input "High" Voltage	V_{IH}	$D_0 \sim D_3$, R3 ~ R4		0.7V_{CC}	–	V_{CC}+0.3	V	
Input "Low" Voltage	V_{IL}	$D_0 \sim D_3$, R3 ~ R4		−0.3	–	0.22V_{CC}	V	
Output "Low" Voltage	V_{OL}	$D_0 \sim D_3$, R3 ~ R4	I_{OL} = 1.6 mA	–	–	0.4	V	
Input/Output Leakage Current	$\|I_{IL}\|$	$D_0 \sim D_3$, R3 ~ R4	V_{in} = 0V to V_{CC}	–	–	1	μA	1

(Note 1) Output buffer current are excluded.

(3) Input/output characteristics for high voltage pin

(V_{CC} = 4.5V to 5.5V, GND = 0V, Ta = −20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Input "High" Voltage	V_{IH}	$D_4 \sim D_{14}$, R1, R2, R_{A1}		0.7V_{CC}	–	V_{CC}+0.3	V	
Input "Low" Voltage	V_{IL}	$D_4 \sim D_{14}$, R1, R2, R_{A1}		V_{CC}-40	–	0.22V_{CC}	V	
Output "High" Voltage	V_{OH}	$D_4 \sim D_{14}$	$-I_{OH}$ = 15mA	V_{CC}-3.0	–	–	V	
			$-I_{OH}$ = 9 mA	V_{CC}-2.0	–	–	V	
		R0 ~ R2	$-I_{OH}$ = 3 mA	V_{CC}-3.0	–	–	V	
			$-I_{OH}$ = 1.8 mA	V_{CC}-2.0	–	–	V	
Output "Low" Voltage	V_{OL}	$D_4 \sim D_{14}$, R0 ~ R2	150kΩ to V_{CC} -40V	–	–	V_{CC}-37	V	
Input/Output Leakage Current	$\|I_{IL}\|$	$D_4 \sim D_{14}$, R0 ~ R2, R_{A1}	V_{in} = V_{CC}-40V to V_{CC}	–	–	20	μA	1

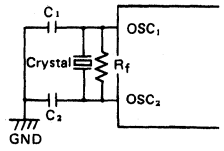
(Note 1) Output buffer current are excluded.

(4) AC characteristics (V_{CC} = 4.5V to 5.5V, GND = 0V, Ta = –20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Oscillation Frequency	f_{osc}	OSC_1, OSC_2	divided-by-8	0.4	4	4.5	MHz	
Instruction Cycle Time	t_{cyc}		divided-by-8	1.78	2	20	μs	
Oscillator Stabilization Time	t_{RC}	OSC_1, OSC_2		–	–	20	ms	1
External Clock "High" Level Width	t_{CPH}	OSC_1	divided-by-8	100	–	–	ns	2
External Clock "Low" Level Width	t_{CPL}	OSC_1	divided-by-8	100	–	–	ns	2
External Clock Rise Time	t_{CPr}	OSC_1		–	–	20	ns	2
External Clock Fall Time	t_{CPf}	OSC_1		–	–	20	ns	2
$\overline{INT_0}$ "High" Level Width	t_{I0H}	$\overline{INT_0}$		2	–	–	t_{cyc}	3
$\overline{INT_0}$ "Low" Level Width	t_{I0L}	$\overline{INT_0}$		2	–	–	t_{cyc}	3
$\overline{INT_1}$ "High" Level Width	t_{I1H}	$\overline{INT_1}$		2	–	–	t_{cyc}	3
$\overline{INT_1}$ "Low" Level Width	t_{I1L}	$\overline{INT_1}$		2	–	–	t_{cyc}	3
RESET "High" Level Width	t_{RSTH}	RESET		2	–	–	t_{cyc}	4
Input Capacitance	C_{in}	all pins	f=1MHz V_{in} = 0V	–	–	15	pF	
RESET Fall Time	t_{RSTf}			–	–	20	ms	4

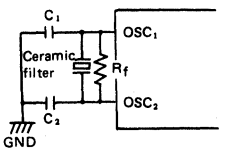
(Note 1) Oscillator stabilization time is the time until the oscillator stabilizes after V_{CC} reaches 4.5V at "Power-on", or after RESET input level goes to "High" by resetting to quit the stop mode by MCU reset on the circuits below. At power ON or recovering from stop mode, apply RESET input more than t_{RC} to obtain the necessary time for oscillator stabilization. When using crystal or ceramic filter oscillator, please ask a crystal oscillator maker's or ceramic filter maker's advice because oscillator stabilization time depends on the circuit constant and stray capacity.

Crystal oscillator



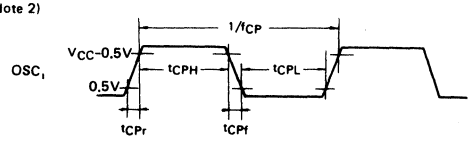
Crystal: 4.194304MHz NC-18C (Nihon Denpa Kogyo)
R_f ; 1MΩ ± 2%
C_1 : 22pF ± 20%
C_2 : 22pF ± 20%

Ceramic filter oscillator

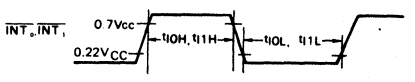


Ceramic filter: CSA4.00MG (Murata)
R_f : 1MΩ ± 2%
C_1 : 30pF ± 20%
C_2 : 30pF ± 20%

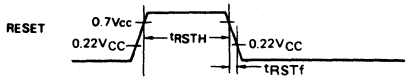
(Note 2)



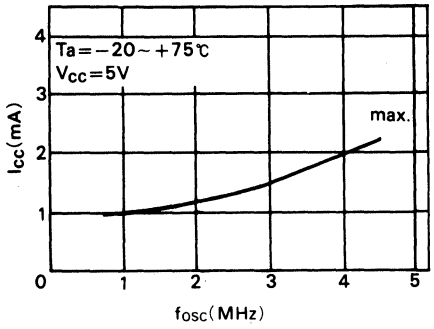
(Note 3)



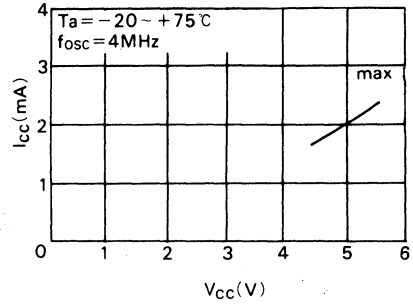
(Note 4)



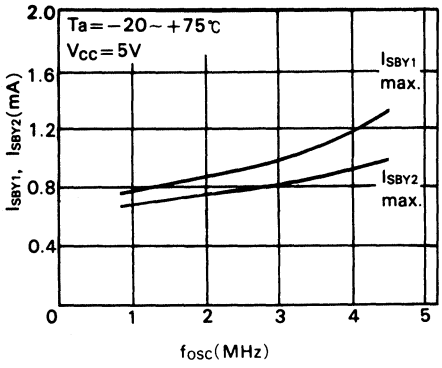
(5) Characteristics curve (reference data)



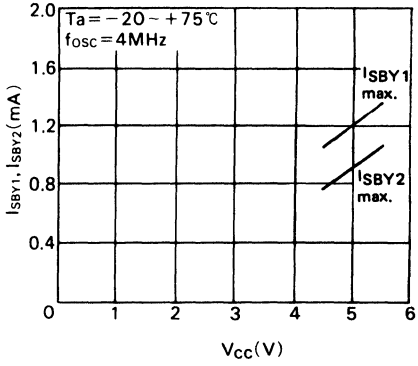
I_{CC} vs. f_{osc} characteristic
(crystal, ceramic resonator)



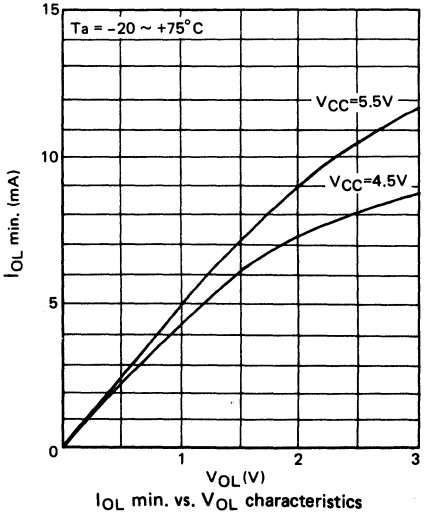
I_{CC} vs. V_{CC} characteristic
(crystal, ceramic resonator)



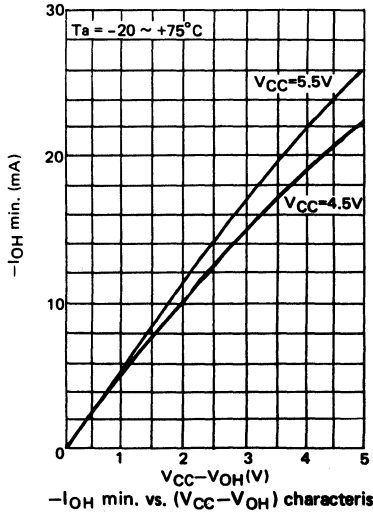
I_{SBY} vs. f_{osc} characteristics
(crystal, ceramic resonator)



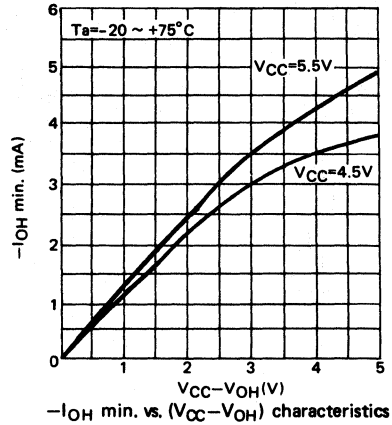
I_{SBY} vs. V_{CC} characteristics
(crystal, ceramic resonator)



I_{OL} min. vs. V_{OL} characteristics
(Standard Pin)



$-I_{OH}$ min. vs. ($V_{CC}-V_{OH}$) characteristics
($D_4 \sim D_{14}$ pins)



$-I_{OH}$ min. vs. ($V_{CC}-V_{OH}$) characteristics
(R0 ~ R2 pins)

12.7 HD40P4181 Electrical Characteristics

(1) DC characteristics (V_{CC} = 4.5V to 5.5V, GND = 0V, Ta = −20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Input "High" Voltage	V_{IH}	$\overline{RESET}$, $\overline{INT_0}$, $\overline{INT_1}$		0.8V_{CC}	–	V_{CC}+0.3	V	
		OSC_1		V_{CC}−0.5	–	V_{CC}+0.3	V	
Input "Low" Voltage	V_{IL}	$\overline{RESET}$, $\overline{INT_0}$, $\overline{INT_1}$		−0.3	–	0.3V_{CC}	V	
		OSC_1		−0.3	–	0.5	V	
Input/Output Leakage Current	$\lvert I_{IL} \rvert$	$\overline{RESET}$, $\overline{INT_0}$, $\overline{INT_1}$, OSC_1	V_{in} = 0V to V_{CC}	–	–	1	μA	1
Current Dissipation in Active Mode	I_{CC}	V_{CC}	V_{CC} = 5V f_{osc} = 8MHz, divide-by-8	–	–	TBD	mA	2, 4
Current Dissipation in Standby Mode	I_{SBY}	V_{CC}	V_{CC} = 5V f_{osc} = 8MHz, divide-by-8	–	–	TBD	mA	3, 4
Current Dissipation in Stop Mode	I_{stop}	V_{CC}	V_{in} ($\overline{TEST}$) = V_{CC}−0.3V to V_{CC} V_{in} (RESET) = 0V to 0.3V	–	–	10	μA	
Stop Mode Retain Voltage	V_{stop}	V_{CC}		2	–	–	V	

(Note 1) Output buffer current are excluded.

(Note 2) The MCU is in the reset state. The input/output current does not flow.
Test Conditions: MCU state; • Reset state in Operation Mode
Pin state; • RESET, $\overline{TEST}$ ... V_{CC} voltage
• $D_0 - D_3$, R3 – R4 ... V_{CC} voltage
• $D_4 - D_{14}$, R0 – R2, R_{A1}, ... V_{CC} – V_{CC}−40V

(Note 3) The timer/counter with the fastest clock and input/output current does not flow.
Test Conditions: MCU state; • Input/Output; Reset state

(Note 4) The consumption of current in operation and standby mode is proportion to f_{osc}.
When $f_{osc} = x$ (MHz), the value of each current is calculated as follows.

$$\text{max. value } (f_{osc} = x) = \frac{x}{8} \times \text{ max. value } (f_{osc} = 8[\text{MHz}]).$$

(2) Input/output characteristics for standard pin
(V_{CC} = 4.5V to 5.5V, GND = 0V, Ta = –20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Input "High" Voltage	V_{IH}	$D_0 - D_3$, R3 – R4		0.7V_{CC}	–	V_{CC}+0.3	V	
Input "Low" Voltage	V_{IL}	$D_0 - D_3$, R3 – R4		–0.3	–	0.3V_{CC}	V	
Output "Low" Voltage	V_{OL}	$D_0 - D_3$, R3 – R4	I_{OL} = 1.6mA	–	–	0.4	V	
Input /Output Leakage Current	$\|I_{IL}\|$	$D_0 - D_3$, R3 – R4	V_{in} = 0V – V_{CC}	–	–	1	μA	1

(Note 1) Output buffer current are excluded.

(3) Input/output characteristics for high voltage pin
(V_{CC} = 4.5V to 5.5V, GND = 0V, Ta = –20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Input "High" Voltage	V_{IH}	$D_4 - D_{14}$, R1 R2, R_{A1}		0.7V_{CC}	–	V_{CC}+0.3	V	
Input "Low" Voltage	V_{IL}	$D_4 - D_{14}$, R1 R2, R_{A1}		V_{CC}–40	–	0.3V_{CC}	V	
Output "High" Voltage	V_{OH}	$D_4 - D_{14}$	$-I_{OH}$ = 15mA	V_{CC}–3.0	–	–	V	
			$-I_{OH}$ = 10mA	V_{CC}–2.0	–	–	V	
			$-I_{OH}$ = 4mA	V_{CC}–1.0	–	–	V	
		R0 – R2	$-I_{OH}$ = 3mA	V_{CC}–3.0	–	–	V	
			$-I_{OH}$ = 2mA	V_{CC}–2.0	–	–	V	
			$-I_{OH}$ = 0.8mA	V_{CC}–1.0	–	–	V	
Output "Low" Voltage	V_{OL}	$D_4 - D_{14}$ R0 – R2,	150kΩ to V_{CC}–40V	–	–	V_{CC}–37	V	
Input/Output Leakage Current	$\|I_{IL}\|$	$D_4 - D_{14}$ R0 – R2, R_{A1}	V_{in} = V_{CC}–40V to V_{CC}	–	–	20	μA	1

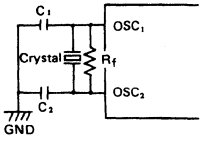
(Note 1) Output buffer current are excluded.

(4) AC characteristics (V_{CC} = 4.5V to 5.5V, GND = 0V, Ta = −20 to +75°C, if not specified.)

Item	Symbol	Pin Name	Test Conditions	Value			Unit	Note
				min	typ	max		
Oscillation Frequency	f_{osc}	OSC_1, OSC_2	divided-by-8	0.4	8	9	MHz	
Instruction Cycle Time	t_{cyc}		divided-by-8	0.89	1	20	μs	
Oscillator Stabilization Time	t_{RC}	OSC_1, OSC_2		–	–	20	ms	1
External Clock "High" Level Width	t_{CPH}	OSC_1	divided-by-8	41	–	–	ns	2
External Clock "Low" Level Width	t_{CPL}	OSC_1	divided-by-8	41	–	–	ns	2
External Clock Rise Time	t_{CPr}	OSC_1		–	–	15	ns	2
External Clock Fall Time	t_{CPf}	OSC_1		–	–	15	ns	2
$\overline{INT_0}$ "High" Level Width	t_{I0H}	$\overline{INT_0}$		2	–	–	t_{cyc}	3
$\overline{INT_0}$ "Low" Level Width	t_{I0L}	$\overline{INT_0}$		2	–	–	t_{cyc}	3
$\overline{INT_1}$ "High" Level Width	t_{I1H}	$\overline{INT_1}$		2	–	–	t_{cyc}	3
$\overline{INT_1}$ "Low" Level Width	t_{I1L}	$\overline{INT_1}$		2	–	–	t_{cyc}	3
RESET "High" Level Width	t_{RSTH}	RESET		2	–	–	t_{cyc}	4
Input Capacitance	C_{in}	all pins	f = 1MHz V_{in} = 0V	–	–	15	pF	
RESET Fall Time	t_{RSTf}			–	–	20	ms	4

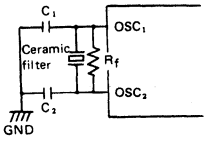
(Note 1) Oscillator stabilization time is the time until the oscillator stabilizes after V_{CC} reaches 4.5V at "Power-on", or after RESET input level goes to "High" by resetting to quit the stop mode by MCU reset on the circuits below. At power ON or recovering from stop mode, apply RESET input more than t_{RC} to obtain the necessary time for oscillator stabilization. When using crystal or ceramic filter oscillator, please ask a crystal oscillator maker's or ceramic filter maker's advice because oscillator stabilization time depends on the circuit constant and stray capacity.

Crystal oscillator



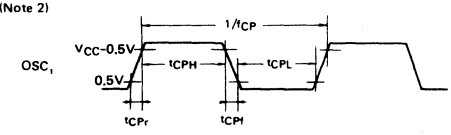
Crystal: 8.388608MHz NC-18 (Nihon Denpa Kogyo)
R_f : 1MΩ ± 20%
C_1 : 10pF ± 20%
C_2 : 10pF ± 20%

Ceramic filter oscillator

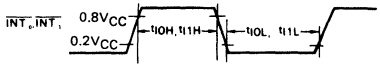


Ceramic filter: CSA8.00MT (Murata)
R_f : 1MΩ ± 20%
C_1 : 30pF ± 20%
C_2 : 30pF ± 20%

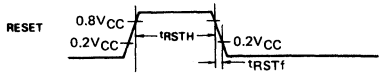
(Note 2)



(Note 3)



(Note 4)



13. PROGRAM DEVELOPMENT PROCEDURE AND SUPPORT SYSTEM

13.1 Overview

The cross assembler and hardware simulator using various types of computers are prepared by the company as supporting systems to develop user's programs. User's programs are mask programmed into ROM and delivered as an LSI by Hitachi.

Fig. 13-1 shows the typical program design procedure and Table 13-1 shows system development support tools for the HMCS400 series used in this process.

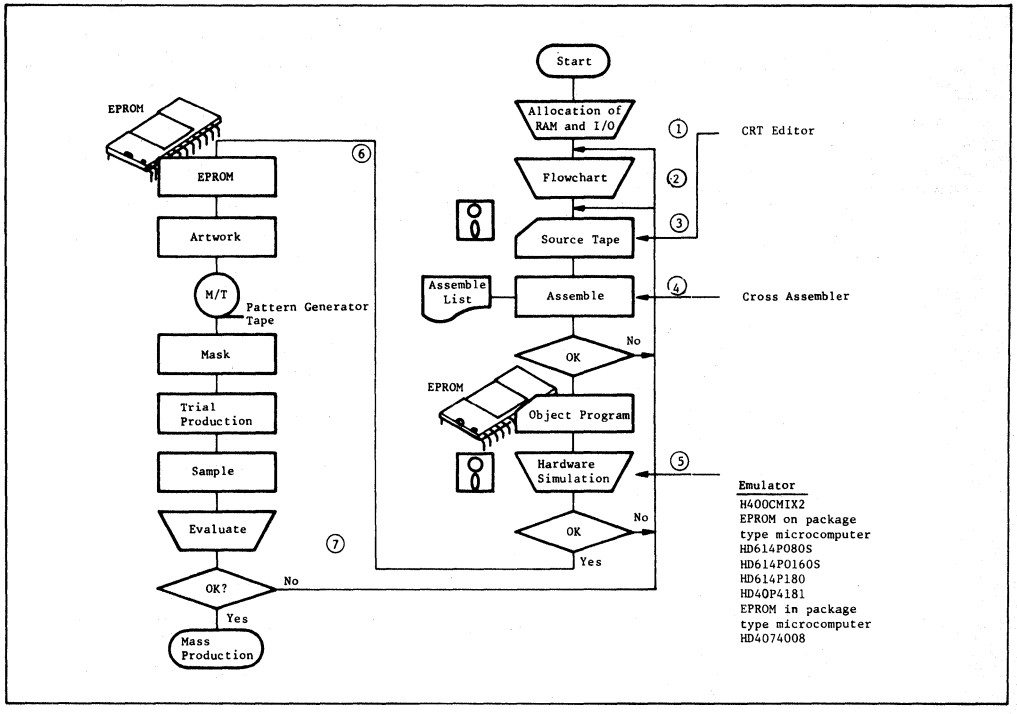


Fig. 13-1 Program Design Procedure

(Description)

① When the user programs the system for the HMCS400 series microcomputer, functional assignment of each I/O pin and allocation of RAM area in accordance with the design system must be specified before actual programming.

② A flowchart is prepared to implement the functions and is coded by using the HMCS400 series' mnemonic codes.

③ Write a source program using the text editor and save it on a floppy disk.

④ Assemble and debug the source program and generate an object program.

⑤ Verify the program through hardware emulation with an emulator, H68SD5/5A, H680SD200 or EPROM on package type microcomputer.

⑥ Forward the completed program to Hitachi in the form of an EPROM. Send also "Single-chip microcomputer order specification" and "Mask option list" at this time.

⑦ ROM and mask option are masked by Hitachi. The LSI is tentatively produced and the sample given to the user. If the user does not detect any programming programs, mass production can be started.

Table 13-1 System Development Support Tool

Hardware System

Partition \ Series	HMCS402C/404C/408C HMCS402AC/404AC/408AC HMCS402CL/404CL/408CL	HMCS412C/414C HMCS412AC/414AC HMCS412CL/414CL
Emulator (Note 1)	H400CMIX2 HS408EML02H HS408EMX22H	
EPROM on-package (Note 2) type microcomputer	HD614P080S HD614P0160S	HD614P180 HD40P4181
EPRON in-package (Note 3) type microcomputer	HD4074008	—

(Note 1) Use the emulator with connecting to the H68SD5/5A or H680SD200. Can be used in stand-alone.

(Note 2) Specifications of power supply and mask option are not applicable.

(Note 3) Specifications of power supply and mask option are not applicable for the HD4074008.

Software

Host computer	Cross assembler	Interface software
H68SD5/5A	S400XAS3F (FDOSIII/IV)(Note 2)	S68EML1-F (Note 2)
H680SD200	S400XAS6M (CP/M-68k) (Note 3)	S680EML1F (Note 3)
INTELLEC® SERIES II MODEL 220/230	S400MDS1F (for ISIS-II®) S400MDS2F (for CP/M-80®) (Note 1)	—
VAX/VMS	S400VAS1F	HMSI SW BOX
IBM PC-DOS	AS400PAI1SF (Note 4)	S31IEM1-F (Note 4)
HP64000	S400MDS2F (for CP/M®)	IN3EVM (Note 5)

(Note 1) ISIS-II® is a registered trademark of Intell.
CP/M-68k and CP/M® are registered trademarks of Digital Research.

(Note 2) This is included in the H400CMIX2.

(Note 3) This is included in the HS408EMX22H.

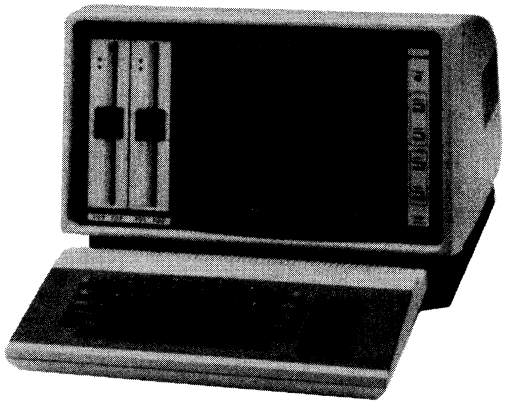
(Note 4) Overseas makers sell it.

(Note 5) Sophia Systems Corp. sell it.

13.2 Development System

13.2.1 H68SD5A Development System

The H68SD5A is a development system capable of developing system programs for the Hitachi 4-bit and 8-bit single-chip microcomputers. It offers high-level functions such as CRT display operations, assembler (based on floppy disk), and debugging with emulator. Software and hardware configurations are shown in Fig. 13-2, 13-3, respectively.



Features

- o Basic systems such as a CRT display, keyboard, and floppy disk drivers are provided at a moderate price.
- o Easy to debug hardware and software by emulator (option) suited for each kind of MCU.
- o The H68SD5A can perform system development through a CRT editor, assembler, linkage editor, emulator, and EPROM writer (Note 1).
- o System configuration corresponding to the purpose of development is provided by easy connection of I/O devices such as a printer (parallel interface), a console typewriter (serial interface), and an EPROM writer. Serial interface of the H68SD5A is completely provided with 3 circuits (1 circuit for H68SD5).
- o Allows program development of other products by exchanging emulator software.

Note 1) Use the following EPROM writers available on the market.
AVAL CORPORATION; PKW-7000, PKW-1000
DATA I/O ; 29A, 29B

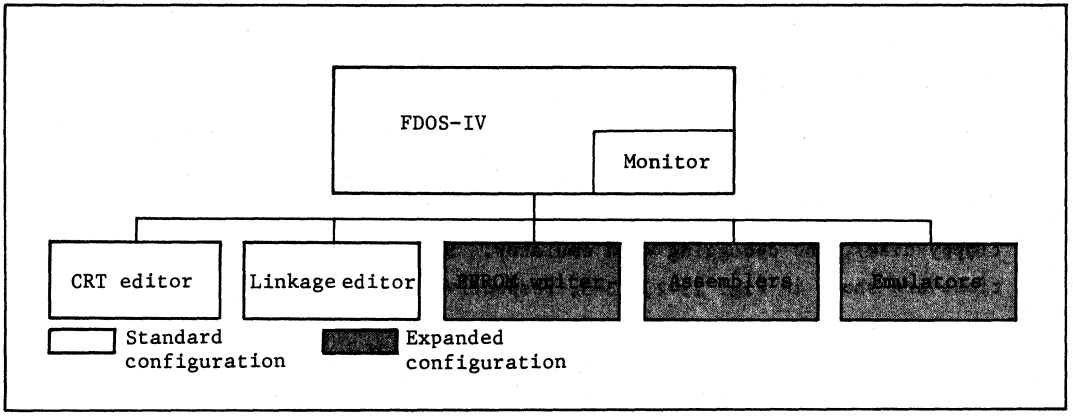


Fig. 13-2 H68SD5 Software Configuration

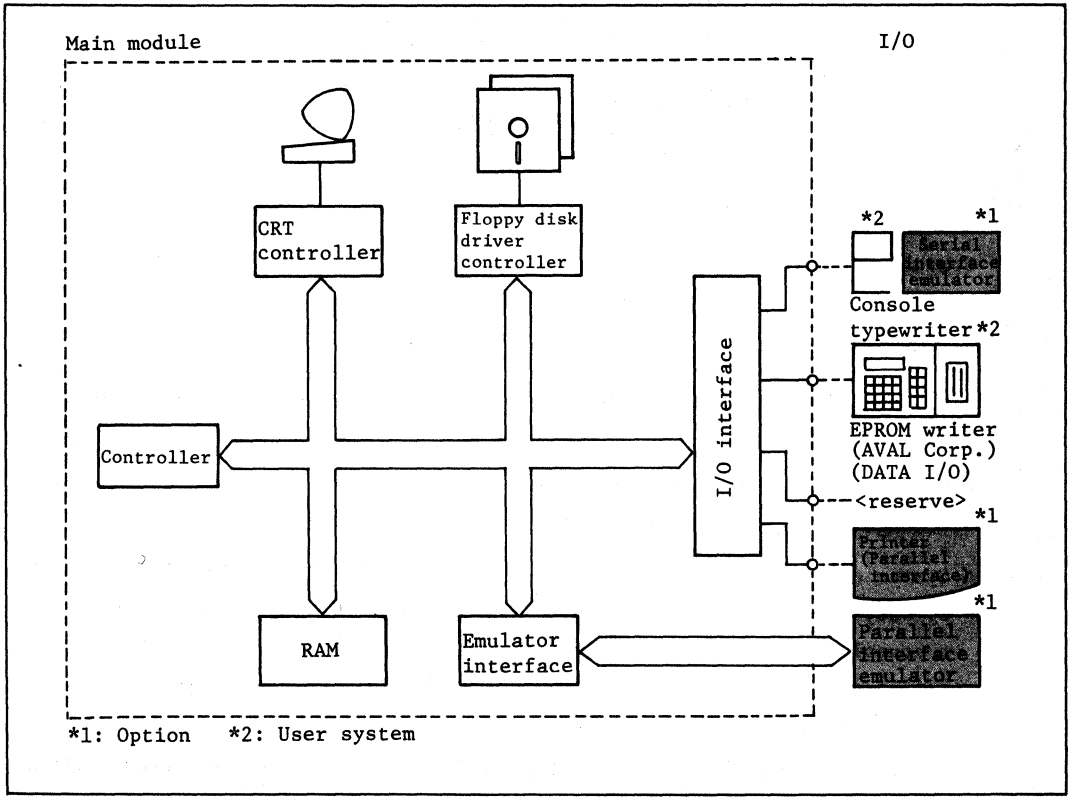


Fig. 13-3 H68SD5 Hardware Configuration

13.2.2 Development System H680SD200

The H680SD200 is a desk top type development support tool devoted to design and develop application systems (4-bit, 8-bit, 16-bit micon system) which make use of the 16-bit microprocessor HD68000. Its front view is shown in Fig. 13-4.

The H680SD200 consists of a CRT, two floppy disk drivers, and control sections in one body. The hardware configuration is shown in Fig. 13-5. The H680SD200 incorporates parallel interface (equivalent to CENTRONICS') and serial interface for EPROM writer. The emulator for the HMCS404, HD6301X, HD6301Y, and ASE for the real time emulator 68000, and ASE for the 64180 can be connected to the H680SD200. Refer to Fig. 13-7 and 13-8.

Software configuration is shown in Fig. 13-6. Assembler, C compiler, and CRT editor operate under the operating system CP/M-68K®, and FORTRAN, super PL/H (option) are provided as high level language compiler for the HD68000.

Features of the H680SD200 are listed below.

(1) General operating system CP/M-68K is applied.

(2) CRT editor which can edit on CRT display is equipped.

(3) 2M bytes of memory can be used by double sided floppy disk driver.

(4) C language, high level language for 68000 is equipped.

(5) High level language super PL/H and FORTRAN can be used for 68000 program development.

(6) Real time emulator 68000ASE for 68000(12.5MHz max) and ASE for 64180 can be used. (Option)

(7) Printer interface (equivalent to CENTRONICS') and EPROM writer interface (equivalent to RS-232C) are provided to make printer-EPROM writer connection easy. Model PKW-7000/PKW-1000 (EPROM writer of AVAL CORPORATION) and Series 22, Model 29A (DATA I/O) can be connected. (Option)

(8) Additional equipment of serial I/O board can supports emulator interface to 4-bit or 8-bit device and VAX-11 interface. (Option)

(9) Assembler super PL/H is supported as 68000 symbolic debugger. (Option)

(10) High speed operation can be realized using RAM disk.

CP/M-68K® is a registered trade mark of Digital Research.

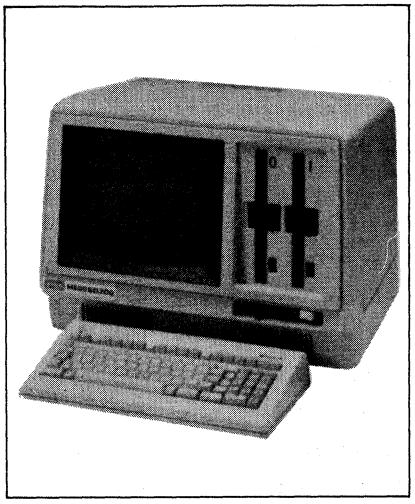


Fig. 13-4 Front View of H680SD200

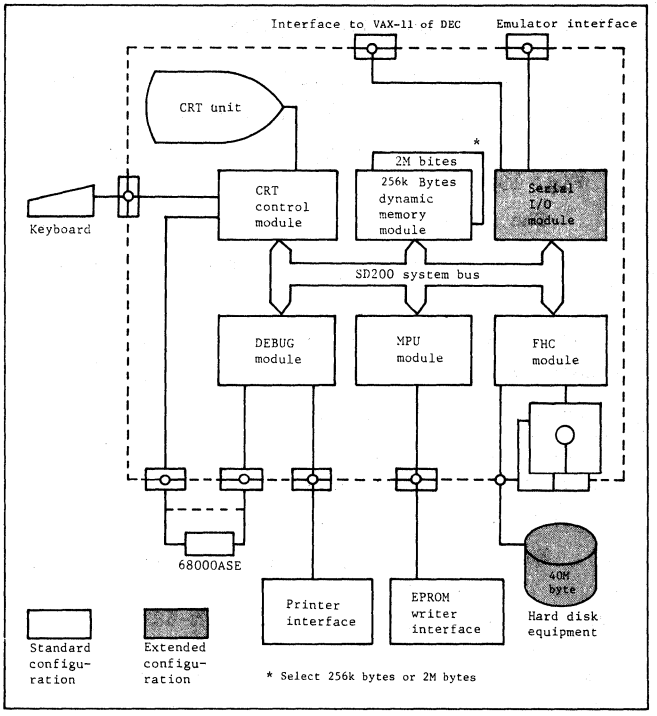


Fig. 13-5 Hardware Configuration of H680SD200

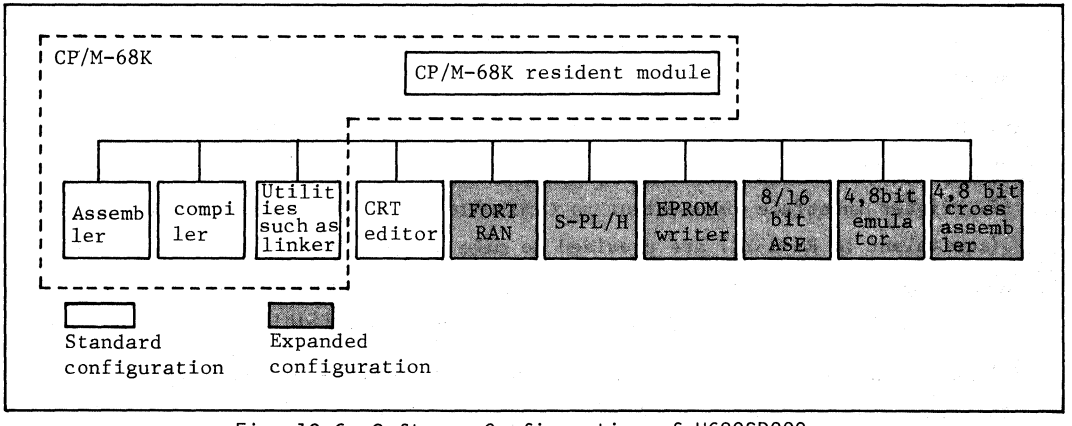


Fig. 13-6 Software Configuration of H680SD200

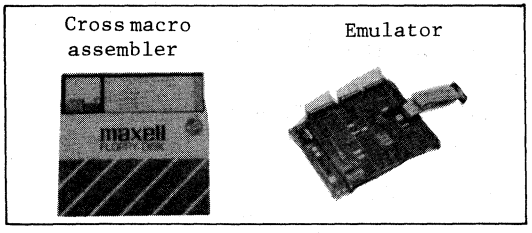


Fig. 13-7 Emulator for 4-bit/8-bit Singlechip

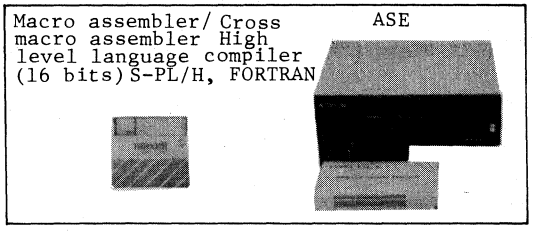


Fig. 13-8 ASE for 8-bit/16-bit Multichip

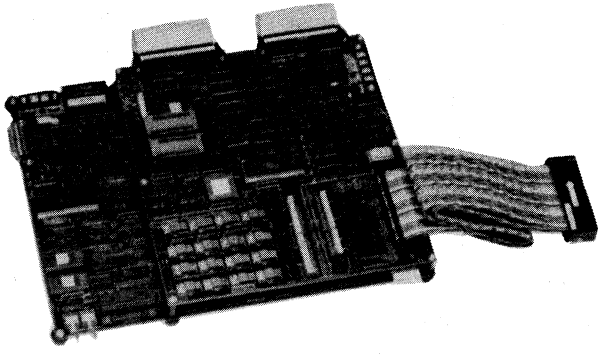
13.3 Emulator

This emulator is a completely integrated hardware and software development system for Hitachi's 4-bit singlechip microcomputer HMCS400 series. It supports the following devices.

HMCS402C/CL/AC, HMCS404C/CL/AC, HMCS408C/CL/AC
HMCS412C/CL/AC, HMCS414C/CL/AC
HMCS424C/CL/AC, HMCS428C/CL/AC

The emulator develops application system by connecting to a host computer or console. It provides three system configurations depending on the host computer or console.

- o H400CMIX2 for H68SD5 or H68SD5A
- o HS408EMX22H for H680SD200
- o HS408EML02H for the host computers or consoles except for above.



13.3.1 Features

- o Designed to aid in development of software and hardware when connected with the user system.
- o Can be connected with host system (H68SD5/5A, H680SD200, IBM-PC, etc.) CRT console, and console typewriter.
- o Takes in and displays the result of the user program execution in real time up to 2,048 cycles.
- o Provides eight external probes, which observe user system operations with real-time trace, useful as breakpoint conditions.
- o Provides HELP function to display all commands used in the emulator.
- o Can debug with the specified frequency.

- o Break function
  - Enables to set four breakpoints by any combination of program counter, instruction word, interrupt processing execution and external prove state.
  - Enables to execute continuously machine cycles 0 to 2000 after the above breakpoint (trigger point).
- o Line assembler and disassembler
  - Can display and/or change the user object program in mnemonics.
  - Displays the user object program in the specified address in mnemonics.
- o Displays the address (in the user object program) of detected instruction.
- o Displays the execution time.
- o Displays the trace data for program testing.
- o Displays and/or changes MCU register, I/O ports and memory contents.
- o Program can be executed on RAM or EPROM based.

The H400CMIX2 (for H68SD5/5A)/ HS408EMX22H (for H680SD200)/ HS408EML02H (for other host systems) emulator supports software and hardware when connected with the HMCS400, 410, 420 series 4-bit Microcomputer Unit (MCU).

Normal development procedure is;

(1) make or change the user system,
(2) translate to user object program by 400 assembler on host system,
(3) download user object program through the serial interface,
(4) set the breakpoints at any conditions,
(5) execute real-time or single-step emulation from the specified address,
(6) display and/or change user object program in mnemonics, MCU internal registers, I/O ports and internal RAMs,
(7) repeat steps (4) to (6) and upload to host system for debugging at next time.

The emulator itself works as the object MCU. The emulator is controlled by host system, CRT console, or console typewriter.

BREAKPOINT FUNCTION

H400CMIX2/HS408EMX22H/HS408EML02H allow the user to set four break conditions (TR, BR1 to BR3) in the user program. Each break condition consists of PC (Program Counter) or AB (Address Bus), DB (Data Bus), Interrupt an eight external probe signals. When one of break condition is detected, the emulator stops the user program, and displays the next PC, next instruction, execution time, MCU register and I/O contents and internal RAM contents. TR, one of break conditions, stops the user program after proceeding the machine cycles designated by COUNT if break condition matches.

REAL-TIME TRACE

H400CMIX2/HS408EMX22H/HS408EML02H can display execution results of up to 2047 machine cycles. They have two display types, one is to display by the instruction unit, the other one is to display by the machine cycle unit. The former one displays only program counter and instruction mnemonics, the latter displays address bus, data bus, interrupt occurrence and eight external probes in each machine cycle.

DISPLAY/CHANGE IN MNEMONICS

H400CMIX2/HS408EMX22H/HS408EML02H have two display and/or change commands. One is I command which is executed in hexadecimal, the other is A command which is executed in mnemonics. Command A is very useful for debugging. And H400CMIX2/H408EMX22H/HS408EML02H display in mnemonic instruction. They are disassemble and display the object program in the specified address with DA command.

DEBUG WITH A SPECIFIED FREQUENCY

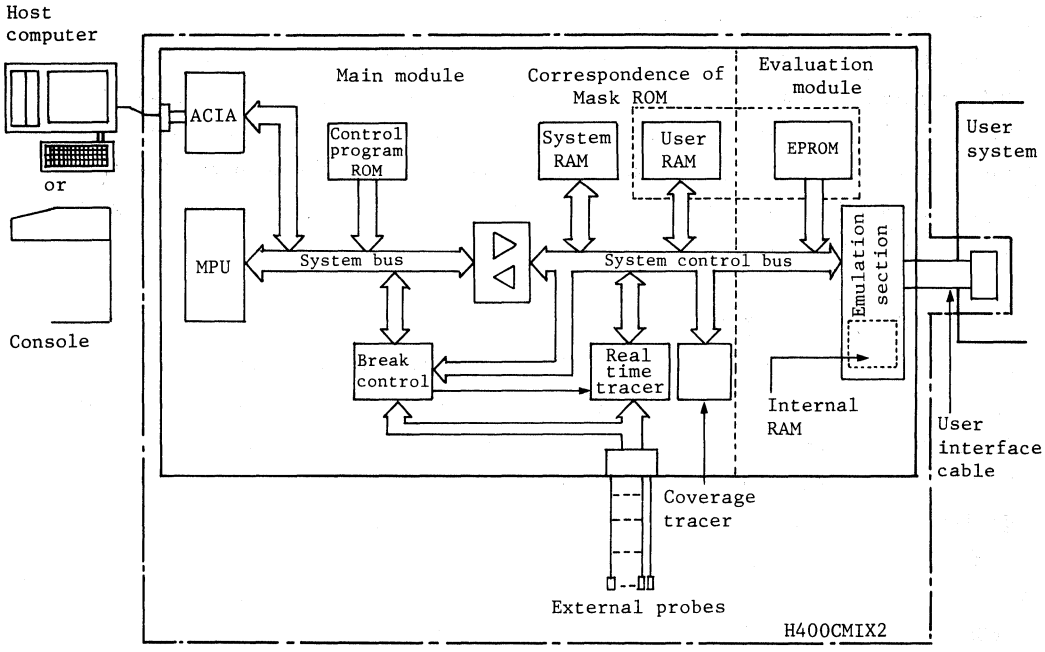
H400CMIX2/HS408EMX22H/HS408EML02H allow some frequency. The frequency is expressed as the clock cycle in user's manual; the cycles should be 1μs, 1.33μs, 2μs, 3μs, 4μs, 5μs, 7μs, 10μs, 20μs. They won't allow other frequency than the above. They select EXT by F command and input the specified frequency to OSC pin.

COVERAGE TRACE

Coverage trace marks the passed address into coverage memory. Coverage function is effective to know which address will be tested. This coverage memory can't be cleared by "G" or "S" command but can be cleared by CO RES command. When starting the test of the user program, clear the coverage memory. For the next step, test the user program with G or S command. G or S command execution marks the passed address into coverage memory. Check the passed address and continue the test until all specified addresses were passed.

EXECUTION ON USER SYSTEM POWER SUPPLY

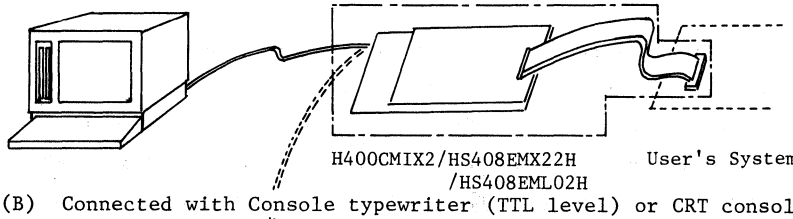
H400CMIX2/HS408EMX22H/HS408EML02H emulate on 3V to 5V. Since H400CMIX2/HS408EMX22H/HS408EML02H themselves work 5V ± 5%, connect V_{CC} to power supply for emulator. However, since evaluation chip works on user system power supply (3V to 5V), connect Vus to user system V_{CC} and V_{disp} to power supply for display (0 to -35V).



Hardware Configuration of the Emulator

13.3.2 System Configuration

(A) Connected with SD5/SD5A/SD200/IBM-PC/the other HOST



(B) Connected with Console typewriter (TTL level) or CRT console (RS-232C level)

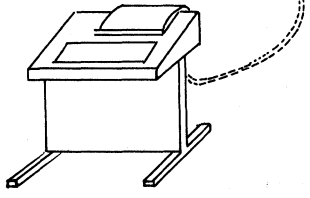


Table 13-2 Connectable Console Typewriter

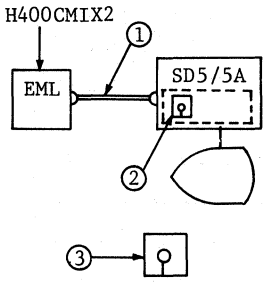
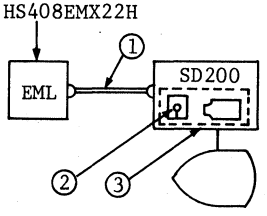
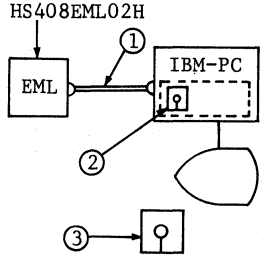
Company Name	TYPE
CASIO COMPUTER CO.	Typuter model 750-T-02
SHARP CO.	Sharpwriter model 300*
CITIZEN WATCH CO.	Protyper model 7652

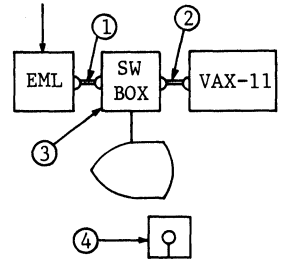
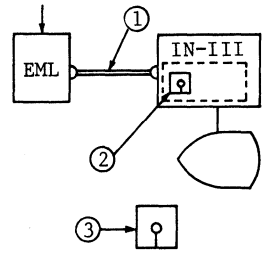
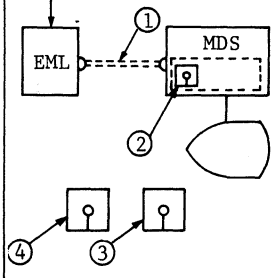
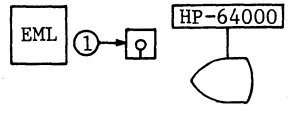
*HITACHI Specification

13.3.3 Emulator commands

COMMAND	DESCRIPTION
A	Displays and modify object program in mnemonics.
TR BR1 to 3	Sets, displays and cancels breakpoints and trigger point.
C	Compares object program.
CO	Coverage tester
DA	Disassembler
F	Sets and displays clock cycle.
G	Executes user program.
HE	Displays the emulator command information.
I	Displays and modifies object program in hexadecimal.
ID	Dumps object program.
IO	Displays and modifies I/O ports.
L	Loads object program.
M	Displays and modifies internal RAM.
MD	Dumps internal RAM.
N	Designates transfer rate.
O	Searches specified bit pattern.
P	Punches object program.
Q	Displays data of real-time trace.
R	Displays and modifies register.
S	Single-step trace of user program.
T	Transfers object program.
U	Sets and displays EPROM/user RAM.
V	Verifies object program.

13.3.4 Host connection configulation

<table>
<tr><th>Host</th><th>Configuration</th><th>Item</th><th>Remarks</th><th colspan="5">1 Cable Connection Table</th></tr>
<tr><td rowspan="4">H68SD5/5A (HITACHI)</td><td rowspan="4">OS:FDOS-III/IV
 
</td><td>① RS-232C Interface Cable</td><td rowspan="2">Included in H400CMIX2</td><td colspan="2">SD5/5A</td><td rowspan="9"></td><td colspan="2">EMULATOR</td></tr>
<tr><td>② Interface Program
S68EML1-F
Cross Assembler
S400XAS3F</td><td>SIGNAL</td><td>no</td><td>no</td><td>SIGNAL</td></tr>
<tr><td rowspan="2">③ Format converter for
H68SD5/5A↔H680SD200
S68CNV1-F</td><td rowspan="2">Provided by HITACHI</td><td>GND</td><td>1</td><td>1</td><td>GND</td></tr>
<tr><td>SEND</td><td>2</td><td>2</td><td>SEND</td></tr>
<tr><td rowspan="4">H680SD200 (HITACHI)</td><td rowspan="4">OS:CP/M-68K
 
</td><td>① RS-232C Interface Cable</td><td rowspan="3">Included in HS408EMX22H</td><td rowspan="4" colspan="7">Same as above</td></tr>
<tr><td>② Interface Program
S680EML1F
Cross Assembler
S400XAS6F</td></tr>
<tr><td>③ SIO Board
H680SI01S</td><td>Included in H680SD200</td></tr>
<tr><td>RECEIVE</td><td>3</td><td>3</td><td>RECEIVE</td></tr>
<tr><td rowspan="5">IBM-PC (IBM)</td><td rowspan="5">OS:PC-DOS
 
</td><td rowspan="2">① RS-232C Interface Cable</td><td rowspan="2">Modify it according to the right</td><td colspan="2">IBM-PC</td><td rowspan="11"></td><td colspan="2">EMULATOR</td></tr>
<tr><td>SIGNAL</td><td>no</td><td>no</td><td>SIGNAL</td></tr>
<tr><td rowspan="2">② Interface Program
S31IEM1-F</td><td rowspan="2">Provided by HITEC-UK/ DESC</td><td>GND</td><td>1</td><td>1</td><td>GND</td></tr>
<tr><td>SEND</td><td>2</td><td>2</td><td>SEND</td></tr>
<tr><td rowspan="3">③ 400 Cross Assembler for IBM-PC 400PASI1F</td><td rowspan="3">Provided by HITEC-AS/ DESC</td><td>RECEIVE</td><td>3</td><td>3</td><td>RECEIVE</td></tr>
<tr><td>RTS</td><td>4</td><td>4</td><td>RTS</td></tr>
<tr><td>CTS</td><td>5</td><td>5</td><td>CTS</td></tr>
<tr><td></td><td>DSR</td><td>6</td><td></td><td></td></tr>
<tr><td></td><td>DTR</td><td>20</td><td></td><td></td></tr>
<tr><td></td><td>GND</td><td>7</td><td>7</td><td>GND</td></tr>
</table>

Host	Configuration	Item	Remarks	1 Cable Connection Table
VAX-11 (DEC)	OS:VMS HS408EML02H 	① RS-232C Interface Cable(1)	Included in HS408EML02H	SW BOX SIGNAL / no — EMULATOR no / SIGNAL GND 1 — 1 GND SEND 2 — 2 SEND RECEIVE 3 — 3 RECEIVE RTS 4 — 4 RTS CTS 5 — 5 CTS GND 7 — 7 GND DCD 8 — 8 DCD DTR 20 — 20 DTR
		② RS-232C Interface Cable(2)	Connecting table is TBD.	
		③ Switch Box *	Provided by HMSI	
		④ 400 Cross Assembler for VAX-11 S400VAS1F	Provided by HITACHI	
IN-III (SOPHIA)	OS:CP/M HS408EML02H 	① RS-232C Interface Cable	Modify it according to the right	IN-III SIGNAL / no — EMULATOR no / SIGNAL GND 1 — 1 GND SEND 2 — 2 SEND RECEIVE 3 — 3 RECEIVE RTS 4 CTS 5 — 5 CTS GND 7 — 7 GND DCD 8 DTR 20
		② Interface Program IN3EVM	Provided by SOPHIA	
		③ 400 Cross Assembler CP/M S400MDS2F	Provided by HITACHI	
MDS (INTEL)	OS:CP/M, ISIS-II HS408EML02H 	① RS-232C Interface Cable	Connecting table is TBD.	—
		② Interface Program (under planning)	Planning by HITEC-UK/DESC	
		③ 400 Cross Assembler CP/M S400MDS2F	Provided by HITACHI	
		④ 400 Cross Assembler ISIS-II S400MDS1F		
HP-64000 (HP)	HS408EML02H OS:CP/M 	① 400 Cross Assembler for HP-64000 YS-1005XS*	Provided by YHP	—

* Under development

13.3.5 Specifications

MODULE	ITEM	SPECIFICATIONS
Main Module	MPU	HD6303X Microcomputer Unit
	Clock	4MHz
	Serial I/O	Interface level: RS-232C level TTL level Baud rates: 300,1200,4800, 9600,19200,BPS (Using HD6850P ACIA)
	User Program execution	Programs are executed from optional address on the MCU.
	User program break	* Break at breakpoint (breakpoints--max. 4 places) * Break after executing the one instruction at a single step * Break with ABORT key or switch * Break at trap
	Real-time trace	Trace memory size: . 2048 machine cycle's worth trace information. Trace information: . Program Counter (PC) . Instruction Address (AB) . Instruction Data (DB) . Interrupt Occurrence . External Probes---8 Display Contents: . Trace information, op-codes and operands listed above are displayed in mnemonic.
	Trap	If an op-code error occurs while executing a user program, address of the next instruction is displayed.
	Module dimensions	Width: 365mm (14.4 in.) Length: 275mm (10.8 in.)

MODULE	ITEM	SPECIFICATIONS
Evaluation Module	MCU	Evaluation chip: HD614088
	Gate array	HD61L033B
	Clock cycle	1, 1.33 2, 3, 4, 5, 7, 10, 20 µs, External
	EPROM sockets	User program debugging area: 16kW (10 bit × 16k)
	Pull-up/ pull-down resistors	Pull-up resistors (47kΩ) or pull-down resistors (200kΩ) are mounted at all ports.
	Power MOS transistors	For high voltage ports (2SJ76 × 16)
	Vdisp power supply	0 to -35V
	Vus Power supply	+3V to +5V
	Module dimensions	Width: 265mm (10.4 in.) Length: 275mm (10.8 in.)
Main Module and Evaluation module	Power supply voltage	DC +5V ± 5%
	Current consumption	4 A(Typ.) 6.5 A(Max.)
	Environmental conditions	Operation: Temperature 15 to 40°C Humidity: 30 to 85% RH (with no condensation) Atmosphere: Must be no corrosive gases present.
		Storage Temperature: 5 to 50°C 30 to 90% RH (with no condensation) Atmosphere: Must be no corrosive gases present.

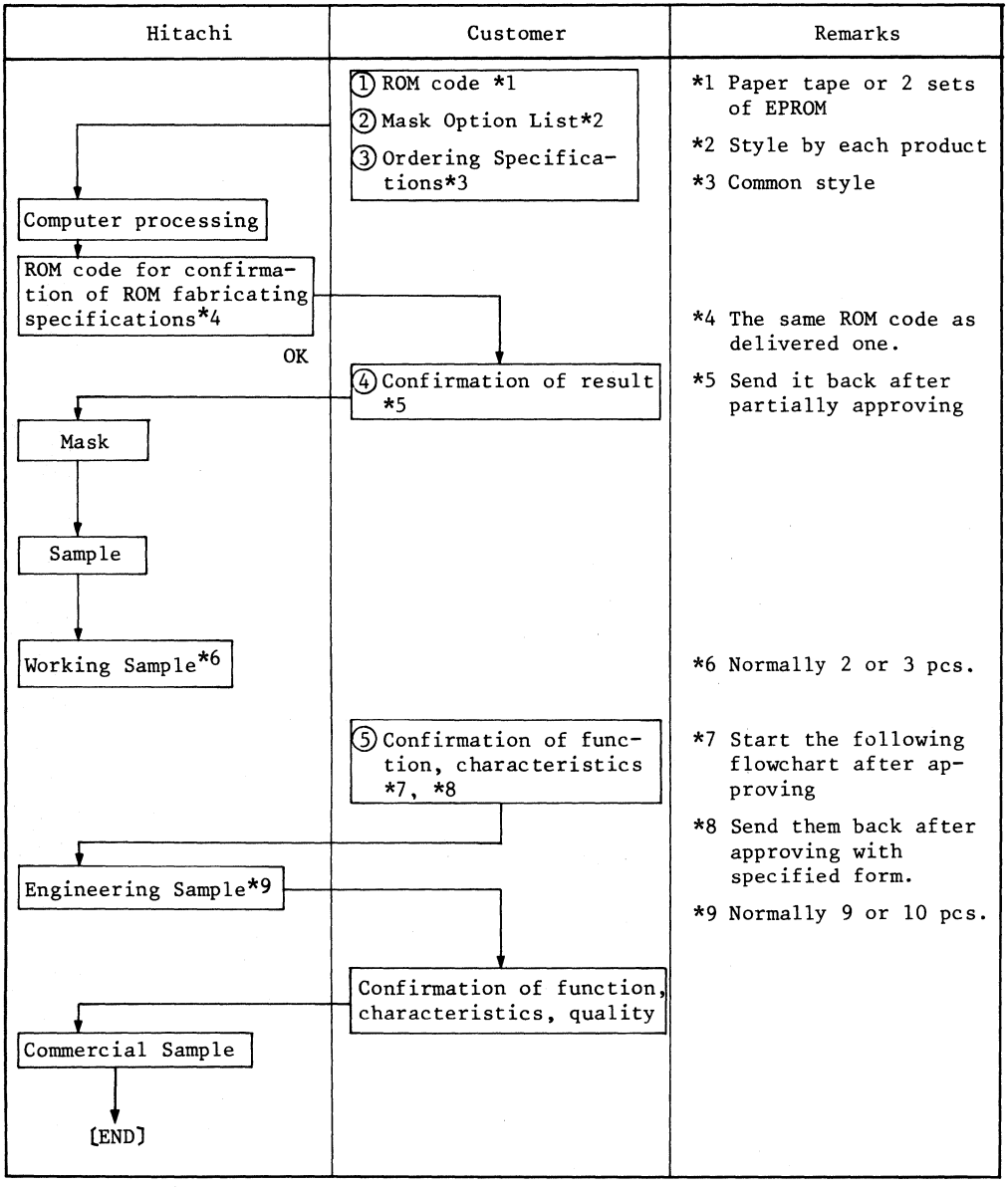
13.3.6 Ordering information

Support Devices	HMCS402C/AC/CL, HMCS404C/AC/CL, HMCS408C/AC/CL, HMCS412C/AC/CL, HMCS414C/AC/CL, HMCS424C/AC/CL, HMCS428C/AC/CL											
Equipment Supplied		Order name	H400CMLX2	H5408EMX22H	H5408EML02H	S68EML1-F	S680EML1F	S400XAS3F	S400XAS6F	S400VAS1F	S400MDS1F	S400MDS2F	S68CNV1-F	Remarks
ITEM	CONTENTS													
Emulator	Emulator boards for above devices		1	1	1	-	-	-	-	-	-	-		
Cables	64 pin User System Interface Cable		1	1	1	-	-	-	-	-	-	-		
	42 pin User System Interface Cable		1	1	1	-	-	-	-	-	-	-		
	Serial I/O Cable (for Host system)		1	1	1	-	-	-	-	-	-	-		
	Power Supply Cable for V_{CC}, Vus, and Vdisp		3	3	3	-	-	-	-	-	-	-		
	GND reinforcement cable		1	1	1	-	-	-	-	-	-	-		
Connector	Serial I/O Connector (for console)		1	1	1	-	-	-	-	-	-	-		
Probes	External Probe for real-time trace		1	1	1	-	-	-	-	-	-	-		
User's Manual	H408EML02HE (for Emulator)		1	1	1	-	-	-	-	-	-	-		
	S68EML1EM (for SD5/5A interface)		1	-	-	1	-	-	-	-	-	-		
	S680EML1EM (for SD200 interface)		-	1	-	-	1	-	-	-	-	-		
	S400XAS3EM		1	-	-	-	-	1	-	-	-	-		
	S400XAS6EM		-	1	-	-	-	-	1	-	-	-		
	S400VAS1EM		-	-	-	-	-	-	-	1	-	-		
	S400MDS1EM		-	-	-	-	-	-	-	-	1	-		
	S400MDS2EM		-	-	-	-	-	-	-	-	-	1		
	S68CNV1-EM		-	-	-	-	-	-	-	-	-	-	1	
Software for SD5/5A	S68EML1-F (FDOS-IV on SD5/5A)		1	-	-	1	-	-	-	-	-	-		
	S400XAS3F (FDOS-IV on SD5/5A)					-	-	1	-	-	-	-		
Software for SD200	S680EML1F (CP/M-68k)		-	1	-	-	1	-	-	-	-	-		
	S400XAS6F (CP/M-68k)					-	-	-	1	-	-	-		
Cross Assembler for VAX-11	S400VAS1F (VMS)		-	-	-	-	-	-	-	1	-	-		
Cross Assembler for MDS	S400MDS1F (ISIS-II)		-	-	-	-	-	-	-	-	1	-		
	S400MDS2F (CP/M)		-	-	-	-	-	-	-	-	-	1		
Converter for SD5/5A - SD200	S68CNV1-F (on SD5/5A)		-	-	-	-	-	-	-	-	-	-	1	
Your Host Computer	H68SD5/5A	OS: FDOS-III/IV	o											
	H680SD200	OS: CP/M-68k		o										
	IBM-PC	OS: PC-DOS			o									S311EM1-F(HITEC-UK/DESC) AS 400PAS11F(HITEC-AS/DESC)
	VAX-11 (DEC)	OS: VMS			o					o				SW BOX (HMSI)
	IN-III (SOPHIA)	OS: CP/M			o							o		IN3EVM (SOPHIA)
	MDS (INTEL)	OS: CP/M			o							o		
	MDS (INTEL)	OS: ISIS-II			o						o			
	HP-64000 (HP)	OS: CP/M			o									TS-1005XS (YHP)

13.4 Single-chip Microcomputer ROM Ordering Procedure

(1) Development flowchart

Single-chip microcomputer device is developed according to the following flowchart after program development.



(Note) Please send in ①, ②, and ③ at ROM ordering, and send back ④, ⑤ after approving.

(2) Information to be submitted

(a) Ordering specifications; standard format for all Hitachi single-chip microcomputer devices. Please enter for the following items. The format is shown on the next page.

Basic ITEM
Environmental Check List
Check List of attached data
Customer

(b) ROM code; Include 2 sets of ROM code identical to the EPROM contents, with ROM code No. entered on them. A program listing is desirable for easy confirmation of program contents.

(3) Change of ROM code

Note that if you change the ROM code once sent in or other specifications, the ROM must be developed from the beginning. The mask charge must be applied again in this case.

(4) Samples and mass production

Working Sample; Sample for confirmation of ROM code and that of mask option. Normally 2 or 3 samples are sent, but not guaranteed as for reliability. Please evaluate and approve immediately because the following sample preparation and mass production are determined after obtaining your evaluation.

Engineering Sample; Sample for evaluating device reliability. 10 pcs are included in mask charge.

Commercial Sample; Sample for set trial production purchased with compensation.

Mass Production; Products for actual mass production. Please enter plan of mass production in full. Refer to Single-chip Microcomputer ROM Ordering Procedure (document No. HMCS·ORD-3M) for details.

Single Chip Microcomputer Ordering Specifications

(1) Basic ITEM (Please fill in blanks or enter check marks: -)

Microcomputer Family	
Application (in full)	
Function (in full)	
ROM Code No.	
ROM Code Media	☐Floppy ☐EPROM (ROM type name)
Outline	☐Plastic DIP ☐Plastic flat package
Operating Temperature	☐Standard ☐J specification (guarantees -40°C ∿ +85°C)
Options	

(2) Environment Check List

This check list is used as data of single chip microcomputer LSI's design reliability, but not used to control its performance assurance. Please enter usual environmental conditions.

Microcomputer Ambinet Temperature	average	°C
	range	°C ∿ °C
Microcomputer Ambient Humidity	average	%
	range	% ∿ %
Power ON Duration	average	hours/day
Max. Applied Voltage to Microcomputer	power supply	max. V
	I/O	max. V
Target Level of Reliability		☐500 fit ☐1000 fit
AQL		☐1.0 % ☐0.65 % ☐0.4 %
Remarks		

(3) Check List of Attached Data (Please fill in blanks or enter check marks:-)

ROM Code	☐ Attached	☐Delivered	☐Delivery data
Mask Option List	☐ Attached	☐Delivered	☐Delivery data

* For Hitachi's use only

LSI Type No.	
Shipping Date of ROM Fabricating specifications	
Approved Date of ROM Fabricating Specifications	

Date of Order	
Customer	
Dept.	
Accepted by	

HMCS402C/AC/CL
HMCS404C/AC/CL
MASK OPTION LIST

Date of Order	
Customer	
Dept.	
Name	
ROM Code Name	
LSI Type Number (Hitachi's entry)	

5V Operation	:☐HMCS402C, ☐HMCS404C
High Speed Operation	:☐HMCS402AC, ☐HMCS404AC
3V Operation	:☐HMCS402CL, ☐HMCS404CL

* Please enter check marks in ☐ (■, ×, ✓).

(1) I/O Option

Note (I/O options masked by ▨ are not available.)

PIN		INPUT/OUTPUT		I/O OPTION				
				A	B	C	D	E
D_0	Standard Pins	Input/Output						
D_1		Input/Output						
D_2		Input/Output						
D_3		Input/Output						
D_4	High Voltage Pins	Input/Output						
D_5		Input/Output						
D_6		Input/Output						
D_7		Input/Output						
D_8		Input/Output						
D_9		Input/Output						
D_{10}		Input/Output						
D_{11}		Input/Output						
D_{12}		Input/Output						
D_{13}		Input/Output						
D_{14}		Input/Output						
D_{15}		Input/Output						
R0	R_{00}	High Voltage Pins	Output					
	R_{01}		Output					
	R_{02}		Output					
	R_{03}		Output					
R1	R_{10}		Input/Output					
	R_{11}		Input/Output					
	R_{12}		Input/Output					
	R_{13}		Input/Output					
R2	R_{20}		Input/Output					
	R_{21}		Input/Output					
	R_{22}		Input/Output					
	R_{23}		Input/Output					
R3	R_{30}	Standard Pins	Input/Output					
	R_{31}		Input/Output					
	R_{32}		Input/Output					
	R_{33}		Input/Output					
R4	R_{40}		Input/Output					
	R_{41}		Input/Output					
	R_{42}		Input/Output					
	R_{43}		Input/Output					
R5	R_{50}		Input/Output					
	R_{51}		Input/Output					
	R_{52}		Input/Output					
	R_{53}		Input/Output					
R6	R_{60}		Output					
	R_{61}		Output					
	R_{62}		Output					
	R_{63}		Output					
R7	R_{70}		Output					
	R_{71}		Output					
	R_{72}		Output					
	R_{73}		Output					
R8	R_{80}		Output					
	R_{81}		Output					
	R_{82}		Output					
	R_{83}		Output					
R9	R_{90}		Input					
	R_{91}		Input					
	R_{92}		Input					
	R_{93}		Input					
RA	R_{A0}		Input					
	R_{A1}		Input	Please Mark on R_{A1}/V_{disp}				

* Please enter "O" in applicable item for I/O option selection.

A; Without Pull-up MOS (NMOS Open Drain) B; With Pull-up MOS

C; CMOS (not be used as Input)

D; Without Pull-down MOS (PMOS Open Drain) E; With Pull-down MOS

(2) R_{A1}/V_{disp}

R_{A1}/V_{disp}
☐ R_{A1}:Without Pull-down MOS (D)
☐ V_{disp}

* Please enter check marks (■, ×, ✓) in applicable item.

(3) Package

Package
☐ DP-64S (shrink package)
☐ FP-64

* Please enter check marks (■, ×, ✓) in applicable item.

Note) R_{A1}/V_{disp} has to be selected as V_{disp} pin exept the case that all High Pins are option D

(4) Divider (DIV)

Clock divide ratio
■ Divided-by-8

(5)

ROM Code Media
☐ EPROM: Emulator Type
☐ EPROM: EPROM On-Package Microcomputer Type

Check List of Application

(A) Oscillator (CPG option)

	☐ 402C/404C (5V Operation)	☐ 402AC/404AC (High Speed Operation)	☐ 402CL/404CL (3V Operation)
CPG option	☐ Resistor (Rf=20kΩ±2%)		
	☐ Ceramic Filter	☐ Ceramic Filter	☐ Ceramic Filter
	☐ Crystal	☐ Crystal	☐ Crystal
	☐ External Clock	☐ External Clock	☐ External Clock

* Please enter check marks (■, ×, ✓) in applicable item.

HMCS408C/AC/CL

MASK OPTION LIST

5V Operation	:☐HMCS408C
High Speed Operation	:☐HMCS408AC
3V Operation	:☐HMCS408CL

* Please enter check marks in ☐ (■, ×, ✓).

Date of Order	
Customer	
Dept.	
Name	
ROM Code Name	
LSI Type Number (Hitachi's entry)	

(1) I/O Option

Note (I/O options masked by ▨ are not available.)

PIN		INPUT/OUTPUT		I/O OPTION A	B	C	D	E
D_0	Standard Pins	Input/Output						
D_1		Input/Output						
D_2		Input/Output						
D_3		Input/Output						
D_4	High Voltage Pins	Input/Output						
D_5		Input/Output						
D_6		Input/Output						
D_7		Input/Output						
D_8		Input/Output						
D_9		Input/Output						
D_{10}		Input/Output						
D_{11}		Input/Output						
D_{12}		Input/Output						
D_{13}		Input/Output						
D_{14}		Input/Output						
D_{15}		Input/Output						
R0	R_{00}	High Voltage Pins	Output					
	R_{01}		Output					
	R_{02}		Output					
	R_{03}		Output					
R1	R_{10}		Input/Output					
	R_{11}		Input/Output					
	R_{12}		Input/Output					
	R_{13}		Input/Output					
R2	R_{20}		Input/Output					
	R_{21}		Input/Output					
	R_{22}		Input/Output					
	R_{23}		Input/Output					

PIN		INPUT/OUTPUT		I/O OPTION A	B	C	D	E
R3	R_{30}	Standard Pins	Input/Output					
	R_{31}		Input/Output					
	R_{32}		Input/Output					
	R_{33}		Input/Output					
R4	R_{40}		Input/Output					
	R_{41}		Input/Output					
	R_{42}		Input/Output					
	R_{43}		Input/Output					
R5	R_{50}		Input/Output					
	R_{51}		Input/Output					
	R_{52}		Input/Output					
	R_{53}		Input/Output					
R6	R_{60}		Output					
	R_{61}		Output					
	R_{62}		Output					
	R_{63}		Output					
R7	R_{70}		Output					
	R_{71}		Output					
	R_{72}		Output					
	R_{73}		Output					
R8	R_{80}		Output					
	R_{81}		Output					
	R_{82}		Output					
	R_{83}		Output					
R9	R_{90}		Input					
	R_{91}		Input					
	R_{92}		Input					
	R_{93}		Input					
RA	R_{A0}		Input					
	R_{A1}	High Voltage Pin	Input	Please Mark on R_{A1}/V_{disp}				

* Please enter "O" in applicable item for I/O option selection.

A; Without Pull-up MOS (NMOS Open Drain) B; With Pull-up MOS

C; CMOS (not be used as Input)

D; Without Pull-down MOS (PMOS Open Drain) E; With Pull-down MOS

(2) R_{A1}/V_{disp}

R_{A1}/V_{disp}
☐ R_{A1}:Without Pull-down MOS (D)
☐ V_{disp}

(3) Package

Package
☐ DP-64S (shrink package)
☐ FP-64

* Please enter check marks (■, ×, ✓) in applicable item.

* Please enter check marks (■, ×, ✓) in applicable item.

Note) R_{A1}/V_{disp} has to be selected as V_{disp} pin even if one high voltage pin is specified as "E".

(4) Divider (DIV)

	Divide-by-4	Divide-by-8	Divide-by-16
HMCS408AC		■	■
HMCS408C			■
HMCS408CL	■		

(5)

ROM Code Media
☐ EPROM: Emulator Type
☐ EPROM: EPROM On-Package Microcomputer Type

Check List of Application

(A) Oscillator (CPG option)

	☐ HMCS408C (5V Operation)	☐ HMCS408AC (High Speed Operation)	☐ HMCS408CL (3V Operation)
CPG option	☐ Ceramic Filter	☐ Ceramic Filter	☐ Ceramic Filter
	☐ Crystal	☐ Crystal	☐ Crystal
	☐ External Clock	☐ External Clock	☐ External Clock

* Please enter check marks (■, ×, ✓) in applicable item.

HMCS412C/AC/CL
HMCS414C/AC/CL
MASK OPTION LIST

5V Operation	:☐ HMCS412C, ☐ HMCS414C
High Speed Operation	:☐ HMCS412AC, ☐ HMCS414AC
3V Operation	:☐ HMCS412CL, ☐ HMCS414CL

* Please enter check marks in ☐ (■, ×, ✓).

Date of Order	
Customer	
Dept.	
Name	
ROM Code Name	
LSI Type Number (Hitachi's entry)	

(1) I/O Option

Note (I/O options masked by ▩ are not available.)

PIN		INPUT/OUTPUT		I/O OPTION A	B	C	D	E
D_0		Standard Pins	Input/Output				▩	▩
D_1			Input/Output				▩	▩
D_2			Input/Output				▩	▩
D_3			Input/Output				▩	▩
D_4		High Voltage Pins	Input/Output	▩	▩	▩		
D_5			Input/Output	▩	▩	▩		
D_6			Input/Output	▩	▩	▩		
D_7			Input/Output	▩	▩	▩		
D_8			Input/Output	▩	▩	▩		
D_9			Input/Output	▩	▩	▩		
D_{10}			Input/Output	▩	▩	▩		
D_{11}			Input/Output	▩	▩	▩		
D_{12}			Input/Output	▩	▩	▩		
D_{13}			Input/Output	▩	▩	▩		
D_{14}			Input/Output	▩	▩	▩		
RO	R_{00}		Output	▩	▩	▩		
	R_{01}		Output	▩	▩	▩		
	R_{02}		Output	▩	▩	▩		
	R_{03}		Output	▩	▩	▩		

PIN		INPUT/OUTPUT		I/O OPTION A	B	C	D	E
R1	R_{10}	High Voltage Pins	Input/Output	▩	▩	▩		
	R_{11}		Input/Output	▩	▩	▩		
	R_{12}		Input/Output	▩	▩	▩		
	R_{13}		Input/Output	▩	▩	▩		
R2	R_{20}		Input/Output	▩	▩	▩		
	R_{21}		Input/Output	▩	▩	▩		
	R_{22}		Input/Output	▩	▩	▩		
	R_{23}		Input/Output	▩	▩	▩		
R3	R_{30}	Standard Pins	Input/Output				▩	▩
	R_{31}		Input/Output				▩	▩
	R_{32}		Input/Output				▩	▩
	R_{33}		Input/Output				▩	▩
R4	R_{40}		Input/Output				▩	▩
	R_{41}		Input/Output				▩	▩
	R_{42}		Input/Output				▩	▩
	R_{43}		Input/Output				▩	▩
RA	R_{A1}	High Voltage Pin	Input	Please Mark on R_{A1}/V_{disp}				

* Please enter "O" in applicable item for I/O option selection.

A; Without Pull-up MOS (NMOS Open Drain) B; With Pull-up MOS

C; CMOS (not be used as Input)

D; Without Pull-down MOS (PMOS Open Drain) E; With Pull-down MOS

(2) R_{A1}/V_{disp}

R_{A1}/V_{disp}
☐ R_{A1}:Without Pull-down MOS (D)
☐ V_{disp}

* Please enter check marks (■, ×, ✓) in applicable item.

Note) R_{A1}/V_{disp} has to be selected as V_{disp} pin even if one high voltage pin is specified as "E".

(3) Divider (DIV)

	Divide-by-4	Divide-by-8	Divide-by-16
HMCS412AC /414AC		■	■
HMCS412C /414C			■
HMCS412CL /414CL	■		

(4)

ROM Code Media
☐ EPROM: Emulator Type
☐ EPROM: EPROM On-Package Microcomputer Type

Check List of Application

(A) Oscillator (CPG option)

	☐ HMCS412C/414C (5V Operation)	☐ HMCS412AC/414AC (High Speed Operation)	☐ HMCS412CL/414CL (3V Operation)
CPG option	☐ Ceramic Filter	☐ Ceramic Filter	☐ Ceramic Filter
	☐ Crystal	☐ Crystal	☐ Crystal
	☐ External Clock	☐ External Clock	☐ External Clock

* Please enter check marks (■, ×, ✓) in applicable item.

HMCS400 SERIES

Section Two

Software Application Notes

SECTION 2

HMCS400 SERIES

Section Two

Software Application Notes

SECTION 2

HITACHI®

400-3-02

PREFACE

HMCS400 is a series of 4-bit single chip microcomputers using an innovative CMOS high breakdown voltage process. This series is much improved over the HMCS40 series in such areas as direct drive of fluorescent character display tube, operating speed, functions, and program development.

APPLICATION NOTES consists of typical application programs for the HMCS400 series to help users better understand the instruction set and to provide them with references for making more customized programs.

Programs described in APPLICATION NOTES have already been debugged. However, please verify operation in actual use.

For additional information reference:

- **Section 1, HMCS400 Series User's Manual**
- **Section 3, HMCS400 Series Hardware Application Notes**

CONTENTS

ASCII CONVERSION

Page

BIT MANIPULATION

COUNTER

DATA COMPARISON

ARITHMETIC OPERATIONS

BCD/HEXADECIMAL CONVERSION

Page

SORT FUNCTION

INSTRUCTION SET

GUIDE TO USING THE HMCS400 SERIES

APPLICATION NOTES

1. Application Programs Explanation Format

Explanation of each program in APPLICATION NOTES is divided into four sections as shown in Fig. 1.1.

- Section 1 — SPECIFICATION
  - FUNCTION
  - ARGUMENTS
  - CHANGES IN CPU REGISTERS AND FLAGS
  - SPECIFICATIONS
  - SPECIFICATIONS NOTES
- Section 2 — DESCRIPTION
  - Function Details
  - User Notes
  - RAM Allocation
  - Sample Application
  - Basic Operation
- Section 3 - FLOWCHART
- Section 4 - PROGRAM LISTING

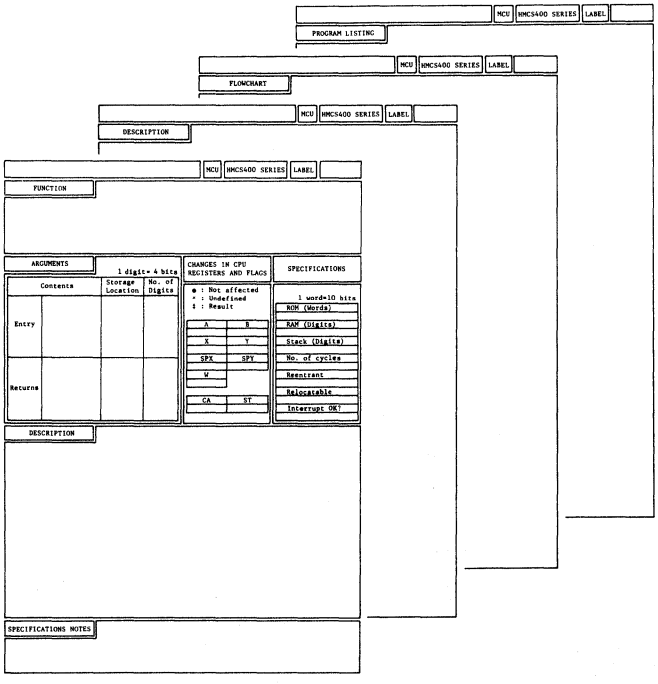


Fig. 1.1 Program Description

Programs in APPLICATION NOTES can be implemented in two ways, i.e. (1) without modification or (2) with modification.

(1) To use a program without modification, you will need:
   (a) The information in section 1
   (b) Function Details, User Notes, RAM Allocation and Sample Application in Section 2
   (c) PROGRAM LISTING in Section 4

(2) To modify a program, you will need:
   All the information in Section 1 to 4; after reading these sections, change the PROGRAM LISTING according to user specifications.

1.1 Symbols

Symbols and abbreviations used in APPLICATION NOTTES are described below.

(1) Operation

( )	= Contents
a → b	= Transfer from "a" to "b"
a ↔ b	= Exchange between "a" and "b"
+	= Addition
-	= Subtraction
×	= Multiplication
/	= Division

(2) Register symbols in MCU

A	= Accumulator
B	= B Register
W	= W Register
X	= X Register
Y	= Y Register
SPX	= SPX Register
SPY	= SPY Register

(3) Flag symbols in MCU

CA	= Carry
ST	= Status

(4) Comparison sign

=	Equal
≠	Not-equal
>	Greater than
<	Less than
≥	Greater than or equal
≤	Less than or equal

(5) Others

' '	= Delineates ASCII characters
:	= Indicates labels of successive addresses
$	= Indicates hexadecimal data
MD($***)	= Specifies a digit in address space ($*** indicates address).
MR($*)	= Specifies a digit in memory registers ($* indicates address).

MSD = Most significant digit in address space

LSD = Least significant digit in address space

1.2 SPECIFICATION Section

The SPECIFICATION Section is shown in Fig. 1.2 (: blocked area in Fig. 1.2). Each numbered item in the figure is described below.

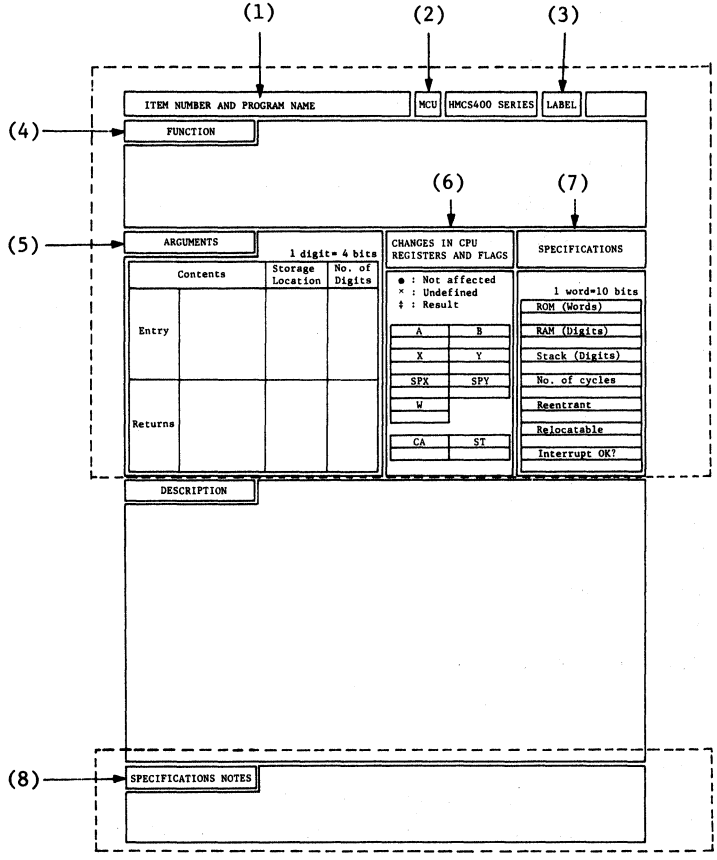
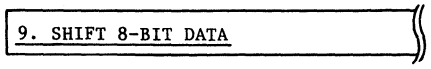


Fig. 1.2 SPECIFICATION Section

(1) ITEM NUMBER AND PROGRAM NAME:

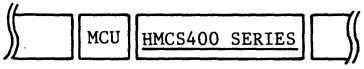
Example:



(2) MCU:

Indicates names of microcomputer series applicable to the program.

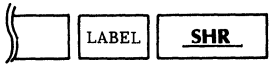
Example:



(3) LABEL:

Indicates the name identifying program entry point. Use this label to call the program.

Example:



(4) FUNCTION:

Describes program function.

Example:

FUNCTION
Shifts 8-bit binary data stored in RAM a specified number of times to the right.

(5) ARGUMENTS:

Describes entry arguments which must be initialized before program execution, and return arguments after execution.

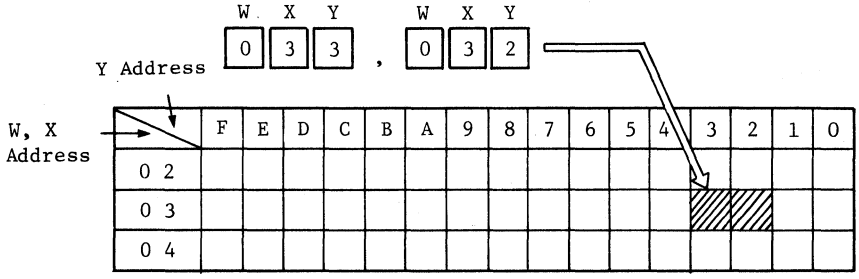
(a) Contents:

Describes arguments' contents, e.g., constant, starting address, string length.

(b) Storage Location:

Indicates registers and RAMs in which arguments must be stored. RAM locations are denoted by "(RAM)".

Note: Absolute storage locations in RAM address space are designated by MD ($WXY, $WXY) using W, X and Y addresses. For example, MD ($033, $032) refers to the marked area in the memory array shown below.



W, X, Y correspond to registers.
W, X, Y which are used to store memory addresses.

(c) No. of Digits:

Indicates arguments' digit length.

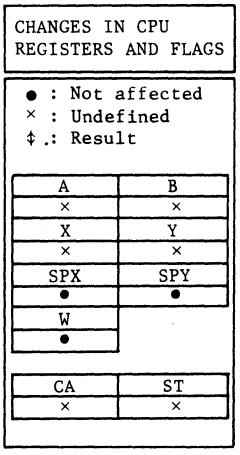
Example:

ARGUMENTS		1 digit= 4 bits	
Contents		Storage Location	No. of Digits
Entry	Unsigned 8-bit binary number to be shifted to the right	MD($033, $032)	2
	No. of shifts	B	1
Returns	Shift result	MD($033, $032)	2

(6) CHANGES IN CPU REGISTERS AND FLAGS:

Describes changes in CPU registers and flag changes in condition code register after executing a program.
The following symbols are used.

- ● : Not affected : Original contents are preserved.
- × : Undefined : Original contents are destroyed.
- ↕ : Result : Contains results of program execution.



Example:

In this example, after executing a program, contents of Accumulator, B register, X register, Y register, carry and status will be destroyed. Thus, contents which will be destroyed should be saved before executing a program.

(7) SPECIFICATIONS:

Describes program operation specifications.

(a) ROM (Words) : Indicates amount of ROM used by the program. 1 word consists of 10 bits.

(b) RAM (Digits): Indicates amount of RAM used by the program. 1 digit consists of 4 bits. (This value does not include memory needed for the stack.)

(c) Stack(Digits):Indicates amount of RAM used by the stack in the program. This memory must be reserved when the program is executed.

(d) No. of cycles:Indicates the maximum number of machine cycles. Calculate the execution time required for program execution as follows:

Execution time (sec) = Number of cycles × cycle time

Cycle time (sec) = 8/External oscillator frequency(Hz)

Note: BRS instruction is regarded as 1 cycle.

(e) Reentrant : Indicates whether a program has a structure which can be called from two or more routines at the same time.

(f) Relocatable : Indicates whether a program can be located in any memory space.

(g) Interrupt OK? : Indicates whether MCU can continue a program normally after serving an interrupt routine. If cannot ("No"), inhibit interrupt before the program is called.

Example:

SPECIFICATIONS	
1 word=10 bits	
ROM (Words)	11
RAM (Digits)	2
Stack (Digits)	0
No. of cycles	46
Reentrant	No
Relocatable	No
Interrupt OK?	Yes

(8) SPECIFICATIONS NOTES:

Explanatory notes for items listed in (7) SPECIFICATIONS.

Example:

SPECIFICATIONS NOTES
"No. of cycles" in "SPECIFICATIONS" indicates the number of cycles required to shift 8 bits of binary data 3 bits to the right.

1.3 DESCRIPTION Section

The DESCRIPTION Section is shown in Fig. 1.3. (┌┐: blocked area in Fig. 1.3). Each numbered item in the figure is explained below.

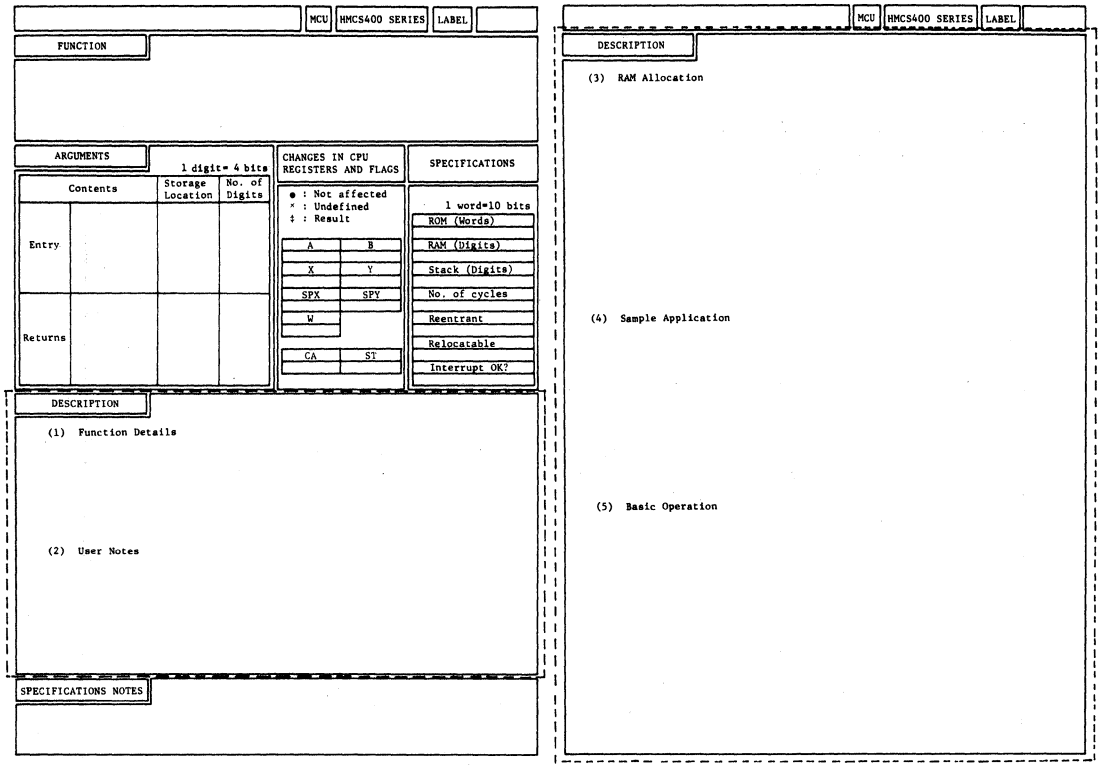


Fig. 1.3 DESCRIPTION Section

(1) Function Details:

Gives internal representation of arguments and results before and after program execution, respectively, and describes basic operation.

(2) User Notes:

Gives precautions and limitations when executing the program.

* Be sure to read these items when using the programs without modification.

Example:

DESCRIPTION

(1) Function Details

(a) Argument details

MD($033,$032): Holds 8-bit binary number to be shifted to the right. After SHR execution, contains shift result.

B : Holds number of shifts.
Contents of B = Number of actual shifts - 1
(See (2) User Notes)

(b) Example of SHR execution is shown in Fig. 1. If entry arguments are as shown in part ① of Fig. 1, 8-bit binary number is shifted to the right as shown in part ② of Fig. 1.
In this case, "0"s are shifted into the 2 leftmost bits.

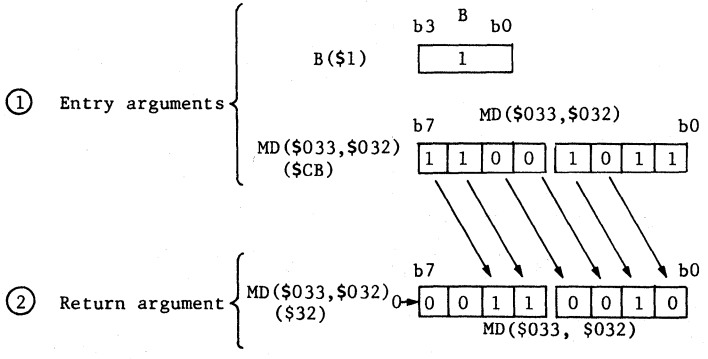


Fig. 1 Example of SHR Execution

(2) User Notes

(a) ST flag is set after SHR execution.

(b) When specifying number of shifts, load B register with number of actual shifts less 1. In part ① of Fig. 1, $1 is held in B register since number of actual shifts is 2.

(c) Number of shifts in B register must be within the range of $0 ≦ B ≦ $7, otherwise MD($033,$032) becomes "0".

(d) Shift operation permits easy multiplication of 8-bit binary number by 2^{-n}. (n = number of shifts)

(3) RAM Allocation:

Provides details of RAM Allocation for program arguments and results.

Example:

(3) RAM Allocation

W,X \ Y	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
02																
03													MSD	LSD		
04																
05																

Fig. 2 RAM Allocation

Label	RAM	Description
—	b7 b0 MSD LSD MD($033, $032)	Holds 8-bit binary number to be shifted to the right before execution. Contains shift result after execution. X and Y addresses are defined by XSFT and YSFT, respectively.

MSD : Most Significant Digit

LSD : Least Significant Digit

(4) Sample Application:

Gives a sample application of calling the program from another routine.

Example:

(4) Sample Application

Shown below is a sample application using SHR with address space allocated as follows.

MD($0A3,$0A2)	:	8-bit binary number to be shifted
MD($0A4)	:	No. of shifts
MD($0A6,$0A5)	:	8-bit binary shift result

```
LWI    $0       ..... Example with W=0.
 :
LAMD   $0A3   ┐
LMAD   $033   ├ ..... Store 8-bit binary number to be shifted
LAMD   $0A2   │       to the right in entry argument.
LMAD   $032   ┘
LAMD   $0A4   ┐ ..... Load number of shifts into entry
LBA           ┘       argument.
CALL   SHR    ..... Call SHR.
LAMD   $033   ┐
LMAD   $0A6   ├ ..... Store shift result, which is contained
LAMD   $032   │       in return argument, in RAM.
LMAD   $0A5   ┘
 :
```

(5) Basic Operation:

Explains basic operation of the program.

Example:

(5) Basic Operation

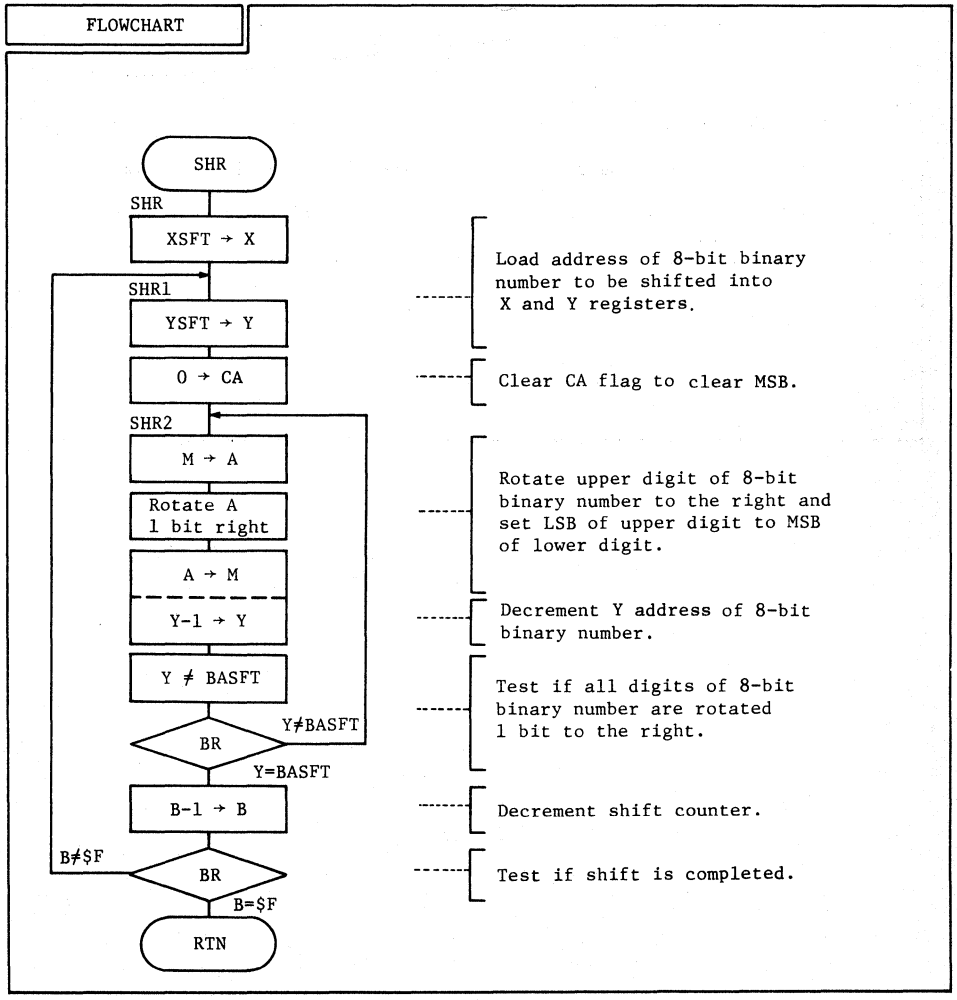
(a) 8-bit binary number is shifted to the right 1 digit at a time.

(b) B register is used to count the number of shifts, and is decremented each time (a) is executed.
Operation loops to (a) until B register becomes $F.

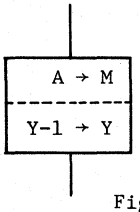
1.4 FLOWCHART Section

An example of the FLOWCHART Section is shown in Fig. 1.4. Flowchart explanatory comments are entered to the right.

Example:



Note:



A multi-instruction command is indicated by a dashed line separating each instruction.

Fig. 1.4 FLOWCHART Section

1.5 PROGRAM LISTING Section

An example of PROGRAM LISTING Section is shown in Fig. 1.5. Included with the program listing is information for modifying the program according to user's requirements.

Example:

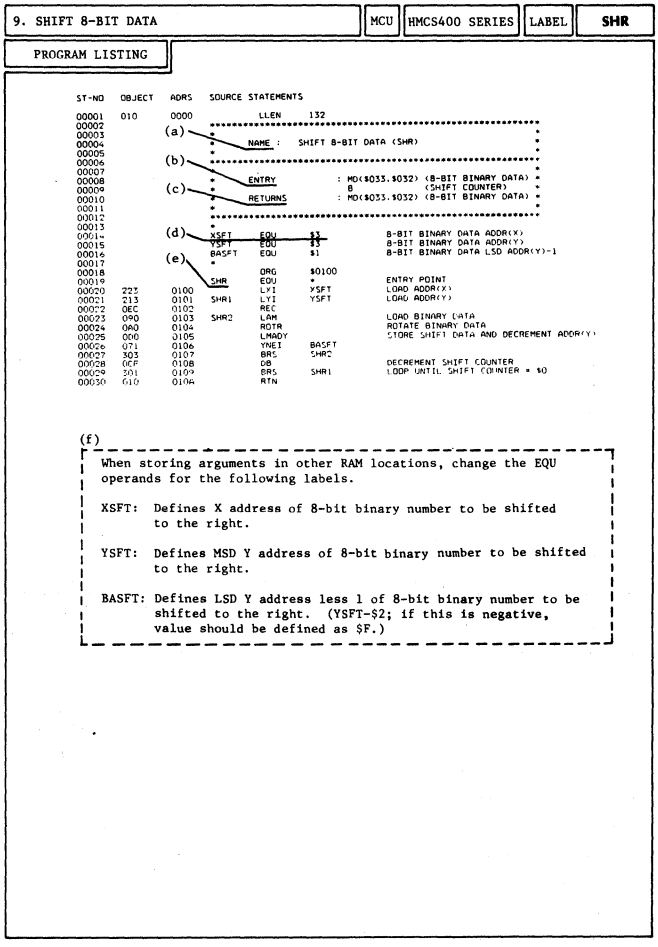


Fig. 1.5 PROGRAM LISTING Section

(a) NAME: Name of a program. ( ) means entry point label.

(b) ENTRY: Shows storage location and contents of entry arguments.

(c) RETURNS: Shows storage location and contents of return arguments.

(d) EQU: Defines RAM and its address by label.

(e) SHR: Shows entry point label.

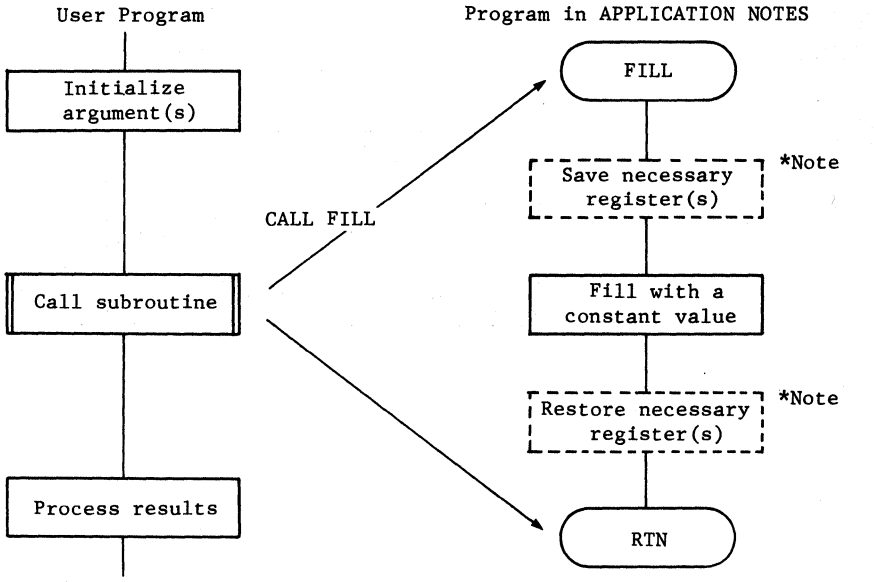
(f) Explains how to modify this application example.

2. Program Execution

The programs in APPLICATION NOTES have been considering efficiency and portability. The following shows how to execute these programs and how to modify them according to user requirements.

2.1 Calling Programs in APPLICATION NOTES from User Programs

The procedure for calling programs in APPLICATION NOTES from user programs is shown in Fig. 2.1. All programs in APPLICATION NOTES written as subroutines and should be called as shown in Fig. 2.1. An example of a user program in which a program in APPLICATION NOTES is called as a subroutine is shown in Fig. 2.2.



*Note: User must save and restore register(s), if necessary.
Refer to the comments in next page.

Fig. 2.1 Procedure for Calling Programs in APPLICATION NOTES

```
User Program

LWI     $0      ..... Example with W = 0.
  ¦
LMAD    $02D    ..... Save register.
LAMD    $0A1
LBA
LAMD    $0A2
LYA
LAMD    $0A3    ..... Initialize arguments.
LXA
LAMD    $0A4
LMAD    $04D
LAMD    $0A5
LMAD    $04C

CALL    FILL    ..... Call program.

LMAD    $02D    ..... Restore register.
  ¦
```

Fig. 2.2 Sample Application

Some programs may destroy register(s) contents before returning to the user program, since register(s) are used not only as argument(s) but as work area for calculation. Usually, register(s) used as work area is saved and restored in subroutine. The programs in APPLICATION NOTES, however, do not save nor restore register content(s).

If register(s) contents need to be saved, users must save and restore register(s) contents as shown in Figs. 2.1 and 2.2.
Refer to the "SPECIFICATIONS" section for each program to determine which registers should be saved as well as for details on subroutine arguments and results.

Also, note that the amount of RAM used for the stack by each program indicated in "SPECIFICATIONS" is in addition to that used when calling the program (4 digits). The entire stack area is 64 digits allowing for a maximum nesting level of 16. Thus, to prevent program malfunctioning, both of the above must be considered when calling (and writing) program subroutines.

2.2 Modifying Programs in APPLICATION NOTES According to User Requirements

The programs in APPLICATION NOTES may be modified depending on user requirements.

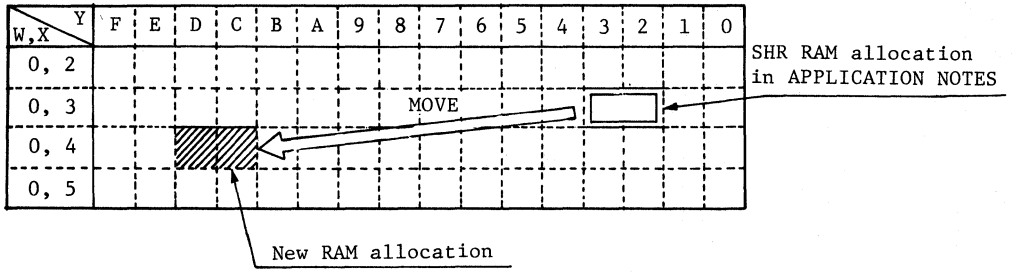


Fig. 2.3 RAM Allocation

For example, to modify RAM allocation for the SHR program as shown in Fig. 2.3, the EQU instruction for the labels shown in Fig. 2.4 must be changed as shown in Fig. 2.5, and the program them reassembled.

RAM allocations that can be modified are described after each program listing, as shown in Fig. 2.4.

```
ST-NO   OBJECT   ADRS   SOURCE STATEMENTS

00001   010      0000           LLEN    132
00002                  ************************************************
00003                  *                                              *
00004                  *      NAME :   SHIFT 8-BIT DATA (SHR)         *
00005                  *                                              *
00006                  ************************************************
00007                  *                                              *
00008                  *      ENTRY          : MD($033.$032) (8-BIT BINARY DATA) *
00009                  *                       B              (SHIFT COUNTER)    *
00010                  *      RETURNS        : MD($033.$032) (8-BIT BINARY DATA) *
00011                  *                                              *
00012                  ************************************************
00013                  *
00014                  XSFT    EQU     $3          8-BIT BINARY DATA ADDR(X)
00015                  YSFT    EQU     $3          8-BIT BINARY DATA ADDR(Y)
00016                  BASFT   EQU     $1          8-BIT BINARY DATA LSD ADDR(Y)-1
00017                  *
00018                          ORG     $0100
00019                  SHR     EQU     *           ENTRY POINT
00020   223      0100   SHR1    LXI     XSFT        LOAD ADDR(X)
00021   213      0101           LYI     YSFT        LOAD ADDR(Y)
00022   0EC      0102           REC
00023   090      0103   SHR2    LAM                 LOAD BINARY DATA
00024   0A0      0104           ROTR                ROTATE BINARY DATA
00025   000      0105           LMADY               STORE SHIFT DATA AND DECREMENT ADDR(Y)
00026   071      0106           YNEI    BASFT
00027   303      0107           BRS     SHR2
00028   0CF      0108           DB                  DECREMENT SHIFT COUNTER
00029   301      0109           BRS     SHR1        LOOP UNTIL SHIFT COUNTER = $0
00030   010      010A           RTN
```

When storing arguments in other RAM locations, change the EQU operands for the following labels.

XSFT: Defines X address of 8-bit binary number to be shifted to the right.

YSFT: Defines MSD Y address of 8-bit binary number to be shifted to the right.

BASFT: Defines LSD Y address less 1 of 8-bit binary number to be shifted to the right. (YSFT-$2; if this is negative, value should be defined as $F.)

Fig. 2.4 Program Listing with Original RAM Allocations

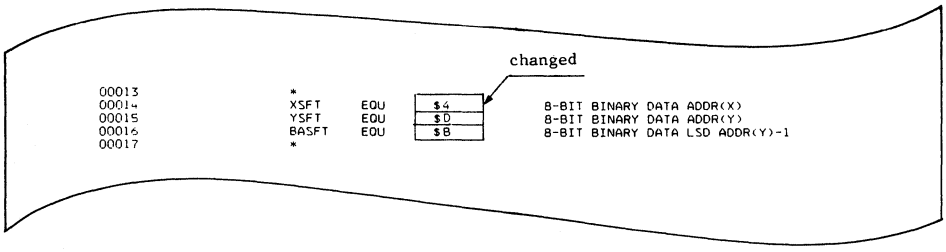


Fig. 2.5 Program Listing with New RAM Allocations

Note that in the program above, W register contents are not specified, but have been assigned as W = 0 elsewhere (W, X, and Y register are all needed for complete specification of a memory location). W register has four possible selections, from 1 to 3. If necessary, change W register contents using the LWI instruction.

PROGRAM APPLICATION EXAMPLES

1. FILL WITH A CONSTANT VALUE

MCU | HMCS400 SERIES | LABEL | **FILL**

FUNCTION

Fills a specified number of bytes in RAM with a constant value.

ARGUMENTS

1 digit= 4 bits

Contents		Storage Location	No. of Digits
Entry	Constant	HCNST, LCNST (RAM)	2
	No. of bytes	B	1
	Starting address	X, Y	2
Returns	——	——	——

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected
× : Undefined
‡ : Result

A	B
×	×
X	Y
×	×
SPX	SPY
●	●
W	
●	

CA	ST
×	×

SPECIFICATIONS

1 word=10 bits

ROM (Words)
14
RAM (Digits)
2
Stack (Digits)
0
No. of cycles
155
Reentrant
No
Relocatable
No
Interrupt OK?
Yes

DESCRIPTION

(1) Function Details

(a) Argument details

HCNST, LCNST(RAM): Holds 1-byte constant to fill an area in RAM.

B : Holds number of bytes of RAM to be filled with constant (Contents of B = actual number of bytes - 1. See (2) User Notes).

X, Y : Holds starting address in RAM.

(b) Example of FILL execution is shown Fig. 1. If entry arguments are as shown in part ① of Fig. 1, $57 in HCNST, LCNST(RAM) is stored in RAM as shown in part ② of Fig. 1.

SPECIFICATIONS NOTES

"No. of cycles" in "SPECIFICATIONS" indicates the number of cycles required to fill 16 bytes of RAM with a 1-byte constant.

DESCRIPTION

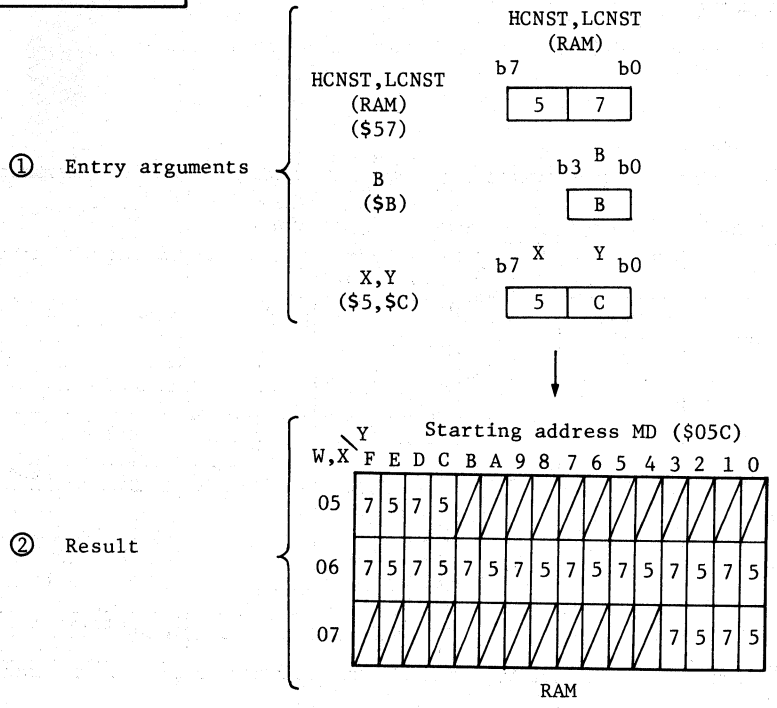


Fig. 1 Example of FILL Execution

(2) User Notes

(a) ST flag is set after FILL execution.

(b) A maximum number of 16 bytes of RAM can be filled by the constant value.

(c) Value specified in B register is the actual number of bytes - 1. In Fig. 1, as the actual byte length is 12($C), 11($B) is initialized in B register.

(d) The program expects an even address initialized in Y register for program operation.

DESCRIPTION

(3) RAM Allocation

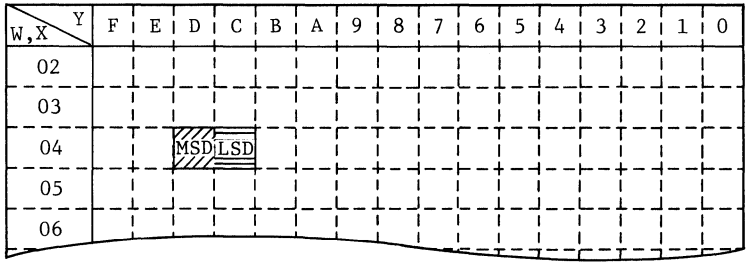


Fig. 2 RAM Allocation

Label	RAM	Description
HCNST	b3 b0 MSD MD($04D)	Upper digit of 1-byte constant to be stored in RAM.
LCNST	b3 b0 LSD MD($04C)	Lower digit of 1-byte constant to be stored in RAM.

DESCRIPTION

(4) Sample Application

Shown below is a sample application using FILL with address space allocated as follows.

MD($0A1)	:	No. of bytes of RAM to be filled with constant
MD($0A3, $0A2)	:	RAM starting address
MD($0A5, $0A4)	:	Constant value

```
        LWI     $0          ... Example with W=0.
         :
        LAMD    $0A1    }   ... Load RAM length into entry argument.
        LBA             }
        LAMD    $0A2    ]
        LYA             ]
        LAMD    $0A3    }   ... Load starting address into entry argument.
        LXA             ]
        LAMD    $0A5    ]
        LMAD    HCNST   ]
        LAMD    $0A4    }   ... Store 1-byte constant in entry argument.
        LMAD    LCNST   ]

        CALL    FILL        ... Call FILL.
         :
```

(5) Basic Operation

(a) X and Y registers are used to point to address in RAM to be filled.

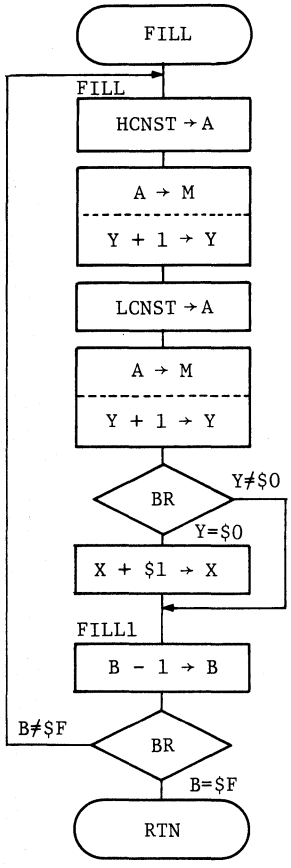
(b) Constant, in HCNST(RAM) and LCNST(RAM), is moved to X and Y address 1 digit at a time using indirect addressing mode.

(c) B register is used as a byte counter indicating how many bytes are left to be filled with the constant. It is decremented after each move and operation loops to (b) until its contents become $F.

(d) Address in X and Y registers is incremented for next storage location.

1. FILL WITH A CONSTANT VALUE	MCU	HMCS400 SERIES	LABEL	FILL

FLOWCHART



Store upper digit of constant in RAM pointed to by X and Y registers.

Increment Y address for next storage location.

Store lower digit of constant in RAM pointed to by X and Y registers.

Increment Y address for next storage location.

Test if Y address is $0.

Increment X address.

Decrement byte counter indicating number of bytes left to be filled with constant.

Test if filling is completed.

PROGRAM LISTING

```
ST-NO   OBJECT   ADRS   SOURCE STATEMENTS

00001   000 000  0000           LLEN     132
00002                  ********************************************************
00003                  *                                                      *
00004                  *       NAME :  FILL WITH A CONSTANT VALUE (FILL)      *
00005                  *                                                      *
00006                  ********************************************************
00007                  *                                                      *
00008                  *       ENTRY     :  HCNST        (UPPER CONSTANT)     *
00009                  *                    LCNST        (LOWER CONSTANT)     *
00010                  *                    B            (BYTE COUNTER)       *
00011                  *                    X,Y          (STARTING ADDRESS)   *
00012                  *       RETURNS   :  NOTHING                           *
00013                  *                                                      *
00014                  ********************************************************
00015                  *
00016                  HCNST    EQU      $04D          UPPER CONSTANT DATA ADDR
00017                  LCNST    EQU      $04C          LOWER CONSTANT DATA ADDR
00018                  *
00019                           ORG      $0100
00020                  FILL     EQU      *             ENTRY POINT
00021   190 04D  0100           LAMD     HCNST         LOAD UPPER CONSTANT
00022   050      0102           LMAIY                  LOAD CONSTANT AND INCREMENT ADDR(Y)
00023   190 04C  0103           LAMD     LCNST         LOAD LOWER CONSTANT
00024   050      0105           LMAIY                  LOAD CONSTANT AND INCREMENT ADDR(Y)
00025   308      0106           BRS      FILL1         LOOP UNTIL BYTE COUNTER = $F
00026   001      0107           XSPX                   INCREMENT ADDR(X)
00027   068      0108           LASPX
00028   281      0109           AI       $1
00029   0E8      010A           LXA
00030   0CF      010B   FILL1    DB                     DECREMENT BYTE COUNTER
00031   300      010C           BRS      FILL
00032   010      010D           RTN
```

When storing arguments in other RAM locations, change the EQU operands for the following labels.

HCNST: Defines upper digit address of 1-byte constant.

LCNST: Defines lower digit address of 1-byte constant.

2. MOVE DATA BLOCK

MCU | HMCS400 SERIES | LABEL | **MOVE**

FUNCTION

Moves data block, stored in ROM, to a specified location in RAM.

ARGUMENTS

1 digit= 4 bits

Contents		Storage Location	No. of Digits
Entry	Source starting address (ROM)	B,A	2
	Destination starting address (X)	X	1
		SPX	1
	Destination starting address (Y)	Y	1
	No. of bytes	SPY	1
Returns	—	—	—

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected
× : Undefined
‡ : Result

A	B
×	×
X	Y
×	×
SPX	SPY
×	×
W	
●	

CA	ST
●	×

SPECIFICATIONS

1 word=10 bits

ROM (Words)
33
RAM (Digits)
2
Stack (Digits)
0
No. of cycles
394
Reentrant
No
Relocatable
No
Interrupt OK?
Yes

DESCRIPTION

(1) Function Details

(a) Argument details

B,A : Holds source starting address (ROM) as a 2-digit hexadecimal number.

X : Holds destination (RAM) starting X address as a 1-digit hexadecimal number to indicate from where upper digits of source data will be stored.

SPX : Holds destination (RAM) starting X address as a 1-digit hexadecimal number to indicate from where lower digits of source data will be stored.
(Contents of SPX register = contents of X register + 1.
See (2) User Notes.)

Y : Holds destination (RAM) starting Y address as a 1-digit hexadecimal number.

SPECIFICATIONS NOTES

"No. of cycles" in "SPECIFICATIONS" indicates the number of cycles required to move 16 bytes of data.

DESCRIPTION

SPY : Holds length of data block in bytes as a 1-digit hexadecimal number.
(Contents of SPY = actual number of bytes - 1.
See (2) User Notes.)

(b) Example of MOVE execution is shown in Fig. 1. If entry arguments are as shown in part ① of Fig. 1, source data block in ROM ($0F12 ∿ $0F18) is moved to destination in RAM MD($05B ∿ $071), MD($06B ∿ $081) as shown in part ② of Fig. 1.

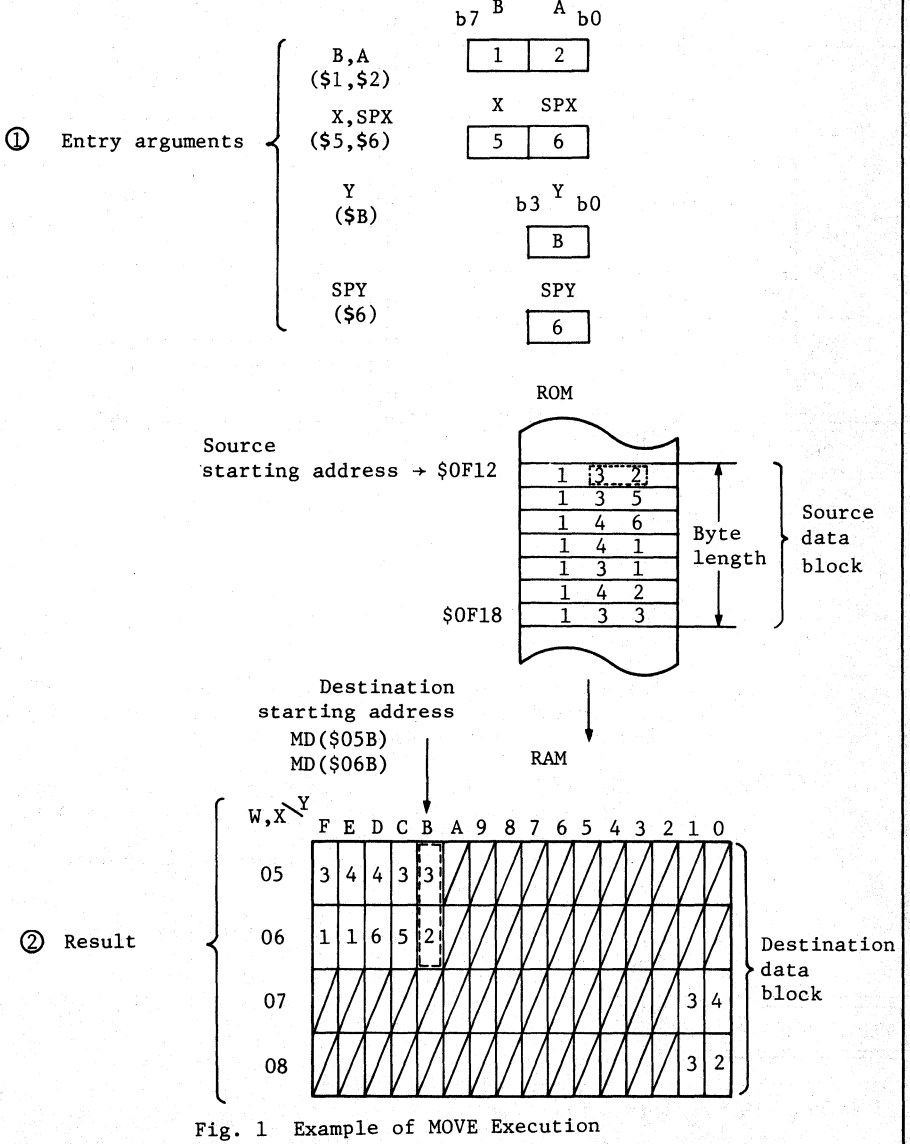


Fig. 1 Example of MOVE Execution

(2) User Notes

(a) ST flag is set after MOVE execution.

(b) When storing the length of the data block in SPY register, the actual number of bytes-1 must be stored. In Fig. 1, as the actual length is 7 bytes, $6 is stored in SPY register.

(c) 1 to 16 bytes of data can be moved.

(d) When storing the destination starting address in SPX register, contents of X register + 1 must be stored, otherwise data cannot be moved successively to RAM. In Fig. 1, as contents of X register is $5, $6 is stored in SPX register.

(e) Bit 8 of all words in the source data table must be set to indicate loading of data into Accumulator and B register when executing the pattern generation instruction (P).

(3) RAM Allocation

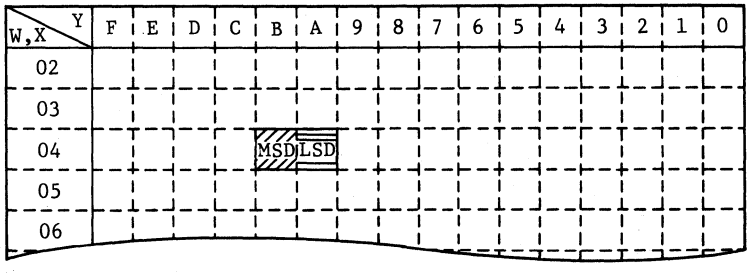


Fig. 2 RAM Allocation

Label	RAM	Description
HSOU	b3 b0 MSD MD ($04B)	Work area for saving contents of B register used to indicate source (ROM) address.
LSOU	b3 b0 LSD MD ($04A)	Work area for saving contents of Accumulator used to indicate source (ROM) address.

DESCRIPTION

(4) Sample Application

Shown below is a sample application using MOVE with address space allocated as follows.

```
MD($0A2,$0A1) : Source starting address
MD($0A5 ∿ $0A3): Destination starting address
MD($0A6)      : Length of data block in bytes
```

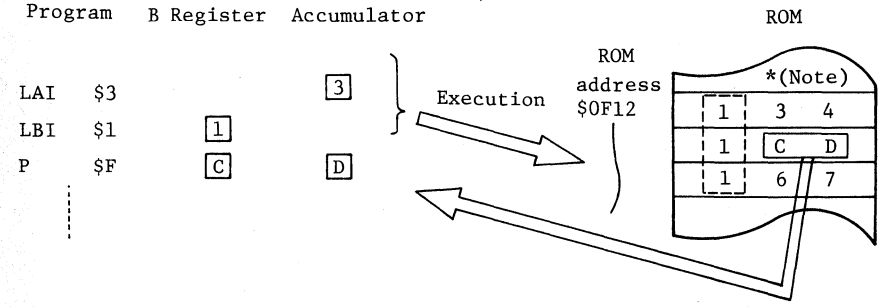
```
        LWI     $0      ..... Example with W=0.
        :
        LAMD    $0A6  ┐
        LYA           } ..... Load length of data block into
        XSPY          ┘       entry argument.
        LAMD    $0A5  ┐
        LXA           │
        XSPX          │
        LAMD    $0A4  } ..... Load destination starting address
        LXA           │       into entry argument.
        LAMD    $0A3  │
        LYA           ┘
        LAMD    $0A2  ┐
        LBA           } ..... Load source starting address into
        LAMD    $0A1  ┘       entry argument.

        CALL    MOVE  ..... Call MOVE.
        :
```

DESCRIPTION

(5) Basic Operation

(a) Data stored at the address indicated by Accumulator and B register is referred to Accumulator and B register, using the table look-up function of the pattern generation instruction (P).



After executing the pattern generation instruction in the above program sequence, $CD is contained in B register and Accumulator ($CD is stored in lower 8 bits of word located at $0F13).
Since data table is allocated from $0F00 ∿ $0FFF, subroutine MOVE uses $F as operand of the pattern generation instruction (P).

*Note:

If dotted area (bit 8) is $1 as shown above, ROM data is referred to Accumulator and B register after executing the P instruction.

Fig. 3 ROM Table and the P Instruction

(b) Accumulator and B register are used as a pointer to the source data.

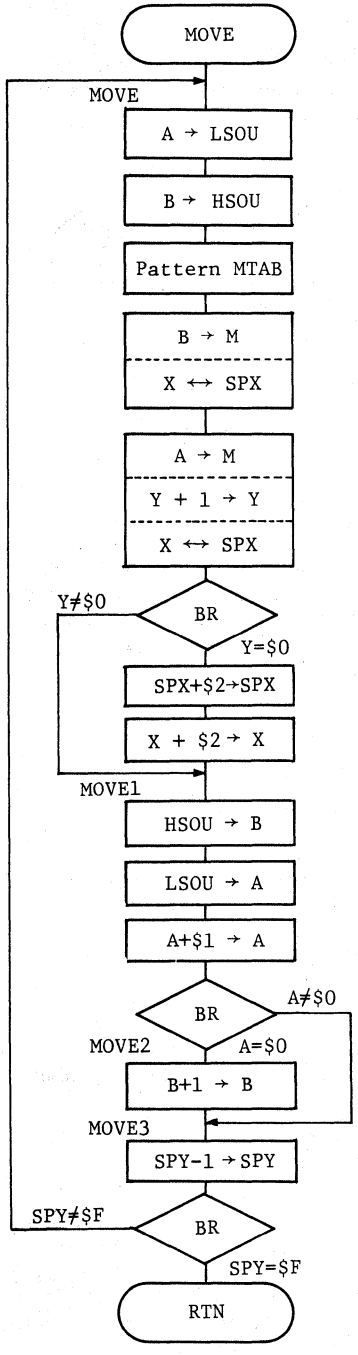
(c) 1 byte of the data word pointed to by Accumulator and B register is moved to the RAM address indicated by X and Y registers.

(d) SPY register is used as a counter indicating number of bytes in data block. SPY register is used as follows:

(i) after every execution of (c), SPY register is decremented

(ii) data continues to be moved 1 byte at a time until SPY register is $F.

FLOWCHART



Save Accumulator and B register indicating source address.

Load source address data indicated by MATB, B register and Accumulator into B register and Accumulator.

Load upper digit of source data into destination.

Load destination X address for lower digit.

Store lower digit of source data in destination.

Increment destination Y address.

Load destination X address for upper digit.

Test if Y register = $0.

Increment destination X address.

Restore Accumulator and B register indicating source address.

Increment Accumulator indicating source address.

Test if B register indicating source address needs incrementing.

Increment B register indicating source address.

Decrement counter indicating byte length of data block.

Test if move is completed.

PROGRAM LISTING

```
ST-NO   OBJECT   ADRS   SOURCE STATEMENTS

00001   000 000  0000           LLEN     132
00002                  ************************************************************
00003                  *                                                          *
00004                  *       NAME : MOVE DATA BLOCK (MOVE)                      *
00005                  *                                                          *
00006                  ************************************************************
00007                  *                                                          *
00008                  *       ENTRY   :  A,B          (SOURCE STARTING ADDRESS)  *
00009                  *                  X,Y          (DESTINATION STARTING ADDRESS) *
00010                  *                  SPY          (NO. OF BYTES)             *
00011                  *       RETURNS :  NOTHING                                 *
00012                  *                                                          *
00013                  ************************************************************
00014                  *
00015                  HSOU     EQU      $04B         WORK AREA FOR REGISTER(B)
00016                  LSOU     EQU      $04A         WORK AREA FOR ACCUMULATOR(A)
00017                  MTAB     EQU      #F           MSD (8-11BIT) OF DATA TABLE ADDRESS
00018                  *
00019                           ORG      $0100
00020                  MOVE     EQU      *            ENTRY POINT
00021   194 04A  0100           LMAD     LSOU         SAVE SOURCE ADDR(A)
00022   048      0102           LAB
00023   194 04B  0103           LMAD     HSOU         SAVE SOURCE ADDR(B)
00024   190 04A  0105           LAMD     LSOU
00025   1BF      0107           P        MTAB         LOAD SOURCE DATA
00026   0C1      0108           XMRX                  LOAD UPPER SOURCE DATA
00027   051      0109           LMAIYX                LOAD LOWER SOURCE DATA AND INCREMENT ADDR(Y)
00028   313      010A           BRS      MOVE1        BRANCH IF ADDR(Y) =/ $0
00029   068      010B           LASPX                 INCREMENT ADDR(SPX)
00030   282      010C           AI       %2
00031   001      010D           XSPX
00032   0E8      010E           LXA
00033   068      010F           LASPX                 INCREMENT ADDR(X)
00034   282      0110           AI       %2
00035   001      0111           XSPX
00036   0E8      0112           LXA
00037   190 04B  0113  MOVE1    LAMD     HSOU         LOAD SOURCE ADDR(A)
00038   0C8      0115           LBA
00039   190 04A  0116           LAMD     LSOU         LOAD SOURCE ADDR(B)
00040   281      0118           AI       $1           INCREMENT SOURCE ADDR(A)
00041   318      0119           BRS      MOVE2        BRANCH IF A = $0
00042   31C      011A           BRS      MOVE3        BRANCH IF A =/ $0
00043   04C      011B  MOVE2    IB                    INCREMENT SOURCE ADDR(B)
00044   002      011C  MOVE3    XSPY
00045   0DF      011D           DY                    DECREMENT LENGTH COUNTER
00046   002      011E           XSPY
00047   300      011F           BRS      MOVE         LOOP UNTIL LENGTH COUNTER = $0
00048   010      0120           RTN
```

When storing arguments in other RAM locations, change the EQU operands for the following labels.

HSOU: Defines address where B register, indicating source data address, is saved.

LSOU: Defines address where Accumulator, indicating source data address, is saved.

MTAB: Defines MSD of the data table address referred to by the pattern generation instruction (P).

FUNCTION

Moves string of data, stored in ROM, to a specified location in RAM using a string terminator ($00).

ARGUMENTS

1 digit= 4 bits

Contents		Storage Location	No. of Digits
Entry	Source starting address (ROM)	B,A	2
	Destination starting address (X)	X	1
		SPX	1
	Destination starting address (Y)	Y	1
Returns	——	——	——

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected
× : Undefined
↕ : Result

A	B
×	×
X	Y
×	×
SPX	SPY
×	×
W	
●	

CA	ST
●	×

SPECIFICATIONS

1 word=10 bits

ROM (Words)
40
RAM (Digits)
2
Stack (Digits)
0
No. of cycles
478
Reentrant
No
Relocatable
No
Interrupt OK?
Yes

DESCRIPTION

(1) Function Details

(a) Argument details

B,A : Holds source (ROM) starting address as a 2-digit hexadecimal number.

X : Holds destination (RAM) starting X address as a 1-digit hexadecimal number to indicate from where upper digit of source data will be stored.

SPX : Holds destination (RAM) starting X address as a 1-digit hexadecimal number to indicate from where lower digit of source data will be stored.
(Contents of SPX register = contents of X register + 1. See (2) User Notes.)

Y : Holds destination (RAM) starting Y address as a 1-digit hexadecimal number.

SPECIFICATIONS NOTES

"No. of cycles" in "SPECIFICATIONS" indicates the number of cycles required to move a string of data with the terminator in the 16th byte.

DESCRIPTION

(b) Example of MOVES execution is shown in Fig. 1. If entry arguments are as shown in part ① of Fig. 1, data in source ROM ($0F12) is moved to destination MD($05B∿$070), MD($06B∿$080) as shown in part ② of Fig. 1. When terminator ($00) is reached MCU terminates moving of data block.

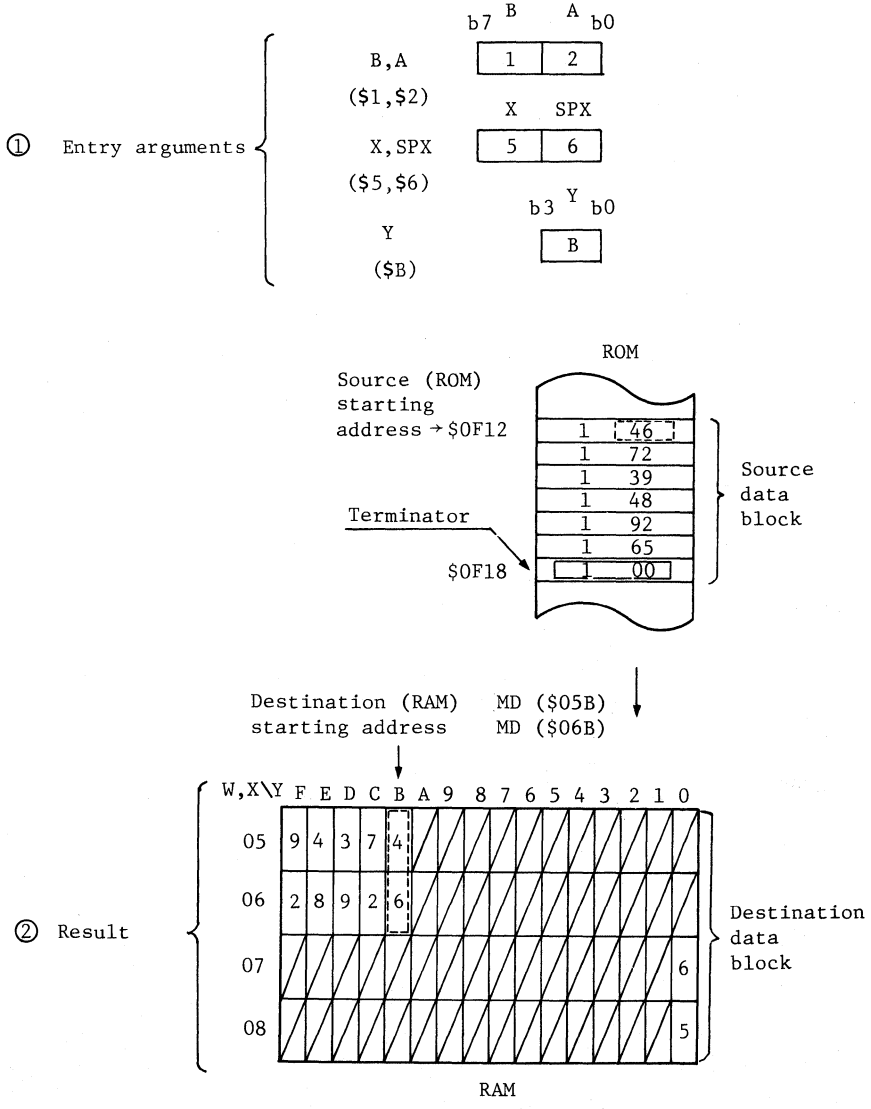


Fig. 1 Example of MOVES Execution

(2) User Notes

(a) ST flag is set after MOVES execution.

(b) When storing the destination starting address in SPX register, contents of X register + 1 must be stored, otherwise data cannot be moved successively to RAM. In Fig. 1, as contents of X register is $5, $6 is stored in SPX register.

(c) Bit 8 of all words in the source data table must be set to indicate loading of data into Accumulator and B register when executing the pattern generation instruction (P).

(3) RAM Allocation

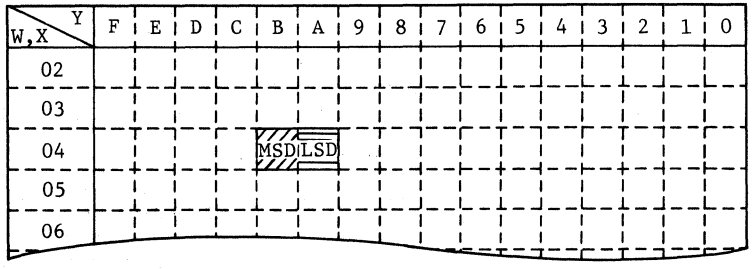


Fig. 2 RAM Allocation

Label	RAM	Description
HSOUR	b3 b0 MSD MD ($04B)	Work area for saving contents of B register used to indicate source (ROM) address.
LSOUR	b3 b0 LSD MD ($04A)	Work area for saving contents of Accumulator used to indicate source (ROM) address.

(4) Sample Application

Shown below is a sample application using MOVES with address space allocated as follows.

MD($0A2,$0A1) : Source starting address
MD($0A5 ∿ $0A3): Destination starting address

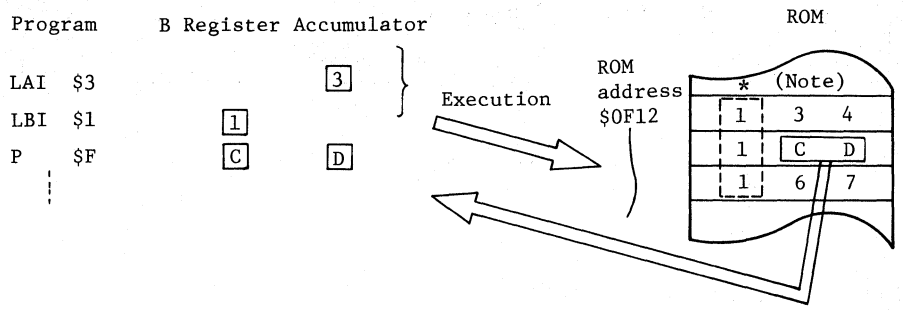
```
LWI     $0        ..... Example with W=0.
 :
LAMD    $0A5   ┐
LXA            │
XSPX           │
LAMD    $0A4   ├ ..... Load destination starting address
LXA            │       into entry argument.
LAMD    $0A3   │
LYA            ┘
LAMD    $0A2   ┐
LBA            ├ ..... Load source starting address into
LAMD    $0A1   ┘       entry argument.

CALL    MOVES  ..... Call MOVES.
 :
```

DESCRIPTION

(5) Basic Operation

(a) Data stored at the address indicated by Accumulator and B register is referred to Accumulator and B register, using the table look-up function of the pattern generation instruction (P).



After executing the pattern generation instruction in the above program sequence, $CD is contained in B register and Accumulator ($CD is stored in lower 8 bits of word located at $0F13). Since data table is allocated from $0F00 ∿ $0FFF, subroutine MOVES uses $F as operand of the pattern generation instruction (P).

*Note:

If dotted area (bit 8) is $1 as shown above, ROM data is referred to Accumulator and B register after executing the P instruction.

Fig. 3 ROM Table and the P Instruction

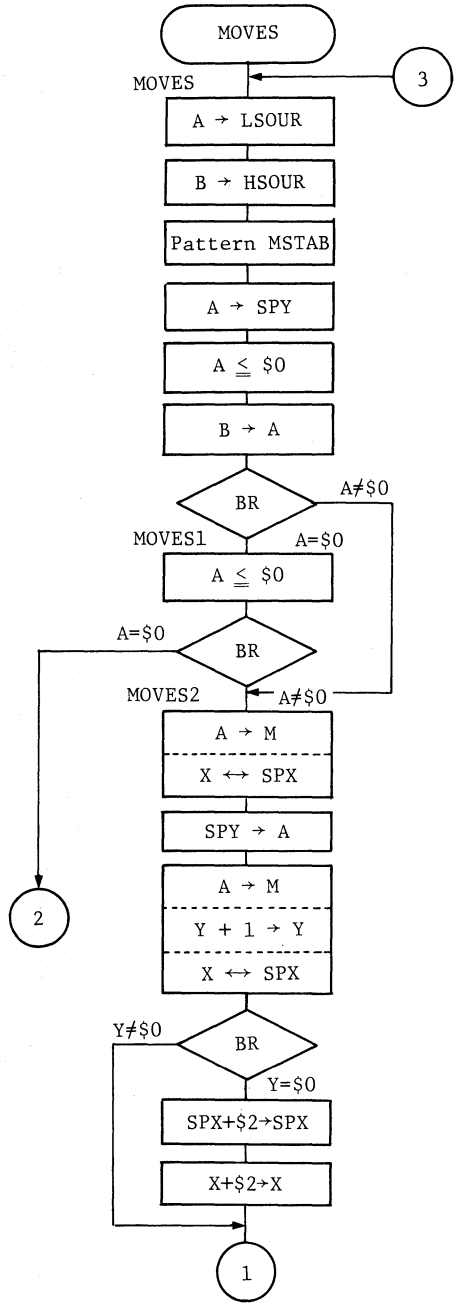
(b) Accumulator and B register are used as a pointer to the source data.

(c) X and Y registers are used as a pointer to the destination location in RAM.

(d) Data block, pointed to by Accumulator and B register, is moved into Accumulator and B register using the pattern generation instruction (P).
Data in Accumulator and B register is tested for terminator ($00).
If not, contents of Accumulator and B register is stored into destination and destination address is then incremented.

(e) Operation loops to (d) until terminator ($00) appears.

FLOWCHART



Save Accumulator and B register indicating source address.

Load source address data, indicated by MSTAB, B register and Accumulator, into B register and Accumulator.

Save Accumulator before test for terminator.

Test if source data is terminator (Accumulator =$0, B register=$0).

If "Yes", terminate subroutine MOVES.

Store upper digit of source data into destination.

Set destination X address for lower digit.

Restore Accumulator.

Store lower digit of source data into destination.

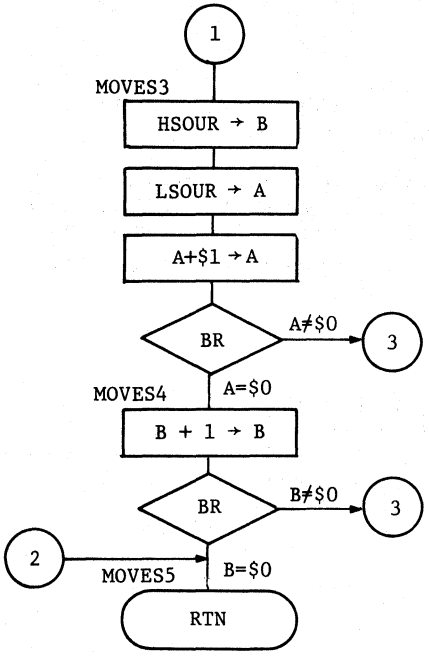
Increment destination Y address.

Increment destination X address for upper digit.

Test if destination Y address is $0.

Increment destination X address.

FLOWCHART



Restore Accumulator and B register indicating source address.

Increment Accumulator indicating source address.

Test if B register, indicating source address, needs incrementing.

Increment B register indicating source address.

Test if move is completed.

PROGRAM LISTING

```
ST-NO   OBJECT   ADRS   SOURCE STATEMENTS

00001   010      0000           LLEN    132
00002                  ********************************************************************
00003                  *                                                                  *
00004                  *      NAME : MOVE STRING (MOVES)                                  *
00005                  *                                                                  *
00006                  ********************************************************************
00007                  *                                                                  *
00008                  *      ENTRY          : A,B     (SOURCE STARTING ADDRESS)         *
00009                  *                       X,Y     (DESTINATION STARTING ADDRESS)    *
00010                  *      RETURNS        : NOTHING                                   *
00011                  *                                                                  *
00012                  ********************************************************************
00013                  *
00014                  HSOUR   EQU     $04B          WORK AREA FOR REGISTER(B)
00015                  LSOUR   EQU     $04A          WORK AREA FOR ACCUMULATOR(A)
00016                  MSTAB   EQU     $F            MSD (8-11BIT) OF DATA TABLE ADDRESS
00017                  *
00018                          ORG     $0100
00019                  MOVES   EQU     *             ENTRY POINT
00020   194 04A  0100          LMAD    LSOUR         SAVE SOURCE ADDR(A)
00021   048      0102          LAB
00022   194 04B  0103          LMAD    HSOUR         SAVE SOURCE ADDR(B)
00023   190 04A  0105          LAMD    LSOUR
00024   1BF      0107          P       MSTAB         LOAD SOURCE DATA
00025   002      0108          XSPY
00026   008      0109          LYA                   LOAD SPY FROM A
00027   002      010A          XSPY
00028   280      010B          ALEI    $0            A =< $0 ?
00029   048      010C          LAB
00030   30F      010D          BRS     MOVES1        BRANCH IF A = $0
00031   311      010E          BRS     MOVES2        BRANCH IF A =/ $0
00032   280      010F  MOVES1  ALEI    $0            A =< $0 ?
00033   327      0110          BRS     MOVES5        BRANCH IF A = $0
00034   095      0111  MOVES2  LMAX                  LOAD UPPER SOURCE DATA
00035   058      0112          LASPY                 LOAD A FROM SPY
00036   051      0113          LMAIYX                LOAD LOWER SOURCE DATA AND INCREMENT ADDR(Y)
00037   310      0114          BRS     MOVES3        BRANCH IF ADDR(Y) = $0
00038   068      0115          LASPX                 INCREMENT ADDR(SPX)
00039   282      0116          AI      %2
00040   001      0117          XSPX
00041   0E8      0118          LXA
00042   068      0119          LASPX                 INCREMENT ADDR(X)
00043   282      011A          AI      $2
00044   001      011B          XSPX
00045   0E8      011C          LXA
00046   190 04B  011D  MOVES3  LAMD    HSOUR         LOAD SOURCE ADDR(A)
00047   0C8      011F          LBA
00048   190 04A  0120          LAMD    LSOUR         LOAD SOURCE ADDR(B)
00049   281      0122          AI      $1            INCREMENT SOURCE ADDR(A)
00050   325      0123          BRS     MOVES4        BRANCH IF A = $0
00051   300      0124          BRS     MOVES         BRANCH IF A =/ $0
00052   04C      0125  MOVES4  IB                    INCREMENT SOURCE ADDR(B)
00053   300      0126          BRS     MOVES         LOOP UNTIL B = $0
00054   010      0127  MOVES5  RTN
```

When storing arguments in other RAM locations, change the EQU operands for the following labels.

HSOUR: Defines address where B register, indicating source data address, is saved.

LSOUR: Defines address where Accumulator, indicating source data address, is saved.

MSTAB: Defines MSD of the data table address referred to by the pattern generation instruction (P).

4. BRANCH FROM TABLE

MCU | HMCS400 SERIES | LABEL | **CCASE**

FUNCTION

Loads service routine starting address, which corresponds to a 1-byte command in RAM, into Accumulator and B register; permits easy decoding and processing of keyboard data or other input data ('case' branching).

ARGUMENTS

1 digit= 4 bits

Contents		Storage Location	No. of Digits
Entry	Command	HCMMD LCMMD (RAM)	2
	Data table starting address	Y,SPY	2
Returns	Service routine starting address	B,A	2
	Command status	ST	(1 bit)

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected
× : Undefined
‡ : Result

A	B
‡	‡
X	Y
●	×
SPX	SPY
●	×
W	
●	

CA	ST
●	‡

SPECIFICATIONS

1 word=10 bits

ROM (Words)
35
RAM (Digits)
3
Stack (Digits)
0
No. of cycles
75
Reentrant
No
Relocatable
No
Interrupt OK?
Yes

DESCRIPTION

(1) Function Details

(a) Argument details

HCMMD,LCMMD(RAM): Holds commands such as ASCII.

Y,SPY : Holds lower 8 bits of data table starting address.

B,A : Holds lower 8 bits of service routine starting address.

ST : Indicates status after CCASE execution.

ST=0 : Indicates data table has the same data (command) as that in HCMMD and LCMMD(RAM).

ST=1 : Indicates data table does not have the same data (command) as that in HCMMD and LCMMD(RAM).

SPECIFICATIONS NOTES

"No. of cycles" in "SPECIFICATIONS" indicates the number of cycles required to process a command located in the 3rd data string in the data table.

DESCRIPTION

(b) Example of CCASE execution is shown in Fig. 1. If entry arguments are as shown in part ① of Fig. 1, data table in Fig. 2 is searched and service routine starting address is contained as shown in part ② of Fig. 1.

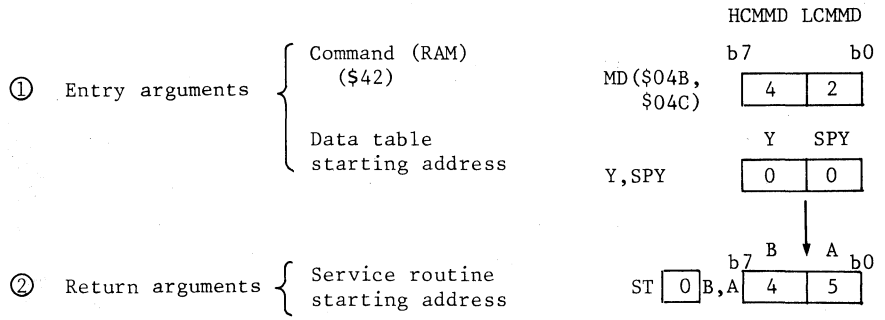


Fig. 1 Example of CCASE Execution

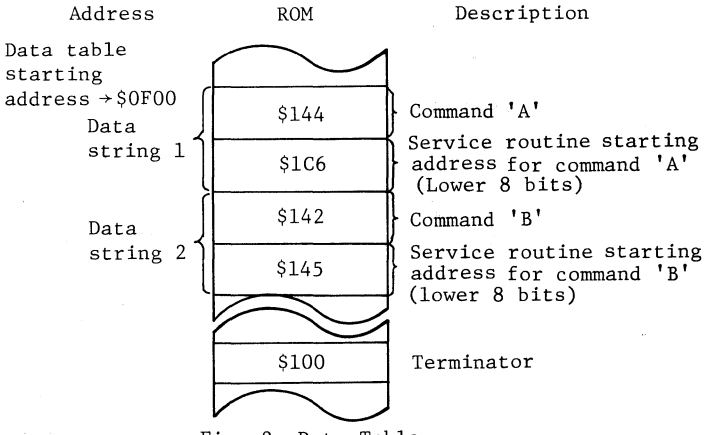


Fig. 2 Data Table

(c) When executing CCASE, data table as shown in Fig. 2 is necessary. Explanation of "Fig. 2 Data Table" is described as follows.

(i) Data table consists of a sequence of 2-word data strings starting at $0F00 and a terminator ($100).

(ii) In the 2-word data string, the first word is the command, and the second word is the lower 8 bits of the service routine starting address.

(2) User Notes

As "$100" is used as a terminator, $0 should not be held in HCMMD,LCMMD(RAM) or in command words in the data table.

(3) RAM Allocation

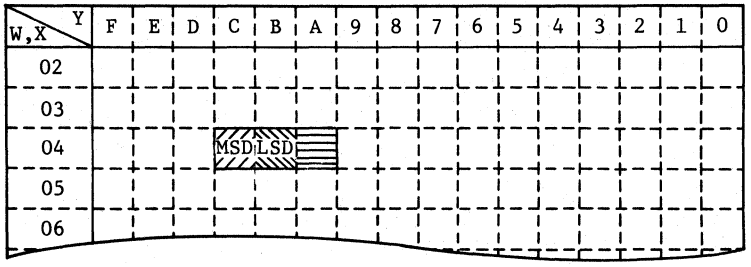


Fig. 3 RAM Allocation

Label	RAM	Description
HCMMD	b3 b0 MSD MD ($04C)	Upper digit of command.
LCMMD	b3 b0 LSD MD ($04B)	Lower digit of command.
CWORK	b3 b0 MD ($04A)	Work area for saving Accumulator.

DESCRIPTION

(4) Sample Application

Shown below is a sample application using CCASE with address space allocated as follows.

```
MD($0AC,$0AB) :  Command to be searched for

MD($0AE,$0AD) :  Data table starting address
```

```
        LWI     $0          ..... Example with W=0.
         :
        LAMD    $0AC        ⎫
        LMAD    HCMMD       ⎬ ..... Store command in entry argument.
        LAMD    $0AB        ⎪
        LMAD    LCMMD       ⎭
        LAMD    $0AE        ⎫
        LBA                 ⎬ ..... Load data table starting address into
        LAMD    $0AD        ⎭       entry argument.
        CALL    CCASE       ..... Call CCASE.
        BRS     ERROR       ..... Branch to ERROR if data table has no data
                                  corresponding to command.
*(Note)
Branch program
to service routine
         :
ERROR   Error program
         :
        ORG     $0F00       ⎫
        DC      $141        ⎪
        DC      $1C6        ⎪
        DC      $142        ⎬ ..... Data table.
        DC      $145        ⎪
         :                  ⎪
        DC      $100        ⎭
         :
```

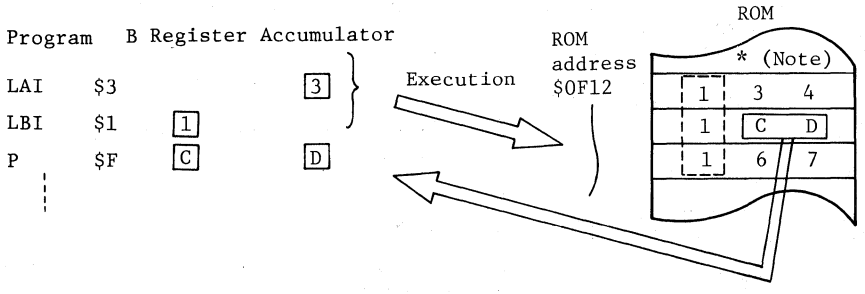
Note: CCASE execution only stores service routine starting address in Accumulator and B register. For actually branching to service routine, use the program sequence described in next page.

DESCRIPTION

```
        CALL    CCASE   ..... Call CCASE.
        BRS     ERROR   ..... If ST flag is set, branch to error program.
        TBR     $E      ..... Branch to service routine indicated by
                               Accumulator (PC3 ∿ PC0), B register
ERROR   Error Program          (PC7 ∿ PC4) and $E(PC11 ∿ PC8).
```

(5) Basic Operation

(a) Data stored at the address indicated by Accumulator and B register is referred to Accumulator and B register, using the table look-up function of the pattern generation instruction (P).



After executing the pattern generation instruction in the above program sequence, $CD is contained in B register and Accumulator ($CD is stored in lower 8 bits of word located at $0F13).

Note:

If blocked area (bit 8) is $1 as shown above, ROM data is referred to Accumulator and B register after executing the P instruction.

Fig. 4 ROM Table and the P Instruction

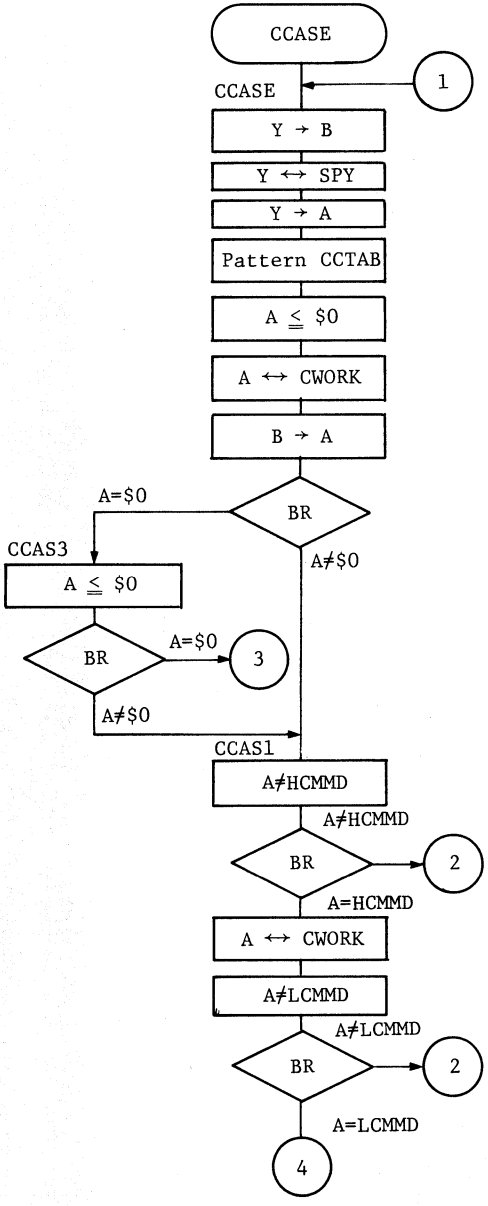
(b) Accumulator and B register are used as a pointer to the data table.

(c) Command in data table is read from the starting address, and compared with the input command using the pattern generation instruction (P).

(d) If the data table command and input command are the same, service routine starting address, which is stored in the address following the command, is loaded into Accumulator and B register using the pattern generation instruction (P). ST flag is then cleared and CCASE terminated.

(e) When data in data table is the terminator, ST flag is set and CCASE terminated.

FLOWCHART



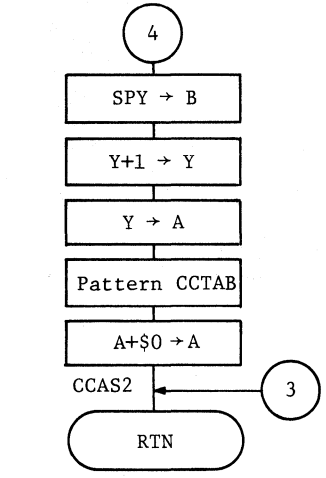
Load data table address into Accumulator and B register.

Load data table contents, specified by CCTAB, B register and Accumulator, into B register and Accumulator.

Test for data table command or terminator (Accumulator =$0, B register=$0).
If terminator, set ST flag and terminate CCASE.

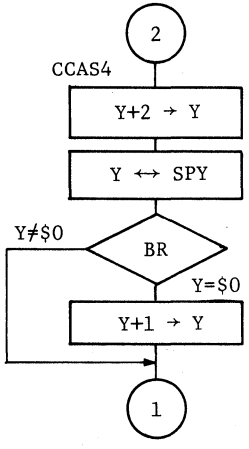
Compare input command with data table command.

FLOWCHART



If input command and data table command are the same, load service routine starting address into Accumulator and B register.

Clear ST flag.



If input command is different from data table command, add "2" to LSD of the data table address to compare with the next data table command.

Test if LSD of data table address generates a carry.

Increment the upper digit of data table address.

PROGRAM LISTING

```
ST-NO   OBJECT   ADRS   SOURCE STATEMENTS

00001   010      0000           LLEN    132
00002                  ***************************************************************
00003                  *                                                             *
00004                  *      NAME : BRANCH FROM TABLE (CCASE)                       *
00005                  *                                                             *
00006                  ***************************************************************
00007                  *                                                             *
00008                  *      ENTRY     : HCMMD,LCMMD   (COMMAND)                    *
00009                  *                  Y,SPY         (DATA TABLE STARTING ADDR)   *
00010                  *      RETURNS   : A,B           (SERVICE ROUTINE STARTING ADDR)*
00011                  *                  ST FLAG       (ST=0:FOUND,ST=1:FALSE)      *
00012                  *                                                             *
00013                  ***************************************************************
00014                  *
00015                  HCMMD   EQU     $04C          UPPER COMMAND ADDR
00016                  LCMMD   EQU     $04B          LOWER COMMAND ADDR
00017                  CWORK   EQU     $04A          WORK AREA ADDR
00018                  CCTAB   EQU     $F            MSD (8-11BIT) OF DATA TABLE ADDRESS
00019                  *
00020                          ORG     $0100
00021                  CCASE   EQU     *             ENTRY POINT
00022   0AF      0100          LAY                   LOAD Y INTO REGISTER(B)
00023   0C8      0101          LBA
00024   002      0102          XSPY                  LOAD Y INTO A
00025   0AF      0103          LAY
00026   1BF      0104          P       CCTAB         LOAD DATA TABLE ADDR INTO REG(B)&A
00027   280      0105          ALEI    $0            A =< $0 ?
00028   180 04A  0106          XMAD    CWORK
00029   048      0108          LAB
00030   319      0109          BRS     CCAS3         BRANCH IF A = $0
00031   104 04C  010A  CCAS1    ANEMD   HCMMD         INPUT COMMAND(B) = DATA TABLE COMMAND?
00032   31C      010C          BRS     CCAS4         BRANCH IF NOT EQUAL
00033   180 04A  010D          XMAD    CWORK
00034   104 04B  010F          ANEMD   LCMMD         INPUT COMMAND(A) = DATA TABLE COMMAND?
00035   31C      0111          BRS     CCAS4         BRANCH IF NOT EQUAL
00036   058      0112          LASPY
00037   0C8      0113          LBA
00038   05C      0114          IY
00039   0AF      0115          LAY
00040   1BF      0116          P       CCTAB         LOAD SERVICE ROUTINE ADDR INTO REG(B)&A
00041   280      0117          AI      $0            CLEAR STATUS FLAG
00042   010      0118  CCAS2    RTN
00043   280      0119  CCAS3    ALEI    $0            A =< $0 ?
00044   318      011A          BRS     CCAS2         BRANCH CCAS2 IF A = $0
00045   30A      011B          BRS     CCAS1         BRANCH CCAS1 IF A =/ $0
00046   05C      011C  CCAS4    IY                    ADD 2 TO REGISTER(Y)
00047   05C      011D          IY
00048   002      011E          XSPY
00049   300      011F          BRS     CCASE         BRANCH IF REGISTER(Y) =/ $0
00050   05C      0120          IY                    INCREMENT REGISTER(Y)
00051   300      0121          BRS     CCASE         BRANCH TO CCASE
00052   010      0122          RTN
```

When storing arguments in other RAM locations, change the EQU operands for the following labels.

HCMMD: Defines upper digit address of 8-bit command.

LCMMD: Defines lower digit address of 8-bit command.

CWORK: Defines address where Accumulator, used as work area, is saved.

CCTAB: Defines MSD of the data table referred to by the pattern generation instruction (P).

FUNCTION

Converts ASCII lowercase data using 7-bit ASCII arguments into uppercase data, loading result into Accumulator and B register.

ARGUMENTS

1 digit= 4 bits

Contents		Storage Location	No. of Digits
Entry	Lowercase (ASCII)	A,B	2
Returns	Uppercase (ASCII)	A,B	2

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected
× : Undefined
↕ : Result

A	B
↕	↕
X	Y
●	×
SPX	SPY
●	●
W	
●	

CA	ST
×	×

SPECIFICATIONS

1 word=10 bits

ROM (Words)	14
RAM (Digits)	0
Stack (Digits)	0
No. of cycles	13
Reentrant	No
Relocatable	No
Interrupt OK?	Yes

DESCRIPTION

(1) Function Details

(a) Argument details

. A,B : Holds ASCII lowercase.
After TPR execution, contains the corresponding uppercase data.

(b) Example of TPR execution is shown in Fig. 1. If entry argument lowercase 'a' ($61) is held in Accumulator and B register as shown in part ① of Fig. 1, lowercase data is converted into uppercase 'A' ($41), and the result is contained in Accumulator and B register.

SPECIFICATIONS NOTES

"No. of cycles" in "SPECIFICATIONS" indicates the number of cycles required to convert $7 and $A held in Accumulator and B register, respectively.

DESCRIPTION

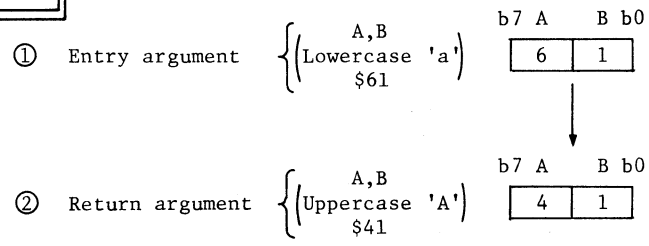


Fig. 1 Example of TPR Execution

(2) User Notes

(a) ST flag is set after TPR execution.

(b) Data other than lowercase ASCII held in Accumulator and B register is destroyed after TPR execution.

(3) RAM Allocation

RAM is not used during TPR execution.

(4) Sample Application

Shown below is a sample application using TPR with address space allocated as follows.

```
MD($0A2,$0A1):  ASCII lowercase
                After TPR execution, ASCII uppercase is contained.
```

```
LWI    $0        ..... Example with W=0.
 ¦
LAMD   $0A1  ┐
LBA          ├  ..... Load lowercase into entry argument.
LAMD   $0A2  ┘

CALL   TPR      ..... Call TPR.

LMAD   $0A2  ┐
LAB          ├  ..... Store uppercase, contained in return
LMAD   $0A1  ┘          argument, in RAM.
 ¦
 ¦
```

DESCRIPTION

(5) Basic Operation

(a) Input data in Accumulator and B register is tested for lowercase, using comparison instruction (YNEI, ALEI).

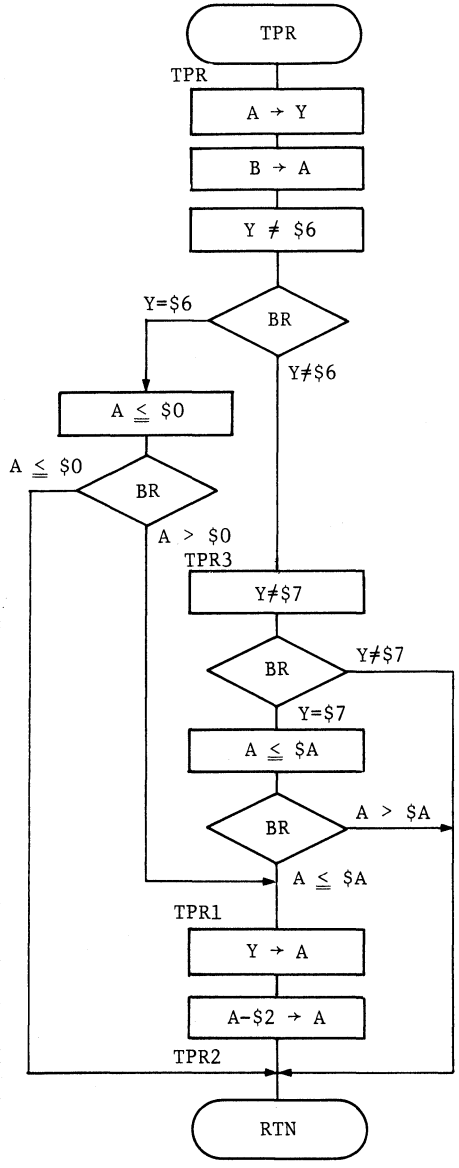
(b) After adding $E to input data using addition instruction (AI), bit 5 of lowercase (bit 1 of Accumulator) is cleared as shown in Fig. 2 and lowercase is converted into uppercase.

	Accumulator				B Register				
bit	7	6	5	4	3	2	1	0	
a($61)	0	1	1	0	0	0	0	1	
b($62)	0	1	1	0	0	0	1	0	
c($63)	0	1	1	0	0	0	1	1	Lowercase (bit 5="1")
⋮									
z($7A)	0	1	1	1	1	0	1	0	
A($41)	0	1	0	0	0	0	0	1	
B($42)	0	1	0	0	0	0	1	0	
C($43)	0	1	0	0	0	0	1	1	Uppercase (bit 5="0")
⋮									
Z($5A)	0	1	0	1	1	0	1	0	

Fig. 2 7-bit ASCII Lowercase and Uppercase

5. CONVERT ASCII LOWERCASE INTO UPPERCASE	MCU	HMCS400 SERIES	LABEL	TPR

FLOWCHART



Load upper digit of ASCII lowercase into Y register.

Load lower digit of ASCII lowercase into Accumulator.

Test if entry argument data is greater than lowercase 'a'. (Lowercase = 'a' ∿ 'z')

Test if entry argument data is less than lowercase 'z'. (Lowercase = 'a' ∿ 'z')

Clear bit 1 of upper digit in lowercase and convert lowercase into uppercase.

PROGRAM LISTING

```
ST-NO   OBJECT   ADRS   SOURCE STATEMENTS

00001   100      0000           LLEN     132
00002                  ************************************************************
00003                  *                                                          *
00004                  *       NAME :  CONVERT ASCII LOWERCASE INTO UPPERCASE (TPR)    *
00005                  *                                                          *
00006                  ************************************************************
00007                  *                                                          *
00008                  *       ENTRY    : A     (UPPER 4 BITS OF ASCII LOWERCASE)       *
00009                  *                  B     (LOWER 4 BITS OF ASCII LOWERCASE)       *
00010                  *     RETURNS    : A     (UPPER 4 BITS OF ASCII UPPERCASE)       *
00011                  *                  B     (LOWER 4 BITS OF ASCII UPPERCASE)       *
00012                  *                                                          *
00013                  ************************************************************
00014                  *
00015                          ORG      $0100
00016                  TPR     EQU      *              ENTRY POINT
00017   0D8      0100          LYA                     LOAD UPPER 4 BITS OF ASCII LOWERCASE
00018   048      0101          LAB                     LOAD LOWER 4 BITS OF ASCII LOWERCASE
00019   076      0102          YNEI     $6             UPPER 4 BITS OF ASCII LOWERCASE=$6 ?
00020   309      0103          BRS      TPR3           BRANCH IF UPPER 4 BITS OF ASCII LOWERCASE =/$6
00021   2B0      0104          ALEI     $0             LOWER 4 BITS OF ASCII LOWERCASE =<$0 ?
00022   308      0105          BRS      TPR2           BRANCH IF LOWER 4 BITS OF ASCII LOWERCASE =<$0
00023   0AF      0106   TPR1   LAY                     LOAD A FROM Y
00024   28E      0107          AI       $E             CONVERT LOWERCASE INTO UPPERCASE
00025   010      0108   TPR2   RTN
00026   077      0109   TPR3   YNEI     $7             UPPER ASCII =$7 ?
00027   308      010A          BRS      TPR2           BRANCH IF UPPER 4 BITS OF ASCII LOWERCASE=/$7
00028   2BA      010B          ALEI     $A             LOWER 4 BITS OF ASCII LOWERCASE=<$A ?
00029   306      010C          BRS      TPR1           BRANCH IF A =< $A
00030   308      010D          BRS      TPR2           BRANCH IF A > $A
```

6. CONVERT ASCII INTO 1-DIGIT HEXADECIMAL

MCU | HMCS400 SERIES | LABEL | **NIBBLE**

FUNCTION

Converts 7-bit ASCII, '0' to '9' or 'A' to 'F' in Y register and Accumulator, into a 1-digit hexadecimal number, loading result into Accumulator.

ARGUMENTS

1 digit= 4 bits

Contents		Storage Location	No. of Digits
Entry	ASCII	Y,A	2
Returns	Hexadecimal number	A	1
	Conversion or no conversion	ST	(1 bit)

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected
× : Undefined
‡ : Result

A	B
‡	●
X	Y
●	×
SPX	SPY
●	●
W	
●	

CA	ST
×	‡

SPECIFICATIONS

1 word=10 bits

ROM (Words)
14
RAM (Digits)
0
Stack (Digits)
0
No. of cycles
13
Reentrant
No
Relocatable
No
Interrupt OK?
Yes

DESCRIPTION

(1) Function Details

(a) Argument details

Y,A : Holds ASCII. After NIBBLE execution, contains a 1-digit hexadecimal number in Accumulator.

ST : Indicates status after NIBBLE execution.

ST=0 : Indicates ASCII range of '0' to '9' or 'A' to 'F'.

ST=1 : Indicates ASCII other than '0' to '9' or 'A' to 'F'.

(b) Example of NIBBLE execution is shown in Fig. 1. If entry argument is as shown in part ① of Fig. 1, data $F, converted from ASCII into a 1-digit hexadecimal number, is contained in Accumulator as shown in part ② of Fig. 1.

SPECIFICATIONS NOTES

"No. in cycles" in "SPECIFICATIONS" indicates the number of cycles required to convert $4 and $6 held in Y register and Accumulator, respectively.

DESCRIPTION

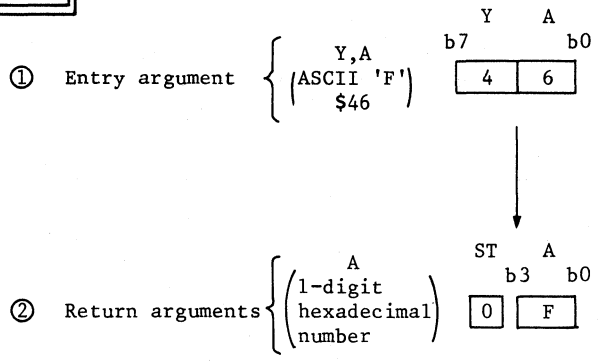


Fig. 1 Example of NIBBLE Execution

(2) User Notes

Data other than ASCII '0' to '9' or 'A' to 'F' held in Accumulator and Y register is destroyed after NIBBLE execution.

(3) RAM Allocation

RAM is not used during NIBBLE execution.

(4) Sample Application

Shown below is a sample application using NIBBLE with address space allocated as follows.

```
MD($0A1,$0A0) :  2-digit ASCII
MD($02B)      :  1-digit hexadecimal number
```

```
        LWI   $0        ..... Example with W=0.
          :
        LAMD  $0A1
        LYA             ..... Load ASCII into entry argument.
        LAMD  $0A0
        CALL  NIBBLE    ..... Call NIBBLE.
        BRS   SKIP      ..... If ASCII is other than '0' to '9' or 'A' to
                              'F', branch to service routine.
        LMAD  $02B      ..... Store a 1-digit hexadecimal number, contained
          :                   in return argument, in RAM.
SKIP  Service routine for
      ASCII other than
      '0' to '9' or 'A' to 'F'
          :
```

DESCRIPTION

(5) Basic Operation

(a) Data in Y register and Accumulator is tested if within range of '0' to '9' (▭ area in Table 1), using comparison instruction (ALEI).

(b) Next, data in Y register and Accumulator is tested if within range of 'A' to 'F' (▭ area in Table 1).

(c) If data is other than above, ST flag is set and operation terminated.

(d) After testing (a) and (b), ASCII is converted into a 1-digit hexadecimal number.

(i) If data is within range of '0' to '9', data in Accumulator is contained as return argument.

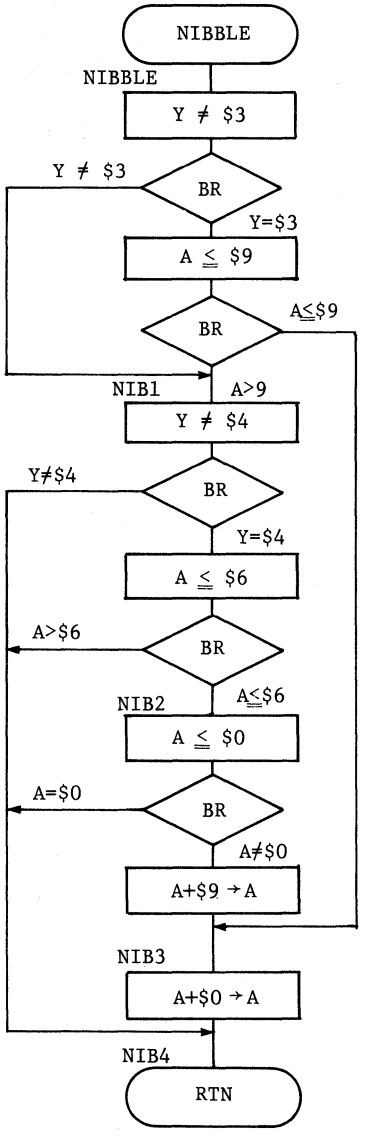
(ii) If data is within range of 'A' to 'F', $9 is added to data in Accumulator to convert ASCII into a 1-digit hexadecimal number.

Table 1 ASCII Table

MSD / LSD	0 000	1 001	2 010	3 011	4 100	5 101	6 110	7 111
0 0000	NUL	DLE	SP	0	@	P	`	p
1 0001	SOH	DC1	!	1	A	Q	a	q
2 0010	STX	DC2	”	2	B	R	b	r
3 0011	ETX	DC3	#	3	C	S	c	s
4 0100	EOT	DC4	$	4	D	T	d	t
5 0101	ENG	NAK	%	5	E	U	e	u
6 0110	ACK	SYN	&	6	F	V	f	v
7 0111	BEL	ETB	’	7	G	W	g	w
8 1000	BS	CAN	(	8	H	X	h	x
9 1001	HT	EM	)	9	I	Y	i	y
A 1010	LF	SUB	*	:	J	Z	j	z
B 1011	VT	ESC	+	;	K	[	k	{
C 1100	FF	FS	,	<	L	\	l	|
D 1101	CR	GS	-	=	M	]	m	}
E 1110	SO	RS	.	>	N	↑	n	∿
F 1111	SI	VS	/	?	O	←	o	DEL

6. CONVERT ASCII INTO 1-DIGIT HEXADECIMAL	MCU	HMCS400 SERIES	LABEL	**NIBBLE**

FLOWCHART



Test if ASCII in entry argument is within range of '0' to '9'.

Test if ASCII in entry argument is within range of 'A' to 'F'.

Convert ASCII into a 1-digit hexadecimal number.

Clear ST flag.

PROGRAM LISTING

```
ST-NO   OBJECT   ADRS   SOURCE STATEMENTS

00001   100      0000           LLEN     132
00002                  ************************************************************
00003                  *                                                          *
00004                  *      NAME :  CONVERT ASCII INTO 1-DIGIT HEXADECIMAL (NIBBLE)*
00005                  *                                                          *
00006                  ************************************************************
00007                  *                                                          *
00008                  *      ENTRY          :  Y  (UPPER ASCII)                  *
00009                  *                        A  (LOWER ASCII)                  *
00010                  *      RETURNS        :  A  (HEXADECIMAL DATA)             *
00011                  *                        ST FLAG (ST=1:CONVERTED,ST=0:FALSE)  *
00012                  *                                                          *
00013                  ************************************************************
00014                  *
00015                        ORG      $0100
00016                  NIBBLE EQU      *              ENTRY POINT
00017   073      0100          YNEI     $3             UPPER ASCII = $3 ?
00018   304      0101          BRS      NIB1           BRANCH IF UPPER ASCII =/ $3
00019   289      0102          ALEI     $9             LOWER ASCII =< $9 ?
00020   30C      0103          BRS      NIB3           BRANCH IF LOWER ASCII =< $9
00021   074      0104   NIB1   YNEI     $4             UPPER ASCII = $4 ?
00022   30D      0105          BRS      NIB4           BRANCH IF UPPER ASCII =/ $4
00023   286      0106          ALEI     $6             LOWER ASCII =< $6 ?
00024   309      0107          BRS      NIB2           BRANCH IF LOWER ASCII =< $6
00025   30D      0108          BRS      NIB4           BRANCH IF LOWER ASCII > $6
00026   280      0109   NIB2   ALEI     $0             LOWER ASCII  =< $0 ?
00027   30D      010A          BRS      NIB4           BRANCH IF ASCII = $0
00028   289      010B          AI       $9             CONVERT ASCII INTO HEXADECIMAL NUMBER
00029   280      010C   NIB3   AI       $0             CLEAR STATUS FLAG
00030   010      010D   NIB4   RTN
```

7. CONVERT 8-BIT BINARY DATA INTO ASCII

MCU | HMCS400 SERIES | LABEL | COBYTE

FUNCTION

Converts 8-bit binary data, held in SPY register and Accumulator, into 2-figure ASCII and stores result in RAM; uses 7-bit ASCII arguments.

ARGUMENTS

1 digit= 4 bits

Contents		Storage Location	No. of Digits
Entry	Upper digit of 8-bit binary number	SPY	1
	Lower digit of 8-bit binary number	A	1
Returns	Upper digit of ASCII	MD($03D, $03C)	2
	Lower digit of ASCII	MD($03B, $03A)	2

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected
× : Undefined
‡ : Result

A	B
×	×
X	Y
×	×
SPX	SPY
●	×
W	
●	

CA	ST
×	×

SPECIFICATIONS

1 word=10 bits

ROM (Words)
19
RAM (Digits)
4
Stack (Digits)
4
No. of cycles
27
Reentrant
No
Relocatable
No
Interrupt OK?
Yes

DESCRIPTION

(1) Function Details

(a) Argument details

SPY,A : Holds 8-bit binary number to be converted into ASCII.
(SPY: Upper digit, A: Lower digit)

MD($03D∿$03A) : Contains 2-figure ASCII.

(b) Example of COBYTE execution is shown in Fig. 1. If entry arguments are as shown in part ① of Fig. 1, data, converted from 8-bit binary number into ASCII, is contained in MD($03D∿$03A) as shown in part ② of Fig. 1.

SPECIFICATIONS NOTES

"No. of cycles" in "SPECIFICATIONS" indicates the number of cycles required to convert $99 into ASCII.

DESCRIPTION

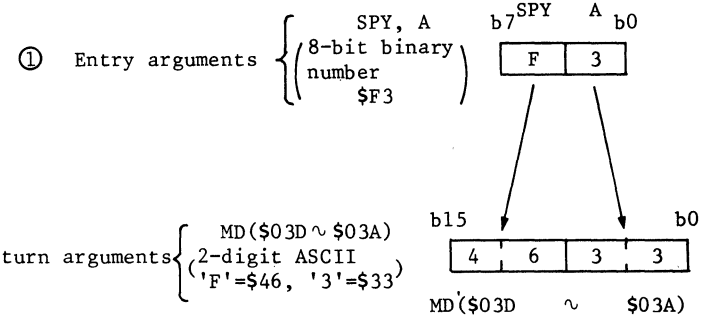


Fig. 1 Example of COBYTE Execution

(2) User Notes

(a) ST flag is set after COBYTE execution.

(b) After COBYTE execution, 8-bit binary number held in SPY register and Accumulator is destroyed. If 8-bit binary number needs to be retained after COBYTE execution, it should be saved in memory before execution.

(3) RAM Allocation

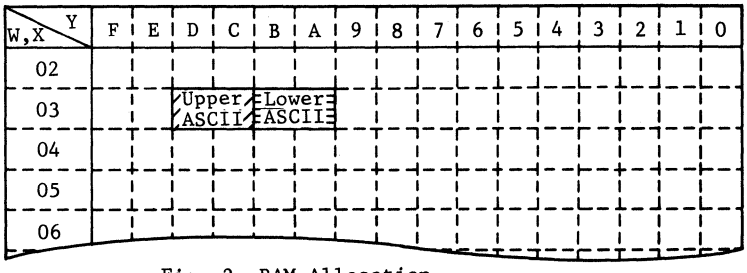


Fig. 2 RAM Allocation

Label	RAM	Description
—	b7 b0 MD($03D,$03C)	Upper ASCII, which is converted from upper 4-bit of 8-bit binary number (held in SPY register). X and Y addresses are defined by XCOB and YCOB, respectively.
—	b7 b0 MD($03B,$03A)	Lower ASCII, which is converted from lower 4-bit of 8-bit binary number (held in Accumulator). X and Y addresses are defined by XCOB and YCOB, respectively.

DESCRIPTION

(4) Sample Application

Shown below is a sample application using COBYTE with address space allocated as follows.

```
MD($0A4,$0A3)   :  8-bit binary data
MD($0AD ∿ $0AA) :  2-figure ASCII
```

```
        LWI    $0        ..... Example with W=0.
         |
         |
        LAMD   $0A4
        LYA                ..... Load 8-bit binary number into entry
        XSPY                     argument.
        LAMD   $0A3
        CALL   COBYTE     ..... Call COBYTE.
        LAMD   $03A
        LMAD   $0AA
        LAMD   $03B
        LMAD   $0AB              Store 2-figure ASCII, contained in return
        LAMD   $03C        ..... argument, in RAM.
        LMAD   $0AC
        LAMD   $03D
        LMAD   $0AD
         |
         |
         |
```

DESCRIPTION

(5) Basic Operation

(a) (i) If data in SPY register and Accumulator is within range of $0 to $9 (area in Table 1), $30 is added to the data.

(ii) If data in SPY register and Accumulator is within range of $A to $F (area in Table 1), $37 is added to the data.

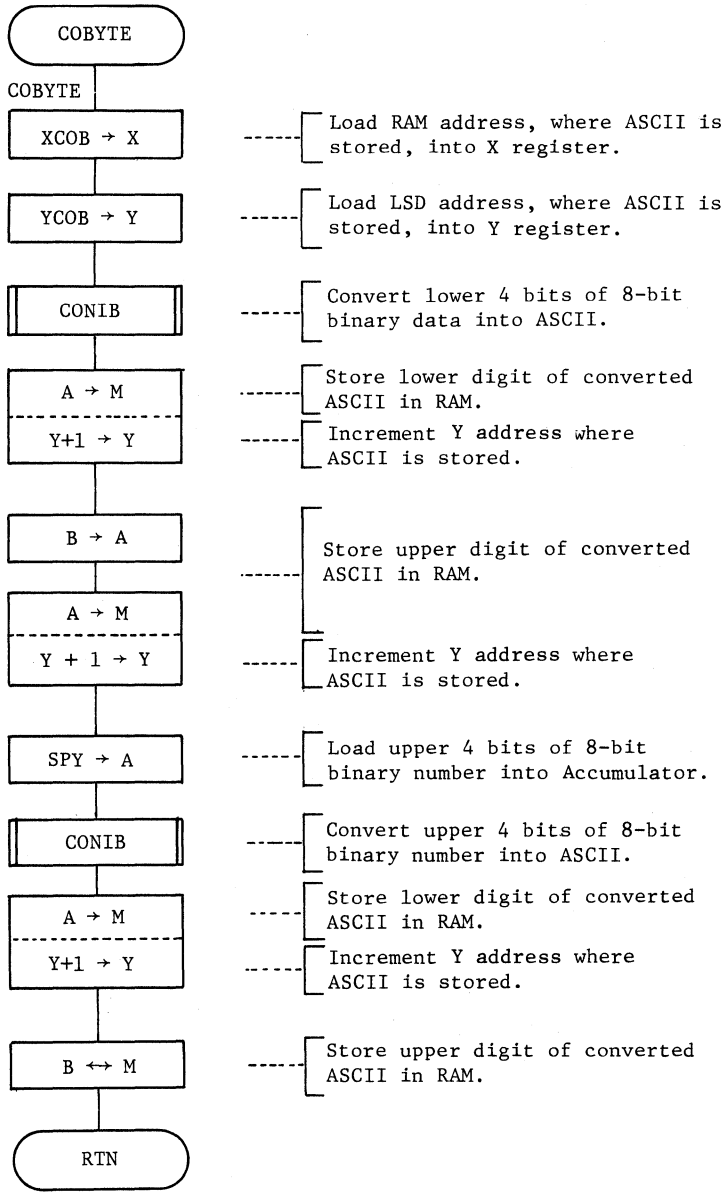
(b) Upper ASCII, converted from data in SPY register, is stored MD($03D,$03C). Lower ASCII, converted from data in Accumulator, is stored in MD($03B,$03A).

Table 1 ASCII Table

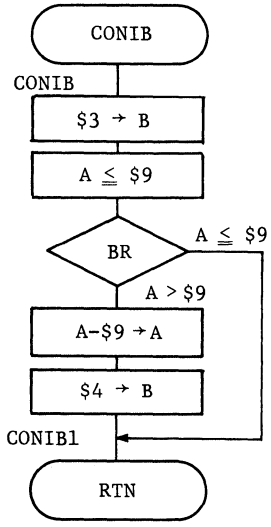
LSD \ MSD	0 000	1 001	2 010	3 011	4 100	5 101	6 110	7 111
0 0000	NUL	DLE	SP	0	@	P	`	p
1 0001	SOH	DC1	!	1	A	Q	a	q
2 0010	STX	DC2	"	2	B	R	b	r
3 0011	ETX	DC3	#	3	C	S	c	s
4 0100	EOT	DC4	$	4	D	T	d	t
5 0101	ENG	NAK	%	5	E	U	e	u
6 0110	ACK	SYN	&	6	F	V	f	v
7 0111	BEL	ETB	'	7	G	W	g	w
8 1000	BS	CAN	(	8	H	X	h	x
9 1001	HT	EM	)	9	I	Y	i	y
A 1010	LF	SUB	*	:	J	Z	j	z
B 1011	VT	ESC	+	;	K	[	k	{
C 1100	FF	FS	,	<	L	\	l	|
D 1101	CR	GS	-	=	M	]	m	}
E 1110	SO	RS	.	>	N	↑	n	~
F 1111	SI	VS	/	?	O	←	o	DEL

7. CONVERT 8-BIT BINARY DATA INTO ASCII	MCU	HMCS400 SERIES	LABEL	COBYTE

FLOWCHART



FLOWCHART



Load $3 as the upper digit of ASCII '0' to '9'.

Test if data is less than $9 or greater than $A. When data is less than $9, contents of B register and Accumulator can be used as ASCII.

When data is greater than $A, subtract $9 from the data and load $4 into B register to convert data into ASCII.

7. CONVERT 8-BIT BINARY DATA INTO ASCII	MCU	HMCS400 SERIES	LABEL	**COBYTE**

PROGRAM LISTING

```
ST-NO   OBJECT   ADRS   SOURCE STATEMENTS

00001   010      0000           LLEN     132
00002                  ************************************************************
00003                  *                                                          *
00004                  *       NAME : CONVERT 8-BIT BINARY DATA INTO ASCII (COBYTE) *
00005                  *                                                          *
00006                  ************************************************************
00007                  *                                                          *
00008                  *       ENTRY        : SPY (UPPER 4 BITS OF 8-BIT BINARY DATA)*
00009                  *                       A   (LOWER 4 BITS OF 8-BIT BINARY DATA)*
00010                  *       RETURNS      : MD($03D.$03C) (UPPER ASCII)          *
00011                  *                      MD($03B.$03A) (LOWER ASCII)          *
00012                  *                                                          *
00013                  ************************************************************
00014                  *
00015                  XCOB     EQU      $3            ASCII ADDR(X)
00016                  YCOB     EQU      $A            LSD ADDR(Y) OF ASCII
00017                  *
00018                           ORG      $0100
00019                  COBYTE   EQU      *             ENTRY POINT
00020   223      0100           LXI      XCOB          LOAD ADDR(X)
00021   21A      0101           LYI      YCOB          LOAD ADDR(Y)
00022   160 10D  0102           CALL     CONIB         CONVERT LOWER 4 BITS BINARY INTO ASCII
00023   050      0104           LMAIY                  STORE UPPER ASCII AND INCREMENT ADDR(Y)
00024   048      0105           LAB
00025   050      0106           LMAIY                  STORE LOWER ASCII
00026   058      0107           LASPY
00027   160 10D  0108           CALL     CONIB         CONVERT UPPER 4 BITS BINARY INTO ASCII
00028   050      010A           LMAIY                  STORE UPPER ASCII AND INCREMENT ADDR(Y)
00029   0C0      010B           XMB                    STORE LOWER ASCII
00030   010      010C           RTN
00031                  *
00032   203      010D   CONIB    LBI      $3            LOAD UPPER ASCII
00033   289      010E           ALEI     $9            ASCII = '0'-'9' OR 'A'-'F' ?
00034   312      010F           BRS      CONIB1        BRANCH IF ASCII = '0'-'9'
00035   287      0110           AI       $7            CONVERT BINARY DATA INTO ASCII ('A'-'F')
00036   204      0111           LBI      $4            LOAD UPPER ASCII
00037   010      0112   CONIB1  RTN
```

When storing arguments in other RAM locations, change the EQU operands for the following labels.

XCOB: Defines X address of ASCII.

YCOB: Defines LSD Y address of ASCII.

8. COUNT LOGICAL "1" BITS

MCU | HMCS400 SERIES | LABEL | HCNT

FUNCTION

Counts number of logical "1" bits in 8-bit data string in HHBIT and LHBIT(RAM), and loads result into B register; permits easy parity checking.

ARGUMENTS

1 digit= 4 bits

Contents		Storage Location	No. of Digits
Entry	8-bit number	HHBIT, LHBIT (RAM)	2
Returns	No. of logical "1" bits	B	1

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected
× : Undefined
‡ : Result

A	B
×	‡

X	Y
●	×

SPX	SPY
●	●

W
●

CA	ST
×	×

SPECIFICATIONS

1 word=10 bits

ROM (Words)
19
RAM (Digits)
2
Stack (Digits)
4
No. of cycles
68
Reentrant
No
Relocatable
No
Interrupt OK?
Yes

DESCRIPTION

(1) Function Details

(a) Argument details

HHBIT,LHBIT(RAM): Holds 8-bit number in which number of logical "1" bits will be counted.

B : Contains number of logical "1" bits in 8-bit number string.

(b) Example of HCNT execution is shown in Fig. 1. If entry argument is as shown in part ① of Fig. 1, number of logical "1" bits is contained in B as shown in part ② of Fig. 1.

(c) Contents of HHBIT and LHBIT(RAM) are saved after HCNT execution.

SPECIFICATIONS NOTES

"No. of cycles" in "SPECIFICATIONS" indicates the number of cycles required to count the number of logical "1" bits in $FF.

DESCRIPTION

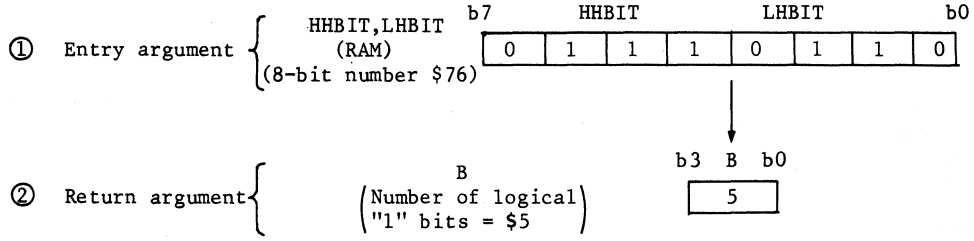


Fig. 1 Example of HCNT Execution

(2) User Notes

ST flag is set after HCNT execution.

(3) RAM Allocation

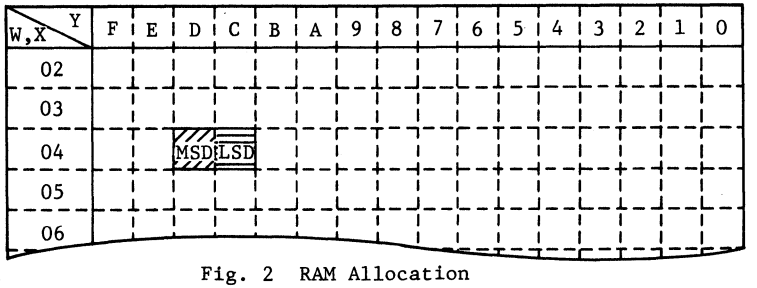


Fig. 2 RAM Allocation

Label	RAM	Description
HHBIT	b3 b0 MSD MD($04D)	Upper digit of 8-bit number for counting number of logical "1" bits.
LHBIT	b3 b0 LSD MD($04C)	Lower digit of 8-bit number for counting number of logical "1" bits.

DESCRIPTION

(4) Sample Application

Shown below is a sample application using HCNT with address space allocated as follows.

```
MD($0AD,$0AC):  8-bit number
MD($0BD)     :  No. of logical "1" bits in 8-bit number string
```

```
        LWI     $0          ..... Example with W=0.
         ¦
         ¦
        LAMD    $0AD       ⎫
        LMAD    HHBIT      ⎬..... Store 8-bit number in entry argument.
        LAMD    $0AC       ⎪
        LMAD    LHBIT      ⎭
        CALL    HCNT        ..... Call HCNT.
        LAB                ⎫
        LMAD    $0BD       ⎭..... Store number of logical "1" bits, contained
                                  in return argument, in RAM.
         ¦
         ¦
```

DESCRIPTION

(5) Basic Operation

(a) In HMCS400 series, when counting number of logical "1" bits is performed with 2 or more digits, the same operation is repeated for each digit. In this program, upper digit of 8-bit data is counted first, and then lower digit is counted.

(b) Y register is used as a counter to count 4 bits.

(c) Data in Accumulator is loaded into CA flag bit by bit using rotation instruction (ROL).

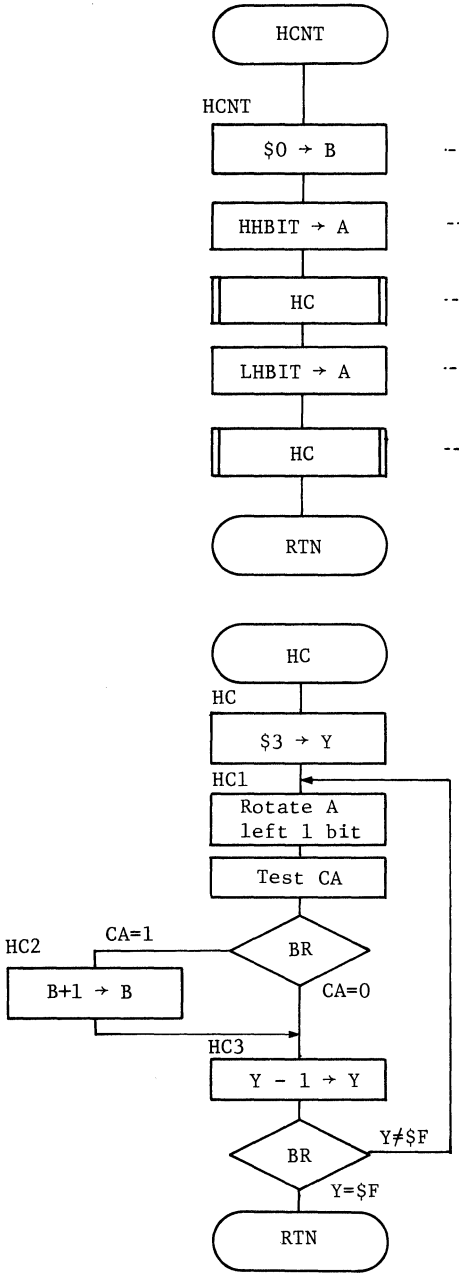
(d) CA flag is tested. When CA flag is "1", B register is incremented. When CA flag is "0", no operation is performed.

(e) Y register is decremented every time (d) is executed.

(f) Operation loops from (c) to (e) until Y register is $F, to obtain number of logical "1" bits in upper digit.

(g) Number of logical "1" bits in lower digit is obtained by repeating steps (b) to (f).

FLOWCHART



Clear counter (HI bit counter) to count the number of logical "1" bits.

Load upper digit of 8-bit data into Accumulator.

Call subroutine HC to count 1-digit logical "1" bits.

Load lower digit of 8-bit data into Accumulator.

Call subroutine HC to count 1-digit logical "1" bits.

Store rotation counter.

Load MSB of 4-bit data into CA flag.

Test whether one of 4-bit data contained in CA flag is "1" or "0".

If data in CA flag is "1", increment HI bit counter.

Decrement rotation counter and test if all bits are counted.

PROGRAM LISTING

```
ST-NO   OBJECT   ADRS   SOURCE STATEMENTS

00001   010      0000           LLEN     132
00002                  ************************************************************
00003                  *                                                          *
00004                  *       NAME : COUNT LOGICAL "1" BITS (HCNT)               *
00005                  *                                                          *
00006                  ************************************************************
00007                  *                                                          *
00008                  *       ENTRY          :HHBIT  (UPPER 4 BITS OF DATA)      *
00009                  *                       LHBIT  (LOWER 4 BITS OF DATA)      *
00010                  *       RETURNS        :B      (NUMBER OF LOGICAL "1" BITS)*
00011                  *                                                          *
00012                  ************************************************************
00013                  *
00014                  HHBIT    EQU      $04D          UPPER 4-BIT DATA ADDR
00015                  LHBIT    EQU      $04C          LOWER 4-BIT DATA ADDR
00016                  *
00017                           ORG      $0100
00018                  HCNT     EQU      *             ENTRY POINT
00019   200      0100           LBI      $0            CLEAR HIGH BIT COUNTER(B)
00020   190 04D  0101           LAMD     HHBIT         LOAD UPPER 4-BIT DATA
00021   160 10A  0103           CALL     HC            CALL 1 DIGIT COUNTER SUBROUTINE
00022   190 04C  0105           LAMD     LHBIT         LOAD LOWER 4-BIT DATA
00023   160 10A  0107           CALL     HC            CALL 1 DIGIT COUNTER SUBROUTINE
00024   010      0109           RTN
00025   213      010A   HC       LYI      $3            LOAD ROTATION COUNTER(Y)
00026   0A1      010B   HC1      ROTL                   ROTATE 8-BIT DATA LEFT 1 BIT
00027   06F      010C           TC                     TEST CARRY
00028   30F      010D           BRS      HC2           BRANCH IF CA = 1
00029   310      010E           BRS      HC3           BRANCH IF CA = 0
00030   04C      010F   HC2      IB                     INCREMENT HIGHT BIT COUNTER(B)
00031   0DF      0110   HC3      DY                     DECREMENT ROTATION COUNTER(Y)
00032   30B      0111           BRS      HC1           LOOP UNTIL ROTATION COUNTER(Y) = $F
00033   010      0112           RTN
```

When storing arguments in other RAM locations, change the EQU operands for the following labels.

HHBIT: Defines upper digit address of 8-bit number.

LHBIT: Defines lower digit address of 8-bit number.

9. SHIFT 8-BIT DATA

MCU | HMCS400 SERIES | LABEL | **SHR**

FUNCTION

Shifts 8-bit binary data stored in RAM a specified number of times to the right.

ARGUMENTS

1 digit= 4 bits

Contents		Storage Location	No. of Digits
Entry	Unsigned 8-bit binary number to be shifted to the right	MD($033, $032)	2
	No. of shifts	B	1
Returns	Shift result	MD($033, $032)	2

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected
× : Undefined
‡ : Result

A	B
×	×
X	Y
×	×
SPX	SPY
●	●
W	
●	

CA	ST
×	×

SPECIFICATIONS

1 word=10 bits

ROM (Words)
11
RAM (Digits)
2
Stack (Digits)
0
No. of cycles
46
Reentrant
No
Relocatable
No
Interrupt OK?
Yes

DESCRIPTION

(1) Function Details

(a) Argument details

MD($033,$032): Holds 8-bit binary number to be shifted to the right. After SHR execution, contains shift result.

B : Holds number of shifts.
Contents of B = Number of actual shifts - 1
(See (2) User Notes)

(b) Example of SHR execution is shown in Fig. 1. If entry arguments are as shown in part ① of Fig. 1, 8-bit binary number is shifted to the right as shown in part ② of Fig. 1.
In this case, "0"s are shifted into the 2 leftmost bits.

SPECIFICATIONS NOTES

"No. of cycles" in "SPECIFICATIONS" indicates the number of cycles required to shift 8 bits of binary data 3 bits to the right.

DESCRIPTION

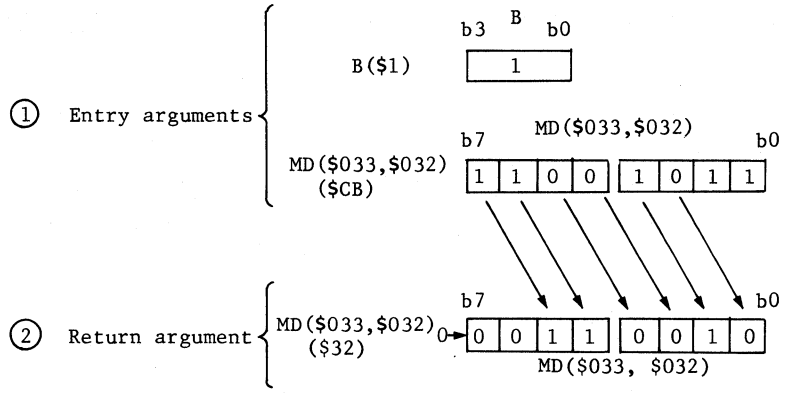


Fig. 1 Example of SHR Execution

(2) User Notes

(a) ST flag is set after SHR execution.

(b) When specifying number of shifts, load B register with number of actual shifts less 1. In part ① of Fig. 1, $1 is held in B register since number of actual shifts is 2.

(c) Number of shifts in B register must be within the range of $\$0 \leqq B \leqq \$7$, otherwise MD($033,$032) becomes "0".

(d) Shift operation permits easy multiplication of 8-bit binary number by 2^{-n}. (n = number of shifts)

DESCRIPTION

(3) RAM Allocation

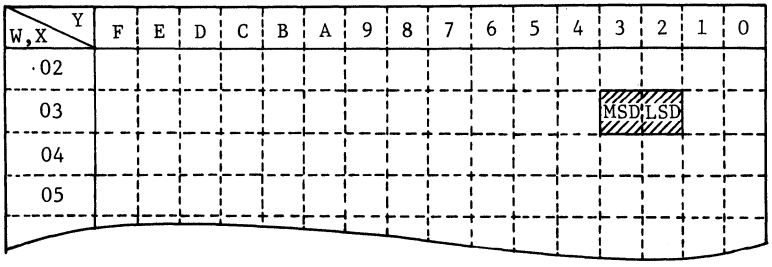


Fig. 2 RAM Allocation

Label	RAM	Description
—	b7 b0 MSD LSD MD($033, $032)	Holds 8-bit binary number to be shifted to the right before execution. Contains shift result after execution. X and Y addresses are defined by XSFT and YSFT, respectively.

DESCRIPTION

(4) Sample Application

Shown below is a sample application using SHR with address space allocated as follows.

MD($0A3,$0A2)	:	8-bit binary number to be shifted
MD($0A4)	:	No. of shifts
MD($0A6,$0A5)	:	8-bit binary shift result

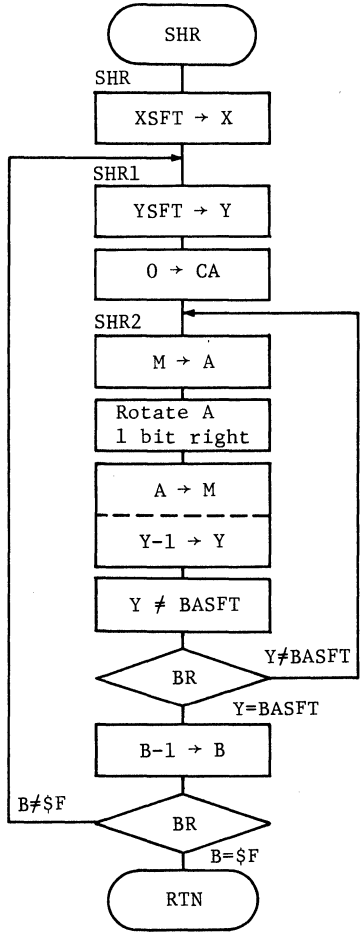
```
        LWI     $0          ..... Example with W=0.
         :
        LAMD    $0A3        ┐
        LMAD    $033        ├..... Store 8-bit binary number to be shifted
        LAMD    $0A2        │      to the right in entry argument.
        LMAD    $032        ┘
        LAMD    $0A4        ┐..... Load number of shifts into entry
        LBA                 ┘      argument.
        CALL    SHR         ..... Call SHR.
        LAMD    $033        ┐
        LMAD    $0A6        ├..... Store shift result, which is contained
        LAMD    $032        │      in return argument, in RAM.
        LMAD    $0A5        ┘
         :
```

(5) Basic Operation

(a) 8-bit binary number is shifted to the right 1 digit at a time.

(b) B register is used to count the number of shifts, and is decremented each time (a) is executed.
Operation loops to (a) until B register becomes $F.

FLOWCHART



Load address of 8-bit binary number to be shifted into X and Y registers.

Clear CA flag to clear MSB.

Rotate upper digit of 8-bit binary number to the right and set LSB of upper digit to MSB of lower digit.

Decrement Y address of 8-bit binary number.

Test if all digits of 8-bit binary number are rotated 1 bit to the right.

Decrement shift counter.

Test if shift is completed.

PROGRAM LISTING

```
ST-NO   OBJECT   ADRS   SOURCE STATEMENTS

00001   010      0000           LLEN     132
00002                  ************************************************************
00003                  *                                                          *
00004                  *       NAME :   SHIFT 8-BIT DATA (SHR)                    *
00005                  *                                                          *
00006                  ************************************************************
00007                  *                                                          *
00008                  *       ENTRY            : MD($033,$032) (8-BIT BINARY DATA) *
00009                  *                          B             (SHIFT COUNTER)     *
00010                  *       RETURNS          : MD($033,$032) (8-BIT BINARY DATA) *
00011                  *                                                          *
00012                  ************************************************************
00013                  *
00014                  XSFT     EQU      $3            8-BIT BINARY DATA ADDR(X)
00015                  YSFT     EQU      $3            8-BIT BINARY DATA ADDR(Y)
00016                  BASFT    EQU      $1            8-BIT BINARY DATA LSD ADDR(Y)-1
00017                  *
00018                           ORG      $0100
00019                  SHR      EQU      *             ENTRY POINT
00020   223      0100           LXI      XSFT          LOAD ADDR(X)
00021   213      0101  SHR1     LYI      YSFT          LOAD ADDR(Y)
00022   0EC      0102           REC
00023   090      0103  SHR2     LAM                    LOAD BINARY DATA
00024   0A0      0104           RCTR                   ROTATE BINARY DATA
00025   0D0      0105           LMADY                  STORE SHIFT DATA AND DECREMENT ADDR(Y)
00026   071      0106           YNEI     BASFT
00027   303      0107           BRS      SHR2
00028   0CF      0108           DB                     DECREMENT SHIFT COUNTER
00029   301      0109           BRS      SHR1          LOOP UNTIL SHIFT COUNTER = $0
00030   010      010A           RTN
```

When storing arguments in other RAM locations, change the EQU operands for the following labels.

XSFT: Defines X address of 8-bit binary number to be shifted to the right.

YSFT: Defines MSD Y address of 8-bit binary number to be shifted to the right.

BASFT: Defines LSD Y address less 1 of 8-bit binary number to be shifted to the right. (YSFT-$2; if this is negative, value should be defined as $F.)

FUNCTION

Increments 4-digit BCD counter; permits easy counting of interrupts (external, timer, etc.).

ARGUMENTS

1 digit= 4 bits

Contents		Storage Location	No. of Digits
Entry	——	——	——
Returns	4-digit BCD counter	MD($03D ∿ $03A)	4
	Overflow or no overflow	CA	(1 bit)

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected
× : Undefined
‡ : Result

A	B
×	●

X	Y
×	×

SPX	SPY
●	●

W
●

CA	ST
‡	×

SPECIFICATIONS

1 word=10 bits

ROM (Words)
10
RAM (Digits)
4
Stack (Digits)
0
No. of cycles
30
Reentrant
No
Relocatable
No
Interrupt OK?
Yes

DESCRIPTION

(1) Function Details

(a) Argument details

MD($03D∿$03A): Used as a 4-digit BCD counter, incremented by every DECNT execution.

CA : Indicates counter status after DECNT execution.

CA = 1 : Indicates counter overflows. (See Fig. 2)

CA = 0 : Indicates counter incremented normally.

(b) Example of DECNT execution is shown in Fig. 1. When DECNT is executed, 4-digit BCD counter is incremented as shown in part ② of Fig. 1.

SPECIFICATIONS NOTES

N/A

DESCRIPTION

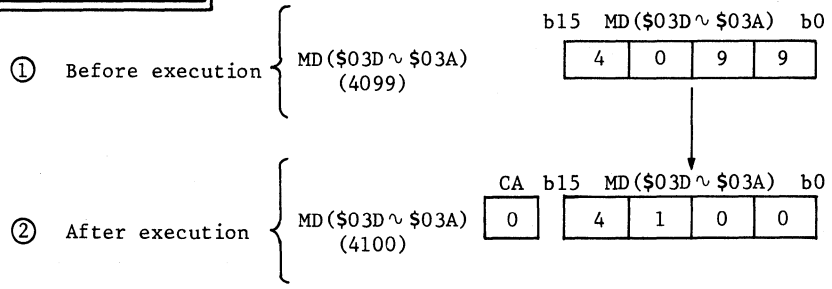


Fig. 1 Example of DECNT Execution

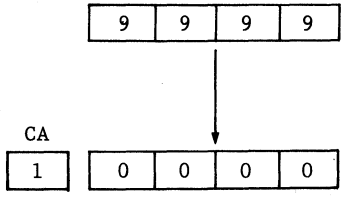


Fig. 2 Example of Counter Overflow

(2) User Notes

(a) ST flag is set after DECNT execution.

(b) If counter overflows as shown in Fig. 2, counter is cleared.

(3) RAM Allocation

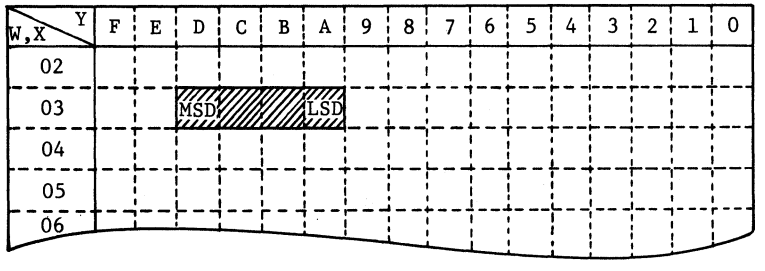


Fig. 3 RAM Allocation

Label	RAM	Description
—	b15 MSD ... LSD b0 MD($03D∿$03A)	4-digit BCD counter. X and Y addresses are defined by XDCNT and YDCNT, respectively.

DESCRIPTION

(4) Sample Application

Shown below is a sample application using DECNT with address space allocated as follows.

```
MD($0AD ∿ $0AA) :  BCD counter
```

```
        LWI     $0          ..... Example with W=0.
         ¦
        CALL    DECNT       ..... Call DECNT.
        TC                  }
        BRS     OVER        } ..... If BCD counter overflows, branch to
                                     service routine.
        LAMD    $03A        ]
        LMAD    $0AA        |
        LAMD    $03B        |
        LMAD    $0AB        } ..... Store counting result of 4-digit BCD counter,
        LAMD    $03C        |       contained in return argument, in RAM.
        LMAD    $0AC        |
        LAMD    $03D        |
        LMAD    $0AD        ]
         ¦
OVER    Service routine in case of overflow
         ¦
```

DESCRIPTION

(5) Basic Operation

(a) X and Y registers are used as a pointer to BCD counter in RAM.

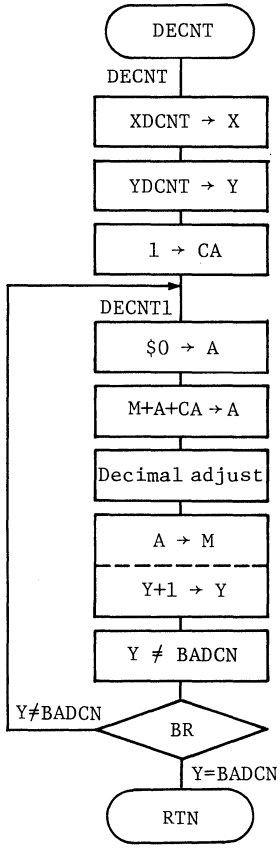
(b) CA flag is set before incrementing.

(c) Formula 1 below and decimal adjustment are performed using the AMC and DAA instructions, respectively. CA flag is set if addition of a digit generates a carry or overflow after the fourth digit.

$$M + \$0 + CA \rightarrow A \quad \ldots\ldots\ldots \text{(Formula 1)}$$

(d) Y register is incremented. Operation loops to (c) until 4-digit process is completed.

FLOWCHART



Load LSD address of 4-digit BCD counter into X and Y registers.

Set CA flag and clear Accumulator to increment BCD counter.

In the first loop, increment LSD in 4-digit BCD counter.
From the second loop on, perform addition considering carry generated from the previous addition.

Adjust addition result into decimal, and load result into RAM for 4-digit BCD counter.

Increment Y address of 4-digit BCD counter.

Test if counting by 4-digit BCD counter is completed.

PROGRAM LISTING

```
ST-NO   OBJECT   ADRS   SOURCE STATEMENTS

00001   010      0000           LLEN    132
00002                  ****************************************************************
00003                  *                                                              *
00004                  *      NAME : 4-DIGIT BCD COUNTER (DECNT)                      *
00005                  *                                                              *
00006                  ****************************************************************
00007                  *                                                              *
00008                  *      ENTRY       : NOTHING                                   *
00009                  *      RETURNS     : MD($03D-$03A)(4-DIGIT BCD COUNTER)        *
00010                  *                    CA FLAG (CA=0:INCREMENTED,CA=1:OVERFLOW)  *
00011                  *                                                              *
00012                  ****************************************************************
00013                  *
00014                  XDCNT   EQU     $3          BCD COUNTER ADDR(X)
00015                  YDCNT   EQU     $A          BCD COUNTER LSD ADDR(Y)
00016                  BADCN   EQU     $E          BCD TERMINATOR FOR ADDR(Y)
00017                  *
00018                          ORG     $0100
00019                  DECNT   EQU     *           ENTRY POINT
00020   223      0100          LXI     XDCNT       LOAD ADDR(X)
00021   21A      0101          LYI     YDCNT       LOAD LSD ADDR(Y)
00022   0EF      0102          SEC                 SET CARRY FLAG
00023   230      0103  DECNT1  LAI     $0          CLEAR A
00024   018      0104          AMC                 INCREMENT BCD COUNTER
00025   0A6      0105          DAA                 CONVERT INTO BCD DATA
00026   050      0106          LMAIY               STORE BCD COUNTER AND BCD DATA
00027   07E      0107          YNEI    BADCN       REG(Y) =/ BCD COUNTER ?
00028   303      0108          BRS     DECNT1      LOOP UNTIL REG(Y) = BCD COUNTER
00029   010      0109          RTN
```

When storing arguments in other RAM locations, change the EQU operands for the following labels.

XDCNT: Defines X address of 4-digit BCD counter.

YDCNT: Defines LSD Y address of 4-digit BCD counter.

BADCN: Defines loop terminator value for comparison with incremented Y address. (YDCNT + $4; if this is overflow, value should be defined as $0.)

11. COMPARE 8-BIT BINARY DATA	MCU	HMCS400 SERIES	LABEL	CMP

FUNCTION

Compares 2 groups of 8-bit binary data and loads result into B register corresponding to >, =, or <; uses unsigned integers as entry arguments.

ARGUMENTS

1 digit= 4 bits

Contents		Storage Location	No. of Digits
Entry	First value	MD($033, $032)	2
	Second value	MD($043, $042)	2
Returns	Comparison result	B	1

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected
× : Undefined
↕ : Result

A	B
×	↕
X	Y
×	×
SPX	SPY
×	●
W	
●	

CA	ST
×	×

SPECIFICATIONS

1 word=10 bits

ROM (Words)
18
RAM (Digits)
4
Stack (Digits)
0
No. of cycles
28
Reentrant
No
Relocatable
No
Interrupt OK?
Yes

DESCRIPTION

(1) Function Details

(a) Argument details

MD($033,$032) : Holds the first value of 8-bit binary number.

MD($043,$042) : Holds the second value of 8-bit binary number.

B : Contains $0, $1 or $2 according to comparison result.

(b) Example of CMP execution is shown in Table 1.

If entry arguments are as shown in Table 1, one of 3 codes ($0, $1, $2) is contained in B register according to comparison result.

(c) Entry arguments are saved after CMP execution.

SPECIFICATIONS NOTES

"No. of cycles" in "SPECIFICATIONS" indicates the number of cycles required to compare 2 values which are the same.

DESCRIPTION

Table 1 Example of CMP Execution

Entry arguments			Return Argument
First value MD($033,$032)	Greater or less	Second value MD($043,$042)	B register
$F6	>	$20	$1
$22	=	$22	$2
$40	<	$F0	$0

(2) User Notes

(a) ST flag is set after CMP execution.

(b) When upper digit is not needed, "0" must be stored in upper digit. Otherwise correct data cannot be obtained since comparison is performed with undefined data in upper digit.

(3) RAM Allocation

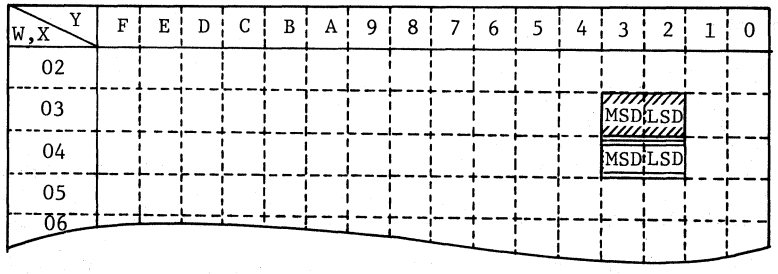


Fig. 1 RAM Allocation

Label	RAM	Description
—	b7 b0 MSD LSD MD($033, $032)	First value of 8-bit binary number. X and Y addresses are defined by XCMD and YCMT, respectively.
—	b7 b0 MSD LSD MD($043, $042)	Second value of 8-bit binary number. X and Y addresses are defined by XCMT and YCMT, respectively.

DESCRIPTION

(4) Sample Application

Shown below is a sample application using CMP with address space allocated as follows.

```
MD($0A3,$0A2)  :  First value of 8-bit binary number
MD($0B3,$0B2)  :  Second value of 8-bit binary number
```

```
LWI    $0        ..... Example with W=0.
 :
 :
LAMD   $0A3  ┐
LMAD   $033  │
LAMD   $0A2  │   ..... Store the first value of 8-bit binary number
LMAD   $032  ┘         in entry argument.

LAMD   $0B3  ┐
LMAD   $043  │
LAMD   $0B2  │   ..... Store the second value of 8-bit binary number
LMAD   $042  ┘         in entry argument.

CALL   CMP    ..... Call CMP.

LAB          ┐
ALEI   $0    │   ..... Branch to service routine for
BRS    SKIP1 ┘         First value < Second value

ALEI   $1    ┐
BRS    SKIP2 ┘   ..... Branch to service routine for
                       First value > Second value

ALEI   $2    ┐
BRS    SKIP3 ┘   ..... Branch to service routine for
                       First value = Second value

JMPL   SKIP4
```

(Continued on next page)

DESCRIPTION

(Continued from previous page)

```
SKIP1   +-----------------------------+
        | Service routine in case of  |
        | First value < Second value  |
        +-----------------------------+
                JMPL    SKIP4

SKIP2   +-----------------------------+
        | Service routine in case of  |
        | First value > Second value  |
        +-----------------------------+
                JMPL    SKIP4

SKIP3   +-----------------------------+
        | Service routine in case of  |
        | First value = Second value  |
        +-----------------------------+

SKIP4           :
                :
```

(5) Basic Operation

(a) In HMCS400 series, when comparison is performed with 2 groups of 2 or more digits, the same operation sequence is repeated for each digit.

(b) After execution of comparison instruction (ALEM), result is contained in B register.

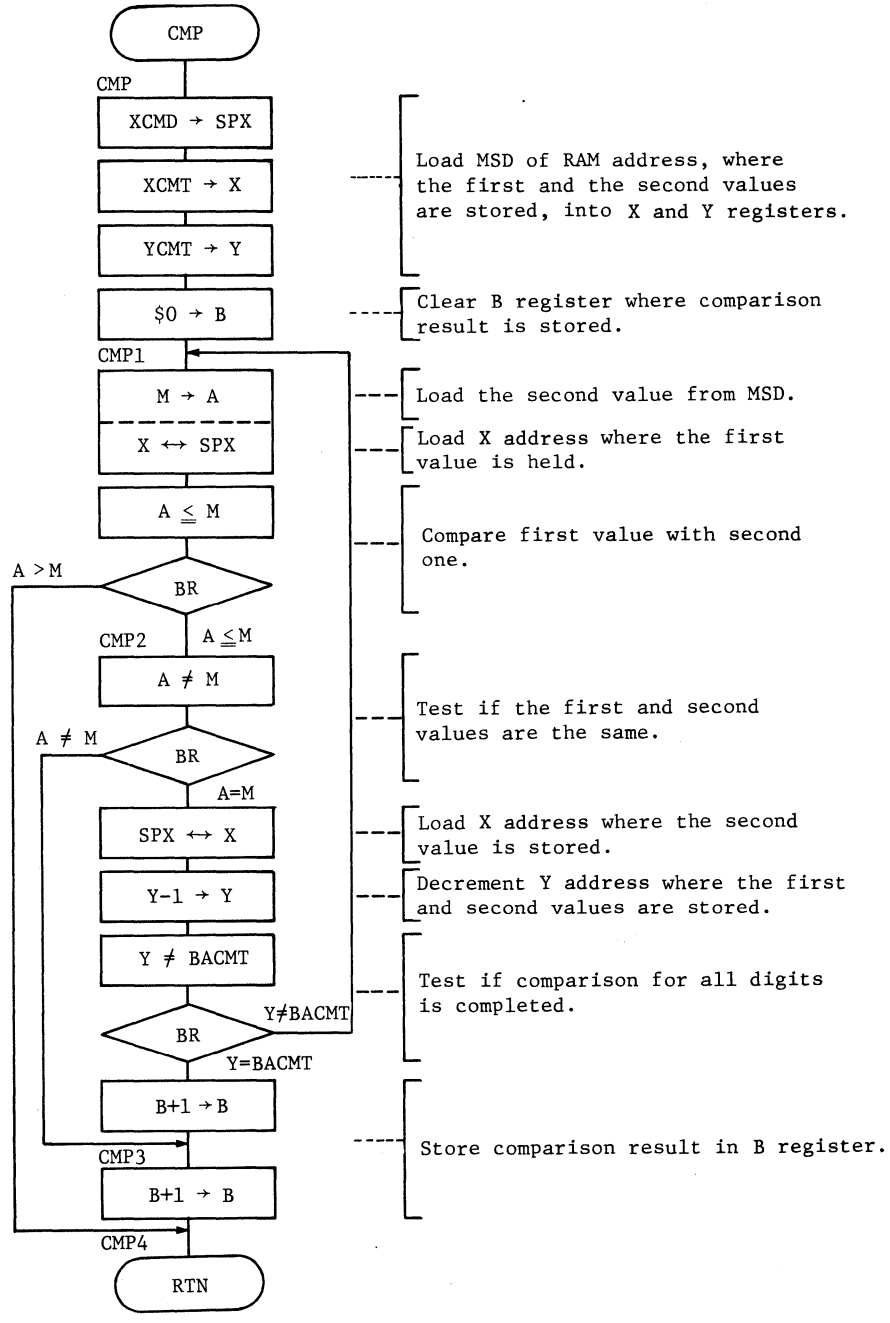
(c) Upper digits are compared using comparison instruction (ALEM).

(i) If upper digits are the same, lower digits are then compared.

(ii) If not the same, CMP is exited.

11. COMPARE 8-BIT BINARY DATA	MCU	HMCS400 SERIES	LABEL	CMP

FLOWCHART



PROGRAM LISTING

```
ST-NO   OBJECT   ADRS   SOURCE STATEMENTS

00001   010      0000           LLEN     132
00002                  ************************************************************
00003                  *                                                          *
00004                  *      NAME : COMPARE 8-BIT BINARY DATA (CMP)              *
00005                  *                                                          *
00006                  ************************************************************
00007                  *                                                          *
00008                  *      ENTRY    : MD($033,$032) (FIRST VALUE)              *
00009                  *                 MD($043,$042) (SECOND VALUE)             *
00010                  *      RETURNS  : B             (COMPARISON RESULT)        *
00011                  *                                                          *
00012                  ************************************************************
00013                  *
00014                  XCMD     EQU      $3            FIRST VALUE ADDR(X)
00015                  XCMT     EQU      $4            SECOND VALUE ADDR (Y)
00016                  YCMT     EQU      $3            MSD ADDR(Y) OF FIRST & SECOND VALUE
00017                  BACMT    EQU      $1            LOOP TERMINATOR FOR ADDR(X) & ADDR(Y)
00018                  *
00019                           ORG      $0100
00020                  CMP      EQU      *             ENTRY POINT
00021   223      0100          LXI      XCMD          LOAD FIRST VALUE ADDR(SPX)
00022   001      0101          XSPX
00023   224      0102          LXI      XCMT          LOAD SECOND VALUE ADDR(X)
00024   213      0103          LYI      YCMT          LOAD SECOND VALUE ADDR(Y)
00025   200      0104          LBI      $0            CLEAR B
00026   091      0105   CMP1   LAMX                    LOAD SECOND VALUE
00027   014      0106          ALEM                    DETERMINE RELATION
00028   309      0107          BRS      CMP2          BRANCH IF A =< M
00029   311      0108          BRS      CMP4          BRANCH IF A > M
00030   004      0109   CMP2   ANEM                    TEST IF A = M ?
00031   310      010A          BRS      CMP3          BRANCH IF A = M
00032   001      010B          XSPX                    LOAD SECOND VALUE ADDR(X)
00033   00F      010C          DY                      DECREMENT ADDR(Y)
00034   071      010D          YNEI     BACMT         TEST LOOP COUNTER
00035   305      010E          BRS      CMP1          BRANCH IF Y =/ LOOP COUNTER
00036   04C      010F          IB                     INCREMENT B (RESULT)
00037   04C      0110   CMP3   IB                     INCREMENT B (RESULT)
00038   010      0111   CMP4   RTN
```

When storing arguments in other RAM locations, change the EQU operands for the following labels.

XCMD: Defines X address of the first value.

XCMT: Defines X address of the second value.

YCMT: Defines MSD Y address of the first and second values.

BACMT: Defines loop terminator value for comparison with decremented Y address. (YCMT-$2; if this is negative, value should be defined as $F.)

FUNCTION

Performs addition of 8-bit binary data in RAM, and stores result in RAM; uses unsigned integers as arguments.

ARGUMENTS

1 digit= 4 bits

Contents		Storage Location	No. of Digits
Entry	Augend	HAUG, LAUG (RAM)	2
	Addend	HADD, LADD (RAM)	2
Returns	Addition result	HAUG, LAUG (RAM)	2
	Carry or no carry	CA	(1 bit)
		ST	(1 bit)

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected
× : Undefined
‡ : Result

A	B
×	●
X	Y
●	●
SPX	SPY
●	●
W	
●	

CA	ST
‡	‡

SPECIFICATIONS

1 word=10 bits

ROM (Words)
14
RAM (Digits)
4
Stack (Digits)
0
No. of cycles
16
Reentrant
No
Relocatable
No
Interrupt OK?
Yes

DESCRIPTION

(1) Function Details

(a) Argument details

HAUG, LAUG(RAM): Holds augend of 8-bit binary number. After ADD execution, contains addition result.

HADD, LADD(RAM): Holds addend of 8-bit binary number.

CA, ST : Indicates whether a carry is generated or not after ADD execution.

CA=1, ST=1 : Indicates a carry is generated in addition result. (See Fig. 2)

CA=0, ST=0 : Indicates no carry is generated in addition result.

SPECIFICATIONS NOTES

N/A

DESCRIPTION

(b) Example of ADD execution is shown in Fig. 1. If entry arguments are as shown in part ① of Fig. 1, addition result is contained in HAUG, LAUG(RAM) as shown in part ② of Fig. 1.

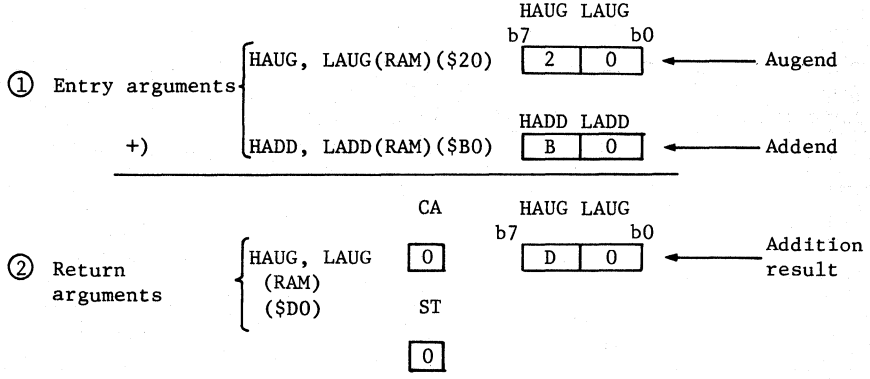


Fig. 1 Example of ADD Execution

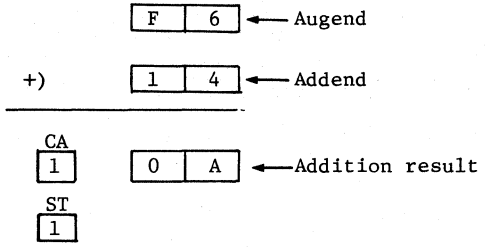


Fig. 2 Example of Addition When Carry is Generated

(2) User Notes

(a) When upper digit is not needed, "0" must be stored in upper digit as shown in Fig. 3. Otherwise correct addition result cannot be obtained since addition is performed with undefined data in upper digit.

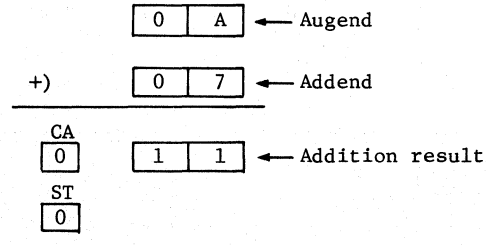


Fig. 3 Example of Addition When Upper Digits are not Needed

DESCRIPTION

(b) After ADD execution, augend is destroyed since addition result is contained in HAUG and LAUG(RAM). If augend needs to be retained after ADD execution, it should be saved in memory before execution.

(3) RAM Allocation

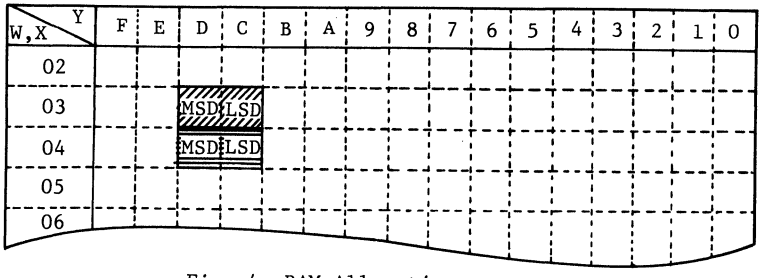


Fig. 4 RAM Allocation

Label	RAM	Description
HAUG	b3 b0 MSD MD($03D)	Upper digit of 8-bit binary augend before execution. Upper digit of 8-bit binary addition result after execution.
LAUG	b3 b0 LSD MD($03C)	Lower digit of 8-bit binary augend before execution. Lower digit of 8-bit binary addition result after execution.
HADD	b3 b0 MSD MD($04D)	Upper digit of 8-bit binary addend.
LADD	b3 b0 LSD MD($04C)	Lower digit of 8-bit binary addend.

DESCRIPTION

(4) Sample Application

Shown below is a sample application using ADD with address space allocated as follows.

```
MD($0AD, $0AC): Augend
                After execution, addition result is contained.

MD($0BD, $0BC): Addend
```

```
        LWI     $0      -------Example with W=0.
        :
        LAMD    $0AC    ]
        LMAD    LAUG    }------- Store augend in entry argument.
        LAMD    $0AD    }
        LMAD    HAUG    ]
        LAMD    $0BC    ]
        LMAD    LADD    }------- Store addend in entry argument.
        LAMD    $0BD    }
        LMAD    HADD    ]

        CALL    ADD     |-------Call ADD.

        BRS     OVER    -------Branch to service routine in case of carry.

        LAMD    LAUG    ]
        LMAD    $0AC    }-------Store addition result, contained in return
        LAMD    HAUG    }       argument, in RAM.
        LMAD    $0AD    ]
        :
OVER    Service routine in
        case of carry
        :
```

DESCRIPTION

(5) Basic Operation

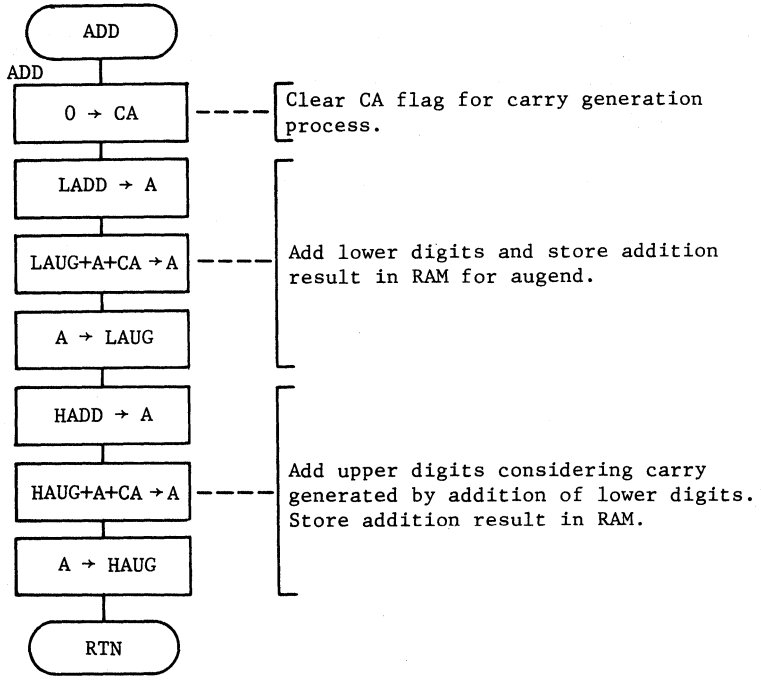
(a) In HMCS400 series, when addition is performed with 2 or more digits, the same operation sequence is repeated for each digit.

(b) CA flag is cleared, and lower digits are added using addition instruction (AMCD) as shown in (Formula 1). If a carry is generated after (Formula 1) execution, CA flag is set.

Augend + Addend + CA → Addition result ..... (Formula 1)

(c) Upper digits are then added using addition instruction (AMCD).

FLOWCHART


ADD
ADD
0 → CA
Clear CA flag for carry generation process.
LADD → A
LAUG+A+CA → A
Add lower digits and store addition result in RAM for augend.
A → LAUG
HADD → A
HAUG+A+CA → A
Add upper digits considering carry generated by addition of lower digits. Store addition result in RAM.
A → HAUG
RTN

PROGRAM LISTING

```
ST-NO   OBJECT   ADRS   SOURCE STATEMENTS

00001   010      0000           LLEN     132
00002                  ****************************************************************
00003                  *                                                              *
00004                  *      NAME : ADD 8-BIT BINARY DATA (ADD)                      *
00005                  *                                                              *
00006                  ****************************************************************
00007                  *                                                              *
00008                  *      ENTRY       : HAUG.LAUG    (AUGEND)                    *
00009                  *                    HADD.LADD    (ADDEND)                    *
00010                  *      RETURNS     : HAUG. LAUG   (ADDITION RESULT)           *
00011                  *                    CA FLAG(CA=0:NORMAL RETURN.CA=1:OVERFLOW)*
00012                  *                                                              *
00013                  ****************************************************************
00014                  *
00015                  HAUG     EQU      $03D          UPPER AUGEND
00016                  LAUG     EQU      $03C          LOWER AUGEND
00017                  HADD     EQU      $04D          UPPER ADDEND
00018                  LADD     EQU      $04C          LOWER ADDEND
00019                  *
00020                           ORG      $0100
00021                  ADD      EQU      *             ENTRY   POINT
00022   0EC      0100           REC                    CLEAR CARRY FLAG
00023   190 04C  0101           LAMD     LADD          LOAD LOWER ADDEND
00024   118 03C  0103           AMCD     LAUG          ADD LOWER ADDEND TO LOWER AUGEND
00025   194 03C  0105           LMAD     LAUG          STORE LOWER ADDITION RESULT
00026   190 04D  0107           LAMD     HADD          LOAD UPPER ADDEND
00027   118 03D  0109           AMCD     HAUG          ADD UPPER ADDEND TO UPPER AUGEND
00028   194 03D  0108           LMAD     HAUG          STORE UPPER ADDITION RESULT
00029   010      010D           RTN
```

When storing arguments in other RAM locations, change the EQU operands for the following labels.

HAUG: Defines upper digit address of augend.

LAUG: Defines lower digit address of augend.

HADD: Defines upper digit address of addend.

LADD: Defines lower digit address of addend.

13. SUBTRACT 8-BIT BINARY DATA

MCU | HMCS400 SERIES | LABEL | **SUB**

FUNCTION

Performs subtraction of 2-digit binary data in RAM, and stores result in RAM; uses unsigned integers as arguments.

ARGUMENTS

1 digit= 4 bits

Contents		Storage Location	No. of Digits
Entry	Minuend	HMIN, LMIN (RAM)	2
	Subtrahend	HSUB, LSUB (RAM)	2
Returns	Subtraction result	HMIN, LMIN (RAM)	2
	Borrow or no borrow	CA	(1 bit)
		ST	(1 bit)

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected
× : Undefined
↕ : Result

A	B
×	●
X	Y
●	●
SPX	SPY
●	●
W	
●	

CA	ST
↕	↕

SPECIFICATIONS

1 word=10 bits

ROM (Words)	14
RAM (Digits)	4
Stack (Digits)	0
No. of cycles	16
Reentrant	No
Relocatable	No
Interrupt OK?	Yes

DESCRIPTION

(1) Function Details

(a) Argument details

HMIN, LMIN(RAM): Holds 8-bit binary minuend. After SUB execution, contains subtraction result.

HSUB, LSUB(RAM): Holds 8-bit binary subtrahend.

CA, ST : Indicates whether a borrow is generated or not after SUB execution

CA=1, ST=1 : Indicates no borrow is generated in subtraction result.

CA=0, ST=0 : Indicates a borrow is generated in subtraction result. (see Fig. 2)

SPECIFICATIONS NOTES

N/A

DESCRIPTION

(b) Example of SUB execution is shown in Fig. 1. If entry arguments are as shown in part ① of Fig. 1, subtraction result is contained in HMIN, LMIN(RAM) as shown in part ② of Fig. 1.

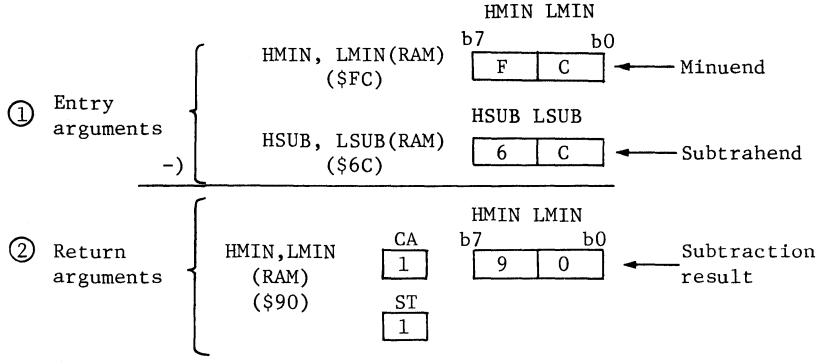


Fig. 1 Example of SUB Execution

(2) User Notes

(a) When subtraction result is negative as shown in Fig. 2 (minuend < subtrahend), the result is in 2's complement.

(b) When upper digit is not needed, "0" must be stored in upper digit as shown in Fig. 3. Otherwise correct subtraction result cannot be obtained since subtraction is performed with undefined data in upper digit.

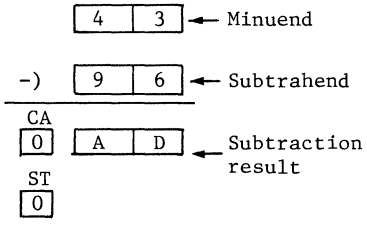


Fig. 2 Example When Borrow is Generated

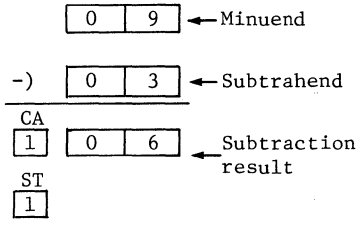


Fig. 3 Example When Upper Digit is Not Used

(c) After SUB execution, minuend is destroyed since subtraction result is stored in HMIN, LMIN(RAM).
If minuend needs to be retained after SUB execution, it should be saved in memory before execution.

DESCRIPTION

(3) RAM Allocation

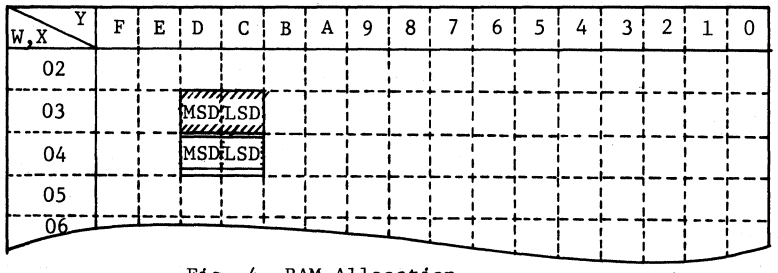


Fig. 4 RAM Allocation

Label	RAM	Description
HMIN	b3 b0 MSD MD($03D)	Upper digit of minuend before execution. Upper digit of subtraction result after execution.
LMIN	b3 b0 LSD MD($03C)	Lower digit of minuend before execution. Lower digit of subtraction result after execution.
HSUB	b3 b0 MSD MD($04D)	Upper digit of subtrahend.
LSUB	b3 b0 LSD MD($04C)	Lower digit of subtrahend.

DESCRIPTION

(4) Sample Application

Shown below is a sample application using SUB with address space allocated as follows.

```
MD($0AD, $0AC): Minuend
                After execution, subtraction result is contained.

MD($0BD, $0BC): Subtrahend
```

```
        LWI     $0          ------- Example with W=0.
         :
        LAMD    $0AD        ┐
        LMAD    HMIN        │
        LAMD    $0AC        ├-------- Store 8-bit binary minuend in entry
        LMAD    LMIN        ┘         argument.
        LAMD    $0BD        ┐
        LMAD    HSUB        ├---------Store 8-bit binary subtrahend in entry
        LAMD    $0BC        │         argument.
        LMAD    LSUB        ┘

        CALL    SUB         ------- Call SUB.

        BRS     WORK        ┐
        BRS     BRROW       ├-------- If a borrow is generated, branch to
                            ┘         service routine.

WORK    LAMD    HMIN        ┐
        LMAD    $0AD        │
        LAMD    LMIN        ├------- Store 8-bit binary subtraction result,
        LMAD    $0AC        ┘        contained in return argument, in RAM.
         :
BRROW   Service routine
        in case of borrow
         :
```

DESCRIPTION

(5) Basic Operation

(a) In HMCS400 series, when subtraction is performed with 2 or more digits, the same operation sequence is repeated for each digit.

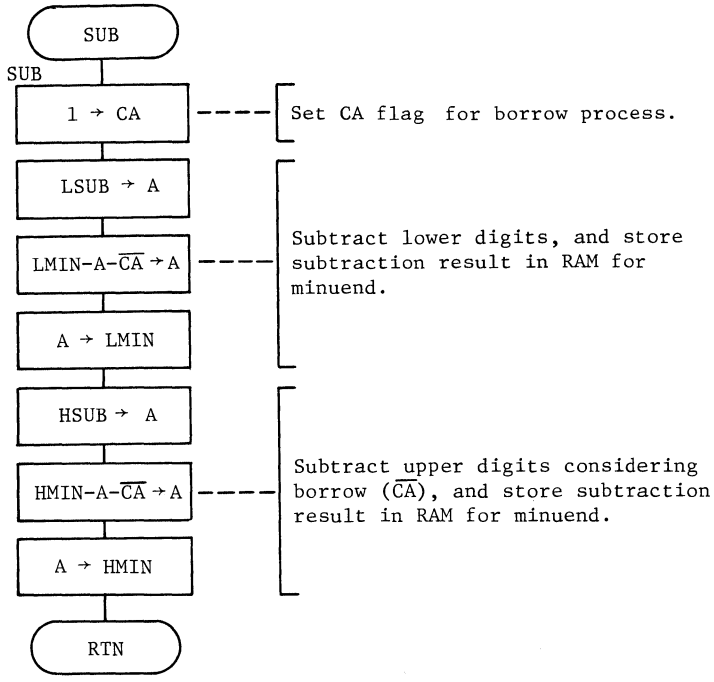
(b) CA flag is set . Subtraction of lower digits is then performed using subtraction instruction (SMCD) as shown in (Formula 1) (CA flag is cleared if borrow is generated).

$$\text{Minuend} - \text{Subtrahend} - \overline{CA} \rightarrow \text{Result} \quad \ldots\ldots \text{(Formula 1)}$$

(c) Subtraction of upper digits is performed using subtraction instruction (SMCD).

13. SUBTRACT 8-BIT BINARY DATA	MCU	HMCS400 SERIES	LABEL	SUB

FLOWCHART



PROGRAM LISTING

```
ST-NO   OBJECT   ADRS   SOURCE STATEMENTS

00001   010      0000           LLEN     132
00002                  ************************************************************
00003                  *                                                          *
00004                  *      NAME : SUBTRACT 8-BIT BINARY DATA (SUB)             *
00005                  *                                                          *
00006                  ************************************************************
00007                  *                                                          *
00008                  *      ENTRY       : HMIN.LMIN(MINUEND)                    *
00009                  *                    HSUB.LSUB(SUBTRAHEND)                 *
00010                  *      RETURNS     : HMIN.LMIN(SUBTRACTION RESULT)         *
00011                  *                    FLAG  (CA=0,ST=0:BORROW               *
00012                  *                           CA=1,ST=1:NORMAL RETURN)       *
00013                  *                                                          *
00014                  ************************************************************
00015                  *
00016                  HMIN     EQU      $03D          UPPER MINUEND
00017                  LMIN     EQU      $03C          LOWER MINUEND
00018                  HSUB     EQU      $04D          UPPER SUBTRAHEND
00019                  LSUB     EQU      $04C          LOWER SUBTRAHEND
00020                  *
00021                           ORG      $0100
00022                  SUB      EQU      *             ENTRY POINT
00023   0EF      0100           SEC                    SET CARRY FLAG
00024   190 04C  0101           LAMD     LSUB          LOAD LOWER SUBTRAHEND
00025   198 03C  0103           SMCD     LMIN          SUBTRACT LOWER SUBTRAHEND FROM LOWER MINUEND
00026   194 03C  0105           LMAD     LMIN          STORE LOWER SUBTRACTION RESULT
00027   190 04D  0107           LAMD     HSUB          LOAD UPPER SUBTRAHEND
00028   198 03D  0109           SMCD     HMIN          SUBTRACT UPPER SUBTRAHEND FROM UPPER MINUEND
00029   194 03D  010B           LMAD     HMIN          STORE UPPER SUBTRACTION RESULT
00030   010      010D           RTN
```

When storing arguments in other RAM locations, change the EQU operands for the following labels.

HMIN: Defines upper digit address of minuend.

LMIN: Defines lower digit address of minuend.

HSUB: Defines upper digit address of subtrahend.

LSUB: Defines lower digit address of subtrahend.

14. MULTIPLY 16-BIT BINARY DATA

MCU | HMCS400 SERIES | LABEL | MUL

FUNCTION

Performs multiplication of 16-bit binary data in RAM, and stores 32-bit binary product in RAM; uses unsigned integers as arguments.

ARGUMENTS

1 digit= 4 bits

Contents		Storage Location	No. of Digits
Entry	Multiplicand	MD($03D ∿ $03A)	4
	Multiplier	MD($049 ∿ $046)	4
Returns	Product (upper 16 bits)	MD($04D ∿ $04A)	4
	Product (lower 16 bits)	MD($049 ∿ $046)	4

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected
× : Undefined
‡ : Result

A	B
×	×
X	Y
×	×
SPX	SPY
×	●
W	
●	

CA	ST
×	×

SPECIFICATIONS

1 word=10 bits

ROM (Words)
29
RAM (Digits)
12
Stack (Digits)
0
No. of cycles
1550
Reentrant
No
Relocatable
No
Interrupt OK?
Yes

DESCRIPTION

(1) Function Details

(a) Argument details

MD($03D ∿ $03A): Holds 16-bit binary multiplicand.

MD($049 ∿ $046): Holds 16-bit binary multiplier. After MUL execution, contains lower 16-bit of product.

MD($04D ∿ $04A): Contains upper 16-bit of product.

(b) Example of MUL execution is shown in Fig. 1. If entry arguments are as shown in part ① of Fig. 1, product is contained in MD($04D ∿ $046) as shown in part ② of Fig. 1.

SPECIFICATIONS NOTES

"No. of cycles" in "SPECIFICATIONS" indicates the number of cycles required to multiply a multiplicand and multiplier both equal to $FFFF.

DESCRIPTION

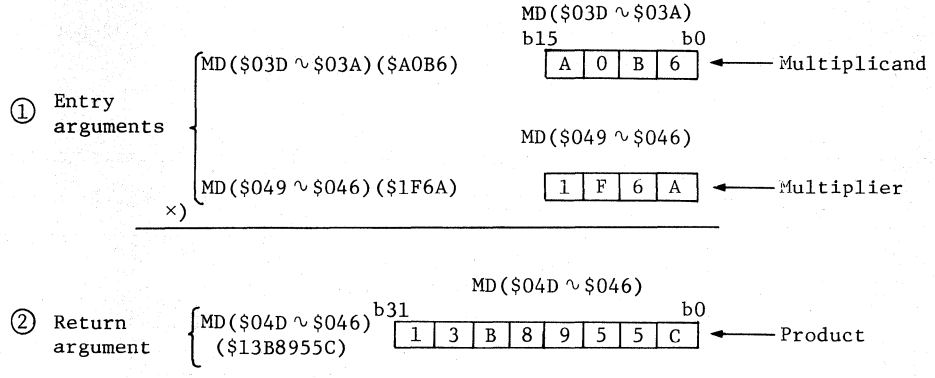


Fig. 1 Example of MUL Execution

(c) Table 1 shows product when $0 is held in entry arguments.

Table 1 Product When Holding "0" in Entry Arguments

Entry Arguments		Return Argument
Multiplicand MD($03D ∿ $03A)	Multiplier MD($049 ∿ $046)	Product MD($04D ∿ $046)
$**** (note)	$0000	$00000000
$0000	$**** (note)	$00000000
$0000	$0000	$00000000

(note) $****: Hexadecimal data

DESCRIPTION

(2) User Notes

(a) ST flag is set after MUL execution.

(b) When upper digits are not needed, "0" must be stored in upper digits as shown in Fig. 2. Otherwise correct product cannot be obtained since multiplication is performed with undefined data in upper digit.

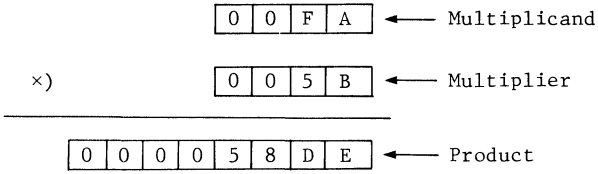


Fig. 2 Multiplication Example When Upper Digit is Not Used

(c) After MUL execution, multiplier is destroyed since lower 4 digits of product is contained in MD($049 ∿ $046). If multiplier needs to be retained after MUL execution, it should be saved in memory before execution.

(3) RAM Allocation

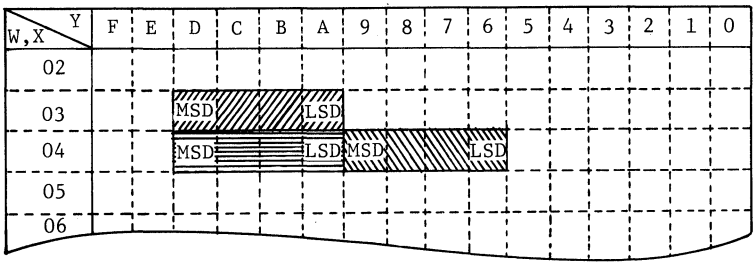


Fig. 3 RAM Allocation

Label	RAM	Description
—	b15 b0 MSD LSD MD($03D ∿ $03A)	16-bit binary multiplicand. X and Y addresses are defined by XMCAN and YMPRO, respectively.
—	b15 b0 MSD LSD MD($04D ∿ $04A)	Upper 16-bit of product. X and Y addresses are defined by XPROW and YMPRO, respectively.
—	b15 b0 MSD LSD MD($049 ∿ $046)	16-bit binary multiplier before execution. Lower 16-bit product after execution. X and Y addresses are defined by XPROW and YMLSB, respectively.

DESCRIPTION

(4) Sample Application

Shown below is a sample application using MUL with address space allocated as follows.

MD($0AD ∿ $0AA) : Multiplicand

MD($0BD ∿ $0BA) : Multiplier

MD($0AD ∿ $0A6) : Product

```
        LWI     $0      --- Example with W=0.
         :
        LXI     $3
        XSPX
        LXI     $A
        LYI     $A      ---  Store multiplicand in entry
WORK1   LAMX                 argument.
        LMAIYX
        YNEI    $E
        BRS     WORK1
```

(Continued on next page)

DESCRIPTION

(Continued from previous page)

```
        LXI     $4
        XSPX
        LXI     $B
        LYI     $6      ---- Store multiplier in entry argument.
WORK2   LAMX
        LMAIYX
        YNEI    $A
        BRS     WORK2

        CALL    MUL     ---- Call MUL.

        LXI     $A
        XSPX
        LXI     $4
        LYI     $6
WORK3   LAMX            ---- Store product, contained in return
        LMAIYX                  argument, in RAM.
        YNEI    $E
        BRS     WORK3
```

DESCRIPTION

(5) Basic Operation

(a) Example of 4-bit binary multiplication is shown in Fig. 4.

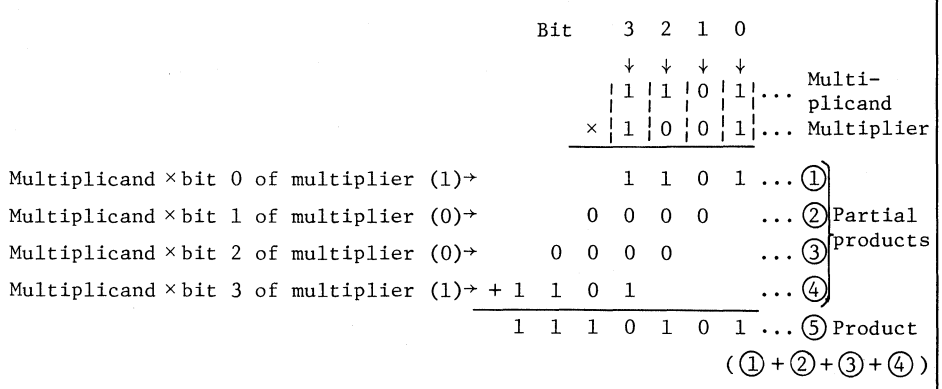


Fig. 4 Multiplication Example ($D × $9 = $75)

Multiplication of 16-bit binary data requires obtaining partial products, as shown in ①, ②, ③ and ④, and adding them. (⑤ in Fig. 4) Each bit of binary data is either "0" or "1". If multiplier bit is "1", its partial product is multiplicand (① and ④ in Fig. 4), while if multiplier bit is "0", its partial product is "0". (② and ③ in Fig. 4)

(b) Program operation is described below according to the multiplication example in Fig. 4.

(i) RAM for upper digit of product is cleared.

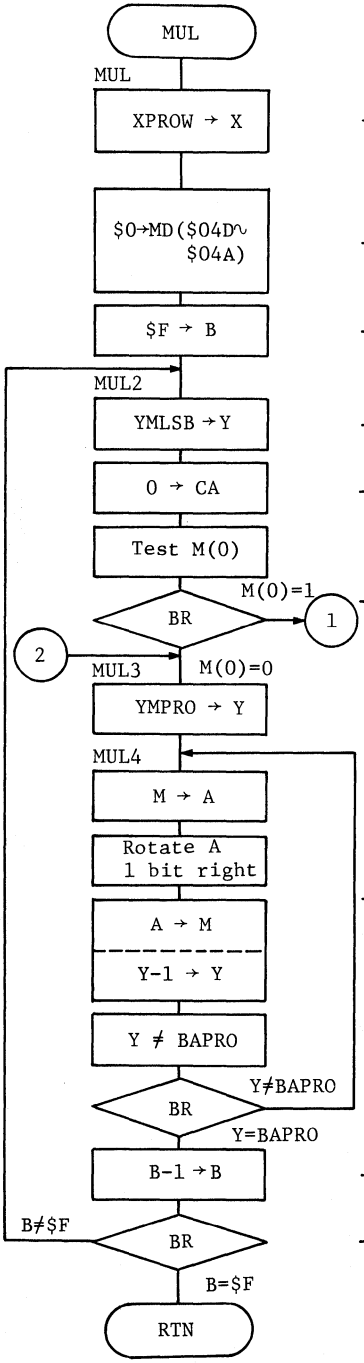
(ii) LSB of multiplier is tested whether it is "0" or "1" to obtain partial product.

(iii) B register is decremented.

(iv) Operation loops (ii) and (iii) until B register is $F.

14. MULTIPLY 16-BIT BINARY DATA	MCU	HMCS400 SERIES	LABEL	MUL

FLOWCHART



Load product and multiplier address into X register.

Clear RAM for product.

Load counter indicating number of partial products.

Load LSD address where multiplier is stored into Y register.

Clear CA flag so that "1" is not set in MSB, when rotating RAM for product.

Test whether LSB of multiplier is "1" or "0".

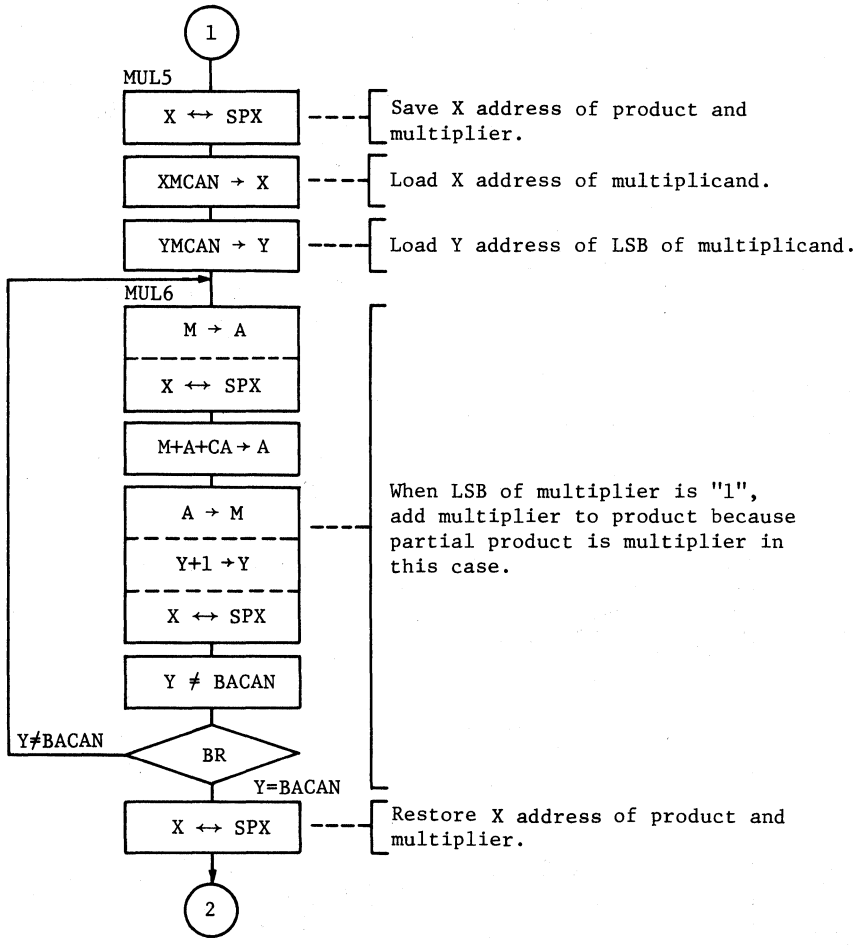
Store LSB of product in MSB of multiplier.

Rotate the next bit to calculate the next partial product into LSB of multiplier.

Decrement counter indicating number of partial products.

Test if all partial products have been acquired.

FLOWCHART


1
MUL5
X ↔ SPX
Save X address of product and multiplier.
XMCAN → X
Load X address of multiplicand.
YMCAN → Y
Load Y address of LSB of multiplicand.
MUL6
M → A
X ↔ SPX
M+A+CA → A
A → M
Y+1 → Y
X ↔ SPX
When LSB of multiplier is "1", add multiplier to product because partial product is multiplier in this case.
Y ≠ BACAN
Y≠BACAN
BR
Y=BACAN
X ↔ SPX
Restore X address of product and multiplier.
2

PROGRAM LISTING

```
ST-NO   OBJECT   ADRS   SOURCE STATEMENTS

00001   010      0000           LLEN    132
00002                  ************************************************
00003                  *                                              *
00004                  *      NAME : MULTIPLY 8-BIT BINARY DATA (MUL) *
00005                  *                                              *
00006                  ************************************************
00007                  *                                              *
00008                  *      ENTRY          : MD($03D-$03A)(MULTIPLICAND)  *
00009                  *                       MD($049-$046)(MULTIPLIER)   *
00010                  *      RETURNS        : MD($04D-$046)(PRODUCT)       *
00011                  *                                              *
00012                  ************************************************
00013                  *
00014                  XMCAN   EQU     $3          MULTIPLICAND ADDR(X)
00015                  XPROW   EQU     $4          MULTIPLIER & PRODUCT ADDR(X)
00016                  YMLSB   EQU     $6          MULTIPLIER & PRODUCT LSD ADDR(Y)
00017                  YMPRO   EQU     $D          MULTIPLICAND & PRODUCT MSD ADDR(Y)
00018                  YMCAN   EQU     $A          MULTIPLICAND LSD ADDR(Y)
00019                  BAPRO   EQU     $5          MULTIPLICAND & PRODUCT LSD ADDR(Y)-1
00020                  BACAN   EQU     $E          MULTIPLICAND & PRODUCT MSD ADDR(Y)+1
00021                  *
00022                          ORG     $0100
00023                  MUL     EQU     *           ENTRY POINT
00024   224      0100           LXI     XPROW       LOAD MULTIPLIER & PRODUCT ADDR(X)
00025   21A      0101           LYI     YMCAN       LOAD MULTIPLICAND ADDR(Y)
00026   290      0102   MUL1    LMIIY   $0          CLEAR PRODUCT
00027   07E      0103           YNEI    BACAN
00028   302      0104           BRS     MUL1
00029   20F      0105           LBI     $F          LOAD BIT COUNTER
00030   216      0106   MUL2    LYI     YMLSB       LOAD MULTIPLIER LSD ADDR INTO REG(Y)
00031   0EC      0107           REC                 CLEAR CARRY
00032   08C      0108           TM      $0          TEST LSB OF PRODUCT = $0 ?
00033   313      0109           BRS     MUL5        BRANCH  IF LSB = $1
00034   210      010A   MUL3    LYI     YMPRO
00035   090      010B   MUL4    LAM
00036   0A0      010C           ROTR                ROTATE PRODUCT AND MULTIPLIER 1-BIT RIGHT
00037   0D0      010D           LMADY
00038   075      010E           YNEI    BAPRO
00039   30B      010F           BRS     MUL4
00040   0CF      0110           DB                  DECREMENT BIT COUNTER
00041   306      0111           BRS     MUL2        LOOP UNTIL BIT COUNTER = $F
00042   010      0112           RTN
00043   001      0113   MUL5    XSPX                SAVE PRODUCT & MULTIPLIER ADDR(X)
00044   223      0114           LXI     XMCAN       LOAD MULTIPLICAND ADDR(X) INTO REG(X)
00045   21A      0115           LYI     YMCAN       LOAD MULTIPLICAND LSB ADDR(Y) INTO REG(Y)
00046   091      0116   MUL6    LAMX
00047   018      0117           AMC                 MULTIPLICAND + PRODUCT -> MULTIPLICAND
00048   051      0118           LMAIYX
00049   07E      0119           YNEI    BACAN
00050   316      011A           BRS     MUL6
00051   001      011B           XSPX
00052   30A      011C           BRS     MUL3        LOAD NEXT LSB
```

When storing arguments in other RAM locations, change the EQU operands for the following labels.

XMCAN: Defines X address of multiplicand.

XPROW: Defines X address of product and multiplier.

YMLSB: Defines LSD Y address of product and multiplier.

YMPRO: Defines MSD Y address of product and multiplicand. (YMLSB + $7)

YMCAN: Defines LSD Y address of multiplicand. (YMLSB + $4)

BAPRO: Defines LSD Y address less 1 of product and multiplicand. (YMLSB-$1; if this is negative, value should be defined as $F.)

BACAN: Defines MSD Y address plus 1 of product and multiplicand. (YMPRO + $1; if this is overflow, value should be defined as $0.)

15. DIVIDE 16-BIT BINARY DATA	MCU	HMCS400 SERIES	LABEL	DIV

FUNCTION

Performs divisions of 16-bit binary data in RAM, and stores result (quotient and remainder) as 16-bit binary data in RAM; uses unsigned integers as arguments.

ARGUMENTS

1 digit= 4 bits

Contents		Storage Location	No. of Digits
Entry	Dividend	MD($03A ∿ $037)	4
	Divisor	MD($04E ∿ $04B)	4
Returns	Quotient	MD($03A ∿ $037)	4
	Remainder	MD($03E ∿ $03B)	4

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected
× : Undefined
‡ : Result

A	B
×	×

X	Y
×	×

SPX	SPY
×	●

W
●

CA	ST
×	×

SPECIFICATIONS

1 word=10 bits

ROM (Words)
38
RAM (Digits)
12
Stack (Digits)
0
No. of cycles
1178
Reentrant
No
Relocatable
No
Interrupt OK?
Yes

DESCRIPTION

(1) Function Details

(a) Argument details

MD($03A∿$037): Holds 16-bit binary dividend. After DIV execution, contains quotient.

MD($04E∿$04B): Holds 16-bit binary divisor.

MD($03E∿$03B): Holds 16-bit binary remainder.

(b) Example of DIV execution is shown in Fig. 1.
If entry arguments are as shown in part ① of Fig. 1, division result is contained in MD($03E ∿ $037).

SPECIFICATIONS NOTES

"No. of cycles" in "SPECIFICATIONS" indicates the number of cycles required to devide $FFFF by $2.

DESCRIPTION

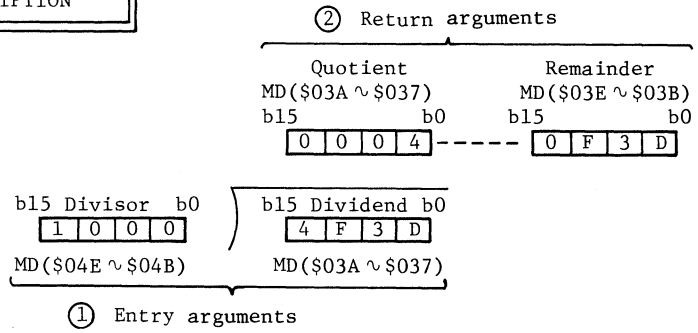


Fig. 1 Example of DIV Execution

(c) Table 1 shows results when $0 is held in entry arguments.

Table 1 Results When Holding "0" in Entry Arguments

Entry Arguments		Return Arguments	
Dividend MD($03A ∿ $037)	Divisor MD($04E ∿ $04B)	Quotient MD($03A ∿ $037)	Remainder MD($03E ∿ $03B)
$**** (note)	$0000	$FFFF	$**** (note)
$0000	$**** (note)	$0000	$0000
$0000	$0000	$FFFF	$0000

(note) $****: Hexadecimal data

(2) User Notes

(a) ST flag is set after DIV execution.

(b) When upper digits are not needed, "0" must be stored in upper digits as shown in Fig. 2. Otherwise correct results cannot be obtained since division is performed with undefined data in upper digit.

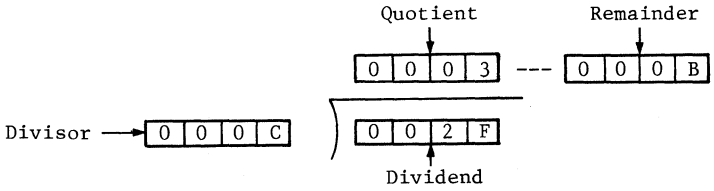


Fig. 2 Division Example When Upper Digit is Not Used

DESCRIPTION

(c) After DIV execution, dividend is destroyed since quotient is contained in MD($03A ∿ $037). If dividend needs to be retained after execution, it should be saved in memory before execution.

(3) RAM Allocation

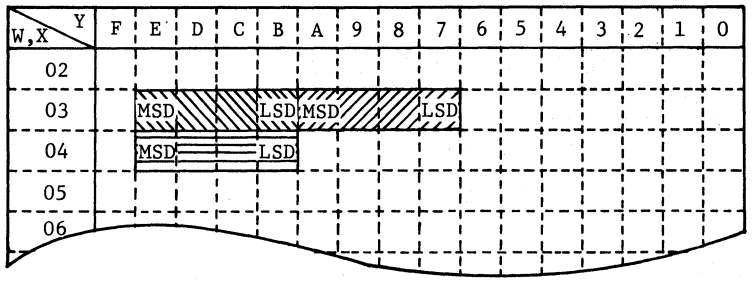


Fig. 3 RAM Allocation

Label	RAM	Description
—	b15 b0 MSD LSD MD($03A ∿ $037)	16-bit binary dividend before execution. 16-bit binary quotient after execution. X and Y addresses are defined by XDEND and YDSOR, respectively.
—	b15 b0 MSD LSD MD($04E ∿ $04B)	16-bit binary divisor. X and Y addresses are defined by XDSOR and YDSOR, respectively.
—	b15 b0 MSD LSD MD($03E ∿ $03B)	Work area where subtraction result (Dividend-Divisor) is contained. Stores 16-bit binary remainder after execution. X and Y addresses are defined by XDEND and YDEND, respectively.

DESCRIPTION

(4) Sample Application

Shown below is a sample application using DIV with address space allocated as follows.

MD($0AA∿$0A7): Dividend. After execution, quotient is contained.

MD($0AE∿$0AB): Divisor. After execution, remainder is contained.

```
        LWI     $0          -----Example with W=0.
         ¦
         ¦
        LXI     $3
        XSPX
        LXI     $A
        LYI     $7          -----Store 16-bit binary dividend in entry
WORK1   LAMX                     argument.
        LMAIYX
        YNEI    $A
        BRS     WORK1
        LXI     $4
        XSPX
        LXI     $A
        LYI     $B          -------Store 16-bit binary divisor in entry
WORK2   LAMX                       argument.
        LMAIYX
        YNEI    $F
        BRS     WORK2

        CALL    DIV         ------Call DIV.

        LXI     $A
        XSPX
        LXI     $3
        LYI     $7          -------Store division result, contained in
WORK3   LAMX                       return arguments, in RAM.
        LMAIYX
        YNEI    $F
        BRS     WORK3
         ¦
```

DESCRIPTION

(5) Basic Operation

(a) In binary data division, quotient and remainder are obtained by repeated subtraction. Fig. 4 shows an example of binary division ($D ÷ $3).

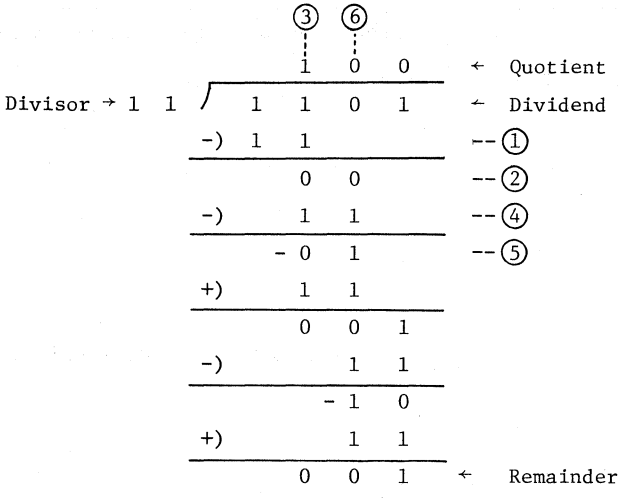


Fig. 4 Division Example ($D ÷ $3)

(b) Referring to the division example in Fig. 4, the program is executed as follows.

(i) RAM is cleared for remainder MD($03E ∿ $03B).

(ii) B register is used as shift counter.

(iii) CA flag is initialized to "1" to set "1" in RAM for quotient.

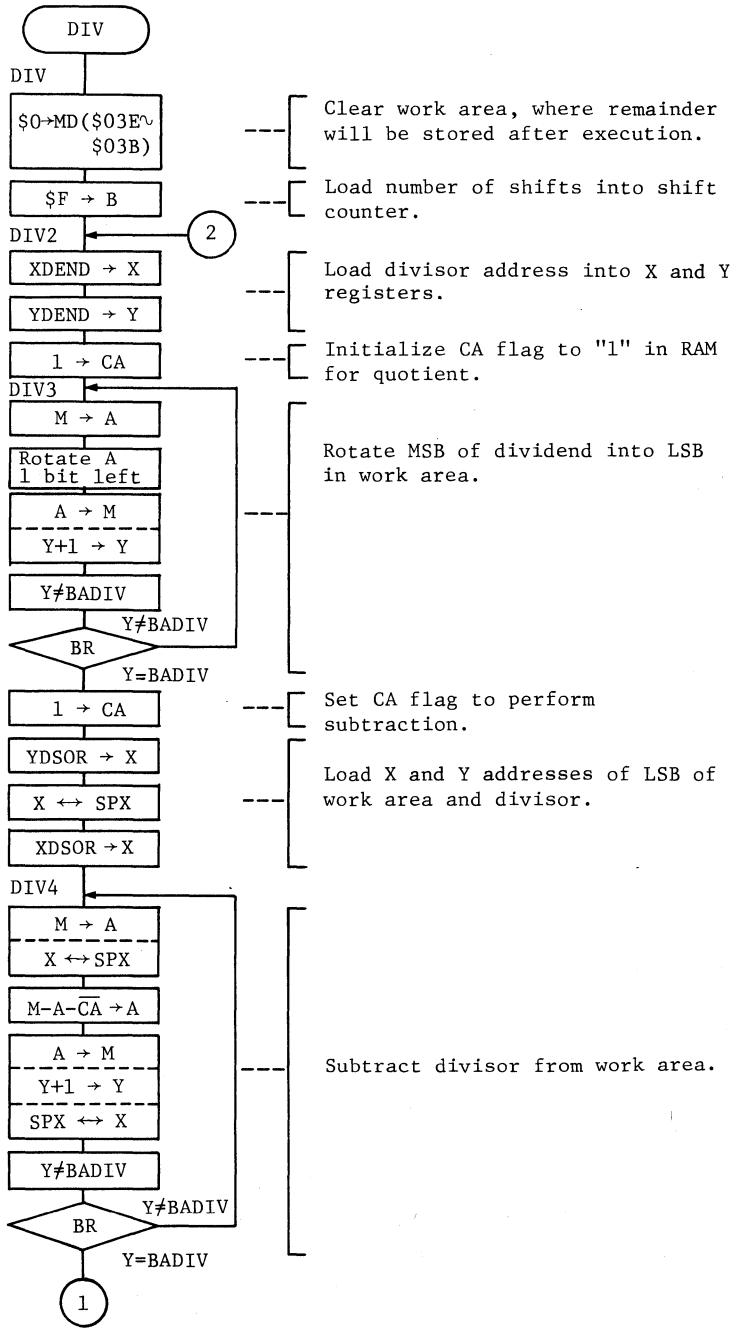
(iv) MD($03A ∿ $037) (dividend) and MD($03E ∿ $03B) are rotated 1 bit left to set "1" in RAM for quotient. At the same time MSB is rotated into LSB.

(v) This is performed because upper bits are fetched one by one from dividend to subtract divisor MD($04E ∿ $04B). Divisor MD($04E ∿ $04B) is then subtracted from MD($03E ∿ $03B). If subtraction result is positive, "1" is retained in the LSB of MD($03A ∿ $037). (Fig. 4 ①→②→③)
If subtraction result is negative, LSB of MD($03A ∿ $037) is cleared and added divisor to subtraction result. (Fig. 4 ④→⑤→⑥)

(vi) Shift counter is decremented.

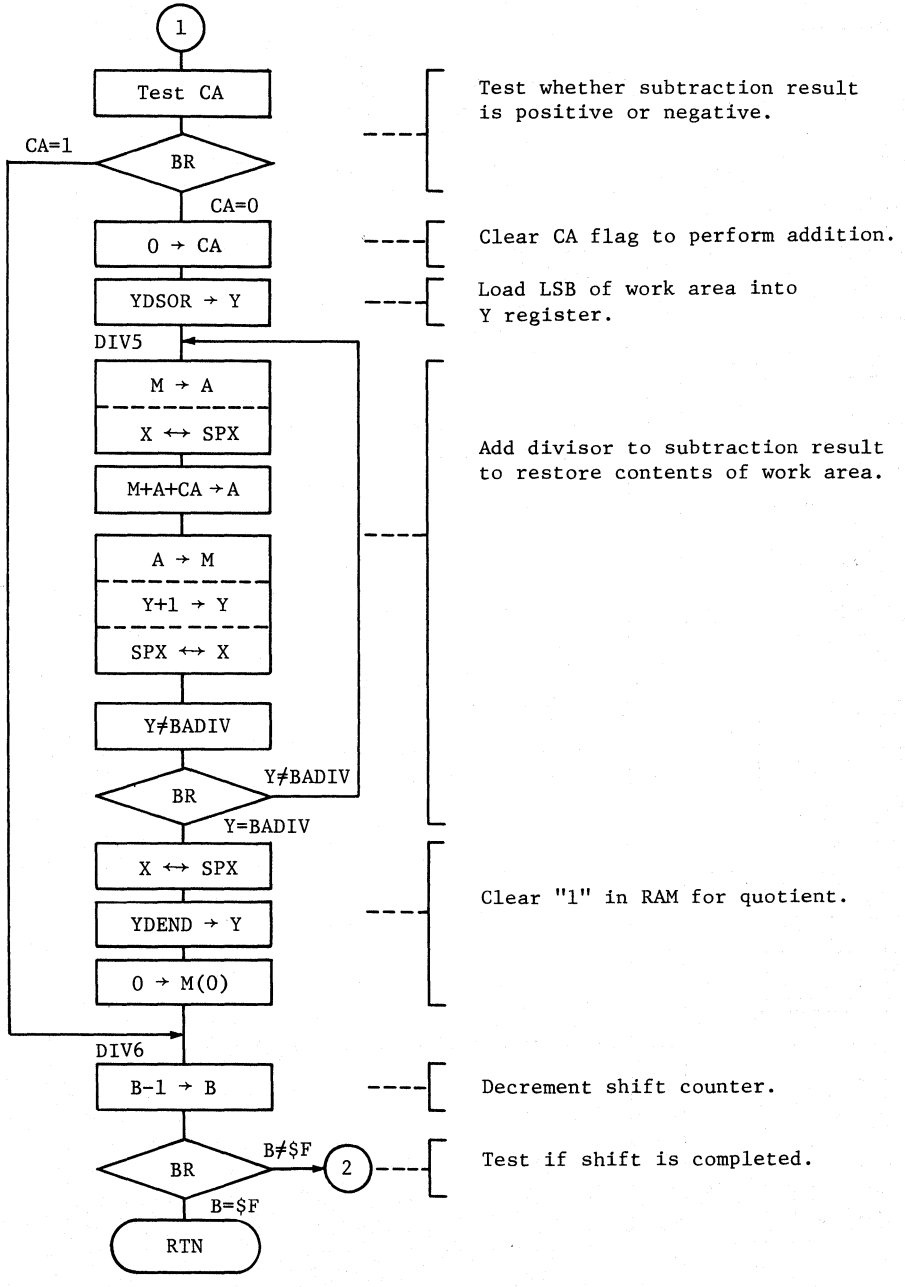
(vii) Operation loops from (iii) to (vi) until shift counter is "0".

FLOWCHART


DIV
DIV
$0→MD($03E∿$03B)
Clear work area, where remainder will be stored after execution.
$F → B
Load number of shifts into shift counter.
DIV2
2
XDEND → X
YDEND → Y
Load divisor address into X and Y registers.
1 → CA
Initialize CA flag to "1" in RAM for quotient.
DIV3
M → A
Rotate A 1 bit left
A → M
Y+1 → Y
Y≠BADIV
Rotate MSB of dividend into LSB in work area.
Y≠BADIV
BR
Y=BADIV
1 → CA
Set CA flag to perform subtraction.
YDSOR → X
X ↔ SPX
XDSOR → X
Load X and Y addresses of LSB of work area and divisor.
DIV4
M → A
X ↔ SPX
M-A-$\overline{CA}$ → A
A → M
Y+1 → Y
SPX ↔ X
Y≠BADIV
Subtract divisor from work area.
Y≠BADIV
BR
Y=BADIV
1

15. DIVIDE 16-BIT BINARY DATA	MCU	HMCS400 SERIES	LABEL	DIV

FLOWCHART



PROGRAM LISTING

```
ST-NO   OBJECT   ADRS   SOURCE STATEMENTS

00001   30A      0000           LLEN     132
00002                  ************************************************
00003                  *                                              *
00004                  *      NAME : DIVIDE 16-BIT BINARY DATA (DIV)  *
00005                  *                                              *
00006                  ************************************************
00007                  *                                              *
00008                  *      ENTRY          : MD($03A-$037)(DIVIDEND)  *
00009                  *                       MD($04E-$04B)(DIVISOR)  *
00010                  *      RETURNS        : MD($03A-$037)(QUOTIENT)  *
00011                  *                       MD($03E-$03B)(REMAINDER)*
00012                  ************************************************
00013                  *
00014                  XDEND    EQU      $3          DIVIDEND, QUOTIENT & REMAINDER ADDR(X)
00015                  XDSOR    EQU      $4          DIVISOR ADDR(X)
00016                  YDEND    EQU      $7          DIVIDEND & QUOTIENT LSD ADDR(Y)
00017                  YDSOR    EQU      $B          DIVISOR & REMAINDER LSD ADDR(Y)
00018                  BADIV    EQU      $F          DIVISOR & REMAINDER MSD ADDR(Y)+1
00019                  *
00020                           ORG      $0100
00021                  DIV      EQU      *           ENTRY POINT
00022   223      0100           LXI      XDEND       CLEAR WORK AREA
00023   21B      0101           LYI      YDSOR
00024   290      0102  DIV1     LMIIY    $0
00025   07F      0103           YNEI     BADIV
00026   302      0104           BRS      DIV1
00027   20F      0105           LBI      $F          LOAD SHIFT COUNTER
00028   223      0106  DIV2     LXI      XDEND       LOAD DIVIDEND ADDR(X)
00029   217      0107           LYI      YDEND       LOAD DIVIDEND ADDR(Y)
00030   0EF      0108           SEC                  SET CARRY FLAG
00031   090      0109  DIV3     LAM                  SHIFT DIVIDEND 1-BIT LEFT-
00032   0A1      010A           RUTL                 TO LOAD MSB OF DIVIDEND INTO LSB OF WORK AREA
00033   050      010B           LMAIY
00034   07F      010C           YNEI     BADIV
00035   309      010D           BRS      DIV3
00036   0EF      010E           SEC                  SET CARRY FLAG
00037   21B      010F           LYI      YDSOR       LOAD DIVISOR ADDR(Y)
00038   001      0110           XSPX
00039   224      0111           LXI      XDSOR       LOAD DIVISOR ADDR(X)
00040   091      0112  DIV4     LAMX                 REMAINDER - DIVISOR -> REMAINDER
00041   098      0113           SMC
00042   051      0114           LMAIYX
00043   07F      0115           YNEI     BADIV
00044   312      0116           BRS      DIV4
00045   06F      0117           TC                   TEST CARRY
00046   323      0118           BRS      DIV6        BRANCH IF REMAIN > DIVIDEND
00047   0EC      0119           REC                  CLEAR CARRY FLAG
00048   21B      011A           LYI      YDSOR       LOAD LSB OF WORK AREA INTO REG(Y)
00049   091      011B  DIV5     LAMX                 REMAINDER + DIVISOR -> REMAINDER
00050   018      011C           AMC
00051   051      011D           LMAIYX
00052   07F      011E           YNEI     BADIV
00053   31B      011F           BRS      DIV5
00054   001      0120           XSPX                 CLEAR LSB OF QUOTIENT
00055   217      0121           LYI      YDEND
00056   088      0122           REM      $0
00057   0CF      0123  DIV6     DB                   DECREMENT SHIFT COUNTER
00058   306      0124           BRS      DIV2        LOOP UNTIL SHIFT COUNTER = $F
00059   010      0125           RTN
```

When storing arguments in other RAM locations, change the EQU operands for the following labels.

XDEND: Defines X address of dividend, quotient and remainder.

XDSOR: Defines X address of divisor.

YDEND: Defines LSD Y address of dividend and quotient.

YDSOR: Defines LSD Y address of divisor and remainder. (YDEND + $4)

BADIV: Defines MSD Y address plus 1 of divisor and remainder. (YDEND + $8; if this is overflow, value should be defined as $0.)

16. ADD 8-DIGIT BCD	MCU	HMCS400 SERIES	LABEL	ADDD

FUNCTION

Performs addition of 8-digit BCD data in RAM, and stores result as 8-digit BCD data in RAM; uses unsigned integers as arguments.

ARGUMENTS

1 digit= 4 bits

Contents		Storage Location	No. of Digits
Entry	Augend	MD($03D ∿ $036)	8
	Addend	MD($04D ∿ $046)	8
Returns	Addition result	MD($03D ∿ $036)	8
	Carry or no carry	CA	(1 bit)

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected
× : Undefined
‡ : Result

A	B
×	●
X	Y
×	×
SPX	SPY
×	●
W	
●	

CA	ST
‡	‡

SPECIFICATIONS

1 word=10 bits

ROM (Words)
12
RAM (Digits)
16
Stack (Digits)
0
No. of cycles
55
Reentrant
No
Relocatable
No
Interrupt OK?
Yes

DESCRIPTION

(1) Function Details

(a) Argument details

MD($03D∿$036): Holds 8-bit BCD augend. After ADDD execution, contains addition result.

MD($04D∿$046): Holds 8-bit BCD addend.

CA : Indicates whether a carry is generated or not after ADDD execution.

CA=1 : Indicates a carry is generated in addition result. (see Fig. 2)

CA=0 : Indicates no carry is generated in addition result.

SPECIFICATIONS NOTES

N/A

(b) Example of ADDD execution is shown in Fig. 1.

If entry arguments are as shown in part ① of Fig. 1, addition result is contained in MD($03D∿$036) as shown in part ② of Fig. 1.

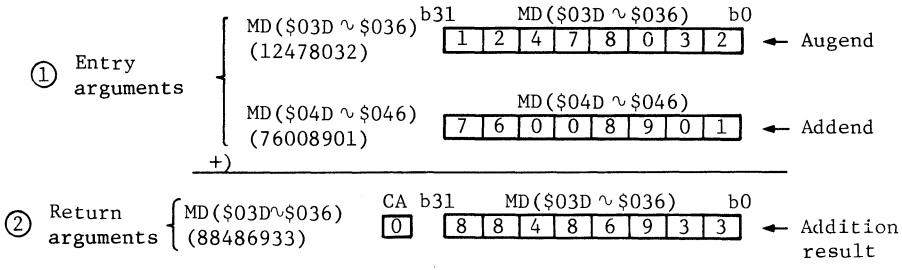


Fig. 1 Example of ADDD Execution

(2) User Notes

(a) ST flag is set after ADDD execution.

(b) When upper digits are not needed, "0" must be stored in upper digits as shown in Fig. 3. Otherwise correct addition result cannot be obtained since addition is performed with undefined data in upper digits.

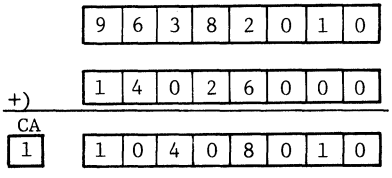


Fig. 2 Addition Example When Carry is Generated

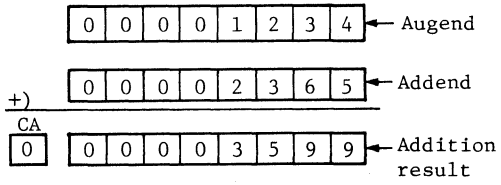


Fig. 3 Addition Example When Upper Digits are Not Needed

(c) After ADDD execution, augend is destroyed since addition result is contained in MD($03D ∿ $036). If augend needs to be retained after ADDD execution, it should be saved in memory before execution.

(d) BCD number must be stored in augend and addend, otherwise correct addition result cannot be obtained.

DESCRIPTION

(3) RAM Allocation

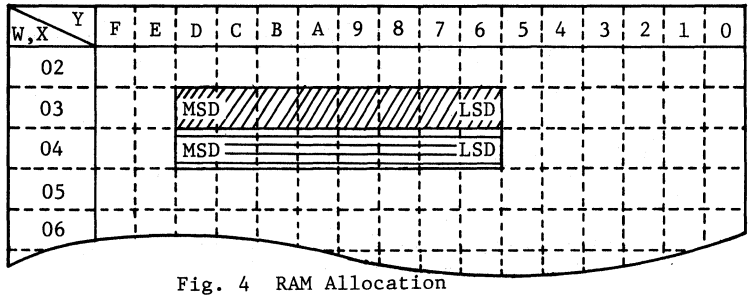


Fig. 4 RAM Allocation

Label	RAM	Description
—	b31 … b0 MSD … LSD MD($03D ∿ $036)	8-digit BCD augend before execution. 8-digit BCD addition result after execution. X and Y addresses are defined by XAUGE and YAUGE, respectively.
—	b31 … b0 MSD … LSD MD($04D ∿ $046)	8-digit BCD addend X and Y addresses are defined by XADDE and YAUGE, respectively.

(4) Sample Application

Shown below is a sample application using ADDD with address space allocated as follows.

MD($0AD ∿ $0A6) : Augend. After execution, addition result is containd.

MD($0B0 ∿ $0B6) : Addend

16. ADD 8-DIGIT BCD	MCU	HMCS400 SERIES	LABEL	**ADDD**

DESCRIPTION

```
        LWI     $0      ------ Example with W=0.
        :
        LXI     $3
        XSPX
        LXI     $A
        LYI     $6      ------- Store 8-digit BCD augend in entry
WORK1   LAMX                    argument.
        LMAIYX
        YNEI    $E
        BRS     WORK1
        LXI     $4
        XSPX
        LXI     $B
        LYI     $6      ------- Store 8-digit BCD addend in entry
WORK2   LAMX                    argument.
        LMAIYX
        YNEI    $E
        BRS     WORK2

        CALL    ADDD    ------ Call ADDD.
        TC
        BRS     OVER    ------- If a carry is generated in addition
                                result, branch to service routine.
        LXI     $A
        XSPX
        LXI     $3
        LYI     $6      ------- Store addition result, contained in
WORK3   LAMX                    return argument, in RAM.
        LMAIYX
        YNEI    $E
        BRS     WORK3
        :
OVER
        Service routine
        in case of carry
        :
```

DESCRIPTION

(5) Basic Operation

(a) In HMCS400 series, when addition is performed with 2 or more digits, the same operation sequence is repeated for each digit.

(b) X and Y registers are used as a pointer to augend and addend.

(c) CA flag is first cleared. Formula 1 is performed on each digit of augend and addend using register indirect addressing mode.

Augend + Addend + (CA) → Accumulator ..... (Formula 1)

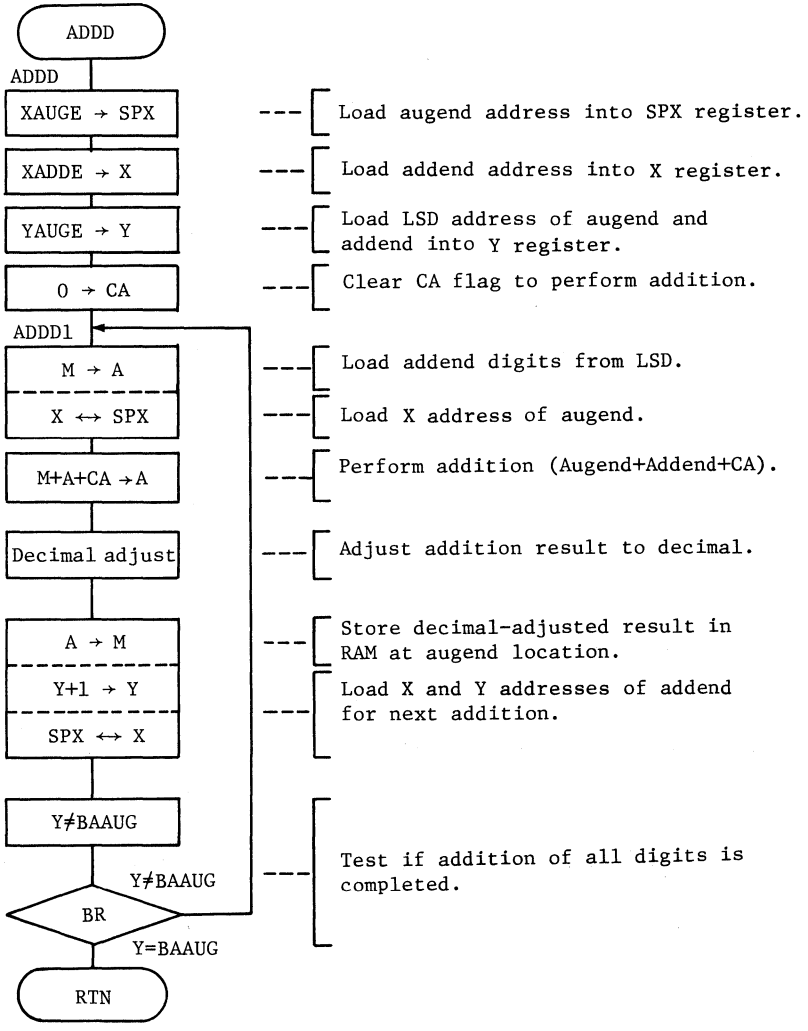
CA flag is added in Formula 1 since digits previously added occasionally generate a carry.

(d) Addition result calculated in (c) is adjusted to a decimal value using the decimal adjust instruction (DAA). The result is then stored in RAM in the augend location.

(e) Y register is incremented every time (c) and (d) is executed.

(f) Operation loops from (c) to (e) until 8-digit addition is completed.

FLOWCHART


ADDD
ADDD
XAUGE → SPX
XADDE → X
YAUGE → Y
0 → CA
ADDD1
M → A
X ↔ SPX
M+A+CA → A
Decimal adjust
A → M
Y+1 → Y
SPX ↔ X
Y≠BAAUG
Y≠BAAUG
BR
Y=BAAUG
RTN
Load augend address into SPX register.
Load addend address into X register.
Load LSD address of augend and addend into Y register.
Clear CA flag to perform addition.
Load addend digits from LSD.
Load X address of augend.
Perform addition (Augend+Addend+CA).
Adjust addition result to decimal.
Store decimal-adjusted result in RAM at augend location.
Load X and Y addresses of addend for next addition.
Test if addition of all digits is completed.

PROGRAM LISTING

```
ST-NO   OBJECT   ADRS   SOURCE STATEMENTS

00001   010      0000           LLEN     132
00002                  ********************************************************
00003                  *                                                      *
00004                  *      NAME : ADD 8-DIGIT BCD (ADDD)                   *
00005                  *                                                      *
00006                  ********************************************************
00007                  *                                                      *
00008                  *     ENTRY  :MD($03D-$036)(AUGEND)                    *
00009                  *             MD($04D-$046)(ADDEND)                    *
00010                  *     RETURNS:MD($03D-$036)(ADDITION RESULT)           *
00011                  *             CA FLAG(CA=0:NORMAL RETURN.CA=1:OVERFLOW)*
00012                  *                                                      *
00013                  ********************************************************
00014                  *
00015                  XAUGE    EQU      $3            AUGEND ADDR(X)
00016                  XADDE    EQU      $4            ADDEND ADDR(X)
00017                  YAUGE    EQU      $6            ADDEND & AUGEND LSD ADDR(Y)
00018                  BAAUG    EQU      $E            ADDEND & AUGEND MSD ADDR(Y)+1
00019                  *
00020                           ORG      $0100
00021                  ADDD     EQU      *             ENTRY POINT
00022   223      0100           LXI      XAUGE         LOAD AUGEND ADDR(SPX)
00023   001      0101           XSPX
00024   224      0102           LXI      XADDE         LOAD ADDEND ADDR(X)
00025   216      0103           LYI      YAUGE         LOAD AUGEND AND ADDEND ADDR(Y)
00026   0EC      0104           REC                    CLEAR CARRY FLAG
00027   091      0105   ADDD1   LAMX                   LOAD ADDEND DATA A
00028   018      0106           AMC                    M+A+CA->A
00029   0A6      0107           DAA                    ADJUST RESULT INTO DECIMAL
00030   051      0108           LMAIYX                 STORE RESULT IN AUGEND
00031   07E      0109           YNEI     BAAUG         TEST IF ADDITION IS COMPLETED
00032   305      010A           BRS      ADDD1         LOOP UNTIL REG(Y) = BAAUG
00033   010      010B           RTN
```

When storing arguments in other RAM locations, change the EQU operands for the following labels.

XAUGE: Defines X address of augend.

XADDD: Defines X address of addend.

YAUGE: Defines LSD Y address of augend and addend.

BAAUG: Defines MSD Y address plus 1 of augend and addend. (YAUGE + $8; if this is overflow, value should be defined as $0.)

17. SUBTRACT 8-DIGIT BCD

MCU | HMCS400 SERIES | LABEL | SUBD

FUNCTION

Performs subtraction of 8-digit BCD data in RAM, and stores result in RAM; uses unsigned integers as arguments.

ARGUMENTS

1 digit= 4 bits

Contents		Storage Location	No. of Digits
Entry	Minuend	MD($03D ∿ $036)	8
	Subtrahend	MD($04D ∿ $046)	8
Returns	Subtraction result	MD($03D ∿ $036)	8
	Borrow or no borrow	CA	(1 bit)

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected
× : Undefined
↕ : Result

A	B
×	●

X	Y
×	×

SPX	SPY
×	●

W
●

CA	ST
↕	↕

SPECIFICATIONS

1 word=10 bits

ROM (Words)
12
RAM (Digits)
16
Stack (Digits)
0
No. of cycles
55
Reentrant
No
Relocatable
No
Interrupt OK?
Yes

DESCRIPTION

(1) Function Details

(a) Argument details

MD($03D∿$036): Holds 8-digit BCD minuend.
After SUBD execution, contains subtraction result.

MD($04D∿$046): Holds 8-digit BCD subtrahend.

CA : Indicates whether a borrow is generated or not after SUBD execution.

CA=1 : Indicates no borrow is generated in subtraction result.

CA=0 : Indicates a borrow is generated in subtraction result. (See Fig. 2)

SPECIFICATIONS NOTES

N/A

DESCRIPTION

(b) Example of SUBD execution is shown in Fig. 1.
If entry arguments are as shown in part ① of Fig. 1.
Subtraction result is contained in MD($03D ∿ $036) as shown in part ② of Fig. 1.

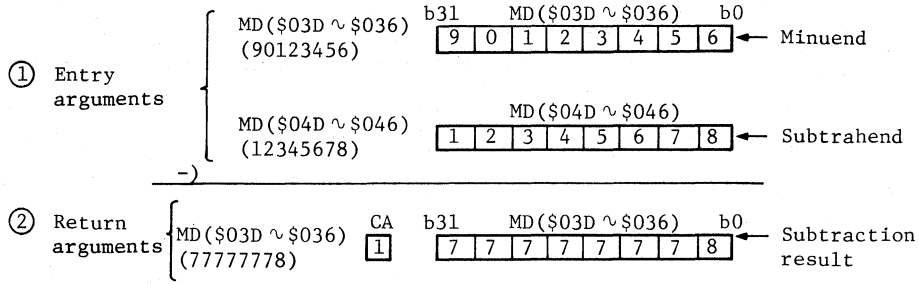


Fig. 1 Example of SUBD Execution

(2) User Notes

(a) ST flag is set after SUBD execution.

(b) When upper digits are not needed, "0" must be stored in upper digits as shown in Fig. 3. Otherwise correct subtraction result cannot be obtained since subtraction is performed with undefined data in upper digits.

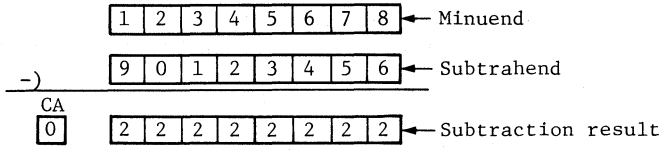


Fig. 2 Subtraction Example When Borrow is Generated

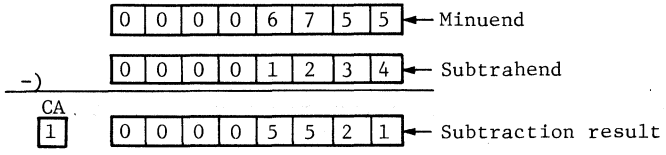


Fig. 3 Subtraction Example When Upper Digits are Not Needed

DESCRIPTION

(c) After SUBD execution, minuend is destroyed since subtraction result is contained in MD($03D ∿ $036). If minuend needs to be retained after SUBD execution, it should be saved in memory before execution.

(d) BCD number must be stored in minuend and subtrahend, otherwise correct subtraction result cannot be obtained.

(3) RAM Allocation

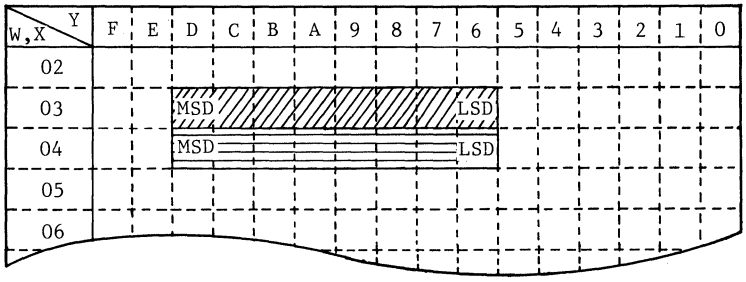


Fig. 4 RAM Allocation

Label	RAM	Description
—	b31 b0 MSD LSD MD($03D ∿ $036)	8-digit binary subtrahend before execution. 8-digit binary subtraction result after execution. X and Y addresses are defined by XSUBT and YSUBT, respectively.
—	b31 b0 MSD LSD MD($04D ∿ $046)	8-digit binary minuend. X and Y addresses are defined by XMINU and YSUBT, respectively.

(4) Sample Application

Shown below is a sample application using SUBD with address space allocated as follows.

```
MD($0AD ∿ $0A6) : Minuend
                  After execution, subtraction result is contained.

MD($0BD ∿ $0B6) : Subtrahend
```

17. SUBTRACT 8-DIGIT BCD	MCU	HMCS400 SERIES	LABEL	SUBD

DESCRIPTION

```
        LWI     $0        ---- Example with W=0.
         ¦
        LXI     $3
        XSPX
        LXI     $A
        LYI     $6        -----+ Store 8-digit BCD minuend in entry
WORK1   LAMX                   | argument.
        LMAIYX                 |
        YNEI    $E             |
        BRS     WORK1     -----+
        LXI     $4
        XSPX
        LXI     $B
        LYI     $6        -----+ Store 8-digit BCD subtrahend in entry
WORK2   LAMX                   | argument.
        LMAIYX                 |
        YNEI    $E             |
        BRS     WORK2     -----+

        CALL    SUBD      ---- Call SUBD.

        TC                     +
        BRS     WORK3     -----+ If borrow is generated in subtraction
        BRS     BRROW          | result, branch to service routine.
WORK3   LXI     $A             |
        XSPX                   |
        LXI     $3             |
        LYI     $6        -----+ Store subtraction result, contained in
WORK4   LAMX                   | return argument, in RAM.
        LMAIYX                 |
        YNEI    $E             |
        BRS     WORK4     -----+
         ¦
BRROW   Service routine
        in case of borrow
         ¦
```

DESCRIPTION

(5) Basic Operation

(a) In HMCS400 series, when subtraction is performed with 2 or more digits, the same operation sequence is repeated for each digit.

(b) X and Y registers are used as a pointer to minuend and subtrahend.

(c) CA flag is first set. Formula 1 is performed for each digit of minuend and subtrahend using register indirect addressing mode.

Minuend - Subtrahend - $(\overline{CA})$ → Accumulator ..... (Formula 1)

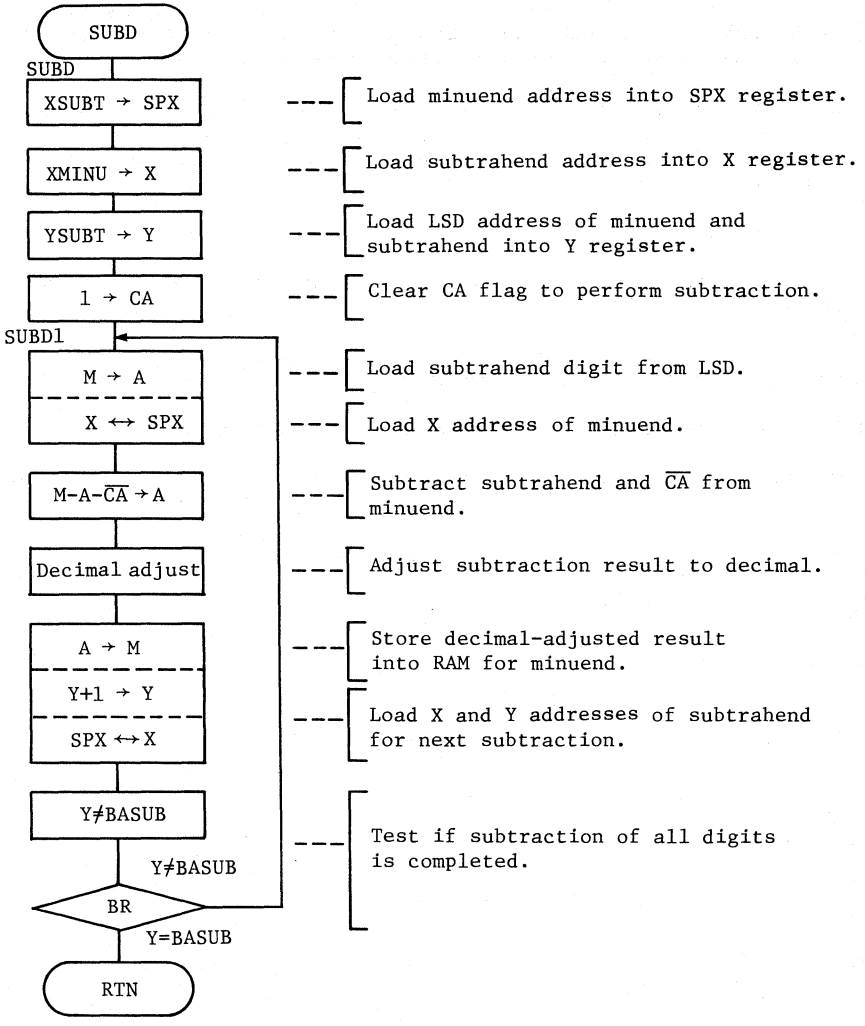
CA flag is subtracted in Formula 1 since subtraction result occasionally generates a borrow.

(d) Subtraction result, calculated in (c), is adjusted to a decimal value using the decimal adjust instruction (DAS).

(e) Y register is incremented every time (c) and (a) are executed.

(f) Operation loops from (c) to (e) until 8-digit subtraction is completed.

FLOWCHART


SUBD
SUBD
XSUBT → SPX
XMINU → X
YSUBT → Y
1 → CA
SUBD1
M → A
X ↔ SPX
M-A-$\overline{CA}$ → A
Decimal adjust
A → M
Y+1 → Y
SPX ↔ X
Y≠BASUB
Y≠BASUB
BR
Y=BASUB
RTN
Load minuend address into SPX register.
Load subtrahend address into X register.
Load LSD address of minuend and subtrahend into Y register.
Clear CA flag to perform subtraction.
Load subtrahend digit from LSD.
Load X address of minuend.
Subtract subtrahend and $\overline{CA}$ from minuend.
Adjust subtraction result to decimal.
Store decimal-adjusted result into RAM for minuend.
Load X and Y addresses of subtrahend for next subtraction.
Test if subtraction of all digits is completed.

PROGRAM LISTING

```
ST-NO   OBJECT   ADRS   SOURCE STATEMENTS

00001   010      0000           LLEN     132
00002                  ********************************************************
00003                  *                                                      *
00004                  *       NAME : SUBTRACT 8-DIGIT BCD (SUBD)             *
00005                  *                                                      *
00006                  ********************************************************
00007                  *                                                      *
00008                  *       ENTRY  : MD($03D-$036)(MINUEND)                *
00009                  *                MD($04D-$046)(SUBTRAHEND)             *
00010                  *       RETURNS: MD($03D-$036)(SUBTRACTION RESULT)     *
00011                  *                CA FLAG(CA=1:NORMAL RETURN,CA=0:BORROW)*
00012                  *                                                      *
00013                  ********************************************************
00014                  *
00015                  XSUBT    EQU      $3            MINUEND ADDR(X)
00016                  XMINU    EQU      $4            SUBTRAHEND ADDR(X)
00017                  YSUBT    EQU      $6            MINUEND & SUBTRAHEND LSD ADDR(Y)
00018                  BASUB    EQU      $E            MINUEND & SUBTRAHEND MSD ADDR(Y)+1
00019                  *
00020                           ORG      $0100
00021                  SUBD     EQU      *             ENTRY POINT
00022   223      0100           LXI      XSUBT         LOAD MINUEND ADDR(SPX)
00023   001      0101           XSPX
00024   224      0102           LXI      XMINU         LOAD SUBTRAHEND ADDR(X)
00025   216      0103           LYI      YSUBT         LOAD MINUEND & SUBTRAHEND ADDR(Y)
00026   0EF      0104           SEC                    SET CARRY FLAG
00027   091      0105   SUBD1   LAMX                   LOAD SUBTRAHEND DATA
00028   098      0106           SMC                    M-A-INV(CA)->A
00029   0AA      0107           DAS                    ADJUST RESULT INTO DECIMAL
00030   051      0108           LMAIYX                 STORE RESULT IN MINUEND
00031   07E      0109           YNEI     BASUB         TEST IF SUBTRACTION IS COMPLETED
00032   305      010A           BRS      SUBD1         LOOP UNTIL Y = BASUB
00033   010      010B           RTN
```

When storing arguments in other RAM locations, change the EQU operands for the following labels.

XSUBT: Defines X address of minuend.

XMINU: Defines X address of subtrahend.

YSUBT: Defines LSD Y address of minuend and subtrahend.

BASUB: Defines MSD Y address plus 1 of minuend and subtrahend. (YSUBT + $8; if this is overflow, value should be defined as $0.)

FUNCTION

Obtains square root of 16-bit binary data in RAM, and store result in RAM; uses unsigned integers as arguments.

ARGUMENTS

1 digit= 4 bits

Contents		Storage Location	No. of Digits
Entry	Number to take square root of	MD($03A ∿ $037)	4
Returns	Square root	MD($04C, $04B)	2

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected
× : Undefined
‡ : Result

A	B
×	×
X	Y
×	×
SPX	SPY
×	●
W	
●	

CA	ST
×	×

SPECIFICATIONS

1 word=10 bits

ROM (Words)
67
RAM (Digits)
12
Stack (Digits)
4
No. of cycles
1492
Reentrant
No
Relocatable
No
Interrupt OK?
Yes

DESCRIPTION

(1) Function Details

(a) Argument details

MD($03A∿$037): Holds 16-bit binary number to take square root of.

MD($04C, $04B): Contains 8-bit binary square root.

(b) Example of SQRT execution is shown in Fig. 1. If entry argument is as shown in part ① of Fig. 1, square root is obtained in MD($04C, $04B) as shown in part ② of Fig. 1.

SPECIFICATIONS NOTES

"No. of cycles" in "SPECIFICATIONS" indicates the number of cycles required to calculate the square root of $FFFF.

DESCRIPTION

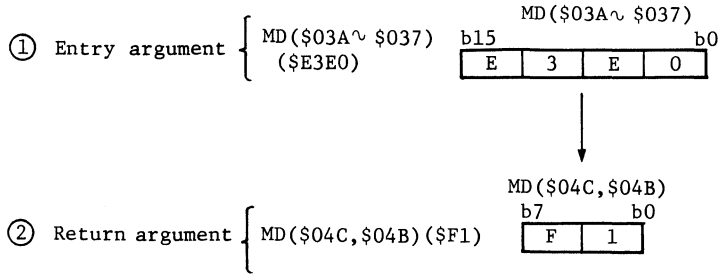


Fig. 1 Example of SQRT Execution

(2) User Notes

(a) ST flag is set after SQRT execution.

(b) When upper digits are not needed, "0" must be stored in upper digits as shown in Fig. 2. Otherwise correct square root cannot be obtained since calculation is performed with undefined data in upper digits.

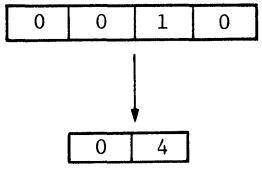


Fig. 2 Example When Upper Digits are Not Used

(c) Values to the right of the binary point are truncated.

DESCRIPTION

(3) RAM Allocation

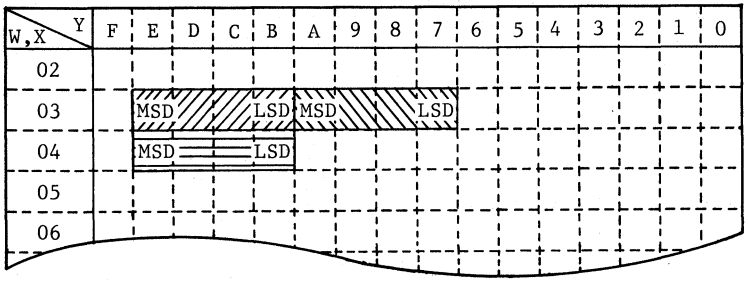


Fig. 3 RAM Allocation

Label	RAM	Description
—	b15 b0 MSD LSD MD($03E ∿ $03B)	Work area for operation on upper 2 bits of data to take square root of. X and Y addresses are defined by XBINA and YSQUR, respectively.
—	b15 b0 MSD LSD MD($03A ∿ $037)	16-bit binary number to take square root of. X and Y addresses are defined by XBINA and YBINA, respectively.
—	b15 b0 MSD LSD MD($04E ∿ $04B)	Work area for square root before execution. Contains 8-bit binary square root in MD($04C, $04B), and "0" in MD($04E, $04D) after execution. X and Y addresses are defined by XSQUA and YSQUR, respectively.

DESCRIPTION

(4) Sample Application

Shown below is a sample application using SQRT with address space allocated as follows.

```
MD($0AA ~ $0A7) : Number to take square root of

MD($0AC, $0AB) : Square root
```

```
        LWI     $0      ----- Example with W=0.
         :
         :
        LXI     $3
        XSPX
        LXI     $A
        LYI     $7      ------- Store 16-bit binary number to be squared
WORK1   LAMX                    in entry argument.
        LMAIYX
        YNEI    $B
        BR      WORK1

        CALL    SQAT    ----- Call SQRT.

        LXI     $A
        XSPX
        LXI     $4
        LYI     $B      ------ Store 8-bit binary square root,
WORK2   LAMX                   contained in return argument, in RAM.
        LMAIYX
        YNEI    $D
        BRS     WORK2
         :
         :
         :
```

(5) Basic Operation

(a) Fig. 4 shows calculation to obtain a square root. In this example, data to take square root of is $22 and square root is $5.

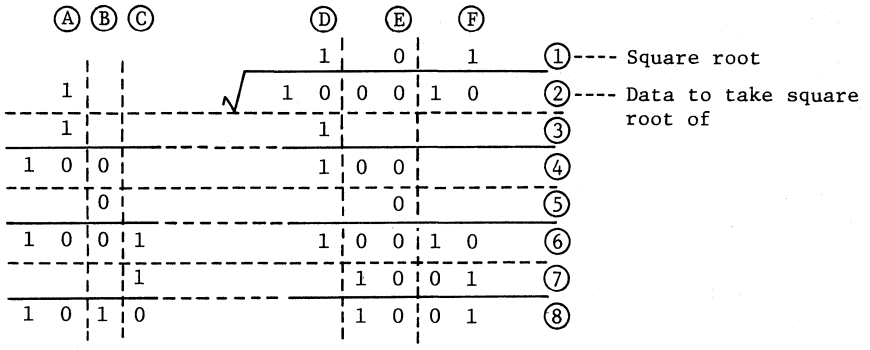


Fig. 4 Calculating a Square Root

(b) Referring to the calculation in Fig. 4, the program is executed as follows.

(i) Square root area MD($04E∿$04B) and work area MD($03E∿$03B) is cleared.

(ii) MD($03A∿$037) and MD($03E∿$03B) is rotated 2 bits left to fetch the upper 2 bits of data to take square root of, this is then loaded into work area MD($03E∿$03B) (Fig. 4 Ⓓ - ②).

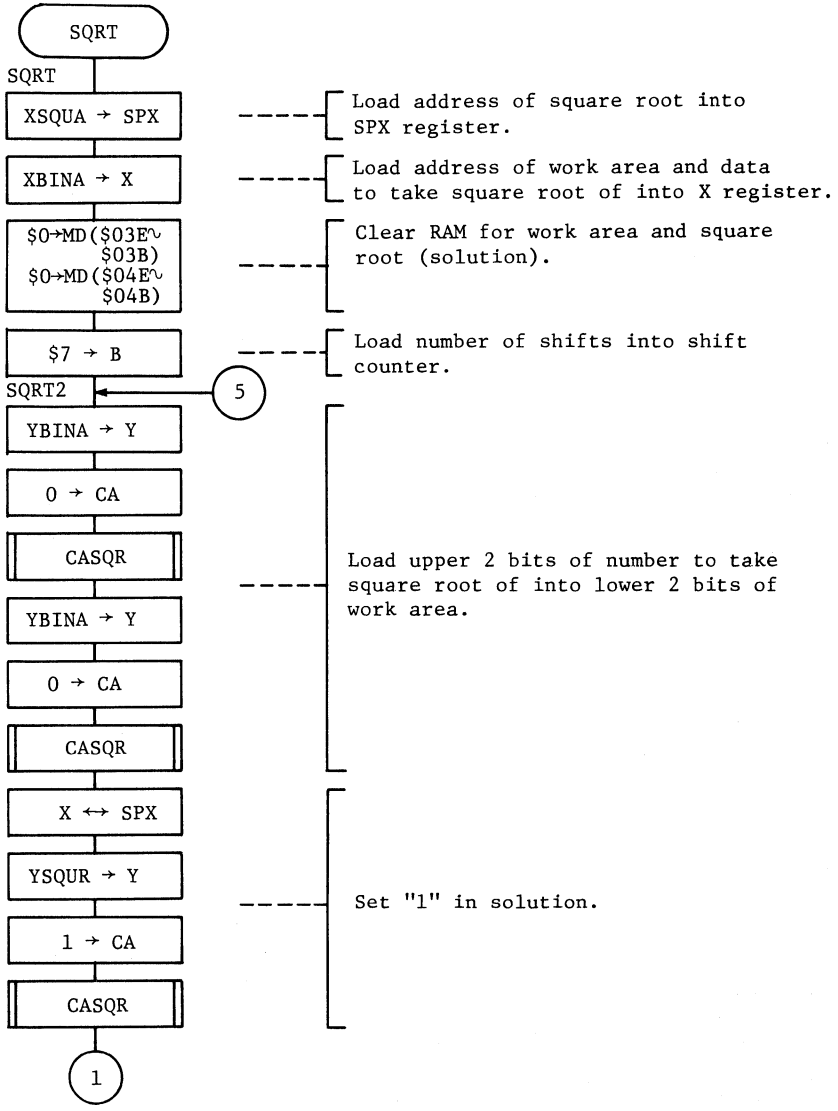
(iii) Square root area is shifted 1 bit left, and "1" is set in LSB (Fig. 4 Ⓐ-②).

(iv) Square root area is subtracted from work area and result stored in work area (Fig. 4 Ⓓ-②, ③, ④).

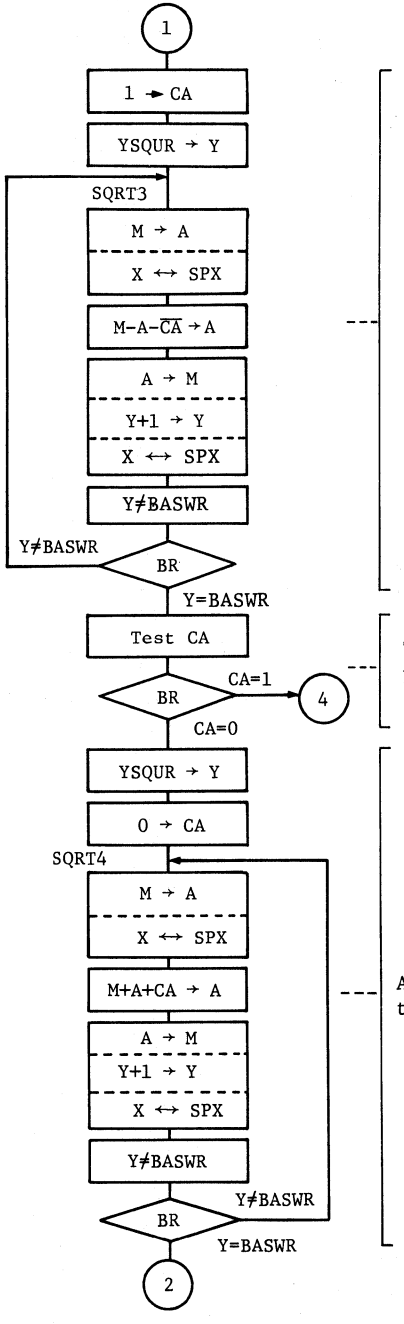
(v) If result is positive, square root area is incremented (Fig. 4 Ⓐ-④). If result is negative, work area is added to square root area, result is stored in square root area, result is stored in square root area and clears "1" setting in (iii) (Fig. 4 Ⓓ, Ⓔ-⑥).

(c) Operation loops from (ii) to (v) 8 times and then square root area is shifted 1 bit right (Fig. 4 Ⓐ, Ⓑ-⑧ is square root).

FLOWCHART


SQRT
SQRT
XSQUA → SPX
XBINA → X
$0→MD($03E∿$03B)
$0→MD($04E∿$04B)
$7 → B
SQRT2
5
YBINA → Y
0 → CA
CASQR
YBINA → Y
0 → CA
CASQR
X ↔ SPX
YSQUR → Y
1 → CA
CASQR
1
Load address of square root into SPX register.
Load address of work area and data to take square root of into X register.
Clear RAM for work area and square root (solution).
Load number of shifts into shift counter.
Load upper 2 bits of number to take square root of into lower 2 bits of work area.
Set "1" in solution.

FLOWCHART

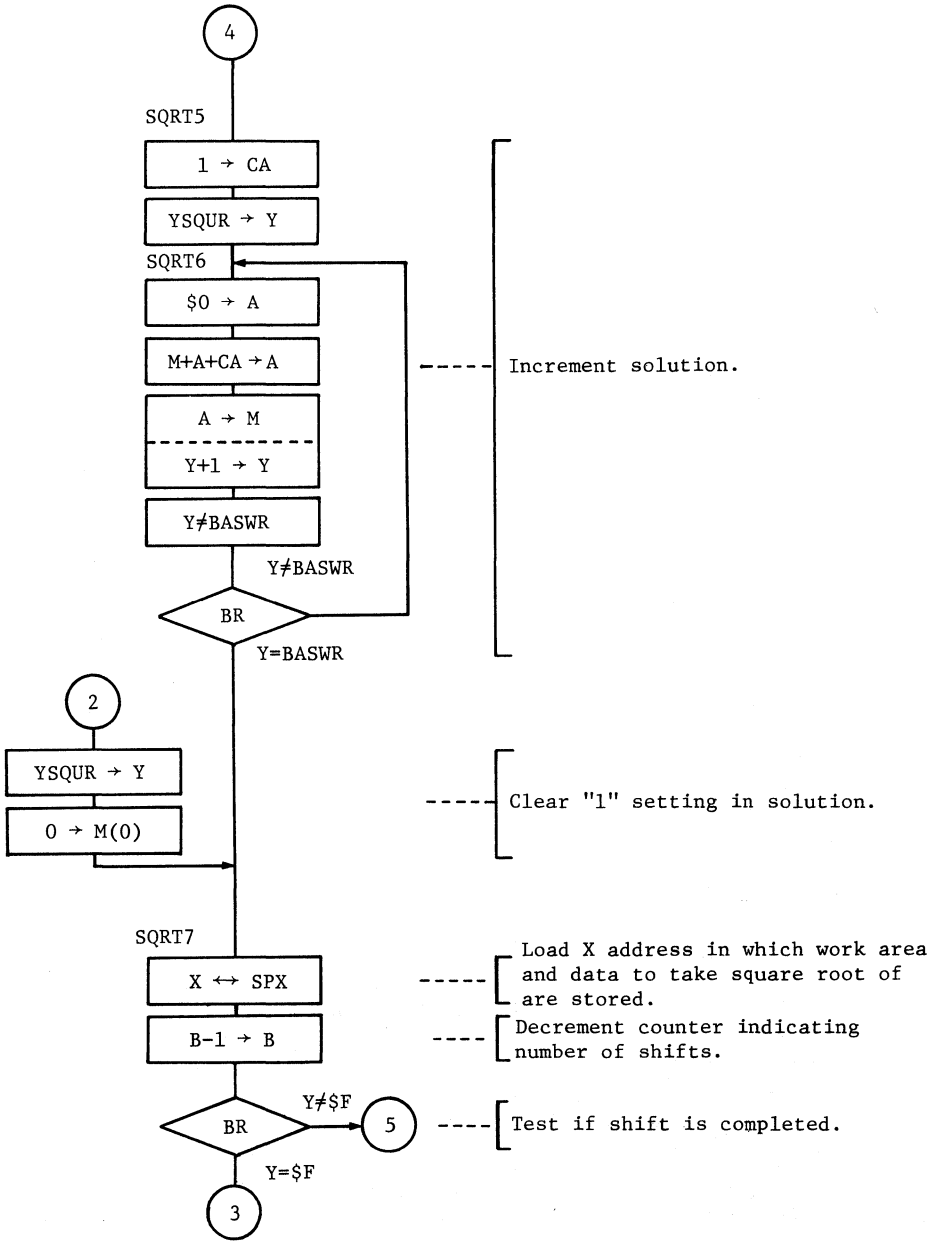


Subtract solution from work area.

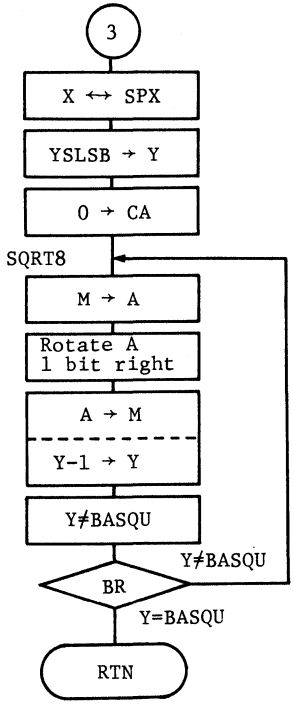
Test whether subtraction result is positive or negative.

Add solution to subtraction result to return to state before subtraction.

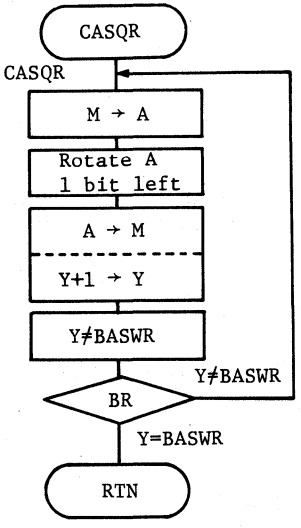
FLOWCHART


4
SQRT5
1 → CA
YSQUR → Y
SQRT6
$0 → A
M+A+CA → A
A → M
Y+1 → Y
Y≠BASWR
Y≠BASWR
BR
Y=BASWR
Increment solution.
2
YSQUR → Y
0 → M(0)
Clear "1" setting in solution.
SQRT7
X ↔ SPX
Load X address in which work area and data to take square root of are stored.
B-1 → B
Decrement counter indicating number of shifts.
Y≠$F
BR
5
Test if shift is completed.
Y=$F
3

FLOWCHART



Shift solution right to adjust digits.



Rotate RAM specified by X and Y addresses 1 bit left.

PROGRAM LISTING

```
ST-NO   OBJECT   ADRS   SOURCE STATEMENTS

00001   010      0000           LLEN     132
00002                  ***************************************************
00003                  *                                                 *
00004                  *       NAME : 16-BIT SQUARE ROOT (SQRT)          *
00005                  *                                                 *
00006                  ***************************************************
00007                  *                                                 *
00008                  *      ENTRY : MD($03A-$037) (DATA TO BE SQUARED)   *
00009                  *    RETURNS : MD($04C.$048) (SQUARE ROOT)          *
00010                  *                                                 *
00011                  ***************************************************
00012                  *
00013                  XBINA    EQU      $3          DATA TO BE SQUARED & WORK AREA ADDR(X)
00014                  XSQUA    EQU      $4          SQUARE ROOT ADDR(X)
00015                  YSQUR    EQU      $B          SQUARE ROOT LSD ADDR(Y)
00016                  YBINA    EQU      $7          DATA TO BE SQUARED LSD ADDR(Y)
00017                  YSLSB    EQU      $E          SQUARE ROOT & WORK AREA MSD ADDR(Y)
00018                  BASWR    EQU      $F          SQUARE ROOT & WORK AREA MSD ADDR(Y)+1
00019                  BASQU    EQU      $A          DATA TO BE SQUARED MSD ADDR(Y)
00020                  *
00021                           ORG      $0100
00022                  SQRT     EQU      *           ENTRY POINT
00023   224      0100           LXI      XSQUA       LOAD SQUARE ROOT ADDR(X) INTO REG(SPX)
00024   001      0101           XSPX
00025   223      0102           LXI      XBINA       LOAD DATA TO BE SQUARED ADDR INTO REG(X)
00026   21B      0103           LYI      YSQUR
00027   230      0104           LAI      $0          CLEAR SQUARE ROOT AREA & WORK AREA
00028   095      0105   SQRT1   LMAX
00029   051      0106           LMAIYX
00030   07F      0107           YNEI     BASWR
00031   305      0108           BRS      SQRT1
00032   207      0109           LBI      $7          LOAD SHIFT COUNTER
00033   217      010A   SQRT2   LYI      YBINA       LOAD UPPER 2-BIT OF DATA TO BE SQUARED-
00034   0EC      010B           REC                  INTO LOWER 2-BIT OF WORK AREA
00035   160 13D  010C           CALL     CASQR
00036   217      010E           LYI      YBINA
00037   0EC      010F           REC
00038   160 13D  0110           CALL     CASQR
00039   001      0112           XSPX                 SET LSB OF SQUARE ROOT
00040   21B      0113           LYI      YSQUR
00041   0EF      0114           SEC
00042   160 13D  0115           CALL     CASQR
00043   0EF      0117           SEC                  WORK AREA - SQUARE ROOT -> WORK AREA
00044   21B      0118           LYI      YSQUR
00045   091      0119   SQRT3   LAMX
00046   098      011A           SMC
00047   051      011B           LMAIYX
00048   07F      011C           YNEI     BASWR       REG(Y) =/ BASWR ?
00049   319      011D           BRS      SQRT3       BRANCH IF REG(Y) =/ BASWR
00050   06F      011E           TC                   TEST CARRY
00051   32A      011F           BRS      SQRT5       BRANCH IF WORK AREA < SQUARE ROOT
00052   21B      0120           LYI      YSQUR       WORK AREA + SQUARE ROOT -> WORK AREA
00053   0EC      0121           REC
00054   091      0122   SQRT4   LAMX
00055   018      0123           AMC
00056   051      0124           LMAIYX
00057   07F      0125           YNEI     BASWR
00058   322      0126           BRS      SQRT4
00059   21B      0127           LYI      YSQUR       CLEAR LSB OF SQUARE ROOT
00060   088      0128           REM      $0
00061   331      0129           BRS      SQRT7
00062   0EF      012A   SQRT5   SEC                  SQUARE ROOT + $1 -> SQUARE ROOT
00063   21B      012B           LYI      YSQUR
00064   230      012C   SQRT6   LAI      $0
00065   018      012D           AMC
00066   050      012E           LMAIY
00067   07F      012F           YNEI     BASWR
00068   32C      0130           BRS      SQRT6
00069   001      0131   SQRT7   XSPX                 DECREMENT SHIFT COUNTER
00070   0CF      0132           DB
00071   30A      0133           BRS      SQRT2       LOOP UNTIL SHIFT COUNTER = $F
00072   001      0134           XSPX                 ADJUST DIGITS
00073   21E      0135           LYI      YSLSB
00074   0EC      0136           REC
00075   090      0137   SQRT8   LAM
00076   0A0      0138           ROTR
00077   0D0      0139           LMADY
00078   07A      013A           YNEI     BASQU
00079   337      013B           BRS      SQRT8
00080   010      013C           RTN
00081   090      013D   CASQR   LAM                  ROTATE MEMORY  1-BIT LEFT
00082   0A1      013E           ROTL
00083   050      013F           LMAIY
00084   07F      0140           YNEI     BASWR
00085   33D      0141           BRS      CASQR
00086   010      0142           RTN
```

PROGRAM LISTING

When storing arguments in other RAM locations, change the EQU operands for the following labels.

XBINA: Defines X address of number to take square root of.

XSQUA: Defines X address of square root.

YSQUR: Defines LSD Y address of square root.

YBINA: Defines LSD Y address of number to take square root of. (YSQUR-$4)

YSLSB: Defines MSD Y address of square root. (YSQUR + $3)

BASWR: Defines MSD Y address plus 1 of work area. (YSQUR + $4; if this is overflow, value should be defined as $0.)

BASQU: Defines MSD Y address of number to take square root of. (YSQUR-$1)

FUNCTION

Converts 2-byte hexadecimal data in RAM into 5-digit BCD data and stores result in RAM; uses unsigned integers as arguments.

ARGUMENTS

1 digit= 4 bits

Contents		Storage Location	No. of Digits
Entry	2-byte hexadecimal	MD($04D∿ $04A)	4
Returns	5-digit BCD	MD($046 ∿ $042)	5

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected
× : Undefined
‡ : Result

A	B
×	×

X	Y
×	×

SPX	SPY
●	●

W
●

CA	ST
×	×

SPECIFICATIONS

1 word=10 bits

ROM (Words)
22
RAM (Digits)
9
Stack (Digits)
0
No. of cycles
885
Reentrant
No
Relocatable
No
Interrupt OK?
Yes

DESCRIPTION

(1) Function Details

(a) Argument details

MD($04D ∿ $04A): Holds a 2-byte hexadecimal number to be converted into BCD number.

MD($046 ∿ $042): Contains a 5-digit BCD number.

(b) Example of HEX execution is shown in Fig. 1.
If argument is as shown in part ① of Fig. 1, a 5-digit BCD number, in this case "52734", is contained in MD($046 ∿ $042), as shown in part ② of Fig. 1.

SPECIFICATIONS NOTES

N/A

DESCRIPTION

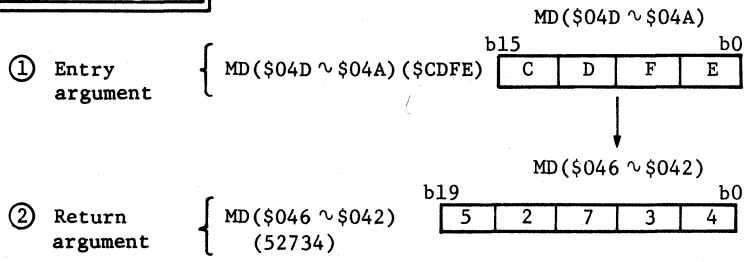


Fig. 1 Example of HEX Execution

(2) User Notes

(a) ST flag is set after HEX execution.

(b) When upper digits are not needed, "0" must be stored in upper digits as shown in Fig. 2. Otherwise correct result cannot be obtained since conversion is performed with undefined data in upper digits.

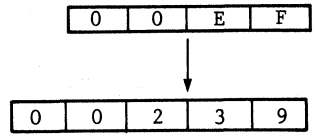


Fig. 2 Conversion Example When Upper Digits are Not Needed

(c) After HEX execution, a 2-byte hexadecimal number is destroyed. If a 2-byte hexadecimal number needs to be retained after HEX execution, it should be saved in memory before execution.

DESCRIPTION

(3) RAM Allocation

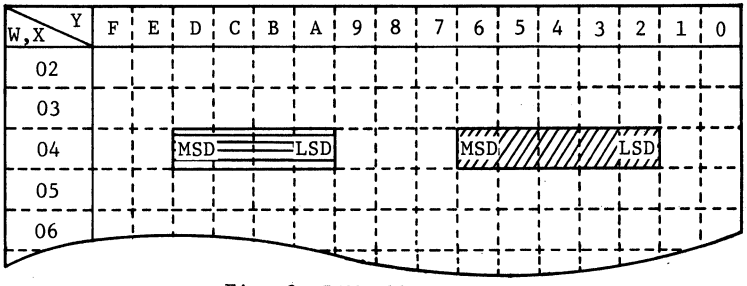


Fig. 3 RAM Allocation

Label	RAM	Description
—	b15 b0 MSD LSD MD($04D ∿ $04A)	2-byte hexadecimal number. X and Y addresses are defined by XHEXD and YHHEX, respectively.
—	b19 b0 MSD LSD MD($046 ∿ $042)	5-digit decimal number. X and Y addresses are defined by XHEXD and YHDEC, respectively.

DESCRIPTION

(4) Sample Application

Shown below is a sample application using HEX with address space allocated as follows.

```
MD($0AD ∿ $0AA): 2-byte hexadecimal number

MD($0A6 ∿ $0A2): 5-digit BCD number
```

```
        LWI     $0      -----Example with W=0.
          ¦
          ¦
        XSPXY
        LXI     $4
        XSPX
        LXI     $A
        LYI     $A      ------ Store a 2-byte hexadecimal number in
WORK1   LAMX                   entry argument.
        LMAIY
        YNEI    $E
        BRS     WORK1

        CALL    HEX     -----Call HEX.

        LXI     $A
        XSPX
        LXI     $4
        LYI     $6      ----- Store 5-digit BCD number, contained in
WORK2   LAMX                   return argument, in RAM.
        LMADY
        YNEI    $1
        BRS     WORK2
          ¦
          ¦
```

DESCRIPTION

(5) Basic Operation

(a) If ABCD is 4-bit binary number, it is expressed as shown in Formula 1 and Formula 2.

$$ABCD = A \times 2^3 + B \times 2^2 + C \times 2^1 + D \times 2^0 \quad \text{..... (Formula 1)}$$

$$= [\{(A \times 2) + B\} \times 2 + C] \times 2 + D \quad \text{..... (Formula 2)}$$

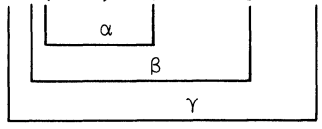


Fig. 4 4-bit Binary (ABCD)

(b) $\alpha = (A \times 2)+B$ is first performed, referring to Formula 2. Second, decimal adjustment is performed. Next, $\beta = (\alpha \times 2)+C$, $\gamma = (\beta \times 2)+D$ is calculated. Each calculation result is decimal adjusted to obtain final 5-digit binary number.

(c) Calculation of $\alpha = (A \times 2)+B$

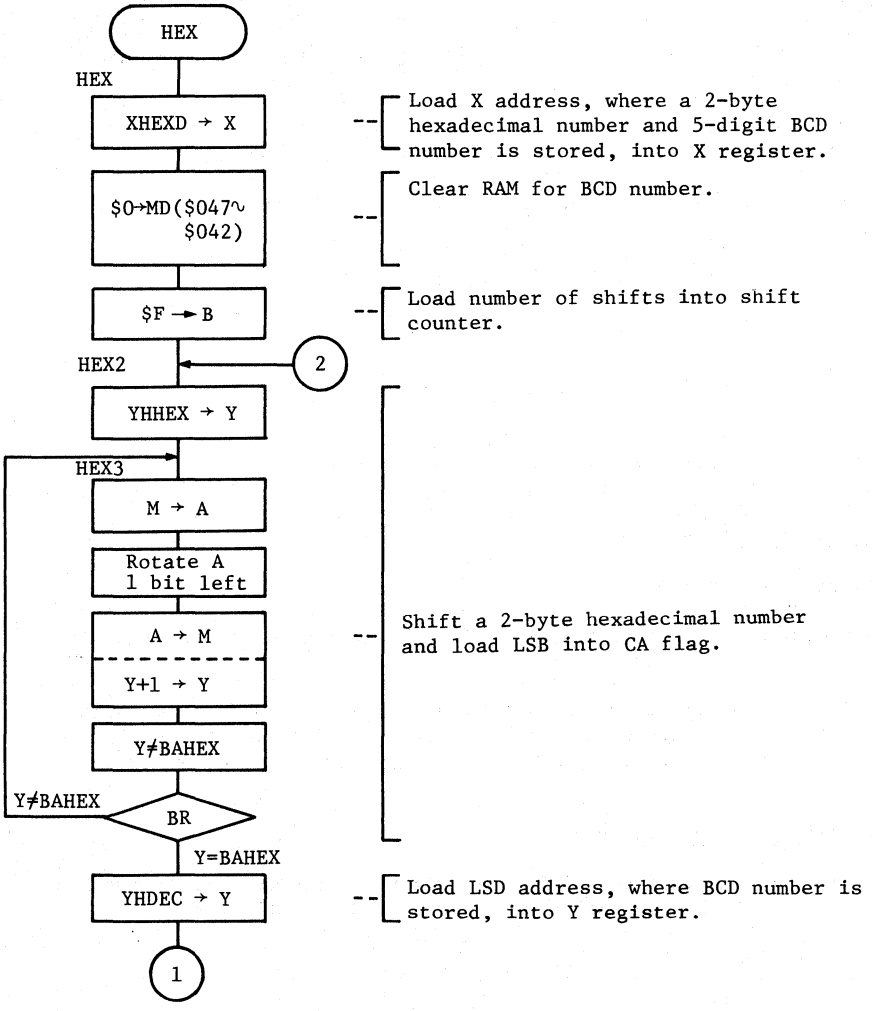
(i) Binary number string MD($04D ∿ $04A) is rotated left, and LSB is loaded into CA flag.

(ii) Decimal number string MD($046 ∿ $042) is shifted left to calculate $(A \times 2)$. Contents of CA flag is added to LSB of decimal number string to calculate $(+ B)$. $\alpha = (A \times 2)+B$ is performed by this process.

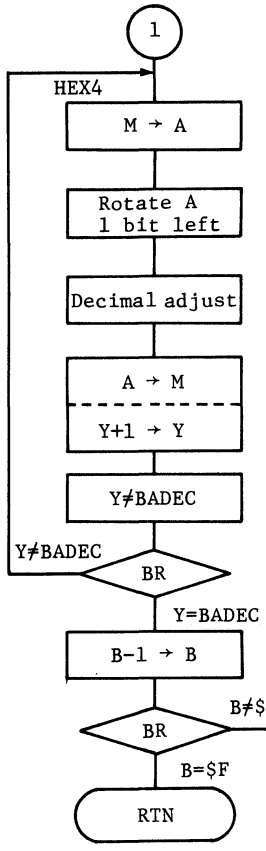
(iii) Addition result is decimal adjusted.

(iv) Operation loops from (i) to (iii) 16 times to complete conversion of a 2-byte hexadecimal number into a 5-digit BCD number.

FLOWCHART


HEX
HEX
XHEXD → X
Load X address, where a 2-byte hexadecimal number and 5-digit BCD number is stored, into X register.
$0→MD($047∿$042)
Clear RAM for BCD number.
$F → B
Load number of shifts into shift counter.
HEX2
2
YHHEX → Y
HEX3
M → A
Rotate A 1 bit left
A → M
Y+1 → Y
Shift a 2-byte hexadecimal number and load LSB into CA flag.
Y≠BAHEX
Y≠BAHEX
BR
Y=BAHEX
YHDEC → Y
Load LSD address, where BCD number is stored, into Y register.
1

FLOWCHART



Double BCD number in RAM and add MSB of hexadecimal number to result.

Adjust result into decimal and store it in RAM as BCD number.

Test if a 5-digit BCD number conversion is completed.

Decrement shift counter.

Test if shifts are completed.

PROGRAM LISTING

```
ST-NO   OBJECT   ADRS   SOURCE STATEMENTS

00001   010      0000           LLEN     132
00002                  ********************************************************************
00003                  *                                                                  *
00004                  *      NAME :  CONVERT 2-BYTE HEXADCIMAL INTO 5-DIGIT BCD (HEX)    *
00005                  *                                                                  *
00006                  ********************************************************************
00007                  *                                                                  *
00008                  *      ENTRY           : MD($04D-$04A) (2-BYTE HEXADECIMAL DATA)   *
00009                  *      RETURNS         : MD($046-$042) (5-DIGIT BCD DATA)          *
00010                  *                                                                  *
00011                  ********************************************************************
00012                  *
00013                  XHEXD    EQU      $4            HEXADECIMAL AND BCD DATA ADDR(X)
00014                  YHHEX    EQU      $A            2-BYTE HEXADECIMAL DATA LSD ADDR(Y)
00015                  YHDEC    EQU      $2            5-DIGIT BCD DATA LSD ADDR(Y)
00016                  BAHEX    EQU      $E            2-BYTE HEXADECIMAL DATA MSD ADDR(Y)+1
00017                  BADEC    EQU      $7            5-DIGIT BCD DATA MSD ADDR(Y)+1
00018                  *
00019                           ORG      $0100
00020                  HEX      EQU      *             ENTRY POINT
00021   224      0100           LXI      XHEXD         LOAD HEXADECIMAL ADDR(X)
00022   212      0101           LYI      YHDEC         LOAD BCD DATA ADDR(Y)
00023   290      0102   HEX1    LMIIY    $0            CLEAR BCD DATA ADDR(Y)
00024   077      0103           YNEI     BADEC
00025   302      0104           BRS      HEX1
00026   20F      0105           LBI      $F            LOAD SHIFT COUNTER
00027   21A      0106   HEX2    LYI      YHHEX         LOAD HEXADECIMAL DATA ADDR(Y)
00028   090      0107   HEX3    LAM                    SHIFT HEXADECIMAL DATA 1-BIT LEFT
00029   0A1      0108           ROTL
00030   050      0109           LMAIY
00031   07E      010A           YNEI     BAHEX
00032   307      010B           BRS      HEX3
00033   212      010C           LYI      YHDEC         LOAD BCD DATA LSD ADDR(Y)
00034   090      010D   HEX4    LAM                    BCD DATA AREA*2+CA->A
00035   0A1      010E           ROTL
00036   0A6      010F           DAA                    CONVERT INTO BCD DATA ?
00037   050      0110           LMAIY                  LOAD DECIMAL DATA
00038   077      0111           YNEI     BADEC         TEST IF CONVERSION IS COMPLETED
00039   30D      0112           BRS      HEX4
00040   0CF      0113           DB                     DECREMENT SHIFT COUNTER
00041   306      0114           BRS      HEX2          LOOP UNTIL SHIFT COUNTER = $F
00042   010      0115           RTN
```

When storing arguments in other RAM locations, change the EQU operands for the following labels.

XHEXD: Defines X address of a 2-byte hexadecimal number and a 5-digit BCD number.

YHHEX: Defines LSD Y address of 2-byte hexadecimal.

YHDEC: Defines LSD Y address of 5-digit decimal.

BAHEX: Defines MSD Y address plus 1 of 2-byte hexadecimal. (YHHEX + $4; if this is overflow, value should be defined as $0.)

BADEC: Defines MSD Y address plus 1 of 5-digit decimal. (YHDEC + $5; if this is overflow, value should be defined as $0.)

FUNCTION

Converts 5-digit BCD data in RAM into 2-byte hexadecimal data, and stores result in RAM; uses unsigned integers as arguments.

ARGUMENTS

1 digit= 4 bits

Contents		Storage Location	No. of Digits
Entry	5-digit BCD	MD($046 ∿ $042)	5
Returns	2-byte hexadecimal	MD($04E ∿ $04B)	4

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected
× : Undefined
‡ : Result

A	B
×	×
X	Y
×	×
SPX	SPY
×	×
W	
●	

CA	ST
×	×

SPECIFICATIONS

1 word=10 bits

ROM (Words)
52
RAM (Digits)
13
Stack (Digits)
0
No. of cycles
546
Reentrant
No
Relocatable
No
Interrupt OK?
Yes

DESCRIPTION

(1) Function Details

(a) Arguments details

MD($046 ∿ $042): Holds a 5-digit BCD number to be converted into hexadecimal.

MD($04E ∿ $04B): Contains a 2-byte hexadecimal number.

① Entry argument { MD($046 ∿ $042) (52734)

② Return argument { MD($04E ∿ $04B) ($CDFE)

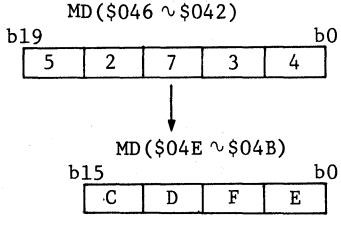


Fig. 1 Example of BCD Execution

SPECIFICATIONS NOTES

"No. of cycles" in "SPECIFICATIONS" indicates the number of cycles required to convert 59999 into hexadecimal data.

DESCRIPTION

(b) Example of BCD execution is shown in Fig. 1.
If entry argument is as shown in part ① of Fig. 1, a 2-byte hexadecimal number is contained in MD($04E ∿ $04B) as shown in part ② of Fig. 1.

(2) User Notes

(a) ST flag is set after BCD execution.

(b) 5-digit BCD number must be less than 65536, otherwise correct result cannot be obtained.

(c) Entry argument must be BCD format, otherwise correct result cannot be obtained.

(3) RAM Allocation

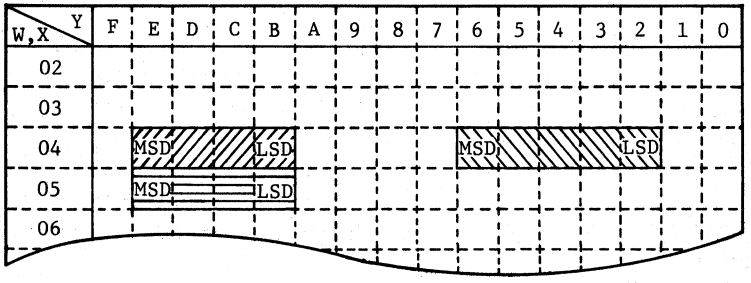


Fig. 2 RAM Allocation

Label	RAM	Description
—	b15 b0 MSD LSD MD($04E ∿ $04B)	2-byte hexadecimal number. X and Y addresses are defined by XBBCD and YBDEC, respectively.
—	b19 b0 MSD LSD MD($046 ∿ $042)	5-digit BCD number. X and Y addresses are defined by XBBCD and YBDEC, respectively.
—	b15 b0 MSD LSD MD($05E ∿ $05B)	Work area for decimal conversion. X and Y addresses are defined by XBWOR and YBDEC, respectively.

DESCRIPTION

(4) Sample Application

Shown below is a sample application using BCD with address space allocated as follows.

MD($0A6 ∿ $0A2): 5-digit BCD number

MD($0AE ∿ $0AB): 2-byte hexadecimal number

```
        LWI     $0      -----Example with W=0.
         ¦
         ¦
        LXI     $4
        XSPX
        LXI     $A
        LYI     $6      ----- Store a 5-digit BCD number in entry
WORK1   LAMX                  argument.
        LMADY
        YNEI    $1
        BRS     WORK1

        CALL    BCD     ----- Call BCD.

        LXI     $A
        XSPX
        LXI     $4
        LYI     $B      ----- Store a 2-byte hexadecimal number,
WORK2   LAMX                  contained in return argument, in RAM.
        LMAIY
        YNEI    $F
        BRS     WORK2
         ¦
         ¦
```

DESCRIPTION

(5) Basic Operation

(a) If ABCD is in 5-digit BCD number, it is expressed as shown in Formula 1 and Formula 2.

$$ABCD = A \times 10^3 + B \times 10^2 + C \times 10^1 + D \times 10^0 \quad \text{...(Formula 1)}$$

$$= [\{(A \times 10) + B\} \times 10 + C] \times 10 + D \quad \text{...(Formula 2)}$$

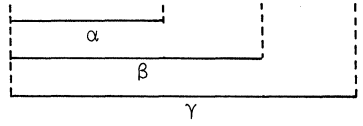


Fig. 3 4-digit BCD (ABCD)

(b) 5-digit BCD can be converted into hexadecimal as follows.

First, $\alpha = (A \times 10) + B$ is calculated to determine α in Fig. 3 (Formula 2).

Next $\beta = (\alpha \times 10) + C$, $\gamma = (\beta \times 10) + D$ are calculated to determine β and γ, respectively.

(c) Calculation of $A \times 10$ is performed using formula 3 and 4;

$$A \times 10 = A \times (2 + 8) \quad \text{..... (Formula 3)}$$

$$= A \times 2(1 + 2^2) \quad \text{..... (Formula 4)}$$

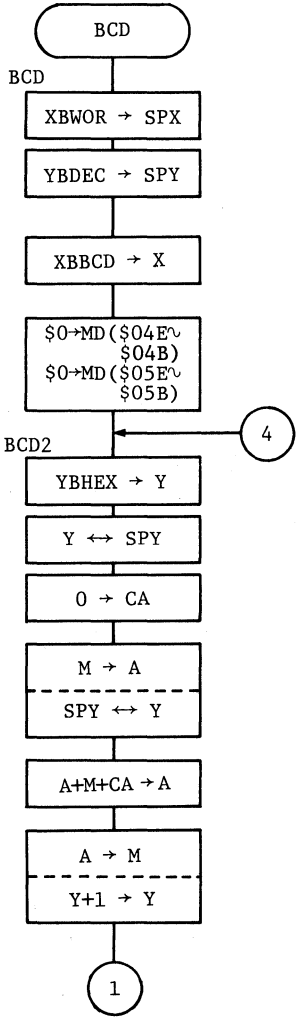
(d) Program processing

(i) To perform Formula 4, MD($04E ∿ $04B) and MD($046 ∿ $042) are used by BCD for arguments, and MD($05E ∿ $05B) is used as work area.

(ii) Referring to (Formula 4), A is first load into MD($04E ∿ $04B), shifted 1 bit left, and the result stored in MD($05E ∿ $05B). Next, MD($05E ∿ $05B) is shifted 2 bits left. Finally, MD($04E ∿ $04B) is added to MD($05E ∿ $05B). $A \times 10$ in (Formula 4) is performed by this process.

(iii) Repeat (i) and (ii) step 5 times to complete conversion from a 5-digit BCD number into 2-byte hexadecimal number.

FLOWCHART



-- Load work area address into SPX register.

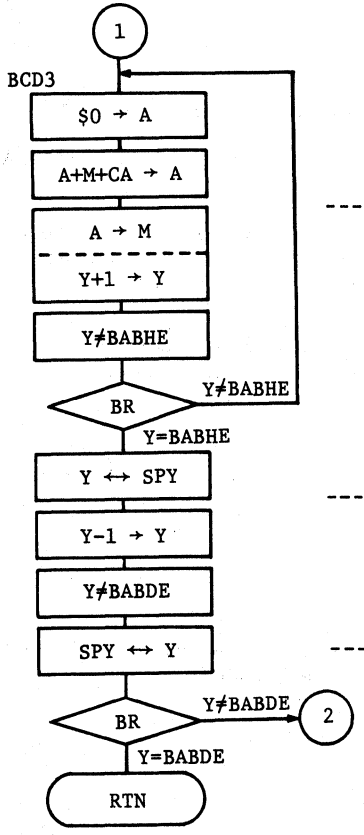
-- Load MSD address of 5-digit BCD RAM into SPY register.

-- Load address, where 5-digit BCD number and a 2-byte hexadecimal number is stored, into X register.

-- Clear work area and RAM for 2-byte hexadecimal number.

-- Add 1 digit of a 5-digit BCD number to 2-byte hexadecimal RAM.

FLOWCHART



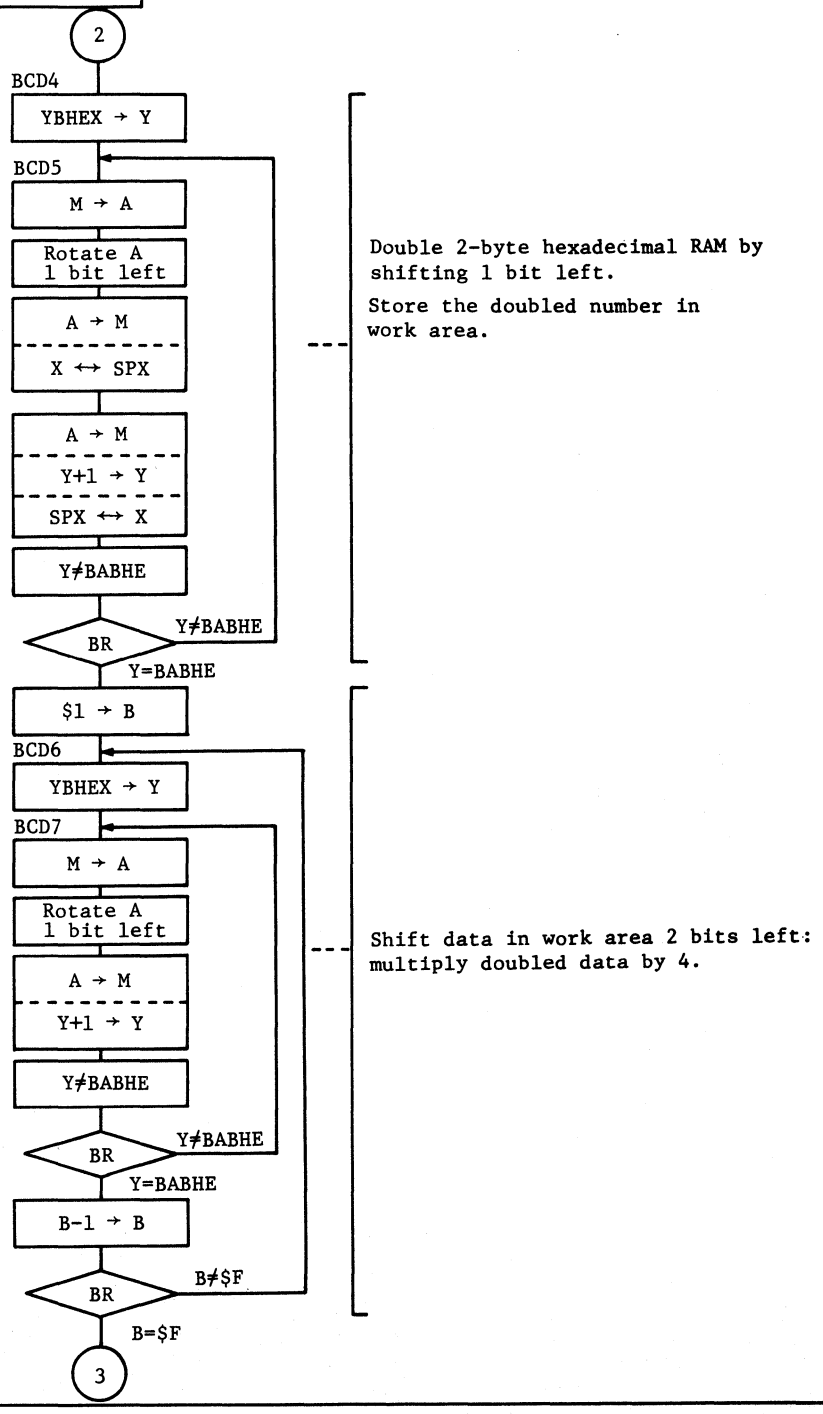
---- Add carry generated in 2-byte hexadecimal RAM.

---- Decrement pointer to digits of a 5-digit BCD number.

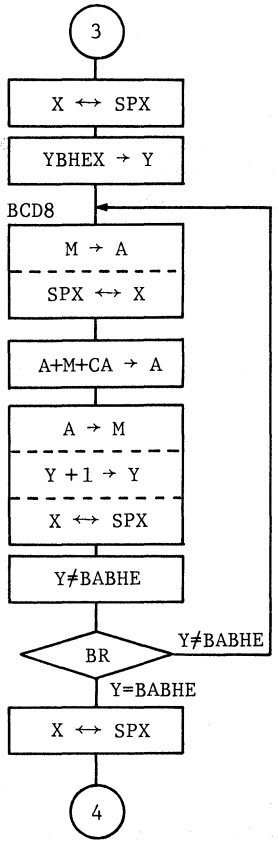
---- Test if 5-digit BCD conversion is completed.

20. CONVERT 5-DIGIT BCD INTO 2-BYTE HEXADECIMAL	MCU	HMCS400 SERIES	LABEL	BCD

FLOWCHART



FLOWCHART



Add doubled 2-byte hexadecimal number to 8-times-multiplied data in work area. Store result in 2-byte hexadecimal RAM.

As a result, 2-byte hexadecimal RAM is multiplied by 10.

Load X address of a 5-digit BCD number.

PROGRAM LISTING

```
ST-NO   OBJECT   ADRS   SOURCE STATEMENTS

00001   010      0000           LLEN     132
00002                  ************************************************************
00003                  *                                                          *
00004                  *      NAME : CONVERT 5-DIGIT BCD INTO 2-BYTE              *
00005                  *                    HEXADECIMAL (BCD)                     *
00006                  ************************************************************
00007                  *                                                          *
00008                  *      ENTRY      :MD($046-$042)(5-DIGIT BCD DATA)         *
00009                  *      RETURNS    :MD($04E-$04B)(2-BYTE HEXADECIMAL DATA)  *
00010                  *                                                          *
00011                  ************************************************************
00012                  *
00013                  XBBCD    EQU      $4          BCD AND HEXADECIMAL DATA ADDR(X)
00014                  XBWOR    EQU      $5          WORK AREA ADDR(X)
00015                  YBHEX    EQU      $B          2-BYTE HEXADECIMAL DATA LSD ADDR(Y)
00016                  YBDEC    EQU      $6          5-DIGITE BCD DATA MSD ADDR(Y)
00017                  BABHE    EQU      $F          2-BYTE HEXADECIMAL DATA MSD ADDR(Y)+1
00018                  BABDE    EQU      $1          5-DIGIT BCD DATA LSD ADDR(Y)-1
00019                  *
00020                           ORG      $0100
00021                  BCD      EQU      *           ENTRY POINT
00022   225      0100           LXI      XBWOR       LOAD WORK AREA ADDR(SPX)
00023   216      0101           LYI      YBDEC       LOAD BCD DATA MSD ADDR(SPY)
00024   003      0102           XSPXY
00025   224      0103           LXI      XBBCD       LOAD BCD & HEXADECIMAL DATA ADDR(X)
00026   21B      0104           LYI      YBHEX       LOAD HEXADECIMAL DATA ADDR(Y)
00027   230      0105           LAI      $0          CLEAR WORK AREA & HEXADECIMAL DATA AREA
00028   095      0106   BCD1    LMAX
00029   051      0107           LMAIYX
00030   07F      0108           YNEI     BABHE
00031   306      0109           BRS      BCD1
00032   21B      010A   BCD2    LYI      YBHEX       LOAD 2-BYTE HEXADECIMAL DATA ADDR(Y)
00033   002      010B           XSPY
00034   0EC      010C           REC                  A + HEXADECIMAL DATA -> HEXADECIMAL DATA
00035   092      010D           LAMY
00036   018      010E           AMC
00037   050      010F           LMAIY
00038   230      0110   BCD3    LAI      $0          ADD CARRY
00039   018      0111           AMC
00040   050      0112           LMAIY
00041   07F      0113           YNEI     BABHE
00042   310      0114           BRS      BCD3
00043   002      0115           XSPY
00044   0DF      0116           DY                   DECREMENT BCD DATA ADDR(Y)
00045   071      0117           YNEI     BABDE       TEST IF BCD CONVERSION IS COMPLETED
00046   002      0118           XSPY
00047   31B      0119           BRS      BCD4
00048   010      011A           RTN
00049   21B      011B   BCD4    LYI      YBHEX
00050   090      011C   BCD5    LAM                  HEXADECIMAL DATA * 2 -> HEXADECIMAL DATA & WORK AREA
00051   0A1      011D           ROTL
00052   095      011E           LMAX
00053   051      011F           LMAIYX
00054   07F      0120           YNEI     BABHE
00055   31C      0121           BRS      BCD5
00056   201      0122           LBI      $1          HEXADECIMAL DATA * 4 -> HEXADECIMAL DATA
00057   21B      0123   BCD6    LYI      YBHEX
00058   090      0124   BCD7    LAM
00059   0A1      0125           ROTL
00060   050      0126           LMAIY
00061   07F      0127           YNEI     BABHE
00062   324      0128           BRS      BCD7
00063   0CF      0129           DB
00064   323      012A           BRS      BCD6
00065   001      012B           XSPX
00066   21B      012C           LYI      YBHEX       HEXADECIMAL DATA + WORK AREA -> HEXADECIMAL DATA
00067   091      012D   BCD8    LAMX
00068   018      012E           AMC
00069   051      012F           LMAIYX
00070   07F      0130           YNEI     BABHE
00071   32D      0131           BRS      BCD8
00072   001      0132           XSPX                 LOAD BCD DATA ADDR(X)
00073   30A      0133           BRS      BCD2
```

PROGRAM LISTING

When storing arguments in other RAM locations, change the EQU operands for the following labels.

XBBCD: Defines X address of a 5-digit BCD number and a 2-byte hexadecimal number.

XBWOR: Defines X address of work area.

YBHEX: Defines LSD Y address of work area and 2-byte hexadecimal.

YBDEC: Defines MSD Y address of 5-digit BCD. (YBHEX-$5)

BABHE: Defines MSD Y address plus 1 of 2-byte hexadecimal. (YBHEX + $4; if this is overflow, value should be defined as $0.)

BABDE: Defines LSD Y address less 1 of 5-digit BCD. (YBHEX-$A; if this is negative, value should be defined as $F.)

FUNCTION

Sorts a specified number of bytes in RAM in descending order; uses unsigned integers as sorting data.

ARGUMENTS

1 digit= 4 bits

Contents		Storage Location	No. of Digits
Entry	No. of bytes to be sorted	A	1
	Starting address of data to be sorted (X)	X, SPX	2
	Starting address of data to be sorted (Y)	Y, SPY	2
Returns	—	—	—

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected
× : Undefined
‡ : Result

A	B
×	×

X	Y
×	×

SPX	SPY
×	×

W
●

CA	ST
×	×

SPECIFICATIONS

1 word=10 bits

ROM (Words)
58
RAM (Digits)
3
Stack (Digits)
0
No. of cycles
233
Reentrant
No
Relocatable
No
Interrupt OK?
Yes

DESCRIPTION

(1) Function Details

(a) Argument details

A : Holds number of bytes to be sorted minus 2 as a 1-byte hexadecimal number.

X, SPX: Holds starting X address of data to be sorted.

Y, SPY: Holds starting Y address of data to be sorted.

(b) Example of SORT execution is shown in Fig. 1.

If entry arguments are as shown in part ① of Fig. 1, sorted data is stored from MD($053) in descending order as shown in part ② of Fig. 1. As number to be sorted is 5 bytes, $3 is loaded into Accumulator as shown in part ① of Fig. 1.

SPECIFICATIONS NOTES

"RAM" and "No. of cycles" in "SPECIFICATIONS" indicates that required to sort 5 bytes of ascending data in descending order.

DESCRIPTION

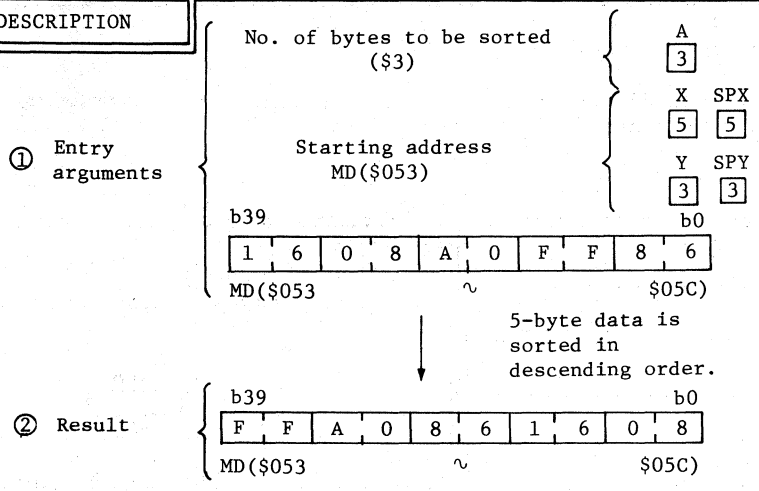


Fig. 1 Example of SORT Execution

(2) User Notes

Number of bytes to be sorted must be loaded as "No. of bytes - $2" into Accumulator for correct loop processing.

(3) RAM Allocation

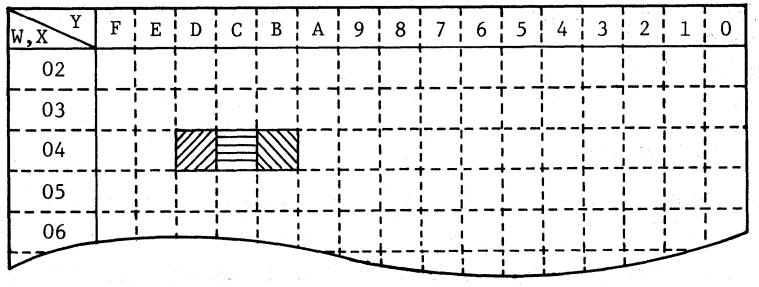


Fig. 2 RAM Allocation

Label	RAM	Description
SCNT1	b3 b0 MD($04D)	Counter indicating number of remaining bytes to be sorted.
SCNT2	b3 b0 MD($04C)	Counter indicating number of remaining values to be compared.
SWORK	b3 b0 MD($04B)	Work area.

DESCRIPTION

(4) Sample Application

Shown below is a sample application using SORT with address space allocated as follows.

```
MD($0AE)       : Number to be sorted
MD($0AD, $0AC): Starting address of number to be sorted
```

```
        LWI     $0      ----- Example with W=0.
        :
        LAMD    $0AD
        LXA
        XSPX
        LXA                   ------ Load starting address of number to be
        LAMD    $0AC                 sorted into entry argument.
        LYA
        XSPY
        LYA
        LAMD    $0AE    ------ Load number of bytes to be sorted
                                into entry argument.
        CALL    SORT    ------ Call SORT.
        :
```

(5) Basic Operation

(a) Fig. 3 shows how 3 units of number are sorted in descending order.

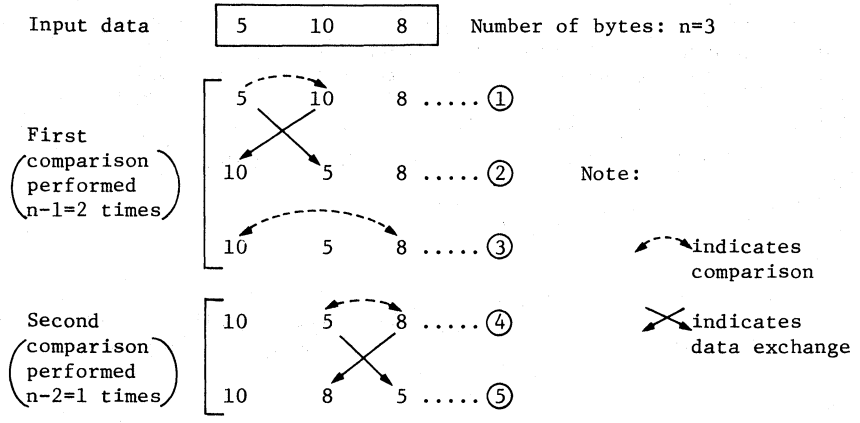


Fig. 3 Sorting Example

(i) Determines the largest value among the three and places it into the left position. (See Fig. 3 ①,② and ③)

(ii) Compares middle and right values and places the larger one in the middle. (See Fig. 3 ④, ⑤)

(b) Program processing

(i) Number of bytes to be sorted (Accumulator) is loaded into SCNT1. SCNT1 is used as sort counter.

(ii) SCNT2 is used as pointer to number in RAM to be sorted. SCNT2 is initialized to SCNT1 value every time SCNT1 is decremented, i.e., as sorting continues, the number of values to be sorted is reduced.

(iii) Value to be compared pointed to by X and Y registers is loaded into Accumulator and B register.

(iv) Address where value is stored is incremented and new value is compared with value in Accumulator and B register.

(v) First, upper digits of values are compared.
If value is larger than or equivalent to compared value, then lower digits are compared.

(vi) If value in memory is larger than compared value in Accumulator and B register, they are exchanged.

(vii) SCNT2 showing number of remaining bytes is decremented; operation loops from (iv) to (vi) until SCNT2 is $0.

DESCRIPTION

(viii) When SCNT2 reaches $0, largest value compared with Accumulator and B register is stored.

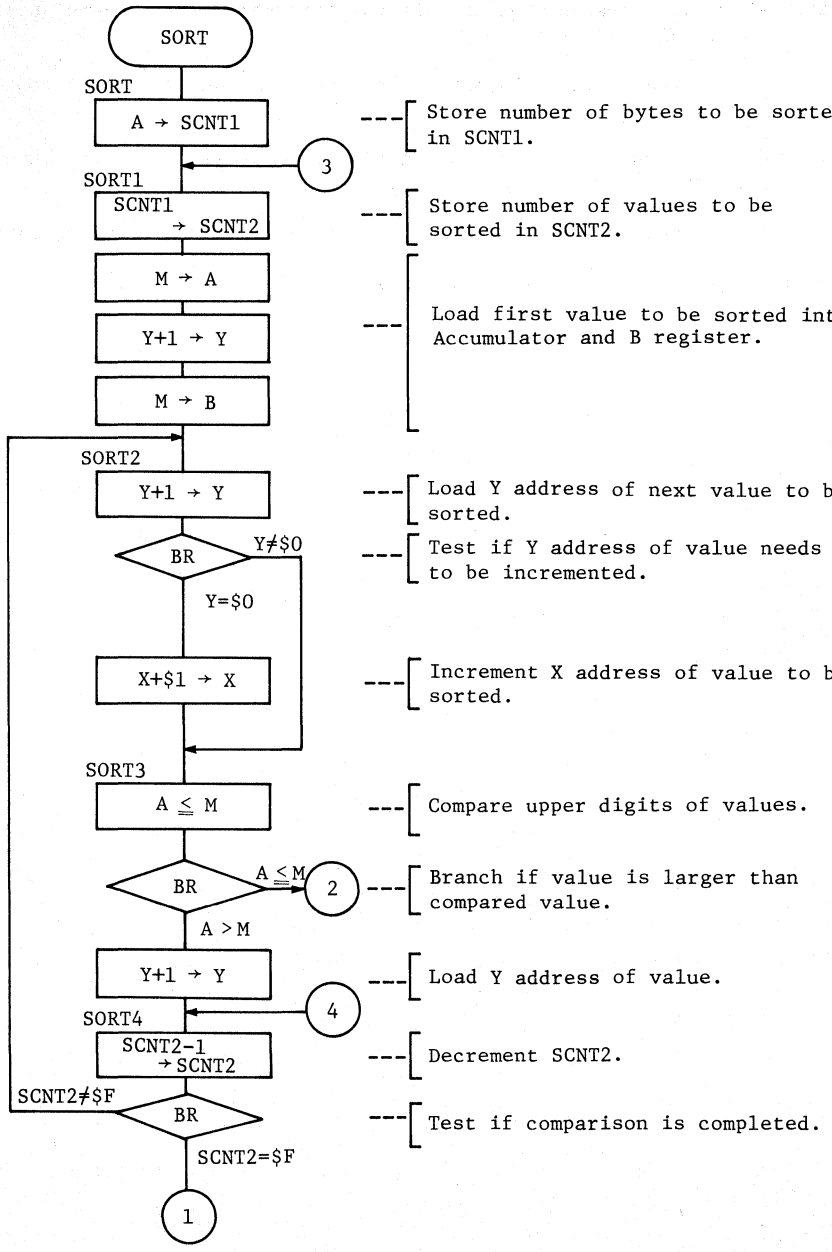
(ix) The largest value is stored in the address specified by X and Y registers.

(x) X and Y addresses specified in (ix) is decremented to indicate address of next value to be compared.

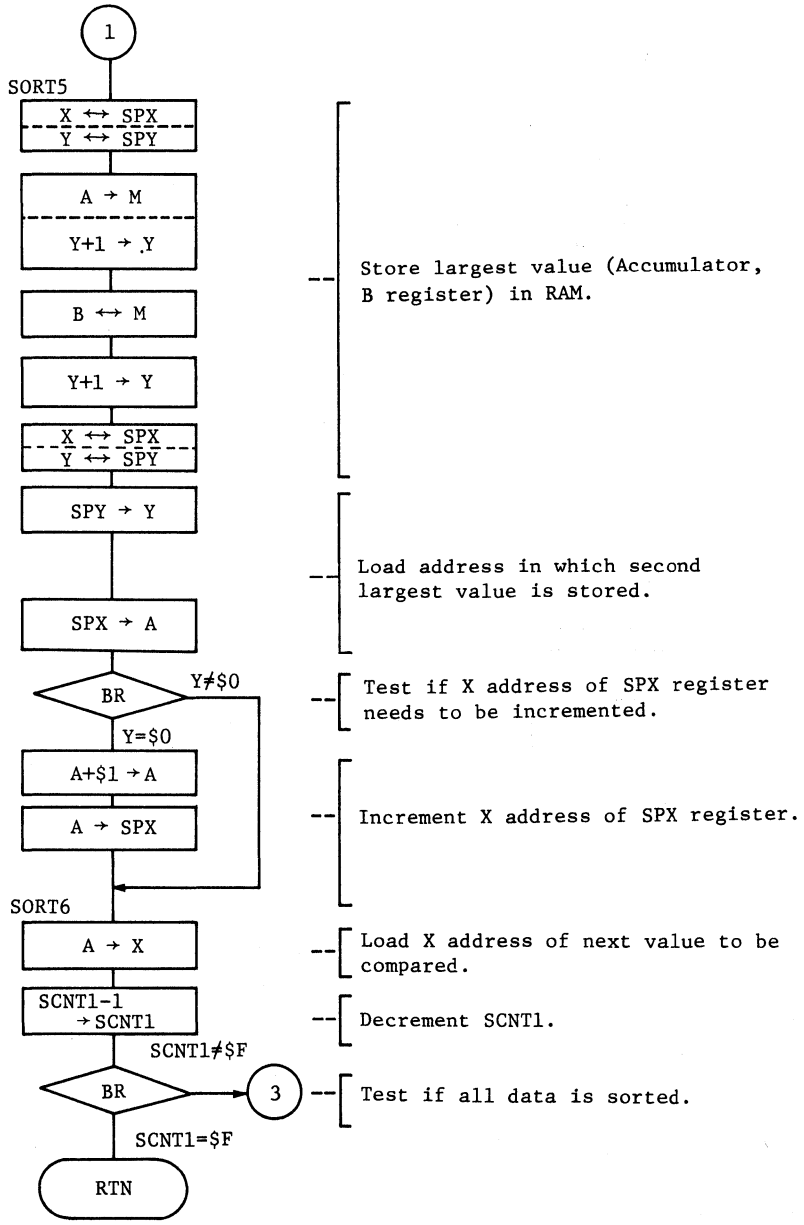
(xi) SCNT1 showing how many data remain to be compared is decremented.

(xii) Operation loops from (iii) to (xi) until SCNT1 is $0.

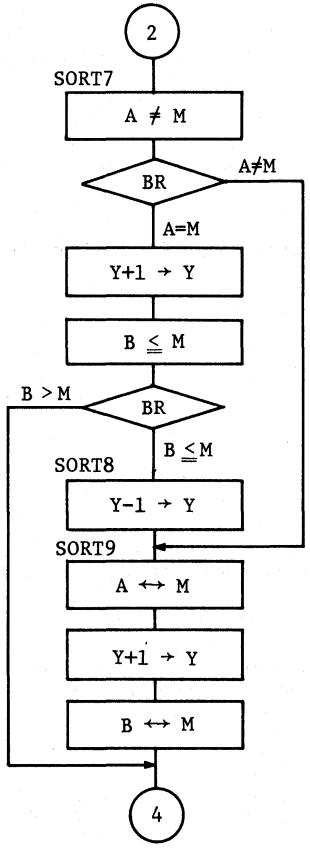
FLOWCHART


SORT
SORT
A → SCNT1
Store number of bytes to be sorted in SCNT1.
3
SORT1
SCNT1 → SCNT2
Store number of values to be sorted in SCNT2.
M → A
Y+1 → Y
M → B
Load first value to be sorted into Accumulator and B register.
SORT2
Y+1 → Y
Load Y address of next value to be sorted.
BR
Y≠$0
Test if Y address of value needs to be incremented.
Y=$0
X+$1 → X
Increment X address of value to be sorted.
SORT3
A ≦ M
Compare upper digits of values.
BR
A ≦ M
2
Branch if value is larger than compared value.
A > M
Y+1 → Y
Load Y address of value.
4
SORT4
SCNT2-1 → SCNT2
Decrement SCNT2.
SCNT2≠$F
BR
Test if comparison is completed.
SCNT2=$F
1

FLOWCHART


1
SORT5
X ↔ SPX
Y ↔ SPY
A → M
Y+1 → Y
B ↔ M
Y+1 → Y
X ↔ SPX
Y ↔ SPY
SPY → Y
SPX → A
BR
Y≠$0
Y=$0
A+$1 → A
A → SPX
SORT6
A → X
SCNT1-1
→SCNT1
SCNT1≠$F
BR
3
SCNT1=$F
RTN
Store largest value (Accumulator, B register) in RAM.
Load address in which second largest value is stored.
Test if X address of SPX register needs to be incremented.
Increment X address of SPX register.
Load X address of next value to be compared.
Decrement SCNT1.
Test if all data is sorted.

FLOWCHART



--- Test if upper digits of values are equivalent.

--- Load Y address of lower digits of value to be sorted.

--- Compare lower digits of values. If value is larger than compared value, branch to SORT8.

--- Reload Y address of upper digits of value.

--- If value is larger than compared value, exchange them.

PROGRAM LISTING

```
ST-NO   OBJECT   ADRS   SOURCE STATEMENTS

00001   30A      0000           LLEN     132
00002                  ************************************************************
00003                  *                                                          *
00004                  *      NAME : SORT (SORT)                                  *
00005                  *                                                          *
00006                  ************************************************************
00007                  *                                                          *
00008                  *      ENTRY    :A       (NO. OF BYTES TO BE SORTED)       *
00009                  *               :X.SPX   (START ADDR(X) OF DATA TO BE SORTED)*
00010                  *                Y.SPY   (START ADDR(Y) OF DATA TO BE SORTED)*
00011                  *      RETURNS  :NOTHING                                   *
00012                  *                                                          *
00013                  ************************************************************
00014                  *
00015                  SCNT1    EQU      $04D          SORTING DATA COUNTER ADDR
00016                  SCNT2    EQU      $04C          COMPARISON DATA COUNTER ADDR
00017                  SWORK    EQU      $04B          WORK AREA ADDR
00018                  *
00019                           ORG      $0100
00020                  SORT     EQU      *             ENTRY POINT
00021   194 04D  0100           LMAD     SCNT1         LOAD NO OF BYTES TO BE SORTED INTO SCNT1
00022   194 04C  0102  SORT1    LMAD     SCNT2         LOAD NO OF BYTES TO BE SORTED INTO SCNT2
00023   090      0104           LAM                    LOAD COMPARISON DATA
00024   05C      0105           IY
00025   040      0106           LBM
00026   05C      0107  SORT2    IY                     LOAD SORTING DATA ADDR(Y)
00027   312      0108           BRS      SORT3
00028   180 04B  0109           XMAD     SWORK         LOAD SORTING DATA ADDR(X)
00029   001      010B           XSPX
00030   068      010C           LASPX
00031   001      010D           XSPX
00032   281      010E           AI       $1            INCREMENT SORTING DATA ADDR(X)
00033   0E8      010F           LXA
00034   180 04B  0110           XMAD     SWORK
00035   014      0112  SORT3    ALEM                   COMPARISON DATA(A) =< SORTING DATA ?
00036   32F      0113           BRS      SORT7         BRANCH IF A =< SORTING DATA
00037   05C      0114           IY
00038   180 04C  0115  SORT4    XMAD     SCNT2         DECREMENT COMPARISON DATA COUNTER
00039   28F      0117           AI       $F            TEST IF COMPARISON IS COMPLETED
00040   180 04C  0118           XMAD     SCNT2
00041   307      011A           BRS      SORT2
00042   003      011B  SORT5    XSPXY                  STORE LARGEST DATA
00043   050      011C           LMAIY
00044   0C0      011D           XMB
00045   05C      011E           IY                     LOAD SECOND LARGEST DATA ADDR(Y)
00046   003      011F           XSPXY
00047   058      0120           LASPY
00048   0D8      0121           LYA
00049   068      0122           LASPX
00050   327      0123           BRS      SORT6
00051   281      0124           AI       $1            INCREMENT ADDR(X) OF SPX
00052   0E8      0125           LXA                    LOAD COMPARISON DATA ADDR(X)
00053   001      0126           XSPX
00054   0E8      0127  SORT6    LXA                    LOAD NEXT DATA ADDR(X)
00055   180 04D  0128           XMAD     SCNT1         DECREMENT SORTING DATA COUNTER
00056   28F      012A           AI       $F            TEST IF ALL DATA IS SORTED
00057   194 04D  012B           LMAD     SCNT1
00058   302      012D           BRS      SORT1
00059   010      012E           RTN
00060   004      012F  SORT7    ANEM                   COMPARISON DATA(A) = SORTING DATA ?
00061   336      0130           BRS      SORT9
00062   05C      0131           IY                     COMPARISON DATA(B) =< SORTING DATA ?
00063   0C4      0132           BLEM
00064   335      0133           BRS      SORT8
00065   315      0134           BRS      SORT4
00066   0DF      0135  SORT8    DY                     LOAD ADDR(Y) OF UPPER DIGIT
00067   080      0136  SORT9    XMA                    EXCHANGE DATA
00068   05C      0137           IY
00069   0C0      0138           XMB
00070   315      0139           BRS      SORT4
```

When storing arguments in other RAM locations, change the EQU operands for the following labels.

SCNT1: Defines work area address indicating number of bytes to be sorted.

SCNT2: Defines work area address indicating number of values to be compared.

SWORK: Defines work area address for saving Accumulator contents.

INSTRUCTION SET

Symbols and Abbreviations

a → b	Transfer from "a" to "b"
a ↔ b	Exchange between "a" and "b"
$\overline{X}$	Logical negation (NOT)
"1"	"High" level
"0"	"Low" level
LSB	Least Significant Bit
MSB	Most Significant Bit
NZ	None Zero
NB	No Borrow
OVF	Overflow
∩	AND
∪	OR
⊕	Exclusive OR
≠	Not Equal
≦	Less than or Equal
DIRECT	Addressing by the instruction operand in the ROM code
REGISTER	Addressing by the contents of Address Register
I/E	Interrupt enable flag
PC	Program counter
SP	Stack pointer
ST	Status flag
CA	Carry flag
A	Accumulator
B	B register
W	W register
X	X register
SPX	SPX register
Y	Y register
SPY	SPY register
M	Memory (RAM)
MR	Memory register
R	Data I/O pin or data register
D	Discrete I/O pin or discrete latch

NZ, NB, OVF: Status is set when NZ, NB, or OVF occurs

Symbolic Operands Used with Instruction Set Mnemonics

Symbol	Contents	Label	D	HD	B
n	RAM digit selection by 2 bits	○	○	○	○
m	RAM (M_R) digit and port selection by 4 bits	○	○	○	○
p	Replacement of contents of PC by 4 bits	○	○	○	○
a	Replacement of contents of PC by 6 bits	○	×	○	○
b	Replacement of contents of PC by 8 bits	○	×	○	○
d	Replacement of PC or RAM direct address by 10 bits	○	×	○	○
i	Immediate data by 4 bits (Note)	○	○	○	○
u	u = p + d (14 bits)	○	×	○	○

D: Decimal HD: Hexadecimal B: Binary

○ .... Can be used × .... Cannot be used

(Note) In case of LWI instruction, 2 bits.

Immediate Instruction

OPERATION	MNEMONIC	OPERATION CODE	FUNCTION	STATUS	WORD/CYCLE
Load A from Immediate	LAI i	1 0 0 0 1 1 i_3 i_2 i_1 i_0	i⟶A		1/1
Load B from Immediate	LBI i	1 0 0 0 0 0 i_3 i_2 i_1 i_0	i⟶B		1/1
Load Memory from Immediate	LMID i,d	0 1 1 0 1 0 i_3 i_2 i_1 i_0 d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	i⟶M		2/2
Load Memory from Immediate, Increment Y	LMIIY i	1 0 1 0 0 1 i_3 i_2 i_1 i_0	i→M,Y+1→Y	NZ	1/1

Register-to-Register Instruction

OPERATION	MNEMONIC	OPERATION CODE	FUNCTION	STATUS	WORD/CYCLE
Load A from B	LAB	0 0 0 1 0 0 1 0 0 0	B→A		1/1
Load B from A	LBA	0 0 1 1 0 0 1 0 0 0	A→B		1/1
Load A from Y	LAY	0 0 1 0 1 0 1 1 1 1	Y→A		1/1
Load A from SPX	LASPX	0 0 0 1 1 0 1 0 0 0	SPX→A		1/1
Load A from SPY	LASPY	0 0 0 1 0 1 1 0 0 0	SPY→A		1/1
Load A from MR	LAMR m	1 0 0 1 1 1 $m_3$$m_2$$m_1$$m_0$	MR(m)→A		1/1
Exchange MR and A	XMRA m	1 0 1 1 1 1 $m_3$$m_2$$m_1$$m_0$	MR(m)↔A		1/1

RAM Address Instruction

OPERATION	MNEMONIC	OPERATION CODE	FUNCTION	STATUS	WORD/CYCLE
Load W from Immediate	LWI i	0 0 1 1 1 1 0 0 i_1 i_0	i⟶W		1/1
Load X from Immediate	LXI i	1 0 0 0 1 0 i_3 i_2 i_1 i_0	i⟶X		1/1
Load Y from Immediate	LYI i	1 0 0 0 0 1 i_3 i_2 i_1 i_0	i⟶Y		1/1
Load X from A	LXA	0 0 1 1 1 0 1 0 0 0	A⟶X		1/1
Load Y from A	LYA	0 0 1 1 0 1 1 0 0 0	A⟶Y		1/1
Increment Y	IY	0 0 0 1 0 1 1 1 0 0	Y+1→Y	NZ	1/1
Decrement Y	DY	0 0 1 1 0 1 1 1 1 1	Y−1→Y	NB	1/1
Add A to Y	AYY	0 0 0 1 0 1 0 1 0 0	Y+A→Y	OVF	1/1
Subtract A from Y	SYY	0 0 1 1 0 1 0 1 0 0	Y−A→Y	NB	1/1
Exchange X and SPX	XSPX	0 0 0 0 0 0 0 0 0 1	X↔SPX		1/1
Exchange Y and SPY	XSPY	0 0 0 0 0 0 0 0 1 0	Y↔SPY		1/1
Exchange X and SPX,Y and SPY	XSPXY	0 0 0 0 0 0 0 0 1 1	X↔SPX,Y↔SPY		1/1

RAM Register Instruction

OPERATION	MNEMONIC	OPERATION CODE	FUNCTION	STATUS	WORD / CYCLE
Load A from Memory	LAM(XY)	0 0 1 0 0 1 0 0 y x	M→A, (X↔SPX, Y↔SPY)		1/1
Load A from Memory	LAMD d	0 1 1 0 0 1 0 0 0 0 d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	M→A		2/2
Load B from Memory	LBM(XY)	0 0 0 1 0 0 0 0 y x	M→B, (X↔SPX, Y↔SPY)		1/1
Load Memory from A	LMA(XY)	0 0 1 0 0 1 0 1 y x	A→M, (X↔SPX, Y↔SPY)		1/1
Load Memory from A	LMAD d	0 1 1 0 0 1 0 1 0 0 d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	A→M		2/2
Load Memory from A, Increment Y	LMAIY(X)	0 0 0 1 0 1 0 0 0 x	A→M, Y+1→Y(X↔SPX)	NZ	1/1
Load Memory from A, Decrement Y	LMADY(X)	0 0 1 1 0 1 0 0 0 x	A→M, Y−1→Y(X↔SPX)	NB	1/1
Exchange Memory and A	XMA(XY)	0 0 1 0 0 0 0 0 y x	M↔A, (X↔SPX, Y↔SPY)		1/1
Exchange Memory and A	XMAD d	0 1 1 0 0 0 0 0 0 0 d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	M↔A		2/2
Exchange Memory and B	XMB(XY)	0 0 1 1 0 0 0 0 y x	M↔B, (X↔SPX, Y↔SPY)		1/1

Note) (XY) and (x) have the meaning as follows:

(1) The instructions with (XY) have 4 mnemonics and 4 object codes for each. (example of LAM (XY) is given below.)

MNEMONIC	y	x	FUNCTION
LAM	0	0	
LAMX	0	1	X↔SPX
LAMY	1	0	Y↔SPY
LAMXY	1	1	X↔SPX, Y↔SPY

(2) The instructions with (x) have 2 mnemonics and 2 object codes for each. (example of LMAIY(X) is given below.)

MNEMONIC	x	FUNCTION
LMAIY	0	
LMAIYX	1	X↔SPX

Arithmetic Instruction

OPERATION	MNEMONIC	OPERATION CODE	FUNCTION	STATUS	WORD/CYCLE
Add Immediate to A	AI i	1 0 1 0 0 0 i_3 i_2 i_1 i_0	A + i → A	OVF	1/1
Increment B	IB	0 0 0 1 0 0 1 1 0 0	B + 1 → B	NZ	1/1
Decrement B	DB	0 0 1 1 0 0 1 1 1 1	B − 1 → B	NB	1/1
Decimal Adjust for Addition	DAA	0 0 1 0 1 0 0 1 1 0			1/1
Decimal Adjust for Subtraction	DAS	0 0 1 0 1 0 1 0 1 0			1/1
Negate A	NEGA	0 0 0 1 1 0 0 0 0 0	$\overline{A}$ + 1 → A		1/1
Complement B	COMB	0 1 0 1 0 0 0 0 0 0	$\overline{B}$ → B		1/1
Rotate Right A with Carry	ROTR	0 0 1 0 1 0 0 0 0 0			1/1
Rotate Left A with Carry	ROTL	0 0 1 0 1 0 0 0 0 1			1/1
Set Carry	SEC	0 0 1 1 1 0 1 1 1 1	1 → CA		1/1
Reset Carry	REC	0 0 1 1 1 0 1 1 0 0	0 → CA		1/1
Test Carry	TC	0 0 0 1 1 0 1 1 1 1		CA	1/1
Add A to Memory	AM	0 0 0 0 0 0 1 0 0 0	M + A → A	OVF	1/1
Add A to Memory	AMD d	0 1 0 0 0 0 1 0 0 0 / d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	M + A → A	OVF	2/2
Add A to Memory with Carry	AMC	0 0 0 0 0 1 1 0 0 0	M + A + CA → A OVF → CA	OVF	1/1
Add A to Memory with Carry	AMCD d	0 1 0 0 0 1 1 0 0 0 / d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	M + A + CA → A OVF → CA	OVF	2/2
Subtract A from Memory with Carry	SMC	0 0 1 0 0 1 1 0 0 0	M − A − $\overline{CA}$ → A NB → CA	NB	1/1
Subtract A from Memory with Carry	SMCD d	0 1 1 0 0 1 1 0 0 0 / d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	M − A − $\overline{CA}$ → A NB → CA	NB	2/2
OR A and B	OR	0 1 0 1 0 0 0 1 0 0	A ∪ B → A		1/1
AND Memory with A	ANM	0 0 1 0 0 1 1 1 0 0	A ∩ M → A	NZ	1/1
AND Memory with A	ANMD d	0 1 1 0 0 1 1 1 0 0 / d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	A ∩ M → A	NZ	2/2
OR Memory with A	ORM	0 0 0 0 0 0 1 1 0 0	A ∪ M → A	NZ	1/1
OR Memory with A	ORMD d	0 1 0 0 0 0 1 1 0 0 / d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	A ∪ M → A	NZ	2/2
EOR Memory with A	EORM	0 0 0 0 0 1 1 1 0 0	A ⊕ M → A	NZ	1/1
EOR Memory with A	EORMD d	0 1 0 0 0 1 1 1 0 0 / d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	A ⊕ M → A	NZ	2/2

Compare Instruction

OPERATION	MNEMONIC	OPERATION CODE	FUNCTION	STATUS	WORD/CYCLE
Immediate Not Equal to Memory	INEM i	0 0 0 0 1 0 i_3 i_2 i_1 i_0	i ≠ M	NZ	1/1
Immediate Not Equal to Memory	INEMD i,d	0 1 0 0 1 0 i_3 i_2 i_1 i_0 / d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	i ≠ M	NZ	2/2
A Not Equal to Memory	ANEM	0 0 0 0 0 0 0 1 0 0	A ≠ M	NZ	1/1
A Not Equal to Memory	ANEMD d	0 1 0 0 0 0 0 1 0 0 / d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	A ≠ M	NZ	2/2
B Not Equal to Memory	BNEM	0 0 0 1 0 0 0 1 0 0	B ≠ M	NZ	1/1
Y Not Equal to Immediate	YNEI i	0 0 0 1 1 1 i_3 i_2 i_1 i_0	Y ≠ i	NZ	1/1
Immediate Less or Equal to Memory	ILEM i	0 0 0 0 1 1 i_3 i_2 i_1 i_0	i ≦ M	NB	1/1
Immediate Less or Equal to Memory	ILEMD i,d	0 1 0 0 1 1 i_3 i_2 i_1 i_0 / d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	i ≦ M	NB	2/2
A Less or Equal to Memory	ALEM	0 0 0 0 0 1 0 1 0 0	A ≦ M	NB	1/1
A Less or Equal to Memory	ALEMD d	0 1 0 0 0 1 0 1 0 0 / d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	A ≦ M	NB	2/2
B Less or Equal to Memory	BLEM	0 0 1 1 0 0 0 1 0 0	B ≦ M	NB	1/1
A Less or Equal to Immediate	ALEI i	1 0 1 0 1 1 i_3 i_2 i_1 i_0	A ≦ i	NB	1/1

RAM Bit Manipulation Instruction

OPERATION	MNEMONIC	OPERATION CODE	FUNCTION	STATUS	WORD/CYCLE
Set Memory Bit	SEM n	0 0 1 0 0 0 0 1 n_1 n_0	1→M(n)		1/1
Set Memory Bit	SEMD n,d	0 1 1 0 0 0 0 1 n_1 n_0 / d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	1→M(n)		2/2
Reset Memory Bit	REM n	0 0 1 0 0 0 1 0 n_1 n_0	0→M(n)		1/1
Reset Memory Bit	REMD n,d	0 1 1 0 0 0 1 0 n_1 n_0 / d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	0→M(n)		2/2
Test Memory Bit	TM n	0 0 1 0 0 0 1 1 n_1 n_0		M(n)	1/1
Test Memory Bit	TMD n,d	0 1 1 0 0 0 1 1 n_1 n_0 / d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0		M(n)	**2/2**

ROM Address Instruction

OPERATION	MNEMONIC	OPERATION CODE	FUNCTION	STATUS	WORD/CYCLE
Branch on Status 1	BR b	1 1 b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0		1	1/1
Long Branch on Status 1	BRL u	0 1 0 1 1 1 p_3 p_2 p_1 p_0 / d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0		1	2/2
Long Jump Unconditionally	JMPL u	0 1 0 1 0 1 p_3 p_2 p_1 p_0 / d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0			2/2
Subroutine Jump on Status 1	CAL a	0 1 1 1 a_5 a_4 a_3 a_2 a_1 a_0		1	1/2
Long Subroutine Jump on Status 1	CALL u	0 1 0 1 1 0 p_3 p_2 p_1 p_0 / d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0		1	2/2
Table Branch	TBR p	0 0 1 0 1 1 p_3 p_2 p_1 p_0			1/1
Return from Subroutine	RTN	0 0 0 0 0 1 0 0 0 0			1/3
Return from Interrupt	RTNI	0 0 0 0 0 1 0 0 0 1	1→I/E CA RESTORE	ST	1/3

Input/Output Instruction

OPERATION	MNEMONIC	OPERATION CODE	FUNCTION	STATUS	WORD/CYCLE
Set Discrete I/O Latch	SED	0 0 1 1 1 0 0 1 0 0	1→D(Y)		1/1
Set Discrete I/O Latch Direct	SEDD m	1 0 1 1 1 0 m_3 m_2 m_1 m_0	1→D(m)		1/1
Reset Discrete I/O Latch	RED	0 0 0 1 1 0 0 1 0 0	0→D(Y)		1/1
Reset Discrete I/O Latch Direct	REDD m	1 0 0 1 1 0 m_3 m_2 m_1 m_0	0→D(m)		1/1
Test Discrete I/O Latch	TD	0 0 1 1 1 0 0 0 0 0		D(Y)	1/1
Test Discrete I/O Latch Direct	TDD m	1 0 1 0 1 0 m_3 m_2 m_1 m_0		D(m)	1/1
Load A from R-Port Register	LAR m	1 0 0 1 0 1 m_3 m_2 m_1 m_0	R(m)→A		1/1
Load B from R-Port Register	LBR m	1 0 0 1 0 0 m_3 m_2 m_1 m_0	R(m)→B		1/1
Load R-Port Register from A	LRA m	1 0 1 1 0 1 m_3 m_2 m_1 m_0	A→R(m)		1/1
Load R-Port Register from B	LRB m	1 0 1 1 0 0 m_3 m_2 m_1 m_0	B→R(m)		1/1
Pattern Generation	P p	0 1 1 0 1 1 p_3 p_2 p_1 p_0			**1/2**

Control Instruction

OPERATION	MNEMONIC	OPERATION CODE	FUNCTION	STATUS	WORD / CYCLE
No Operation	NOP	0000000000			1/1
Start Serial	STS	0101001000			1/1
Stand-by Mode	SBY	0101001100			1/1
Stop Mode	STOP	0101001101			1/1

Op-Code Map

R9 H / R8 L	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	NOP	XSPX	XSPY	XSP XY	AN EM			AM				ORM			
	1	RTN	RTNI			ALEM			AMC				EORM			
	2	INEM i(4)														
	3	ILEM i(4)														
	4	LBM(XY)			BNEM				LAB				IB			
	5	LMAIY(X)			AYY				LASPY				IY			
	6	NEGA			RED				LASPX							TC
	7	YNEI i(4)														
	8	XMA(XY)			SEM n(2)				REM n(2)				TM n(2)			
	9	LAM(XY)			LMA(XY)				SMC				ANM			
	A	ROTR	ROTL			DAA			DAS							LAY
	B	TBR p(4)														
	C	XMB(XY)			BLEM				LBA							DB
	D	LMADY(X)			SYY				LYA							DY
	E	TD			SED				LXA				REC			SEC
	F	LWI i(2)														
1	0	LBI i(4)														
	1	LYI i(4)														
	2	LXI i(4)														
	3	LAI i(4)														
	4	LBR m(4)														
	5	LAR m(4)														
	6	REDD m(4)														
	7	LAMR m(4)														
	8	AI i(4)														
	9	LMIIY i(4)														
	A	TDD m(4)														
	B	ALEI i(4)														
	C	LRB m(4)														
	D	LRA m(4)														
	E	SEDD m(4)														
	F	XMRA m(4)														

R9 H / R8 L	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0			ANEMD				AMD				ORMD			
	1			ALEMD				AMCD				EORMD			
	2	INEMD i(4)													
	3	ILEMD i(4)													
	4	COMB			OR				STS				SBY	STOP	
	5	JMPL p(4)													
	6	CALL p(4)													
	7	BRL p(4)													
	8	XMAD			SEMD n(2)				REMD n(2)				TMD n(2)		
	9	LAMD			LMAD				SMCD				ANMD		
	A	LMID i(4)													
	B	P p(4)													
	C	CAL a(6)													
	D														
	E														
	F														
1	0	BR b(8)													
	1														
	2														
	3														
	4														
	5														
	6														
	7														
	8														
	9														
	A														
	B														
	C														
	D														
	E														
	F														

... 1-word/2-cycle Instruction

... 1-word/3-cycle Instruction

... RAM Direct Address Instruction (2-word/2-cycle)

... 2-word/2-cycle Instruction

HMCS400 SERIES

Section Three

Hardware Application Notes

SECTION 3

HARDWARE APPLICATION NOTE

PREFACE

The HMCS402C, HMCS404C, and HMCS408C are 4-bit single chip microcomputers all incorporating the latest CMOS high pressure resistant processes, and are capable of driving directly flourescent display tubes. Their performance characteristics have been substantially upgraded from the previous HMCS40 series in terms of operation speed, functions and program efficiency.

A CPU, clock oscillator, ROM, RAM, I/O timer and serial interface (SCI) are all outfitted on a single chip enabling the product to perform over a wide range from small to large scale applications.

These Hardware Notes deal with application examples utilizing the special functions of the HMCS402C, HMCS404C and HMCS408C. It has been compiled to assist system hardware designers by providing application examples with circuit diagrams, timing charts and program listings. (The examples have been written for the HMCS404C but can be applied to the HMCS402C and HMCS408C.)

Application systems examples in these Notes should be tested by actual operation before being put to practical use.

For additional information reference:

- **Section 1, HMCS400 Series User's Manual**
- **Section 2, HMCS400 Series Software Application Notes**

LCD Driver Application

Low Power Dissipation/Fail Safe Applications

INSTRUCTION SET

CONTENTS

APPLICATION NOTE GUIDE

SYSTEM APPLICATION EXAMPLES

APPLICATION NOTE GUIDE

1. Explanation of Symbols

Symbols and abbreviations used in these Notes are as follows.

(1) Operation

$a \rightarrow b$ = transfer from a to b

$a \leftrightarrow b$ = exchange a with b

+	= addition		
-	= subtraction	$\wedge$	= logical product (AND)
$\times$	= multiplication	$\vee$	= logical sum (OR)
/	= division	$\oplus$	= exclusive OR

(2) Registers within the MCU

A	= Accumulator	B	= B register
W	= W register	X	= X register
SPX	= SPX register	SPY	= SPY register

(3) Flags within the MCU

CA	= Carry	ST	= Status

(4) Other Symbols

= = equivalence symbol

$\neq$ = not equal to

: = denotes labels having consecutive addresses

$ = denotes hexadecimal digits

$>, <, \geq, \leq$ = comparison symbols

MD ($***) = denotes 1 digit in RAM area as used in direct addressing mode ($*** specifies the address location)

MR ($*) = denotes 1 digit in memory registers as used in memory register addressing mode ($* specifies the address number)

MSD = denotes the highest digit in RAM

2. Application Example Configuration

This chapter explains the configuration of each system application example following this chapter.

Each application example in APPLICATION NOTES is divided into 5 sections, as shown in figure 1.

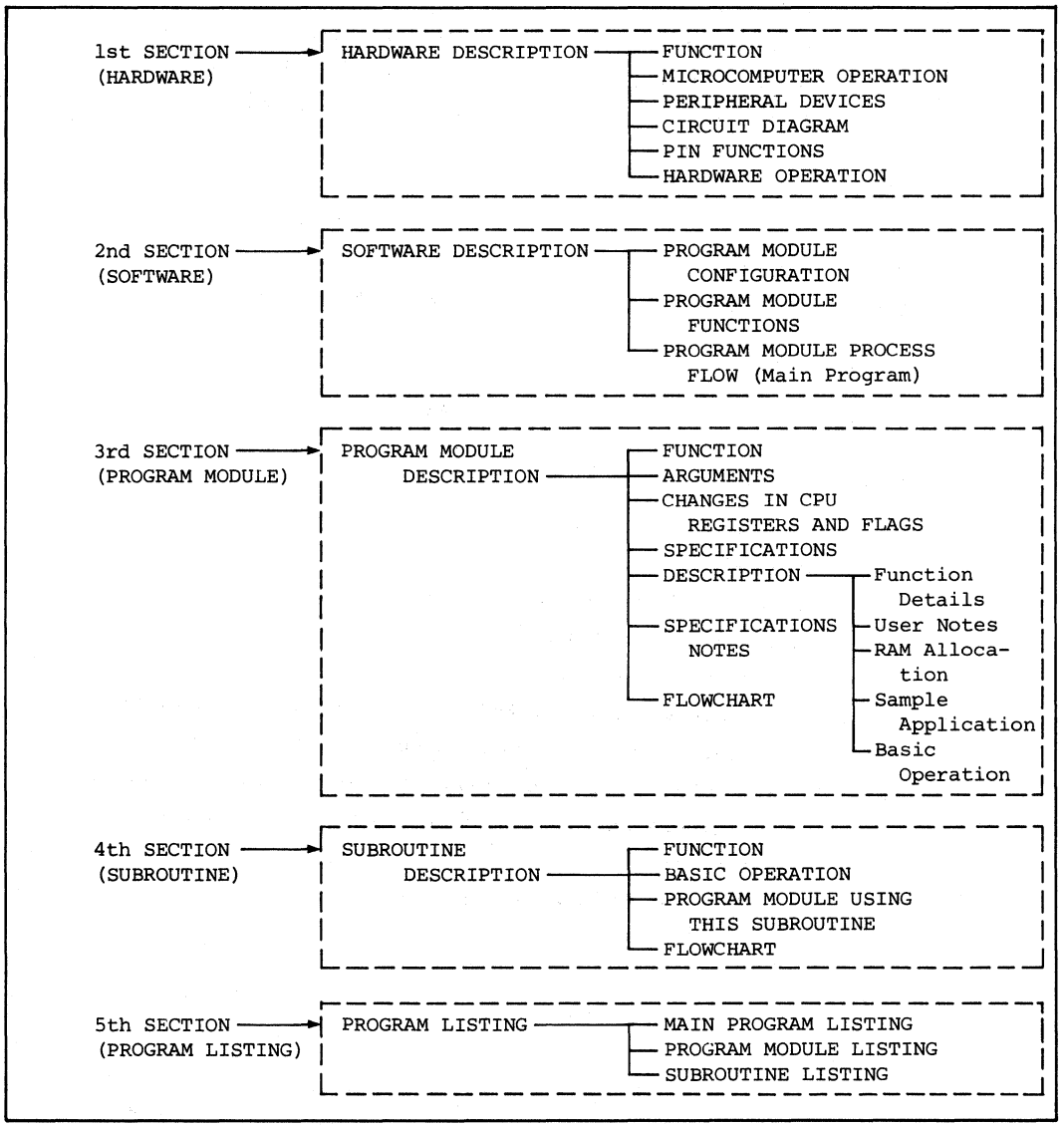


Figure 1. Application Example Configuration

(1) 1st Section (Hardware)

Describes functions, circuit diagram, hardware operation for each hardware application example and making specific use of HMCS400 series characteristic functions.

(2) 2nd Section (Software)

Describes program module configuration which controls hardware application example explained in the 1st Section. Also shows main program of sample application.

(3) 3rd Section (Program Module)

Describes program modules except main program, presented in the 2nd Section, in detail. Each program module is described in the same formal so that users can use them independently.

(4) 4th Section (Subroutine)

Describes subroutine used by each program module. When using program modules explained in the 3rd Section, refer to these subroutines, if necessary.

(5) 5th Section (Program Listing)

Provides program listings for sample application explained in the 1st section.

A detailed explanation of all five sections follows.

3. 1st Section (Hardware)

3.1 Function

Describes system specifications for the hardware used in a particular application.

Example:

14.1.1 Function

Controls LCD module H2570 and displays "CMOS MCU HMCS400" on the liquid crystal display.

3.2 Microcomputer Operation

Describes typical functions of the microcomputer used in a particular application.

Example:

14.1.2 Microcomputer Operation

(1) Controls HD44780 (hereafter abbreviated LCD-II) data bus through ports R4 and R5.

(2) Controls LCD-II control signals (Signals RS, R/W and E) through port D.

(3) To control LCD-II data bus and control signals by HMCS404C software, there are no restrictions in terms of timing.

(4) From HMCS404C display data is transmitted to LCD-II in the form of ASCII code. Liquid crystal driver HD44100 and the liquid crystal display are automatically controlled by LCD-II which is in turn controlled by the HMCS404C.

3.3 Peripheral Devices

Describes typical functions of the peripheral devices used in a particular application.

Example:

14.1.3 Peripheral Devices

(1) LCD controller driver HD44780 (LCD-II): Controls dot matrix LCD of LCD module H2570.

(2) LCD driver HD44100: Drives LCD of LCD module H2570.

(3) LCD module H2570: Provides a display of 16 characters × 1 row.

3.4 Circuit Diagram

Describes the circuit diagram for the hardware example.

Note) All microcomputers described in APPLICATION NOTES use the plastic DIP type package.

Example:

14.1.4 Circuit Diagram

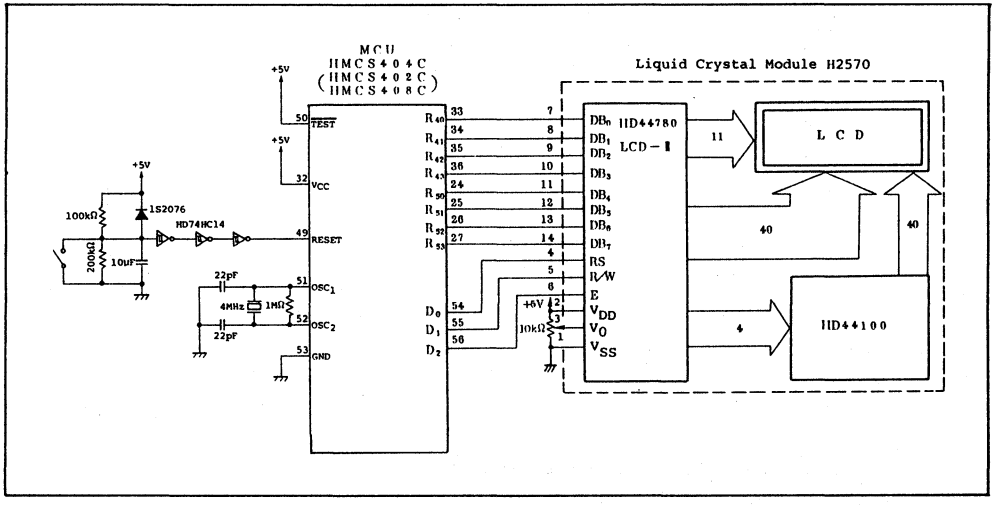


Fig. 14.1. H2570 Control Circuit

I/O options can be selected for the HMCS400 series.
In the circuits shown in this manual, CMOS is selected for standard output ports, with pull-up MOS is selected for standard input ports, with pull-down MOS is selected for high voltage ports.

3.5 Pin Functions

Describes interface between microcomputer and the external circuit using a table.

Example:

14.1.5 Pin Functions

Pin functions at the connecting interface of HMCS404C and LCD-II are shown in Table 14.1.

Table 14.1. Pin Functions

Pin Name (HMCS404C)	Input/ Output	Active level	Function	Pin Name (LCD-II)
D_0	Input/ Output	Low	Selects instruction register	RS
		High	Selects data register	
D_1		Low	Data writing (Microcomputer → LCD-II)	R/W
		High	Data reading (Microcomputer ← LCD-II)	
D_2		High	Enable signal	E
R_{40}		-	Data line	DB0
R_{41}		-		DB_1
R_{42}		-		DB_2
R_{43}		-		DB_3
R_{50}		-		DB_4
R_{51}		-		DB_5
R_{52}		-		DB_6
R_{53}				DB_7

"Active Level" in the table indicates the following:

High : Logical 1
Low : Logical 0
— : Logical 1 or 0

3.6 Hardware Operation

Describes hardware operation for controlling an external circuit using a timing chart.

Example:

14.1.6 Hardware Operation

Control signals from both the HMCS404C and LCD-II is performed with the timing shown in Fig. 14.2.

① Data is written to LCTC at the falling edge of E.

② Data from LCD-II can be read during period T_1.

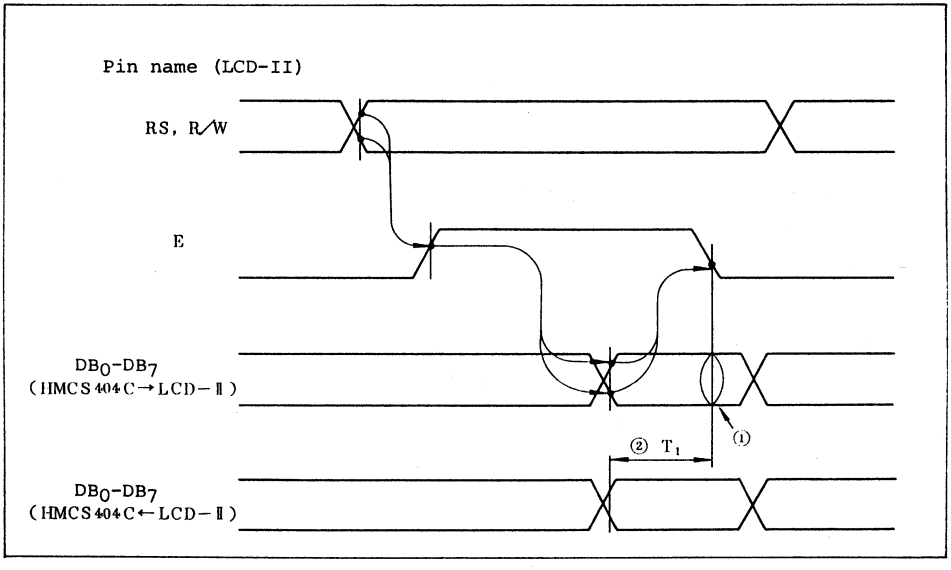


Fig. 14.2. HMCS404C ↔ LCD-II Interface Timing Chart

4. 2nd Section (Software)

4.1 Program Module Configuration

Describes program module configuration to control the hardware application example. Each program module is numbered. No. of main program is 0, and the other program modules are numbered from 1 to N.

Example:

14.2.1 Program Module Configuration

Fig. 14.3 shows the program module configuration for performing display on the liquid crystal.

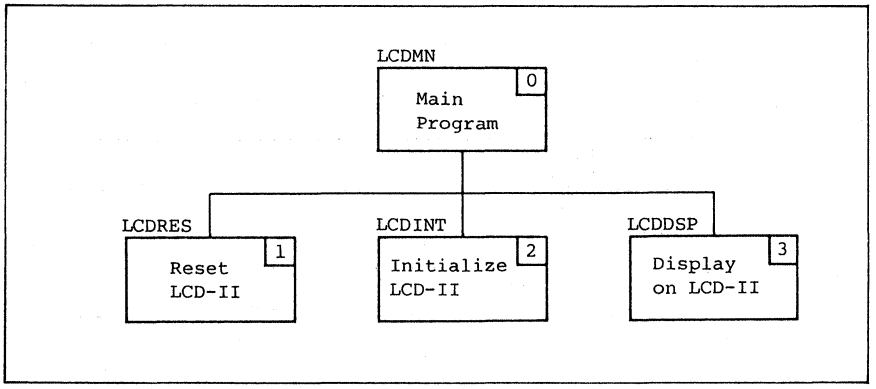


Fig. 14.3. Program Module Configuration

4.2 Program Module Functions

Describes function of each program module using a table. "No." in the table matches "No." in the Program Module Configuration.

Example:

14.2.2 Program Module Functions

Outline of program module functions is shown in Table 14.2.

Table 14.2. Program Module Functions

No.	Program Module Name	Label	Function
0	Main Program	LCDMN	Main program which conducts character display on H2570
1	Reset LCD-II	LCDRES	Resets LCD-II according to instructions
2	Initialize LCD-II	LCDINT	Initializes LCD-II display mode
3	Display on LCD	LCDDSP	Sends ASCII code to LCD-II and displays on H2570

4.3 Program Module Process Flow (Main Program)

Describes sample main program to execute program modules, explained in (1) Program Module Configuration.

Example:

14.2.3 Program Module Process Flow (Main Program)

An example of conducting character display on H2570 using the module shown in Fig. 14.3 is shown in Fig. 14.4. Execution of the main program of Fig. 14.4 results in liquid crystal demonstration display as shown in Fig. 14.5.

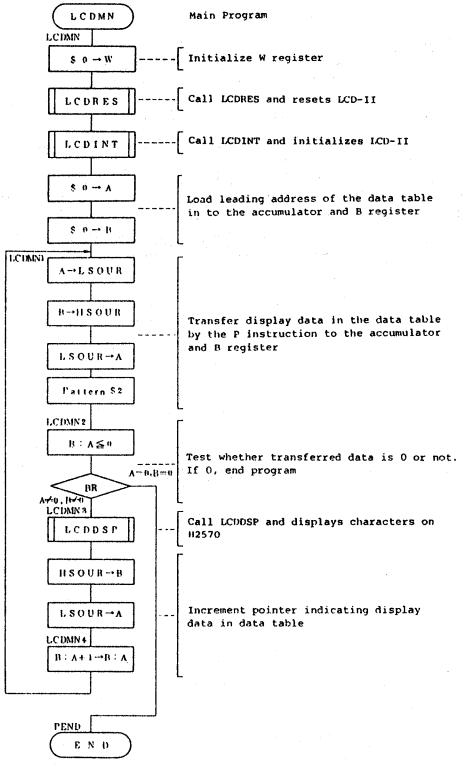
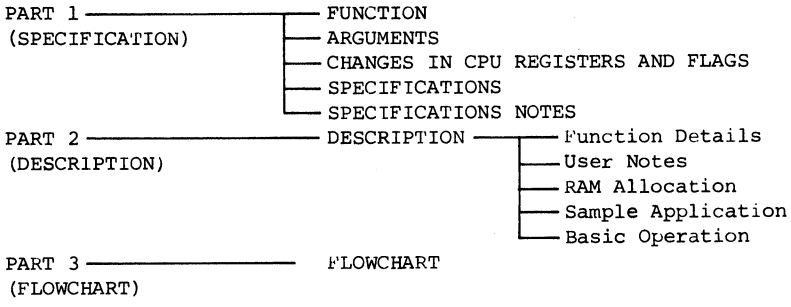


Fig. 14.4. Example of Program Module Sample Application

Fig. 14.5. Example of Liquid Crystal Display

5. 3rd Section (Program Module)

The 3rd Section consists of the parts as shown in figure 2.



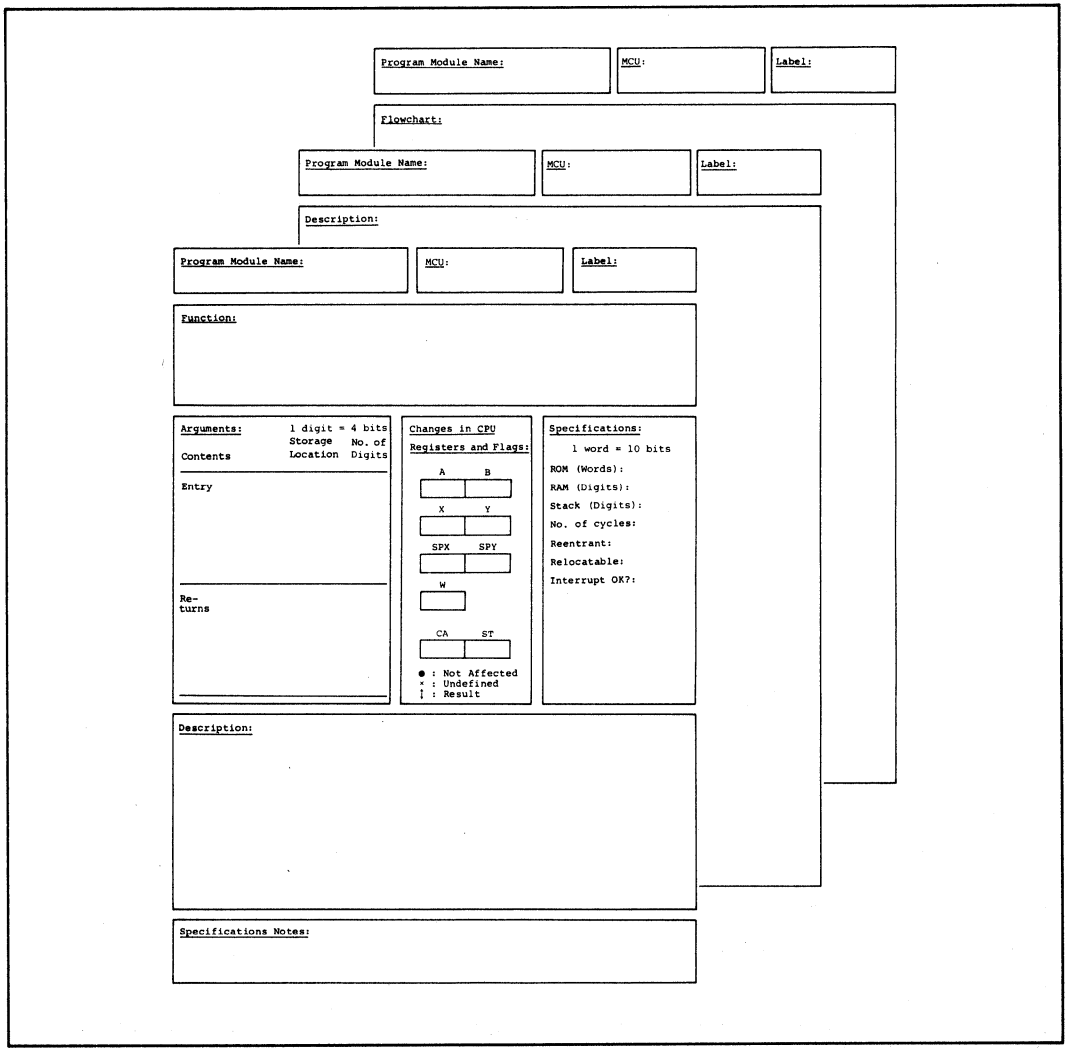


Figure 2. Program Module Section

5.1 Specification

The Specification Part is shown in figure 3. ( ): blocked off area in figure 3). This part explains function, arguments, changes in CPU registers and flags, specifications and specifications notes. Each numbered item in the figure is described below.

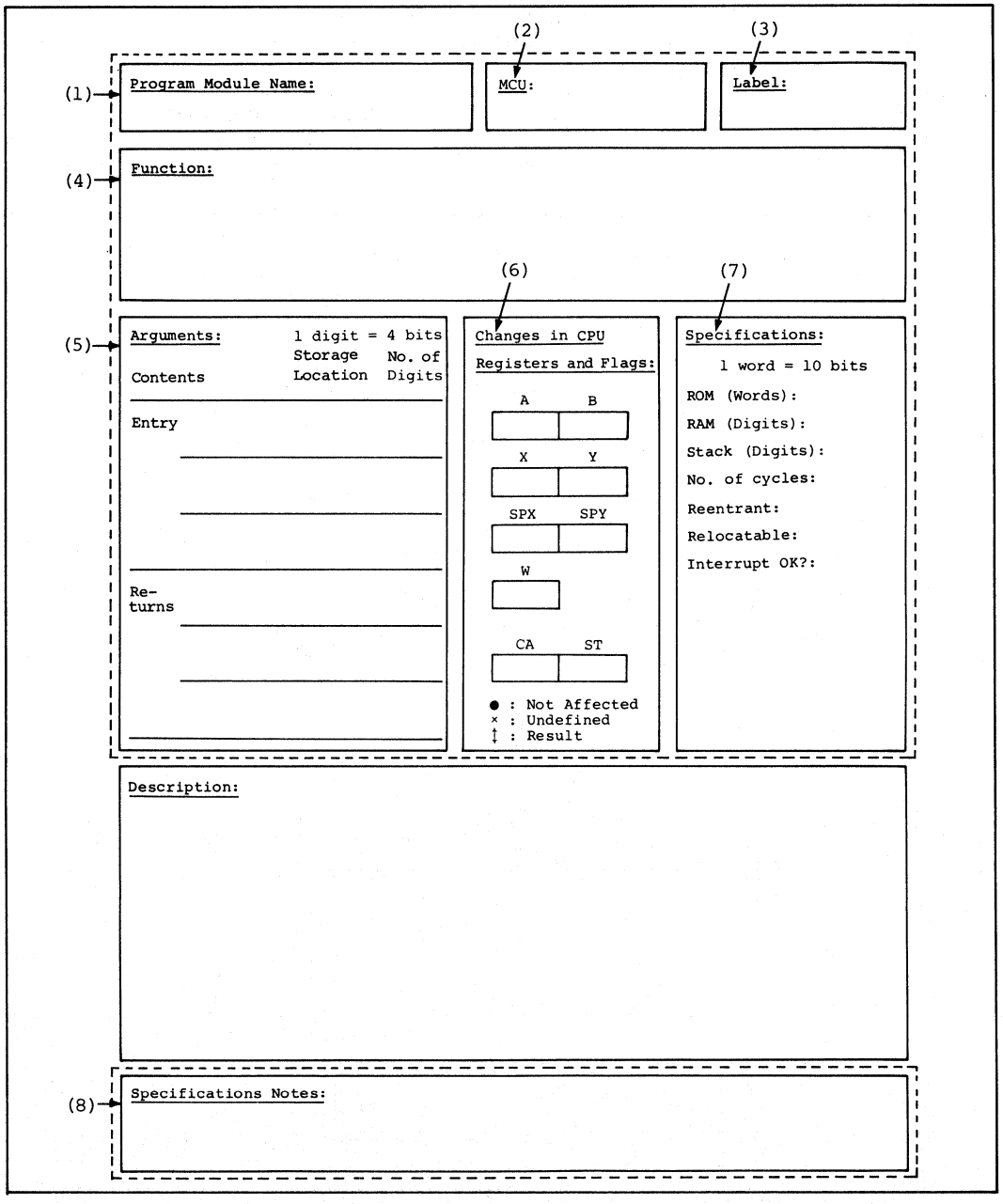


Figure 3. Specification Part

(1) PROGRAM MODULE NAME: Indicates the name of the module.

Example:

Program Module Name:
Display on LCD

(2) MCU: Indicates the names of microcomputers for which this module can be applied.

MCU: HMCS402C/ HMCS404C/HMCS408C

(3) LABEL: Indicates the name identifying program entry point. When using the program without modification, use this label to call the program.

Example:

Label: LCDDSP

(4) FUNCTION: Describes program function.

Example:

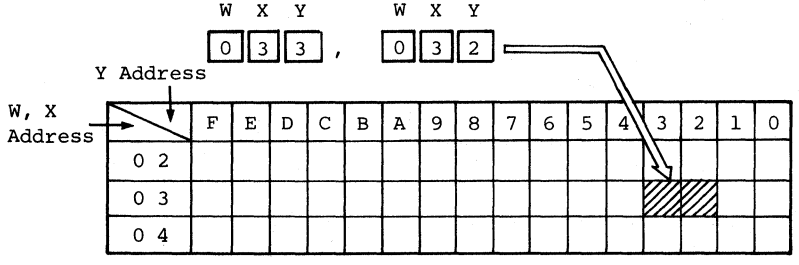
Function: Writes ASCII code in DDRAM of LCD-II and displays graphically on liquid crystal.

(5) ARGUMENTS: Describes entry arguments which must be initialized before program execution, and return arguments after execution.

(a) Contents: Describes arguments' contents, e.g., constant, starting address, string length.

(b) Storage Location: Indicates registers and RAMs in which arguments must be stored. RAM locations are denoted by "(RAM)".

Note: Absolute storage locations in RAM address space are designated by MD ($WXY, $WXY) using W, X and Y addresses. For example, MD ($033, $032) refers to the marked area in the memory array shown below.



W, X, Y correspond to registers.

W, X, Y which are used to store memory addresses.

(c) No. of Digits: Indicates arguments' digit length.

Example:

Arguments:		1 digit = 4 bits	
Contents		Storage Location	No. of Digits
Entry Display data (ASCII code)	Upper	B	1
	Lower	A	1
Returns	___	___	___

(6) CHANGES IN CPU REGISTERS AND FLAGS: Describes changes in CPU registers after execution of a program module as well as flag changes in CA and ST.

Symbols and abbreviations used here are as follows:

(a) CPU registers

A:	Accumulator	B:	B register
W:	W register		
X:	X register	Y:	Y register
SPX:	SPX register	SPY:	SPY register

(b) Flags

CA:	Carry	ST:	Status

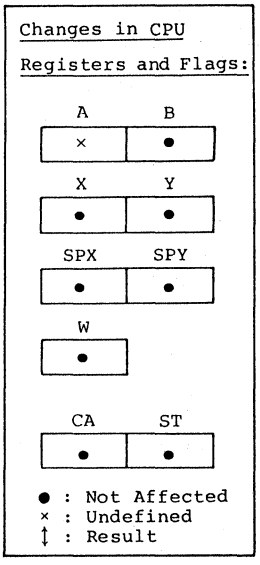
(c) Status of CPU registers and flags after execution of the program module:

Not Affected: Previous contents are retained after executing the program module.

Undefined: Previous contents are destroyed after executing the program module.

Result: A result is stored after executing the program module.

Example:



Note: In the example shown, the contents of the Accumulator, B register and Y register are destroyed upon executing the program module. Thus, whenever necessary, registers which will be destroyed should be saved before executing a program.

(7) SPECIFICATIONS: Describes program operation specifications.

(a) ROM (Words): Indicates amount of ROM used by the program. 1 word consists of 10 bits.

(b) RAM (Digits): Indicates amount of RAM used by the program. 1 digit consists of 4 bits. (This value does not include memory needed for the stack.)

(c) Stack (Digits): Indicates amount of RAM used by the stack in the program. This memory must be reserved when the program is executed.

(d) No. of cycles: Indicates the maximum number of machine cycles. Calculate the execution time required for program execution as follows:

Execution time (sec) = Number of cycles × cycle time

Cycle time (sec) = 8/External oscillator frequency (Hz)

Note: BRS instruction is regarded as 1 cycle.

(e) Reentrant: Indicates whether a program has a structure which can be called from two or more routines at the same time.

(f) Relocatable: Indicates whether a program can be located in any memory space.

(g) Interrupt OK?: Indicates whether MCU can continue a program normally after serving an interrupt routine. If cannot ("No"), inhibit interrupt before the program is called.

Example:

Specifications:	
1 word = 10 bits	
ROM (Words):	13
RAM (Digits):	0
Stack (Digits):	0
No. of cycles:	29
Reentrant:	No
Relocatable:	No
Interrupt OK?:	No

(8) SPECIFICATIONS NOTES: Explanatory notes for items listed in (7) SPECIFICATIONS.

Example:

Specifications Notes:

5.2 Description

The Description Part is shown in figure 4. ( ): blocked off area in figure 4). This part explains function details, user notes, RAM allocation, sample application and basic operation. Each numbered item in the figure is described below.

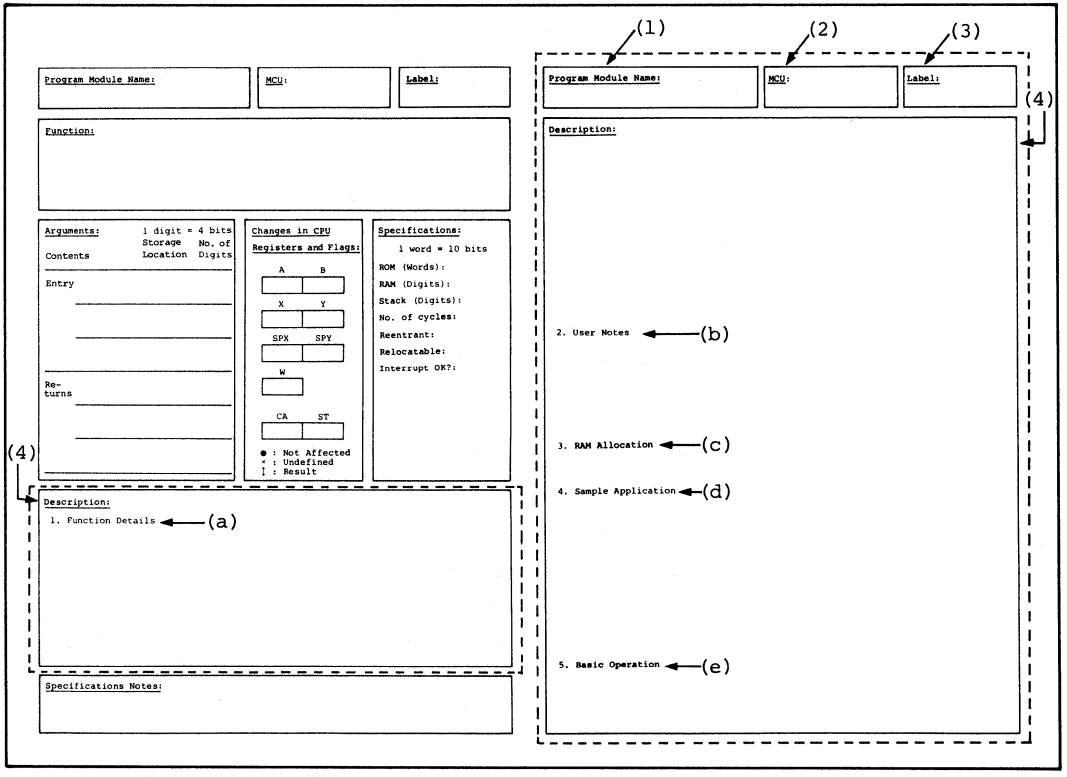


Figure 4. Description Part

(1) PROGRAM MODULE NAME
(2) MCU
(3) LABEL
} Same as "5.1 Specification"

(4) DESCRIPTION: Describes function details, user notes, RAM allocation, sample application and basic operation of the program module.

(a) Function Details: Describes detailed functions of the program module referring to the execution example.

Example:

Description:

1. Function Details

(1) Argument details

A: Stores the lower 4 bits of the ASCII code of the graphic display.

B: Stores the upper 4 bits of the ASCII code of the graphic display.

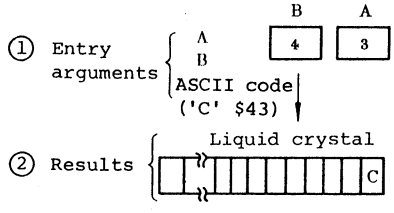


Fig. 14.7. Example of LCDDSP Execution

(b) User Notes: Describes notes and limitations when executing the program module.

*Be sure to read these items when using program modules without modification.

Example:

2. User Notes

(1) Resets LCD-II.

(2) Initializes LCD-II display mode.

(3) Stores entry arguments.

(c) RAM Allocation: Describes labels and contents of RAM used in program module.

Example:

3. RAM Allocation

RAM is not used by program module LCDDSP.

(d) Sample Application: Shows a sample application for actual execution of the program.

Note: Initializing stack pointer is not shown in this part.

Example:

4. Sample Application

```
        :
CALL    LCDRES    ..... Execute LCDRES and reset LCD-II
CALL    LCDINT    ..... Execute LCDINT and initialize LCD-II
                        display mode
```

(e) Basic Operation: Explains how a program module is executed.

Example:

5. Basic Operation

(1) Checks LCD-II busy flag.

(2) Controls signals RS, R/W and E by software through port D and outputs display data.

5.3 Flowchart

The Flowchart Part is shown in figure 5. This part gives the program module flowchart.

Example:

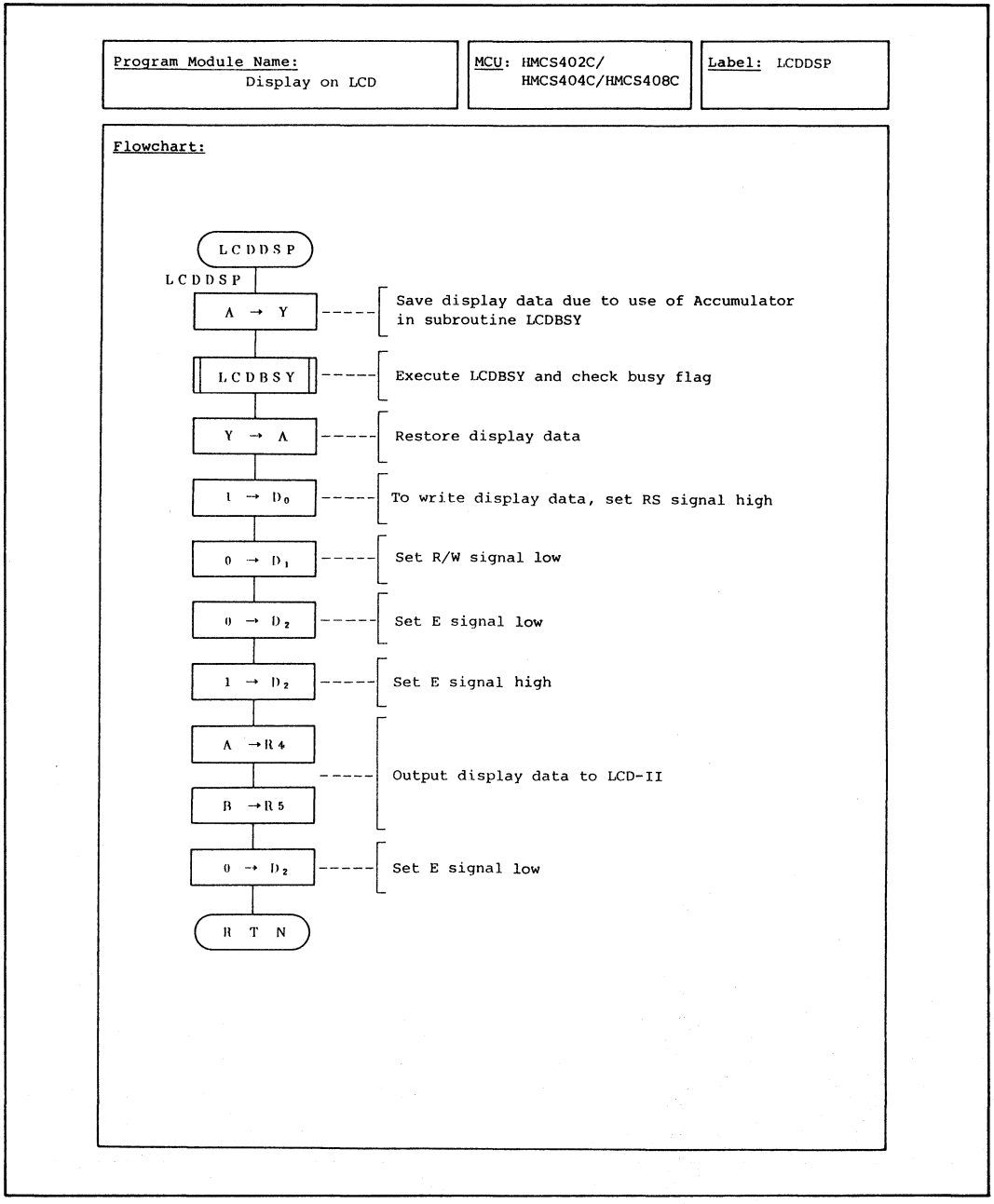


Figure 5. Flowchart Part

6. 4th Section (Subroutine)

The Subroutine Section is shown in figure 6. Each numbered item is described as follows.

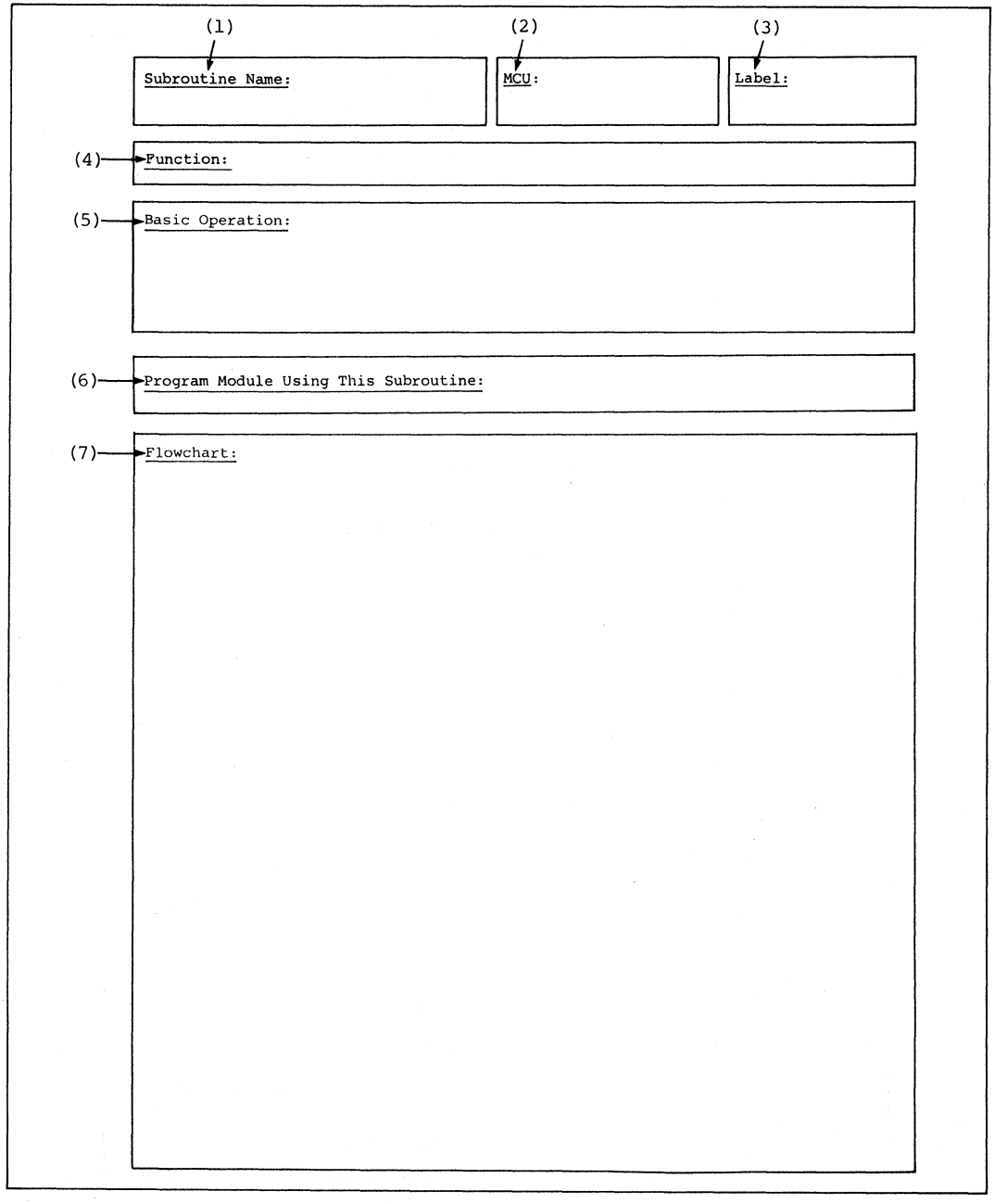


Figure 6. Subroutine Section

(1) SUBROUTINE NAME:

Example:

Subroutine Name: Busy Check

(2) MCU: Indicates microcomputer or microprocessor applicable to the subroutine.

Example:

MCU: HMCS402C/ HMCS404C/HMCS408C

(3) LABEL: Indicates the name identifying subroutine entry point. When using the subroutine without modification, use this label to call the subroutine.

Example:

Label: LCDBSY

(4) FUNCTION: Describes subroutine function.

Example:

Function: Checks if LCD-II is busy and waits for ready state.

(5) BASIC OPERATION: Explains how a subroutine is executed.

Example:

Basic Operation:	(1) Since, while LCD-II is in operation, it will not allow access from microcomputers, checks must be made on the LCD-II busy flag. (2) Control of signals RS, R/W and E through port D is performed by software together with busy flag check.

(6) PROGRAM MODULE USING THIS SUBROUTINE: Lists program modules using the subroutine.

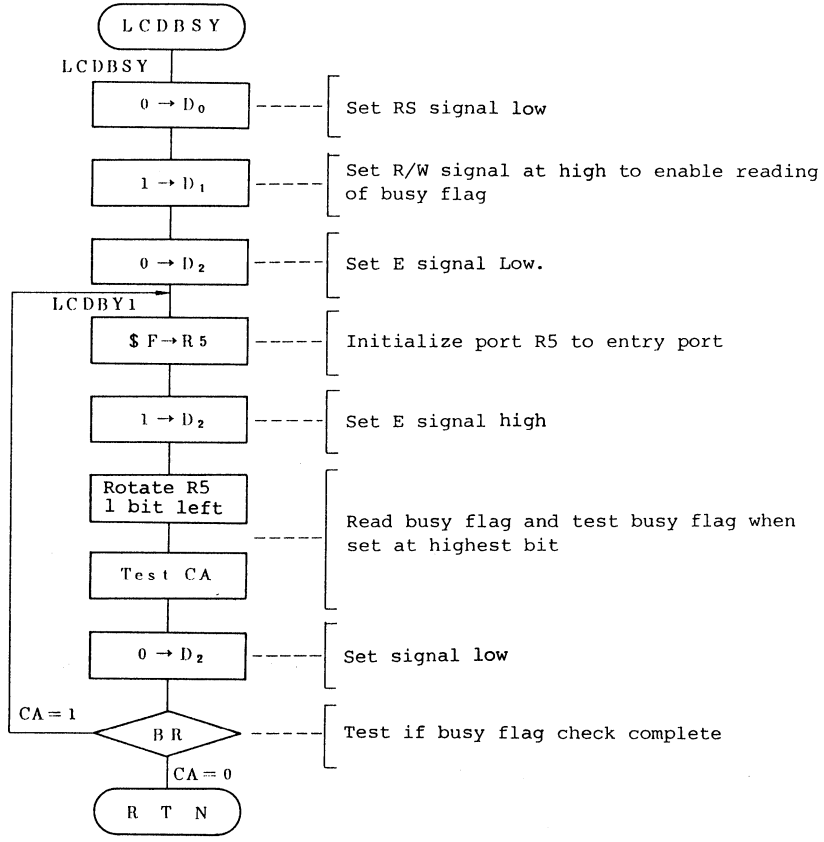
Example:

Program Module Using This Subroutine: LCDDSP, LCDINT, LCDRES

(7) FLOWCHART: Gives subroutine flowchart.

Example:

Flowchart:



7. 5th Section (Program Listing)

The Program Listing Section explains RAM allocation and CPU register allocation, and gives program module and subroutine listings.

(1) RAM Allocation: RAM used in program modules or subroutines is allocated as shown below.

Example:

```
00003                   *
00004                   ****   RAM ALLOCATION   ****************************************
00005                   *
00006                   LSOUR    EQU      $020          WORK AREA FOR ACCA
00007                   HSOUR    EQU      $021          WORK AREA FOR B REGISTER
```

(a) The title "RAM ALLOCATION" is followed by the actual RAM allocation used.

(b) RAM label.

(2) Vector Address: Describes vector address allocation.

Example:

```
00008                   ************************************************************
00009                   *                                                          *
00010                   *              VECTOR ADDRESSES                            *
00011                   *                                                          *
00012                   ************************************************************
00013                   *
00014                           ORG      $0000
00015                   *
00016   150 040  0000           JMPL     LCDMN         RESET
00017   150 040  0002           JMPL     LCDMN         INTO
00018   150 040  0004           JMPL     LCDMN         INT1
00019   150 040  0006           JMPL     LCDMN         TIMER A
00020   150 040  0008           JMPL     LCDMN         TIMER B
00021                   *
00022                           ORG      $000C
00023                   *
00024   150 040  000C           JMPL     LCDMN         SERIAL
```

(a) The title is always "VECTOR ADDRESSES".

(b) Indicates the end of a program.
This can be moved, if necessary.

(3) Main Program: Gives main program listing of a sample application.

Example:

```
00025                   ****************************************************************
00026                   *                                                              *
00027                   *              MAIN PROGRAM : LCDMN                            *
00028                   *                                                              *
00029                   ****************************************************************
00030                   *
00031                            ORG     $0040
00032                   *
00033   0F0      0040   LCDMN    LWI     $0          INITIALIZE W REGISTER
00034   160 064  0041           CALL    LCDRES      RESET LCD-II
00035   160 07E  0043           CALL    LCDINT      INITIALIZE LCD-II
00036   230      0045           LAI     $0          LOAD DATA TABLE STARTING ADDR INTO ACCA
00037   200      0046           LBI     $0          LOAD DATA TABLE STARTING ADDR INTO B REG
00038   194 020  0047   LCDMN1  LMAD    LSOUR       ACCA ---> LSOUR
00039   048      0049           LAB
00040   194 021  004A           LMAD    HSOUR       B REGISTER ---> HSOUR
00041   190 020  004C           LAMD    LSOUR       LSOUR ---> ACCA
00042   1B2      004E           P       $2          MOVE DISPLAY TO ACCA & B REGISTER
00043   0D8      004F           LYA
00044   2B0      0050           ALEI    $0          TEST IF ACCA=0
00045   048      0051           LAB
00046   354      0052           BR      LCDMN2      IF 0, BRANCH TO LCDMN2
00047   356      0053           BR      LCDMN3      IF NOT 0, BRANCH TO LCDMN3
00048   2B0      0054   LCDMN2  ALEI    $0          TEST IF B REGISTER=0
00049   363      0055           BR      PEND        IF 0, BRANCH TO PEND
00050   0AF      0056   LCDMN3  LAY
00051   160 090  0057           CALL    LCDDSP      DISPLAY FIGURE ON LCD-II
00052   190 021  0059           LAMD    HSOUR       HSOUR ---> B REGISTER
00053   0C8      005B           LBA
00054   190 020  005C           LAMD    LSOUR       LSOUR ---> ACCA
00055   281      005E           AI      $1          ACCA + $1 ---> ACCA
00056   361      005F           BR      LCDMN4      BRANCH IF ACCA =$F
00057   347      0060           BR      LCDMN1
00058   04C      0061   LCDMN4  IB                  INCREMENT B REGISTER
00059   347      0062           BR      LCDMN1      LOOP UNTIL B REGISTER = $F
```

(a) The title is always "MAIN PROGRAM". Label after color shows entry point label.

(b) Entry point label.

(4) Program Module: Gives program module listing of a sample application.

Example:

```
00060   363      0063   PEND     BR      PEND
00061                   ****************************************************************
00062                   *                                                              *
00063                   *       NAME : LCDRES (RESET LCD-II)                           *
00064                   *                                                              *
00065                   ****************************************************************
00066                   *                                                              *
00067                   *      ENTRY : NOTHING                                         *
00068                   *    RETURNS : NOTHING                                         *
00069                   *                                                              *
00070                   ****************************************************************
00071   202      0064   LCDRES   LBI     $2          INITIALIZE LOOP COUNTER
00072   239      0065   LCDRS1  LAI     $9          EXECUTE 15MS SOFTWARE TIMER
00073   21F      0066   LCDRS2  LYI     $F
00074   002      0067   LCDRS3  XSPY
00075   21F      0068           LYI     $F
00076   000      0069   LCDRS4  NOP
```

(a) Program module title is always followed by the entry point label in parenthesis and description of entry and return arguments.

(b) Entry point label.

(5) Subroutine: Gives subroutine listing.

Example:

```
00145   010     009C            RTN
00146                   ********************************************************************
00147                   *                                                                  *
00148                   *          NAME : LCDBSY (CHECK BUSY FLAG)                         *
00149                   *                                                                  *
00150                   ********************************************************************
00151   260     009D    LCDBSY  REDD    $0              RS=0
00152   2E1     009E            SEDD    $1              R/W=1
00153   262     009F            REDD    $2              E=0
00154   23F     00A0    LCDBY1  LAI     $F              SELECT R PORT AS INPUT
00155   2D5     00A1            LRA     $5
00156   2E2     00A2            SEDD    $2              E=1
00157   255     00A3            LAR     $5              READ BUSY FLAG
00158   0A1     00A4            ROTL
```

(a) Subroutine title is followed by the entry point label in parenthesis.

(b) Entry point label.

(6) Data Table: Describes data table used in the main program, program modules and subroutines.

Example:

```
00159   06F     00A5            TC                      TEST CARRY
00160   262     00A6            REDD    $2              E=0
00161   3A0     00A7            BR      LCDBY1          LOOP UNTIL BUSY FLAG=0
00162   010     00A8            RTN
00163                   ********************************************************************
00164                   *                                                                  *
00165                   *          DATA TABLE                                              *
00166                   *                                                                  *
00167                   ********************************************************************
00168                   *
00169                           ORG     $0100
00170                   *
00171   10C     0100    $00     DC      $10C
00172   118     0101    $01     DC      $118
00173   191     0102    $02     DC      $191
00174   107     0103    $03     DC      $107
00175   101     0104    $04     DC      $101
00176   108     0105    $05     DC      $108
00177                   *
00178                           ORG     $0200
00179                   *
00180   143     0200    $00     DC      $143
00181   14D     0201    $01     DC      $14D
00182   14F     0202    $02     DC      $14F
00183   153     0203    $03     DC      $153
00184   120     0204    $04     DC      $120
00185   14D     0205    $05     DC      $14D
00186   143     0206    $06     DC      $143
00187   155     0207    $07     DC      $155
00188   120     0208    $08     DC      $120
00189   148     0209    $09     DC      $148
00190   14D     020A    $0A     DC      $14D
00191   143     020B    $0B     DC      $143
00192   153     020C    $0C     DC      $153
00193   134     020D    $0D     DC      $134
00194   130     020E    $0E     DC      $130
00195   130     020F    $0F     DC      $130
00196   100     0210    $10     DC      $100
00197                   *
00198                           END
```

(a) The title is always "DATA TABLE".

(b) Data table label.

8. Program Module Execution

The programs in APPLICATION NOTE have been written considering efficiency and portability. The following shows how to execute these programs and how to modify them according to user requirements.

The procedure for calling programs in APPLICATION NOTE from user programs is shown in figure 7. All programs in APPLICATION NOTE are written as subroutines and should be called as shown. An example of a user program in which a program in APPLICATION NOTE is called as a subroutine is shown in figure 8.

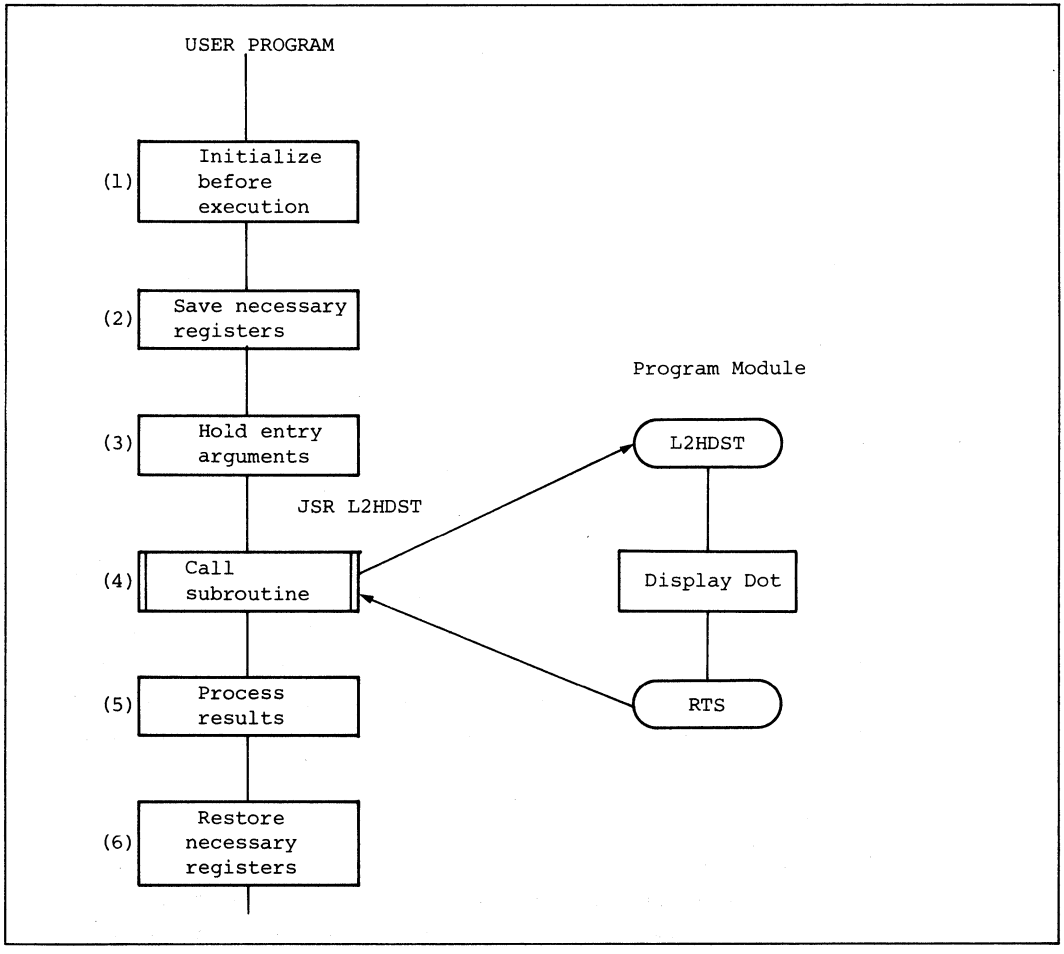


Figure 7. Procedure for Calling Program Module in APPLICATION NOTE

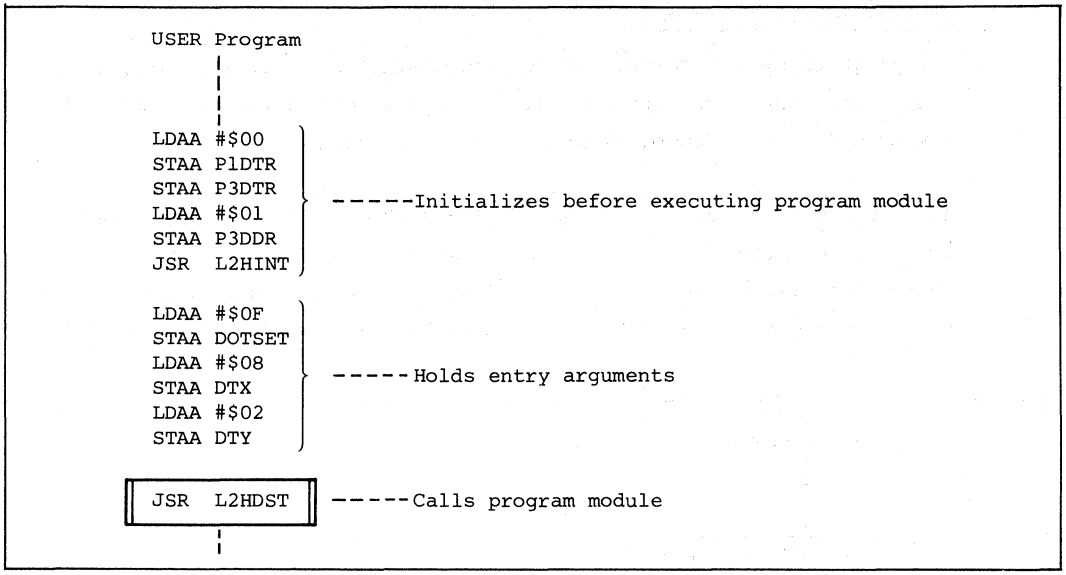


Figure 8. Example of How to Execute a Program Module

Explanation of figure 7.

(1) Initialize

Examples of items requiring initialization are input/output ports, control registers and counters used by the program module. Refer to Program Module Sample Application for data details.

(2) Save registers

The program modules use CPU registers for arithmetic operations destroying the original contents. Thus register contents should be saved if needed. Refer to the "CHANGES OF CPU REGISTERS AND FLAGS" column in "SPECIFICATIONS" (Part 1 in the 3rd SECTION-PROGRAM MODULE DESCRIPTION) for register status after a program module is executed.

(3) Hold entry arguments

Holds entry arguments in CPU registers or memory before calling a program module in the user program. Refer to "ARGUMENTS" in SPECIFICATIONS (Format 1 in 3RD SECTION - Program Module) for details.

(4) Call subroutine

Program module is called.

(5) Process result

After a program module is executed, the results returned in the return arguments must be processed as required. Refer to "ARGUMENTS" in SPECIFICATIONS (Format 1 in 3RD SECTION - Program Module) for details.

(6) Restore necessary registers

Registers saved in (2) should be restored here. Note that when a program module is used as a subroutine, the stack area shown in SPECIFICATIONS (Format 1 in 3RD SECTION - Program Module) is necessary in addition to the stack area required by the subroutine calls in the user program. When any subroutine is called, this stack area must be reserved.

SYSTEM APPLICATION EXAMPLES

1.1 HARDWARE DESCRIPTION

1.1.1 Function

Measures the input AC power frequency (50 Hz, 60 Hz) by transforming the single phase 100 VAC wave configuration into a pulse wave configuration having the same frequency.

1.1.2 Microcomputer Operation

Tests input at port D_3 from the pulse conversion circuit and turns on LED corresponding to 50 Hz or 60 Hz using ports D_0 and D_1; this measurement is performed only once after system reset.

1.1.3 Peripheral Devices

1.1.4 Circuit Diagram

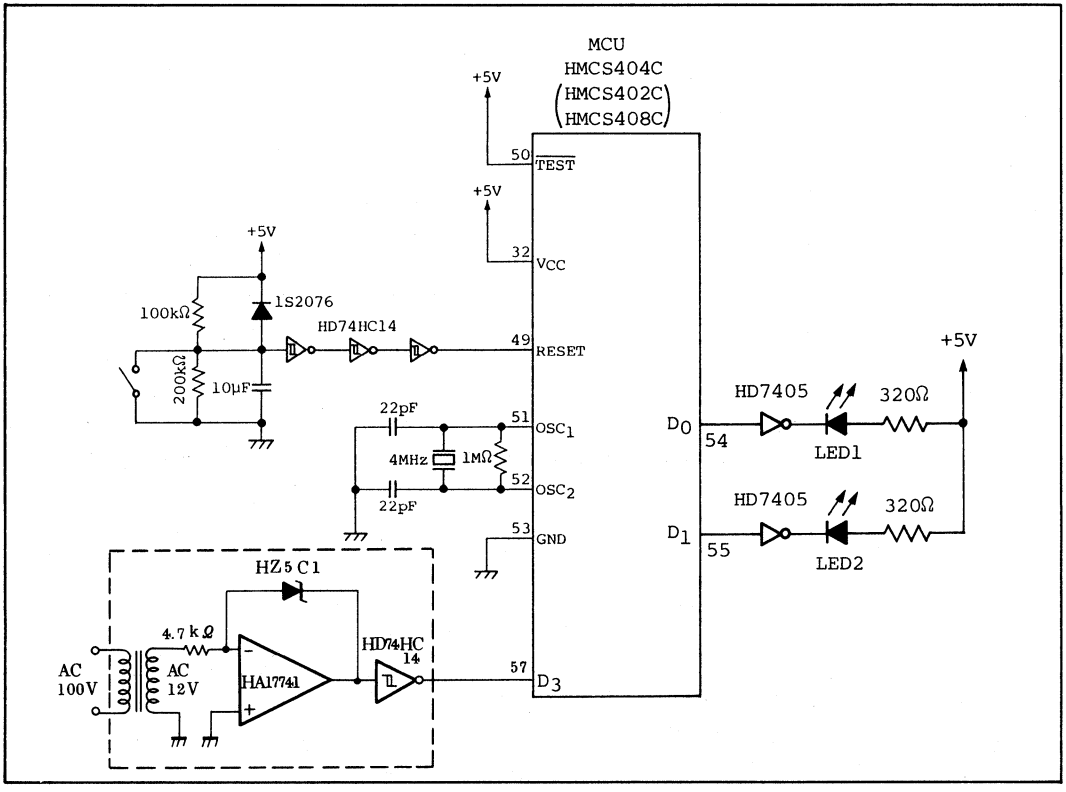


Fig. 1.1. Measurement of AC Frequency

1.1.5 Pin Functions

Pin function for connecting interface of HMCS404C and pulse conversion circuit are shown in Table 1.1.

Table 1.1. Pin Function

Pin Name (HMCS404C)	Input/ Output	Active Level (High or Low)	Function	Program Label
D_3	Input	Low	AC Pulse Signal Input	PORTD3
D_0	Output	High	Measurement Result (60 Hz) Output	-
D_1	Output	High	Measurement Result (50 Hz) Output	-

1.1.6 Hardware Operation

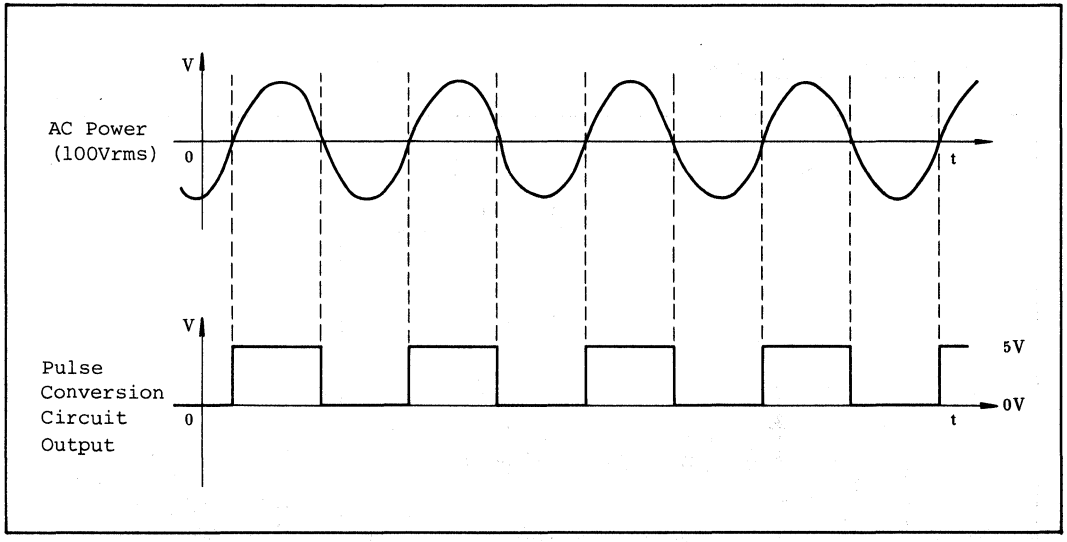


Fig. 1.2. Timing Chart for AC and Pulse Conversion Circuit Output

1.2.1 Program Module Configuration

Fig. 1.3 shows the program modules for measuring AC frequency.

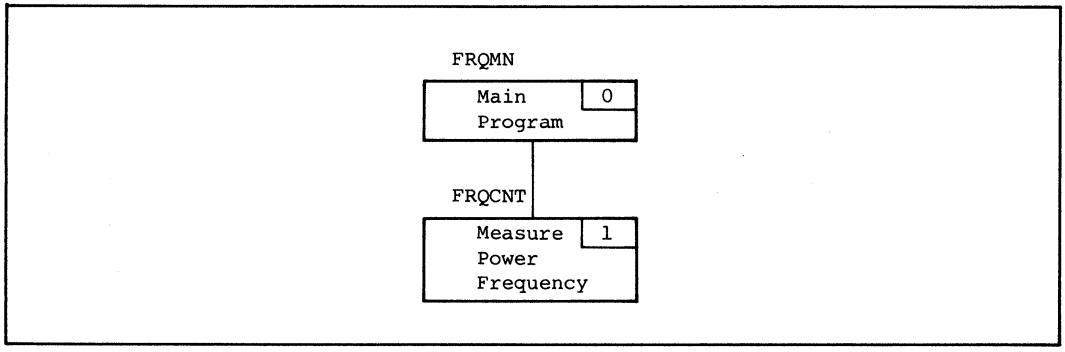


Fig. 1.3. Program Module Configuration

1.2.2 Program Module Functions

Program module functions are summarized in Table 1.2.

Table 1.2. Program Module Functions

No.	Program Module Name	Label	Function
0	Main Program	FRQMN	Main program which measures AC power frequency and turns on LED corresponding to the above frequency
1	Measure Power Frequency	FRQCNT	Measures AC power frequency and stores the 50 Hz, 60 Hz frequency results into RAM

1.2.3 Program Module Process Flow (Main Program)

Fig. 1.4 shows an example of measuring AC power frequency by using the modules shown in Fig. 1.3.

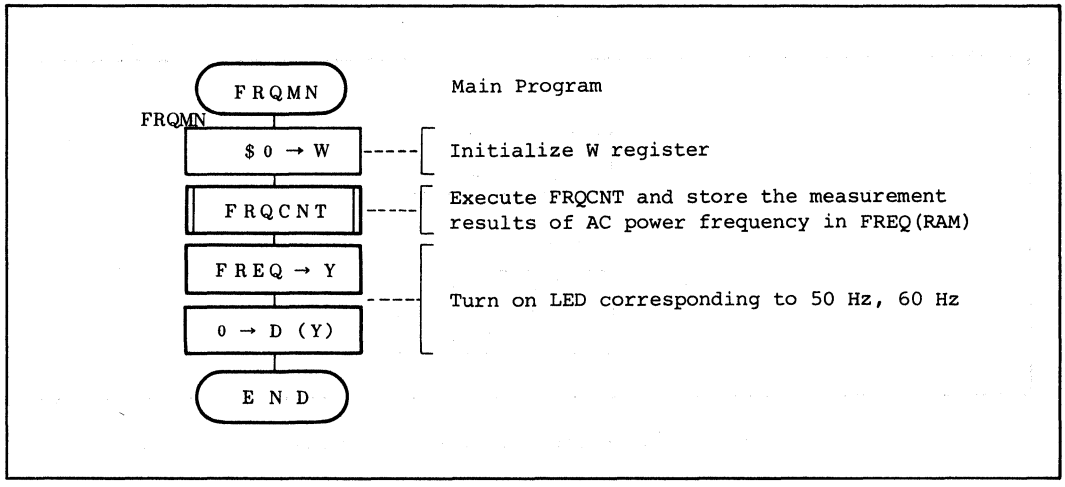


Fig. 1.4. Program Module Application Example

1.3 Program Module Description

Program Module Name:	MCU:	Label:
Measure Power Frequency	HMCS402C/ HMCS404C/HMCS408C	FRQCNT

Function:

Tests whether pulse frequency as inputted from port D_3 is 50 Hz or 60 Hz.

Arguments:

	Contents	Storage Location	1 digit = 4 bits No. of Digits
Entry	—	—	—
Returns	Frequency measurement result	FREQ	1

Changes in CPU Registers and Flags:

A	B
×	×

X	Y
●	×

SPX	SPY
●	×

W
●

CA	ST
×	×

● : Not Affected
× : Undefined
↕ : Result

Specifications:

1 word = 10 bits
ROM (Words): 32
RAM (Digits): 2
Stack (Digits): 0
No. of cycles: 4640
Reentrant: No
Relocatable: No
Interrupt OK?: No

Description:

1. Function Details

(1) Argument details

FREQ(RAM): Contain flag indicating result of frequency measurement. Table 1.3 shows flag functions.

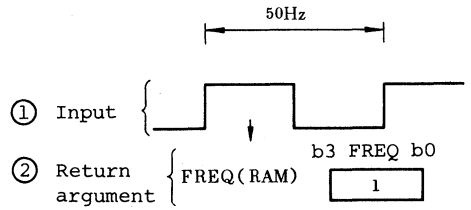


Fig. 1.5. Example of FRQCNT Execution

Specifications Notes:

Program Module Name:	MCU: HMCS402C/ HMCS404C/HMCS408C	Label:
Measure Power Frequency		FRQCNT

Description:

Table 1.3. Flag Functions

Label	RAM $021	Function
FREQ	0	Indicates that frequency is 60 Hz
	1	Indicates that frequency is 50 Hz

(2) Fig. 1.5 shows an example of program module FRQCNT execution. If power supply of 50 Hz is measured as shown in part ① of Fig. 1.5, result is contained in FREQ(RAM) as shown in part ② of Fig. 1.5.

(3) Program module FRQCNT calls subroutines shown in Table 1.4.

Table 1.4. Subroutines Called in FRQCNT

Subroutine Name	Label	Function
Delay 9.2 ms	DLY920	Executes 9.2 ms software timer

2. Users Notes

Due to frequency being measured by software timer, uses oscillator frequency of 4 MHz.

3. RAM Allocation

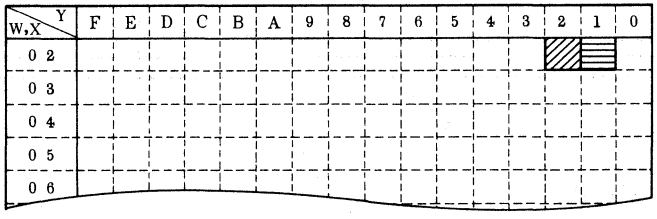


Fig. 1.6. RAM Allocation

Label	RAM	Description
FREQ	b3 b0 MD($021)	Stores the results of frequency measurement
SBFREQ	b3 b0 MD($022)	Stores first measurement results

Program Module Name:	MCU: HMCS402C/ HMCS404C/HMCS408C	Label:
Measure Power Frequency		FRQCNT

Description:

4. Sample Application

```
        ⋮
    CALL FRECNT      ..... Call subroutine FRECNT
        ⋮
    LAMD  FREQ
    LYA              ..... Turn on LED corresponding to FREQ(RAM)
    SED                    measurement results
        ⋮
```

5. Basic Operation

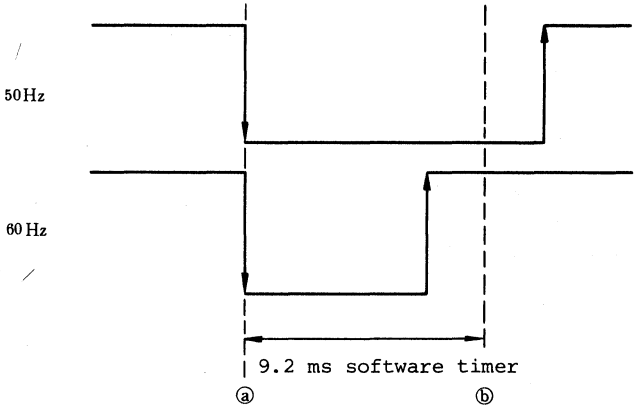


Fig. 1.7. Frequency Measurement Outline

(1) Using port D_3, the falling edge of the pulse is tested (Fig. 1.7. ⓐ).

(2) After testing the falling edge of the pulse, calls subroutine DLY920 for executing 9.2 ms software timer.

(3) After executing software timer, tests condition of port D_3.

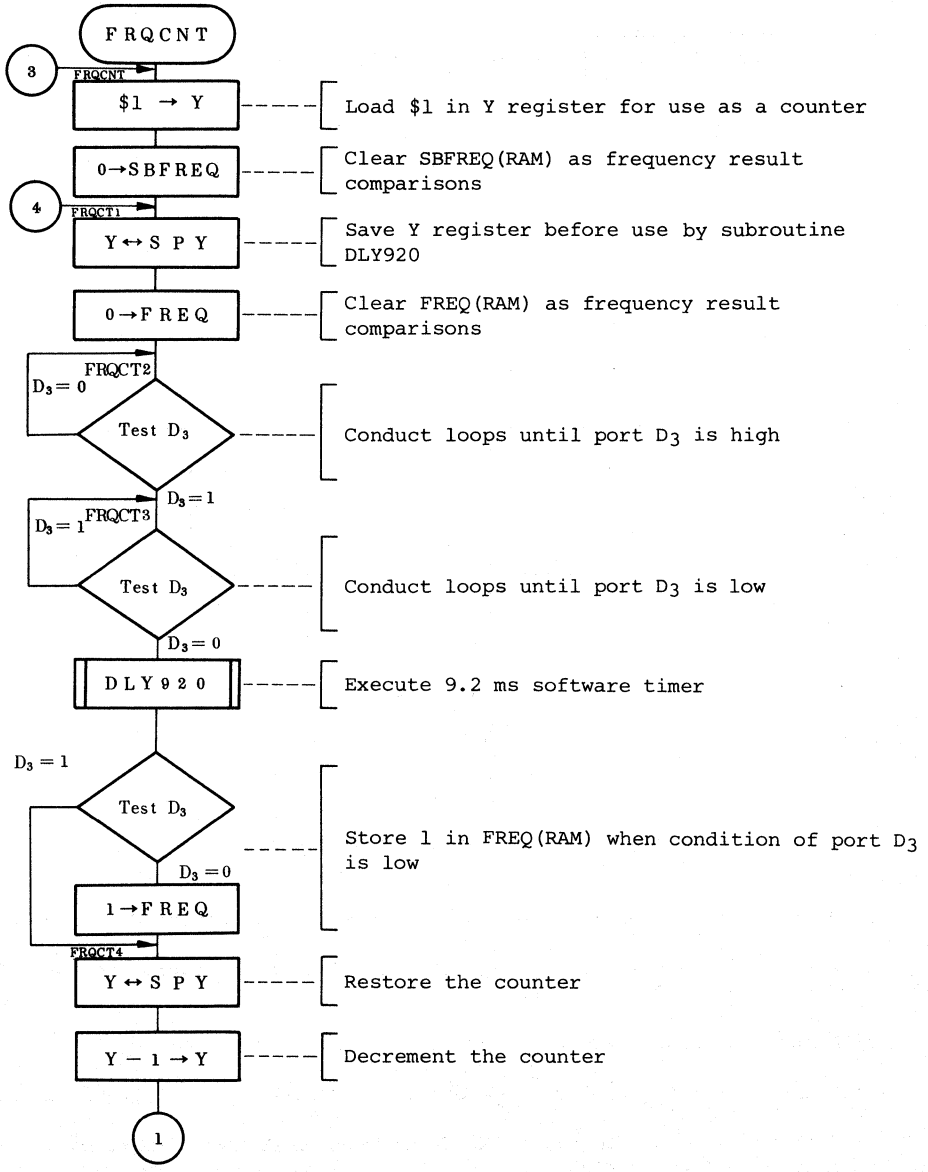
(4) Stores 1 in FREQ(RAM) when condition of port D_3 (Fig. 1.7. ⓑ) is low.

(5) Nothing is to be stored in FREQ(RAM) when condition of port D_3 (Fig. 1.7. ⓑ) is high.

(6) Performs tests (1) through (5) twice, and when results agree, the frequency measurement is completed.

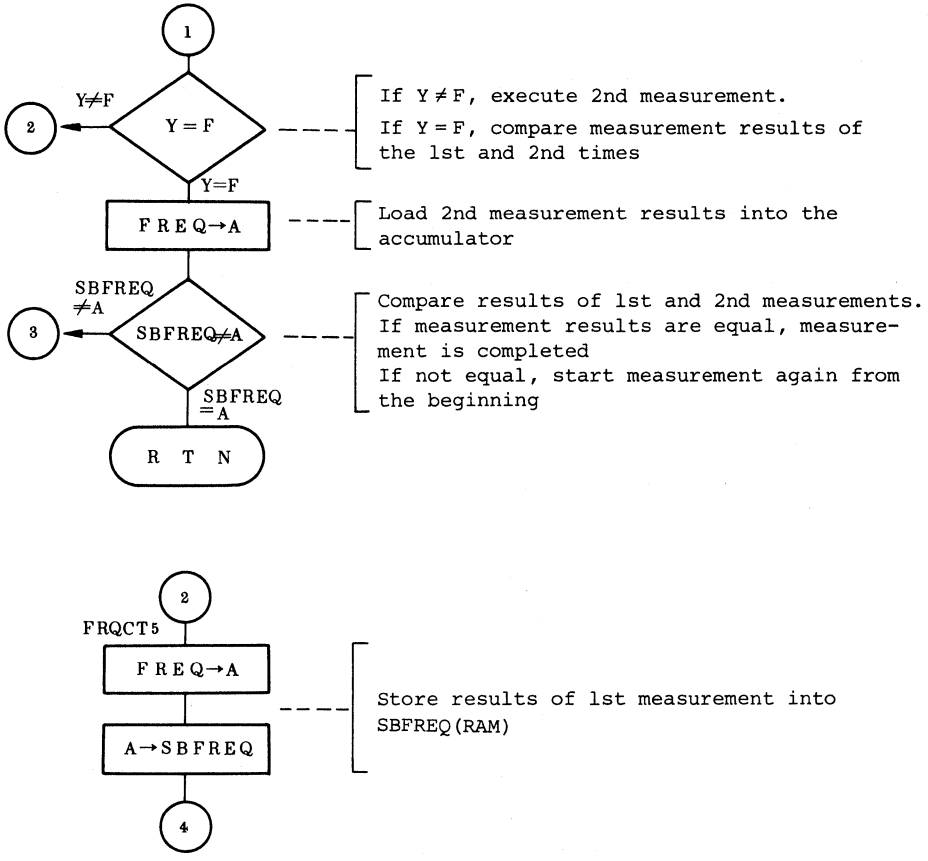
Program Module Name:	MCU: HMCS402C/ HMCS404C/HMCS408C	Label:
Measure Power Frequency		FRQCNT

Flowchart:



Program Module Name:	MCU: HMCS402C/ HMCS404C/HMCS408C	Label:
Measure Power Frequency		FRQCNT

Flowchart:



1.4 Subroutine Description

Subroutine Name:	MCU:	Label:
Delay 9.2 ms	HMCS402C/ HMCS404C/HMCS408C	DLY920

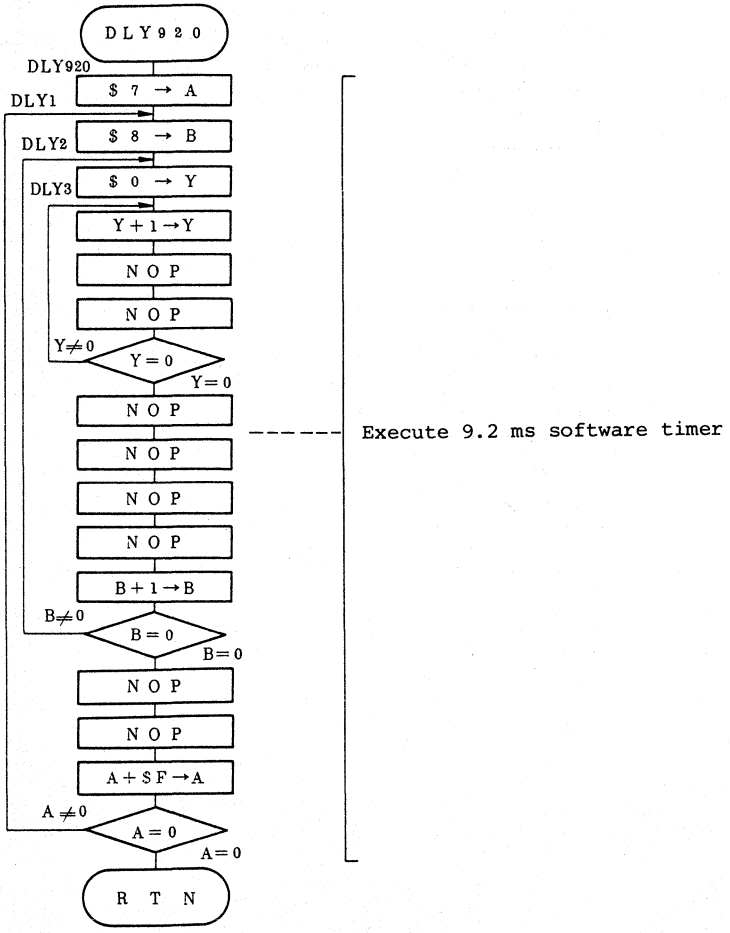
Function:

9.2 ms software timer.

Basic Operation:

Executes after falling edge of port D_3 signal after which status of port D_3 is tested.

Flowchart: FRQCNT



1.5 PROGRAM LISTING

```
ST-NO   OBJECT   ADRS   SOURCE STATEMENTS

00001                          LLEN     132
00002                          TITLE    ZERO CROSS
00003                  *
00004                  ****   RAM ALLOCATION   ****************************************
00005                  *
00006                  FREQ     EQU      $020          FREQUENCY RESULT
00007                  SBFREQ   EQU      $021          SUB FREQUENCY RESULT
00008                  *
00009                  ****   SYMBOL DEFINITIONS   *************************************
00010                  *
00011                  PORTD3   EQU      $3            INPUT POWER PULSE SIGNAL
00012                  ***************************************************************
00013                  *                                                             *
00014                  *               VECTOR ADDRESSES                              *
00015                  *                                                             *
00016                  ***************************************************************
00017                  *
00018                          ORG      $0000
00019                  *
00020   150 010  0000          JMPL     FRQMN         RESET
00021   150 010  0002          JMPL     FRQMN         INT0
00022   150 010  0004          JMPL     FRQMN         INT1
00023   150 010  0006          JMPL     FRQMN         TIMER-A
00024   150 010  0008          JMPL     FRQMN         TIMER-B
00025                          ORG      $000C
00026   150 010  000C          JMPL     FRQMN         SERIAL
00027                  ***************************************************************
00028                  *                                                             *
00029                  *               MAIN PROGRAM : FRQMN                          *
00030                  *                                                             *
00031                  ***************************************************************
00032                  *
00033                          ORG      $0010
00034                  *
00035   0F0      0010  FRQMN    LWI      $0            INITIALIZE W REGISTER
00036   160 018  0011          CALL     FRQCNT        MEASURE POWER FREQUENCY
00037   270      0013          LAMR     $0            LOAD FREQUENCY RESULT INTO ACCUMULATOR
00038   0D8      0014          LYA                    LOAD ACCUMULATOR INTO Y REGISTER
00039   064      0015          RED                    TURN ON LED
00040   150 016  0016  PEND    JMPL     PEND          END OF PROGRAM
00041                  ***************************************************************
00042                  *                                                             *
00043                  *       NAME : FRQCNT (MEASURE POWER FREQUENCY)               *
00044                  *                                                             *
00045                  ***************************************************************
00046                  *                                                             *
00047                  *      ENTRY : NOTHING                                        *
00048                  *    RETURNS : FREQ (50HZ=1,60HZ=0)                           *
00049                  *                                                             *
00050                  ***************************************************************
00051   211      0018  FRQCNT   LYI      1             INITIALIZE COUNTER
00052   1A0 021  0019          LMID     0,SBFREQ      CLEAR SUBFREQUENCY RESULT
00053   002      001B  FRQCT1   XSPY
00054   1A0 020  001C          LMID     0,FREQ        CLEAR FREQUENCY RESULT
00055   2A3      001E  FRQCT2   TDD      PORTD3        LOOP UNTIL PORT D3 IS HIGH
00056   321      001F          BR       FRQCT3
00057   31E      0020          BR       FRQCT2
```

```
00058  2A3       0021   FRQCT3   TDD      PORTD3          LOOP UNTIL PORT D3 IS LOW
00059  321       0022            BR       FRQCT3
00060  160 038   0023            CALL     DLY920
00061  2A3       0025            TDD      PORTD3          TEST PORT D3
00062  329       0026            BR       FRQCT4
00063  1A1 020   0027            LMID     1,FREQ          SET 50HZ FLAG
00064  002       0029   FRQCT4   XSPY
00065  0DF       002A            DY                       DECREMENT
00066  332       002B            BR       FRQCT5
00067  190 020   002C            LAMD     FREQ            COMRARE FREQ WITH SBFREQ
00068  104 021   002E            ANEMD    SBFREQ
00069  318       0030            BR       FRQCNT          IF NOT EQUAL, BRANCH TO FRQCNT
00070  010       0031            RTN
00071  190 020   0032   FRQCT5   LAMD     FREQ            STORE FREQ IN SBFRQ
00072  194 021   0034            LMAD     SBFREQ
00073  150 01B   0036            JMPL     FRQCT1
00074                   ********************************************************************
00075                   *                                                                  *
00076                   *      NAME : DLY920 (DELAY 9.2MSEC)                               *
00077                   *                                                                  *
00078                   ********************************************************************
00079  237       0038   DLY920   LAI      7               EXECUTE 9.2MSEC SOFTWARE TIMER
00080  208       0039   DLY1     LBI      8
00081  210       003A   DLY2     LYI      0
00082  05C       003B   DLY3     IY
00083  000       003C            NOP
00084  000       003D            NOP
00085  33B       003E            BR       DLY3
00086  000       003F            NOP
00087  000       0040            NOP
00088  000       0041            NOP
00089  000       0042            NOP
00090  04C       0043            IB
00091  33A       0044            BR       DLY2
00092  000       0045            NOP
00093  000       0046            NOP
00094  28F       0047            AI       15
00095  339       0048            BR       DLY1
00096  010       0049            RTN
00097                   *
00098                   END
```

2.1 HARDWARE DESCRIPTION

2.1.1 Function

Performs analog to digital conversion in the range from 0 to 5V by HMCS404C MCU control of a resistant ladder type D/A converter. Results of A/D conversion are expressed in BCD (Binary Coded Decimal).

2.1.2 Microcomputer Operation

Outputs data from $00 - $FF from ports R7 and R8 to control resistant ladder type DA converter, and then compares analogue output (V_{ref}) and input (V_{in}) voltages of the D/A converter and inputs results into port D0.

2.1.3 Peripheral Devices

2.1.4 Circuit Diagram

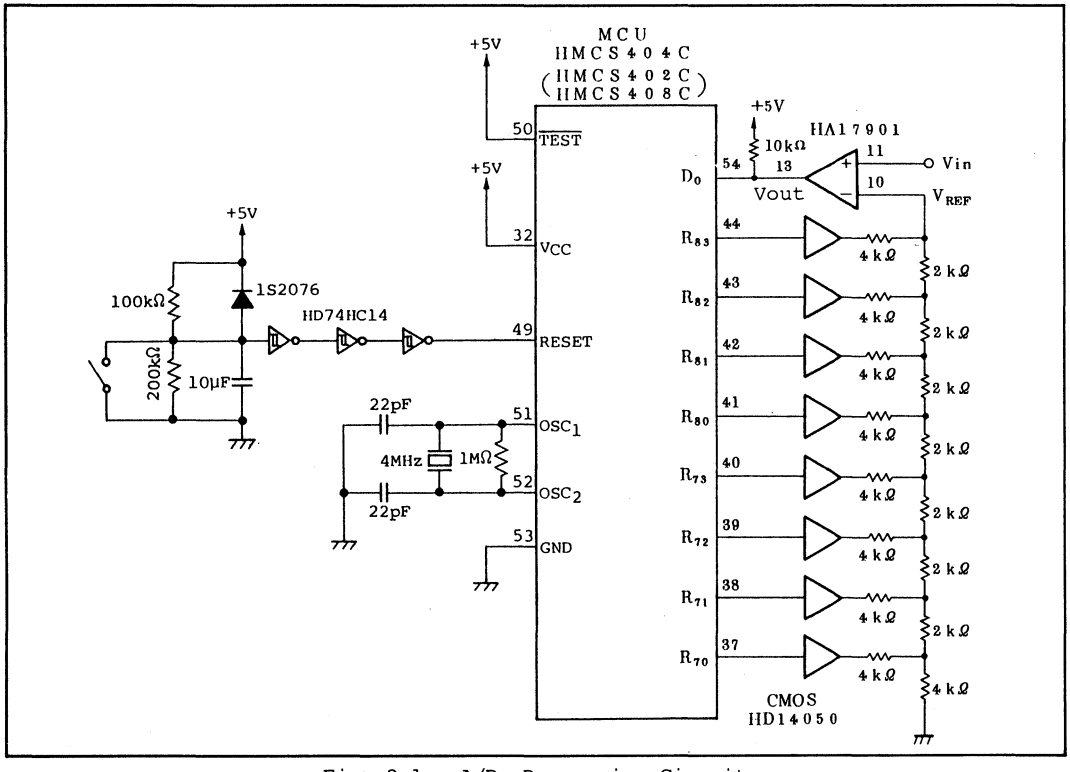


Fig. 2.1. A/D. Donversion Circuit

2.1.5 Pin Functions

Pin functions for A/D conversion are shown in Table 2.1.

Table 2.1. Pin Functions

Pin Name (HMCS404C)	Input/ Output	Active Level (High or Low)	Function
D_0	Input	-	Inputs comparator V_{out} output
R_{70}	Output	-	Changes digital output in the range from $FF - $00
R_{71}	Output	-	
R_{72}	Output	-	
R_{73}	Output	-	
R_{80}	Output	-	
R_{81}	Output	-	
R_{82}	Output	-	
R_{83}	Output	-	

2.1.6 Hardware Operation

Fig. 2.2 shows timing chart of comparator V_{in}, V_{ref} and V_{out} when 3.4 V has been inputted in V_{in}.

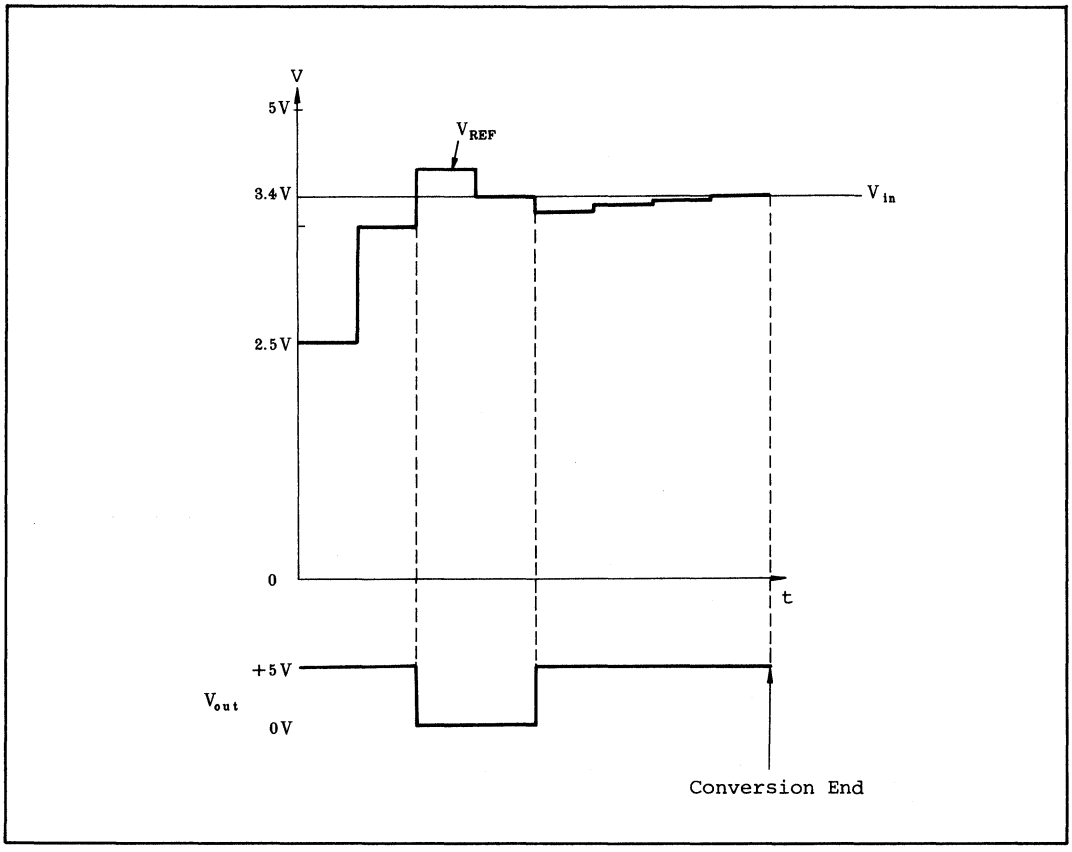


Fig. 2.2. A/D Conversion Timing Chart

2.2.1 Program Module Configuration

The program module configuration to conduct A/D conversion is shown in Fig. 2.3.

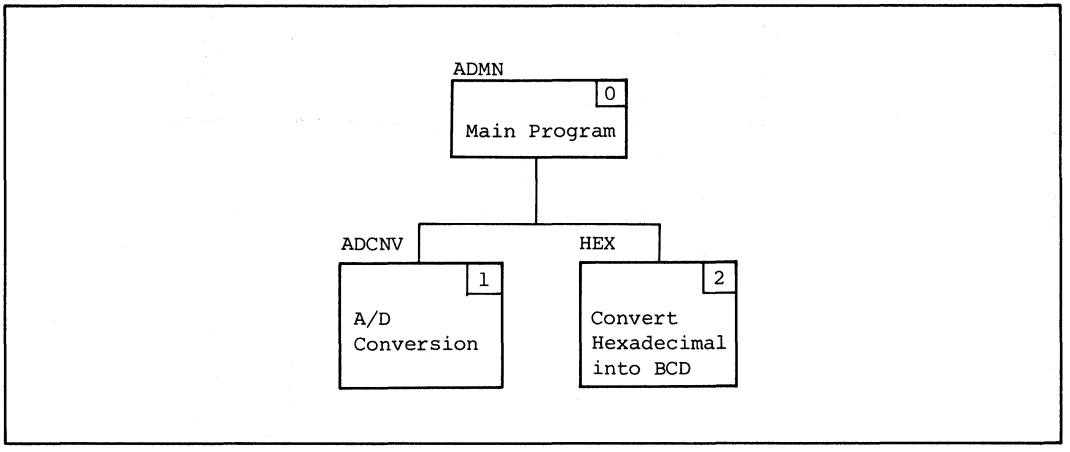


Fig. 2.3. Program Module Configuration

2.2.2 Program Module Functions

Program module functions are summarized in Table 2.2.

Table 2.2. Program Module Functions

No.	Program Module Name	Label	Function
0	Main Program	ADMN	Main program which calculates value of A/D conversion into 3 decimal digits
1	A/D Conversion	ADCNV	Conducts A/D conversion
2	Convert Hexadecimal into BCD	HEX	Converts 2 hexadecimal bytes into 5 decimal digits. For details, refer to HMCS400 Series Application Note (Software Edition) on Software HEX

2.2.3 Program Module Process Flow (Main Program)

Fig. 2.4 shows an example of executing A/D conversion using the modules shown in Fig. 2.3. Execution of the main program in Fig. 2.4 determines the value of A/D conversion in 3 decimal digits.

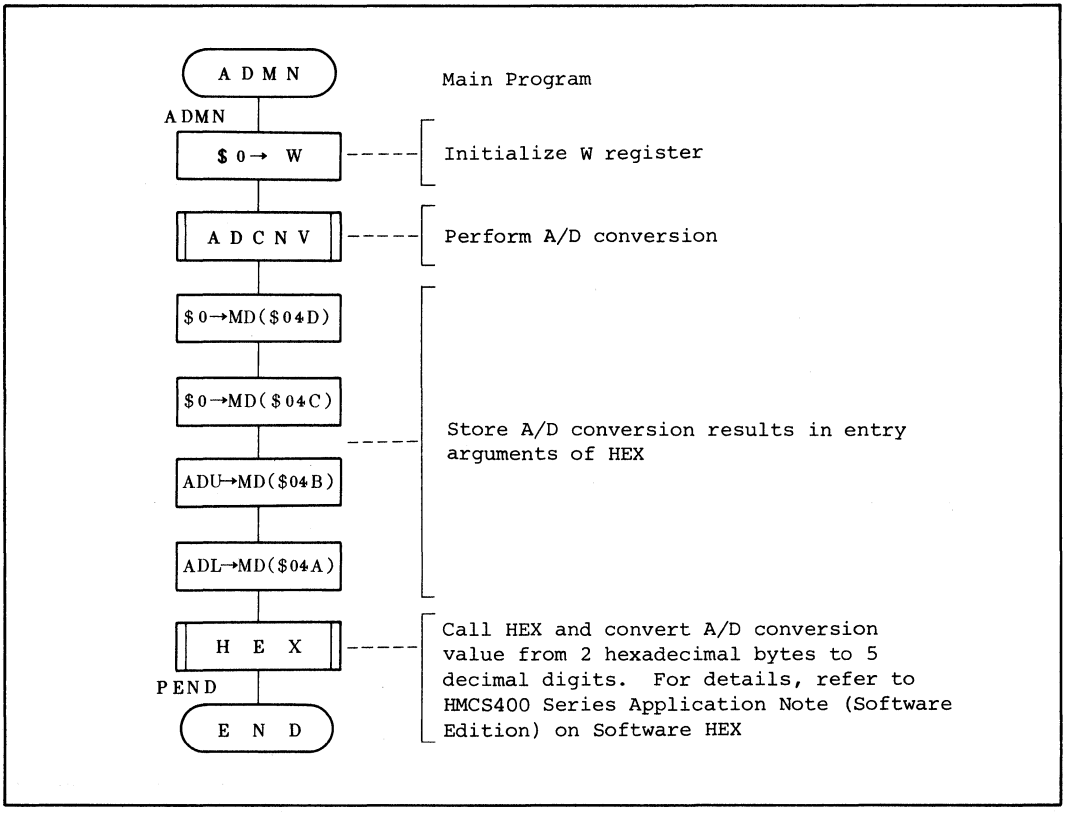


Fig. 2.4. Example of Executing A/D Conversion

Program Module Name: A/D Conversion

MCU: HMCS402C/ HMCS404C/HMCS408C

Label: ADCNV

Function:

Controls op-amp and resistor ladder and performs A/D conversion.

Arguments:

Contents		1 digit = 4 bits Storage Location	No. of Digits
Entry			
Returns	A/D conversion result	ADU(RAM)	1
		ADL(RAM)	1

Changes in CPU Registers and Flags:

A	B
×	●

X	Y
×	×

SPX	SPY
●	●

W
●

CA	ST
●	×

● : Not Affected
× : Undefined
↕ : Result

Specifications:

1 word = 10 bits

ROM (Words): 77
RAM (Digits): 2
Stack (Digits): 0
No. of cycles: 88
Reentrant: No
Relocatable: No
Interrupt OK?: No

Description:

1. Function Details

(1) Argument details
ADU(RAM), ADL(RAM): Holds 2 hexadecimal digits of A/D conversion results.

(2) Fig. 2.5 shows an example of program If Vin is as shown in part ① of Fig. 2.5, A/D conversion results are stored in ADU(RAM), ADL(RAM) as shown in part ② of Fig. 2.5.

① Entry arguments { $V_{in} = 1.0$ V

② Return arguments { ADU, ADL (RAM)

b7	ADU,ADL	b0
3		3

Fig. 2.5. Example of ADCNV Execution

Specifications Notes: Number of cycles is the value when comparator V_{in} is 0 V.

Program Module Name: A/D Conversion	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: ADCNV

Description:

2. User Notes

Software timer is executed with due consideration for delays in HA17901.

3. RAM Allocation

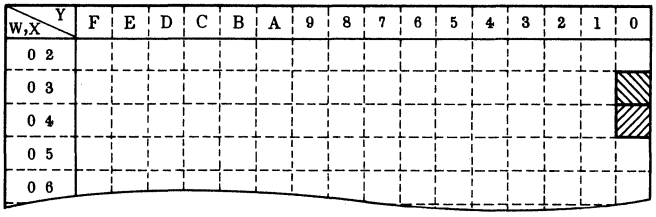


Fig. 2.6. RAM Allocation

Label	RAM	Description
ADU	b3 b0 MD($030)	Stores A/D conversion output results
ADL	b3 b0 MD($040)	

Program Module Name:	MCU: HMCS402C/ HMCS404C/HMCS408C	Label:
A/D Conversion		ADCNV

Description:

4. Sample Application

```
WORKU   EQU   $030
WORKL   EQU   $040
        :
        CALL  ADCNV
        :
        LAMD  ADU
        LMAD  WORKU
        LAMD  ADL
        LMAD  WORKL
        :
```

Allocate RAM area to store 2 hexadecimal digits of A/D conversion results

Call subroutine ADCNV

Store A/D conversion results in return arguments into user program RAM area

5. Basic Operation

(1) The binary search method used in A/D conversion is described below. As shown in Fig. 2.7, when searching for a certain number, the entire range of numbers is consecutively divided into 1/2, 1/4, 1/8, etc., each time performing a numerical comparison to determine which side of the division the searched for number must lie in. The maximum number of times (A) that this process must be performed is expressed by the formula $\log_2 N = A$. Binary search greatly speeds up the A/D conversion process.

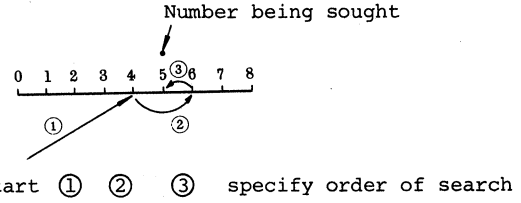


Fig. 2.7. Binary Search Method

(2) Procedures are described below.

(a) Use ADU(RAM), ADL(RAM) as outputting data buffer to ports R7 and R8.

(b) Clear ADU(RAM), ADL(RAM) beforehand.

Program Module Name:	MCU: HMCS402C/ HMCS404C/HMCS408C	Label:
A/D Conversion		ADCNV

Description:

(c) In the order from the highest bit in ADU(RAM), ADL(RAM) set 1 by the set memory bit (SEM instruction) and output ADU(RAM), ADL(RAM) data to ports R8 and R7.

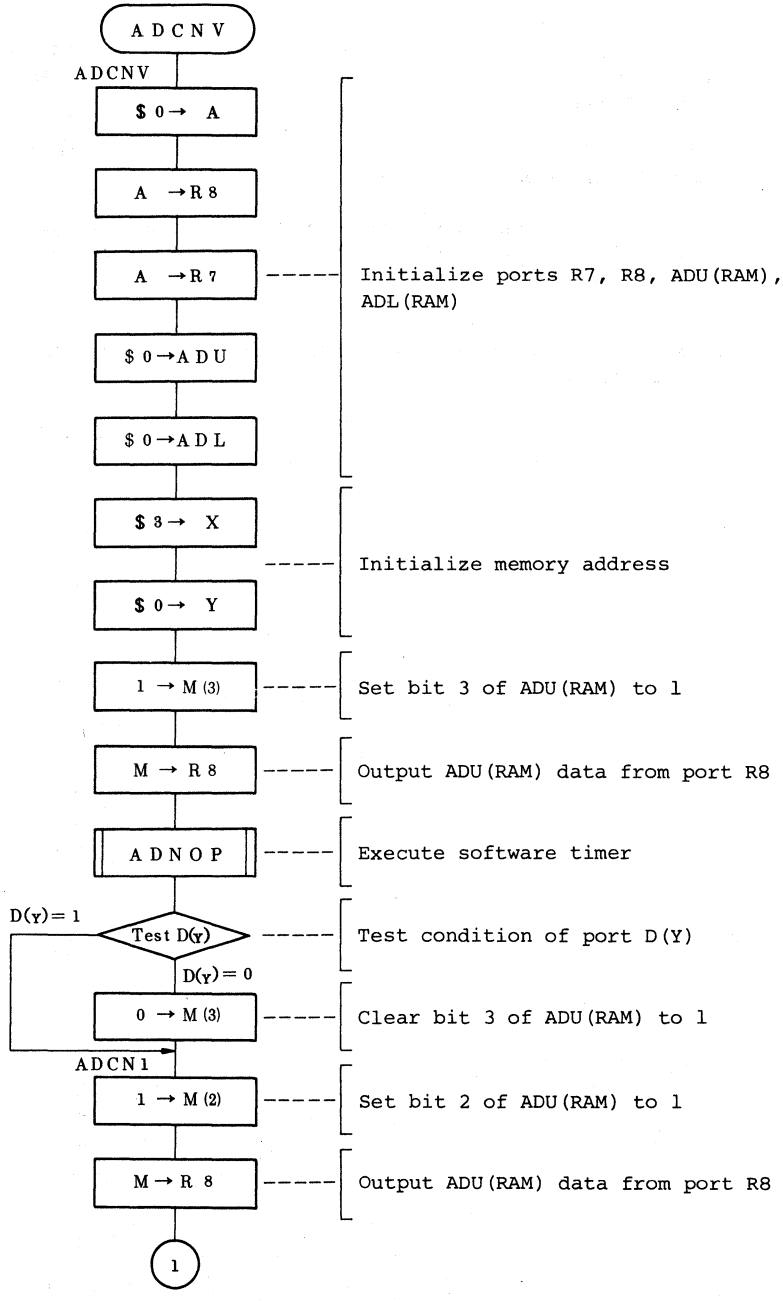
(d) Test whether V_{out} input from comparator is high or low every time there is an output from port R8 or R7.

(e) If high, $V_{in} > V_{ref}$; continue next output under same conditions. If low, $V_{in} < V_{ref}$; conduct next output after clearing bits set in ADU(RAM), ADL(RAM).

(3) Software timer is being executed with due consideration for delay time of the comparator.

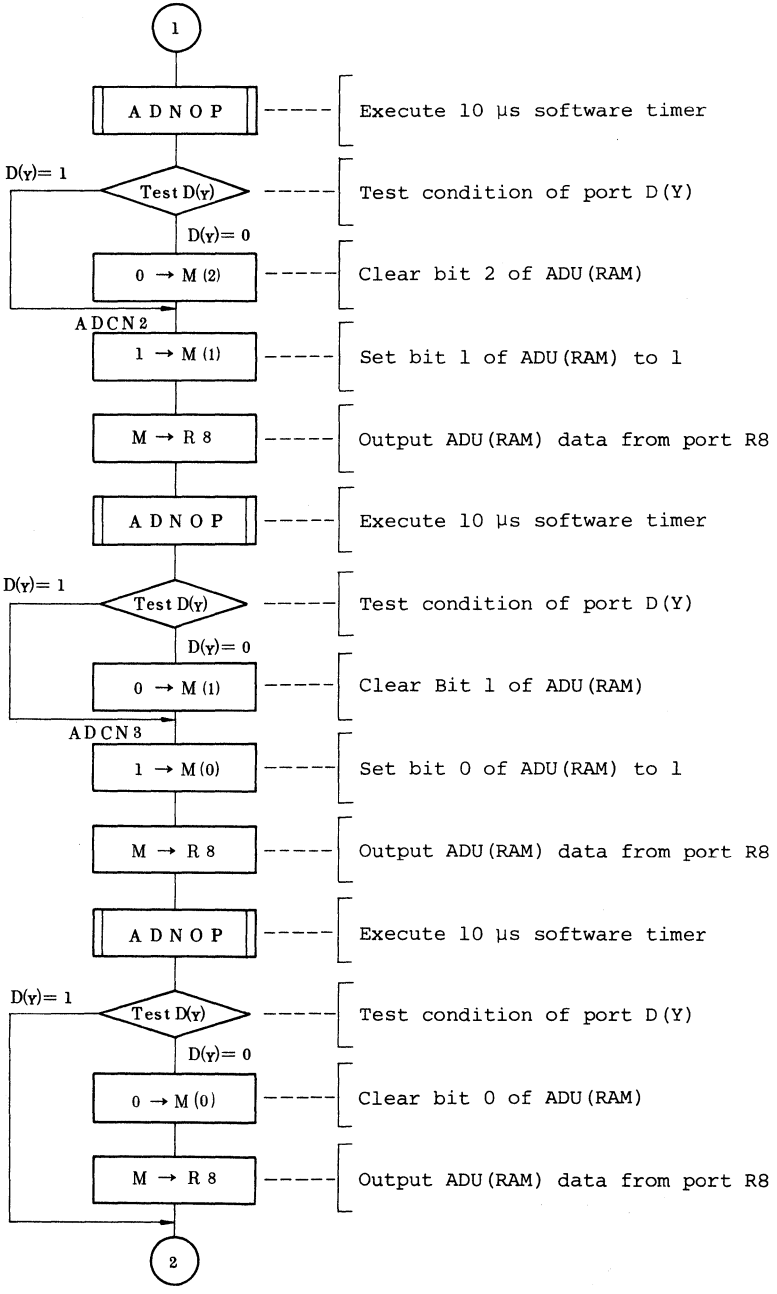
Program Module Name:	MCU: HMCS402C/ HMCS404C/HMCS408C	Label:
A/D Conversion		ADCNV

Flowchart:



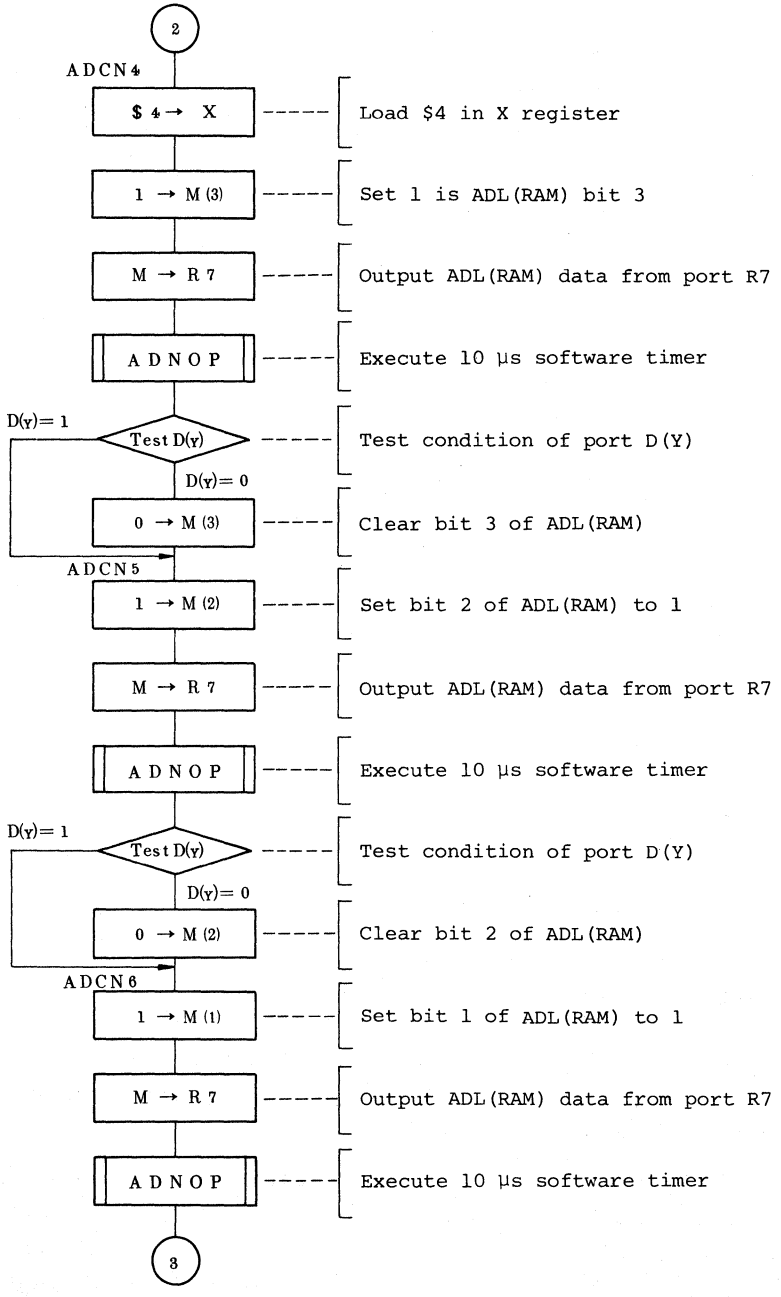
Program Module Name:	MCU:	Label:
A/D Conversion	HMCS402C/ HMCS404C/HMCS408C	ADCNV

Flowchart:



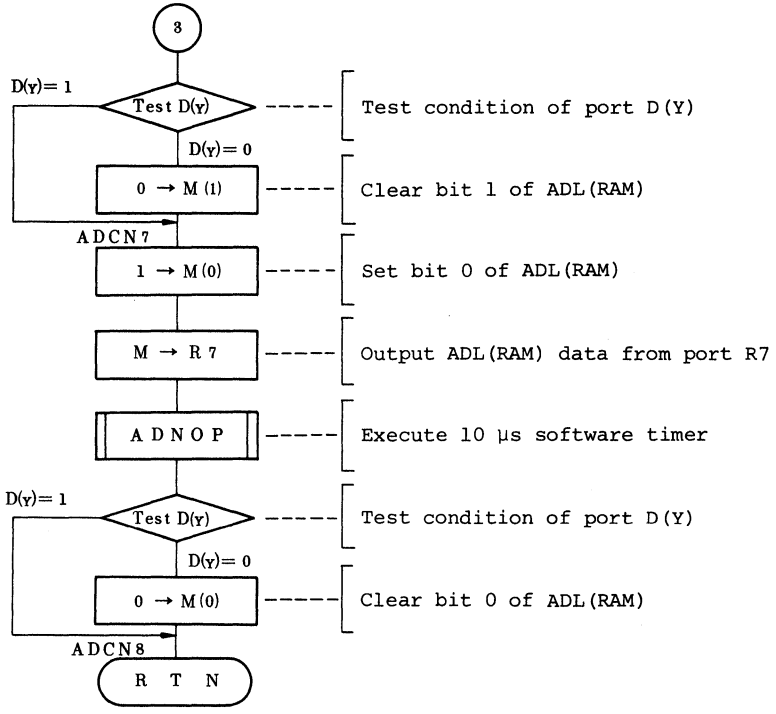
Program Module Name:	MCU:	Label:
A/D Conversion	HMCS402C/ HMCS404C/HMCS408C	ADCNV

Flowchart:



Program Module Name: A/D Conversion	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: ADCNV

Flowchart:



2.4 SUBROUTINE DESCRIPTION

Subroutine Name:	MCU:	Label:
Software Timer	HMCS402C/ HMCS404C/HMCS408C	ADNOP

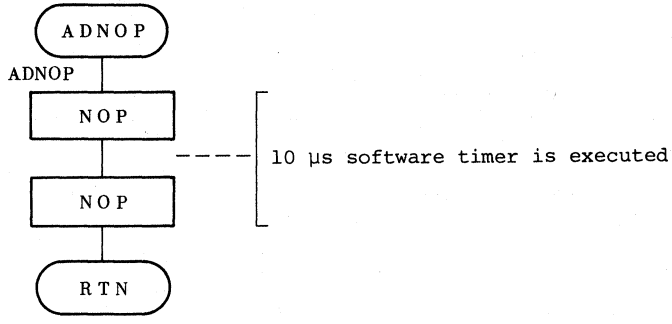
Function:

Executes 10 µs software timer by NOP instruction.

Basic Operation:

1. Executes software timer in consideration of delay time for comparator HA17901.
2. Executes of NOP instruction (2 times) and RTN instruction resulting in execution of the 10 µs software timer.

Flowchart: ADCNV



2.5 PROGRAM LISTING

```
ST-NO   OBJECT    ADRS   SOURCE STATEMENTS

00001   010       0000           LLEN     132
00002                            TITLE    AD CONVERSION
00003                  *
00004                  ****   RAM ALLOCATION   ********************************
00005                  *
00006                  ADU      EQU      $030          UPPER A/D CONVERSION FLAG
00007                  ADL      EQU      $040          LOWER A/D CONVERSION FLAG
00008                  XHEXD    EQU      $4            HEXADECIMAL AND DECIMAL DATA ADDR(X)
00009                  YHHEX    EQU      $A            2-BYTE HEXADECIMAL DATA ADDR(Y)
00010                  YHDEC    EQU      $2            5-DIGIT BCD DATA LSD ADDR(Y)
00011                  BAHEX    EQU      $E            2-BYTE HEXADECIMAL DATA MSD ADDR(Y)+1
00012                  BADEC    EQU      $7            5-DIGIT BCD DATA MSD ADDR(Y)+1
00013                  ************************************************************
00014                  *                                                          *
00015                  *               VECTOR ADDRESSES                           *
00016                  *                                                          *
00017                  ************************************************************
00018                  *
00019                           ORG      $0000
00020                  *
00021   150 010   0000         JMPL     ADMN          RESET
00022   150 010   0002         JMPL     ADMN          INT0
00023   150 010   0004         JMPL     ADMN          INT1
00024   150 010   0006         JMPL     ADMN          TIMER-A
00025   150 010   0008         JMPL     ADMN          TIMER-B
00026   000       000A         NOP
00027   000       000B         NOP
00028   150 010   000C         JMPL     ADMN          SERIAL
00029                  ************************************************************
00030                  *                                                          *
00031                  *               MAIN PROGRAMN : ADMN                       *
00032                  *                                                          *
00033                  ************************************************************
00034                  *
00035                           ORG      $0010
00036                  *
00037   0F0       0010   ADMN     LWI      $0            INITIALIZE W REGISTER
00038   160 022   0011           CALL     ADCNV         A/D CONVERTION
00039   1A0 04D   0013           LMID     $0,$04D       LOAD RESULT INTO HEX ENTRY ARGUMENT
00040   1A0 04C   0015           LMID     $0,$04C
00041   190 030   0017           LAMD     ADU
00042   194 04B   0019           LMAD     $04B
00043   190 040   001B           LAMD     ADL
00044   194 04A   001D           LMAD     $04A
00045   160 06F   001F           CALL     HEX           CONVERT RESULT INTO BCD DATA
00046   321       0021   PEND     BR       PEND          END OF PROGRAM
00047                  ************************************************************
00048                  *                                                          *
00049                  *          NAME : ADCNV (AD CONVERT)                       *
00050                  *                                                          *
00051                  ************************************************************
00052                  *                                                          *
00053                  *         ENTRY : NOTHING                                  *
00054                  *       RETURNS : ADU (UPPER A/D CONVERSION FLAG)          *
00055                  *                 ADL (LOWER A/D CONVERSION FLAG)          *
00056                  *                                                          *
00057                  ************************************************************
```

```
00058  230      0022  ADCNV   LAI     $0
00059  2D8      0023          LRA     $8          INITIALIZE R8 PORT
00060  2D7      0024          LRA     $7          INITIALIZE R7 PORT
00061  1A0 030  0025          LMID    $0.ADU      INITIALIZE RAM
00062  1A0 040  0027          LMID    $0.ADL
00063  223      0029          LXI     $3          DEFINE RAM STARTING ADDRESS
00064  210      002A          LYI     $0
00065  087      002B          SEM     $3          SET BIT 3 OF WORKU(RAM)
00066  090      002C          LAM                 OUTPUT WORKU(RAM) DATA
00067  2D8      002D          LRA     $8
00068  160 085  002E          CALL    ADNOP       TIME DELAY
00069  0E0      0030          TD                  COMPARE OUTPUT DATA WITH INPUT VOLTAGE
00070  333      0031          BR      ADCN1
00071  08B      0032          REM     $3          CLEAR BIT 3
00072  086      0033  ADCN1   SEM     $2          SET BIT 2
00073  090      0034          LAM                 OUTPUT DATA
00074  2D8      0035          LRA     $8
00075  160 085  0036          CALL    ADNOP       TIME DELAY
00076  0E0      0038          TD                  COMPARE OUTPUT DATA WITH INPUT VOLTAGE
00077  338      0039          BR      ADCN2
00078  08A      003A          REM     $2          CLEAR BIT 2
00079  085      003B  ADCN2   SEM     $1          SET BIT 1
00080  090      003C          LAM                 OUTPUT DATA
00081  2D8      003D          LRA     $8
00082  160 085  003E          CALL    ADNOP       TIME DELAY
00083  0E0      0040          TD                  COMPARE OUTPUT DATA WITH INPUT VOLTAGE
00084  343      0041          BR      ADCN3
00085  089      0042          REM     $1          CLEAR BIT 1
00086  084      0043  ADCN3   SEM     $0          SET BIT 0
00087  090      0044          LAM                 OUTPUT DATA
00088  2D8      0045          LRA     $8
00089  160 085  0046          CALL    ADNOP       TIME DELAY
00090  0E0      0048          TD                  COMPARE OUTPUT DATA WITH INPUT VOLTAGE
00091  34D      0049          BR      ADCN4
00092  088      004A          REM     $0          CLEAR BIT 0
00093  090      004B          LAM                 OUTPUT DATA
00094  2D8      004C          LRA     $8
00095  224      004D  ADCN4   LXI     $4          CONVERT RAM ADDRESS
00096  087      004E          SEM     $3          SET BIT 3
00097  090      004F          LAM                 OUTPUT DATA
00098  2D7      0050          LRA     $7
00099  160 085  0051          CALL    ADNOP       TIME DELAY
00100  0E0      0053          TD                  COMPARE OUTPUT DATA WITH INPUT VOLTAGE
00101  356      0054          BR      ADCN5
00102  08B      0055          REM     $3          CLEAR BIT 3
00103  086      0056  ADCN5   SEM     $2          SET BIT 2
00104  090      0057          LAM                 OUTPUT DATA
00105  2D7      0058          LRA     $7
00106  160 085  0059          CALL    ADNOP       TIME DELAY
00107  0E0      005B          TD                  COMPARE OUTPUT DATA WITH INPUT VOLTAGE
00108  35E      005C          BR      ADCN6
00109  08A      005D          REM     $2          CLEAR BIT 2
00110  085      005E  ADCN6   SEM     $1          SET BIT 1
00111  090      005F          LAM                 OUTPUT DATA
00112  2D7      0060          LRA     $7
00113  160 085  0061          CALL    ADNOP       TIME DELAY
00114  0E0      0063          TD                  COMPARE OUTPUT DATA WITH INPUT VOLTAGE
```

```
00115   366       0064            BR        ADCN7
00116   089       0065            REM       $1              CLEAR BIT 1
00117   084       0066   ADCN7    SEM       $0              SET BIT 0
00118   090       0067            LAM                       OUTPUT DATA
00119   2D7       0068            LRA       $7
00120   160 085   0069            CALL      ADNOP           TIME DELAY
00121   0E0       006B            TD                        COMPARE OUTPUT DATA WITH INPUT VOLTAGE
00122   36E       006C            BR        ADMN8
00123   088       006D            REM       $0              CLEAR BIT 0
00124   010       006E   ADMN8    RTN
00125                    ****************************************************************
00126                    *                                                              *
00127                    *          NAME : HEX (CONVERT 2-BYTE HEXADCIMALS              *
00128                    *                       INTO 5-DIGIT BCD)                      *
00129                    *                                                              *
00130                    ****************************************************************
00131                    *                                                              *
00132                    *         ENTRY : MD($04D-$04A) (2-BYTE HEXADECIMAL DATA)      *
00133                    *       RETURNS : MD($046-$042) (5-DIGIT BCD DATA)             *
00134                    *                                                              *
00135                    ****************************************************************
00136   224       006F   HEX      LXI       XHEXD           LOAD HEXADECIMAL ADDR(X)
00137   212       0070            LYI       YHDEC           LOAD BCD DATA ADDR(Y)
00138   290       0071   HEX1     LMIIY     $0              CLEAR BCD DATA ADDR(Y)
00139   077       0072            YNEI      BADEC
00140   371       0073            BRS       HEX1
00141   20F       0074            LBI       $F              LOAD SHIFT COUNTER
00142   21A       0075   HEX2     LYI       YHHEX           LOAD HEXADECIMAL DATA ADDR(Y)
00143   090       0076   HEX3     LAM                       SHIFT HEXADECIMAL DATA 1 BIT LEFT
00144   0A1       0077            ROTL
00145   050       0078            LMAIY
00146   07E       0079            YNEI      BAHEX
00147   376       007A            BRS       HEX3
00148   212       007B            LYI       YHDEC           LOAD BCD DATA LSD ADDR(Y)
00149   090       007C   HEX4     LAM                       BCD DATA AREA *2+CA->A
00150   0A1       007D            ROTL
00151   0A6       007E            DAA                       CONVERT INTO BCD DATA ?
00152   050       007F            LMAIY                     LOAD DECIMAL DATA
00153   077       0080            YNEI      BADEC           TEST IF CONVERSION IS COMPLETED
00154   37C       0081            BRS       HEX4
00155   0CF       0082            DB                        DECREMENT SHIFT COUNTER
00156   375       0083            BRS       HEX2            LOOP UNTIL SHIFT COUNTER = $F
00157   010       0084            RTN
00158                    ********************************************************
00159                    *                                                      *
00160                    *          NAME : ADNOP (TIME DELAY)                   *
00161                    *                                                      *
00162                    ********************************************************
00163   000       0085   ADNOP    NOP
00164   000       0086            NOP
00165   010       0087            RTN
00166                    *
00167                    END
```

SECTION 3. PULSE OUTPUT DUTY CONTROL

3.1 HARDWARE DESCRIPTION

3.1.1 Function

Performs duty control of pulses output from the HMCS404C MCU in the range from 0 - 100%, increasing the duty rate 10% every 0.7 sec. The output pulses are input to an amplifier and integration circuit, producing output voltages from 0 V to 5 V in 0.5 units.

3.1.2 Microcomputer Operation

Outputs high, low pulses from port D4 using timer B with interrupt routines. The high and low period of these pulses are varied with TMB (Timer Mode B register).

3.1.3 Peripheral Devices

HD14050B: Operational Amplifier - Prevents the fluctuation of analog output voltage caused by the load in user system.

HA17458: Integration Circuit - Converts digital pulses output from the MCU to analog.

3.1.4 Circuit Diagram

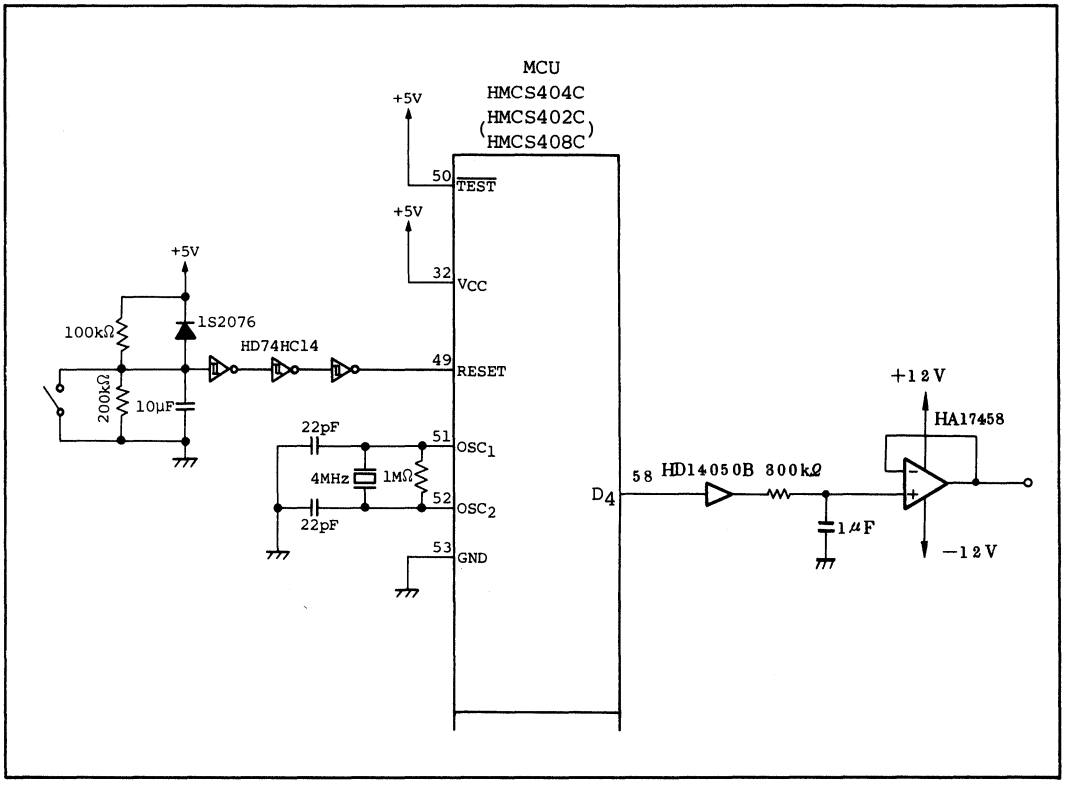


Fig. 3.1. Duty Pulse Output

3.1.5 Pin Functions

Pin function for pulse output is shown in Table 3.1.

Table 3.1. Pin Function

Pin Name (HMCS404C)	Input/ Output	Active Level (High or Low)	Function
D4	Output	-	Output pulse

3.1.6 Hardware Operation

Outputs pulses with 0 - 100% duty rate every 0.7s, increasing the duty rate 10% each time from port D_4 on the HMCS404C. The pulse output and DA conversion are shown in Fig. 3.2.

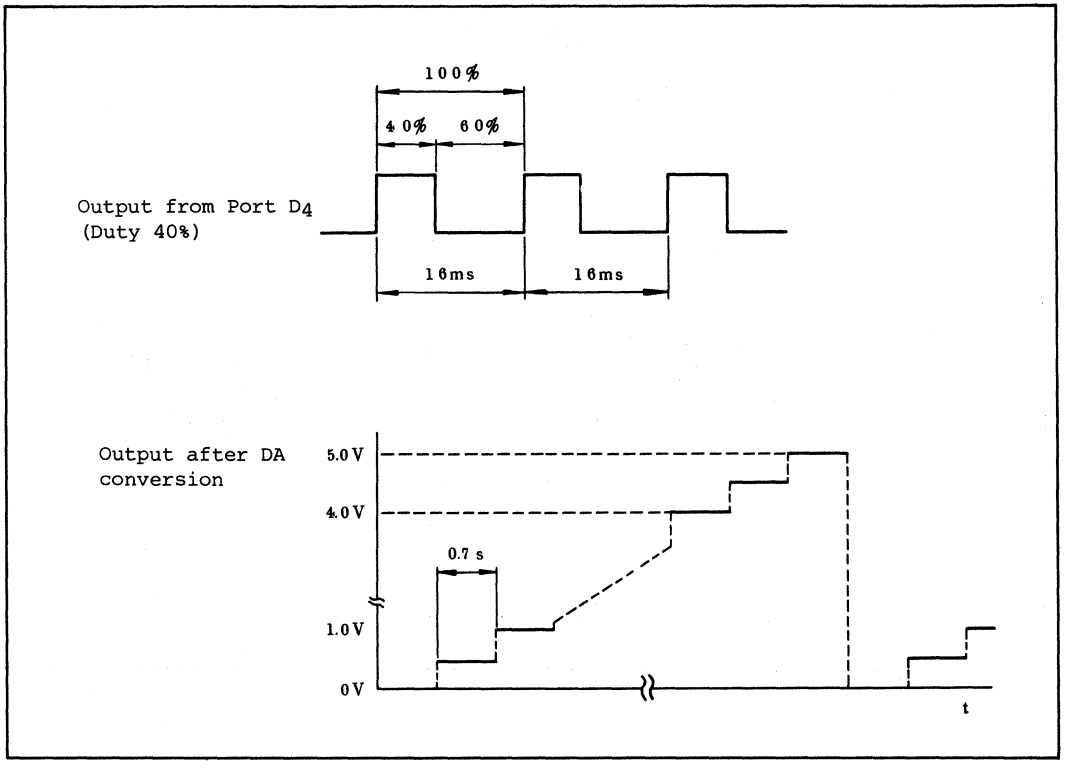


Fig. 3.2. Pulse Output and Waveform after DA Conversion

3.2.1 Program Module Configuration

The program module configuration for pulse output is shown in Fig. 3.3.

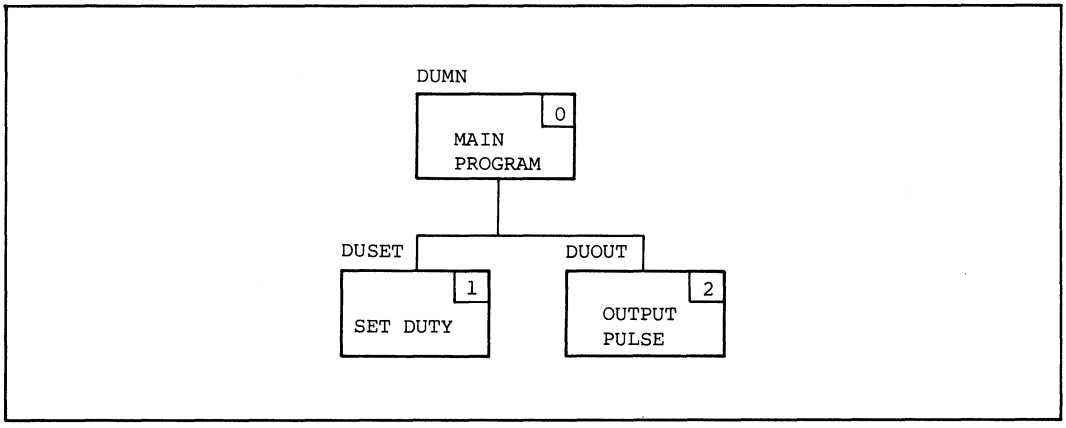


Fig. 3.3. Program Module Configuration

3.2.2 Program Module Functions

Program module functions are summarized in Table 3.2.

Table 3.2. Program Module Functions

No.	Program Module Name	Label	Function
0	Main Program	DUMN	Outputs pulse with 0 - 100% duty rate
1	Set Duty	DUSET	Sets duty
2	Output Pulse	DUOUT	Outputs pulse from I/O port

3.2.3 Program Module Process Flow (Main Program)

The flowchart in Fig. 3.4 is an example of D/A conversion by controlling pulse output, performed by the program modules in Fig. 3.3. The main program in Fig. 3.4 changes pulse output with 0 - 100% duty rate every 0.7 seconds, increasing the duty rate 10% each time.

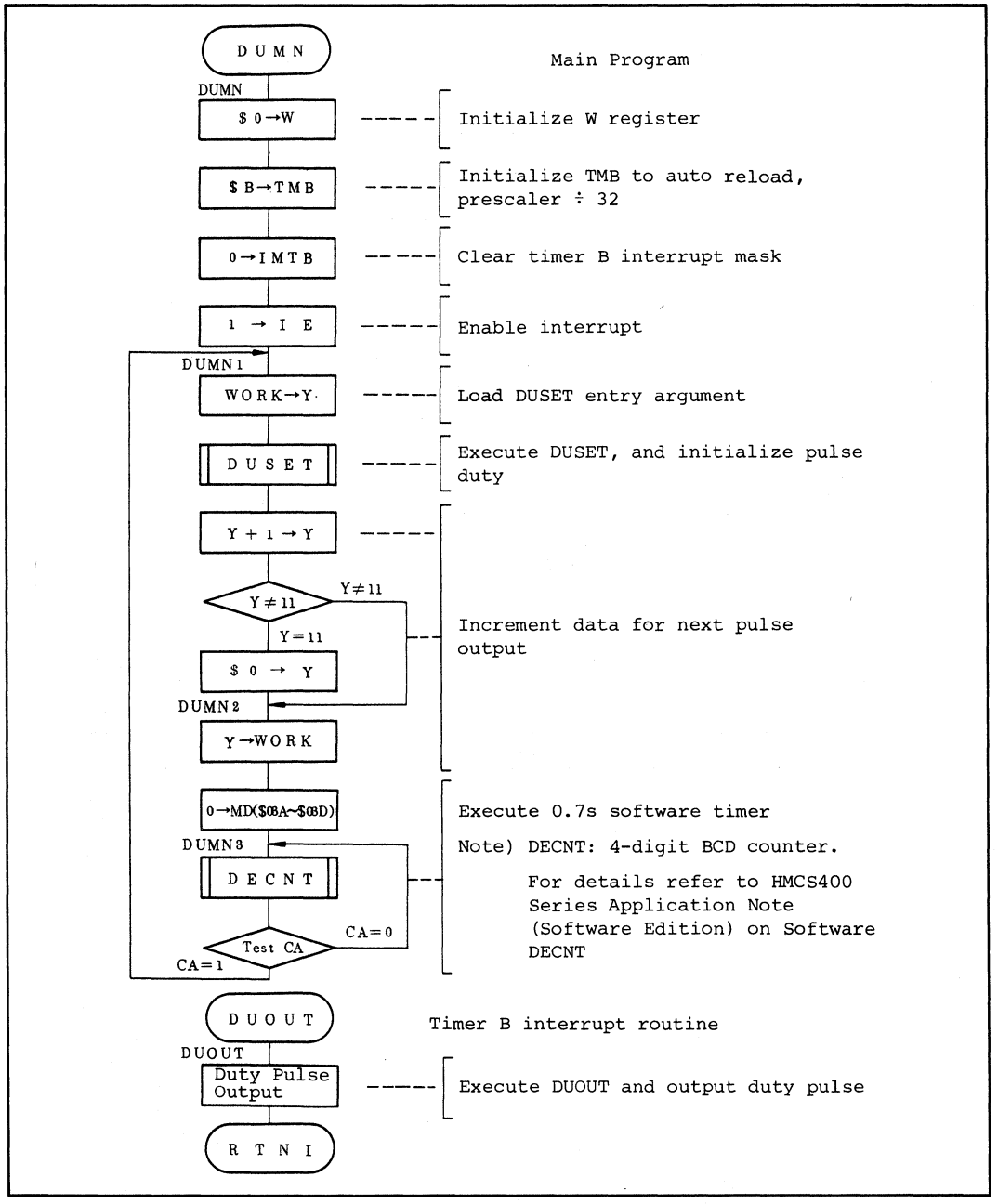


Fig. 3.4. Program Module Sample Application

Program Module Name:	MCU:	Label:
Set Duty	HMCS402C/ HMCS404C/HMCS408C	DUSET

Function: Stores high and low output period corresponding to 1-digit hexadecimal duty stored in entry argument.

Arguments:

Contents		1 digit = 4 bits Storage Location	No. of Digits
Entry	Duty	Y	1
Re-turns	High output width	HTIMEU HTIMEL (RAM)	2
	Low output width	LTIMEU LTIMEL (RAM)	2
	Output pulse mode flag	HOUTF LOUTF (RAM)	2

Changes in CPU Registers and Flags:

A	B
×	×

X	Y
●	×

SPX	SPY
●	●

W
●

CA	ST
×	×

● : Not Affected
× : Undefined
↕ : Result

Specifications:

1 word = 10 bits

ROM (Words): 44
RAM (Digits): 7
Stack (Digits): 0
No. of cycles: 30
Reentrant: No
Relocatable: No
Interrupt OK?: No

Description:

1. Function Details

(1) Argument details

Y: Holds duty as 1-digit hexadecimal number

HTIMEU, HTIMEL (RAM): Contains high output period

LTIMEU, LTIMEL (RAM): Contains low output period

HOUTF, LOUTF (RAM): Contains flag indicating what output is performed; low consecutive output, high consecutive output, or pulse output. Table 3.3 shows flag functions

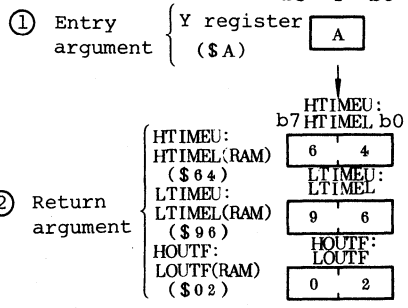


Fig. 3.5. Example of Program Module DUSET Execution

Specifications Notes:

Program Module Name: Set Duty	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: DUSET

Description:

Table 3.3. Flag Functions

HOUTF	LOUTF	Function
1	0	Outputs high consecutively from port D_4
0	0	Outputs pulses from port D_4
0	1	Outputs low consecutively from port D_4

(2) Program module DUSET calls neither program modules nor subroutines.

2. User Notes

Data set in Y must be in the range $0 \leq Y \leq 10$.
Setting data outside of this range will make high and low pulse width measurement impossible.

Program Module Name:	MCU: HMCS402C/ HMCS404C/HMCS408C	Label:
Set Duty		DUSET

Description:

3. RAM Allocation

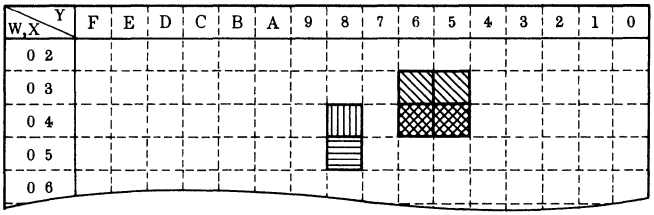


Fig. 3.6. RAM Allocation

Label	RAM	Description
HOUTF	b3 b0 MD($0,$048)	Flag indicating pulse mode from port D_4
LOUTF	b3 b0 MD($0,$058)	
HTIMEU:HTIMEL	b7 b0 MD($036,$035)	Stores upper and lower data of high output width
LTIMEU:LTIMEL	b7 b0 MD($046,$045)	Stores upper and lower data of low output width

Program Module Name:	MCU:	Label:
Set Duty	HMCS402C/ HMCS404C/HMCS408C	DUSET

Description:

4. Sample Application

```
        :
LAMD    WORK    } ..... Initialize duty
LAY             }
CALL    DUSET   ..... Call subroutine DUSET
        :
```

5. Basic Operation

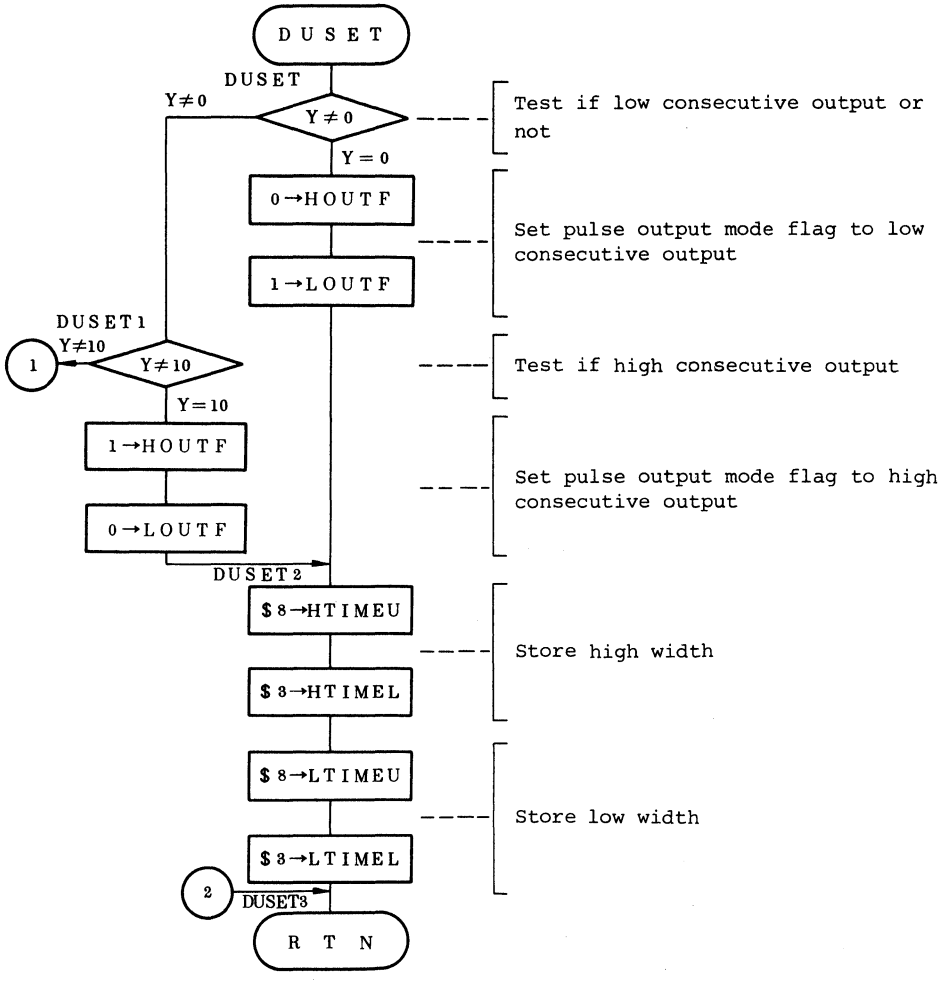
(1) Y is used as a table index pointer to indicate high or low output pulse width corresponding to duty.

(2) Test if duty in Y is 0% or 100% and if so, store 0 in HTIMEU, LTIMEU(RAM) or 10 in HTIMEL, LTIMEL(RAM).

(3) If duty set in Y is other than 0% or 100%, store high and low output width in HTIMEU, HTIMEL(RAM) and LTIMEU, LTIMEL(RAM), respectively, by pattern command.

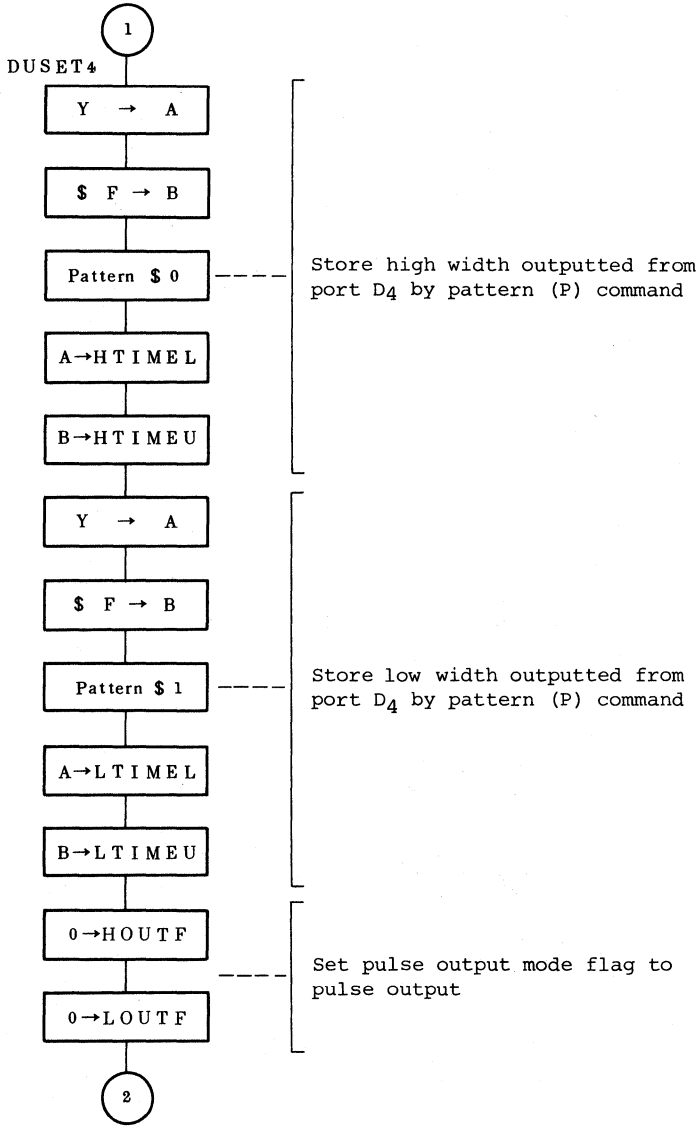
Program Module Name: Set Duty	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: DUSET

Flowchart:



Program Module Name:	MCU:	Label:
Set Duty	HMCS402C/ HMCS404C/HMCS408C	DUSET

Flowchart:



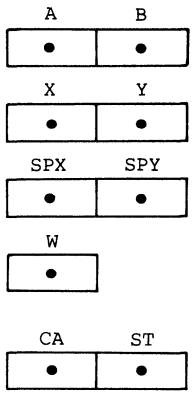
Program Module Name:	MCU: HMCS402C/ HMCS404C/HMCS408C	Label:
Output Pulse		DUOUT

Function: Output duty pulse from port.

Arguments:

Contents		1 digit = 4 bits Storage Location	No. of Digits
Entry	High output width	HTIMEU HTIMEL (RAM)	2
	Low output width	LTIMEU LTIMEL (RAM)	2
	Output pluse mode flag	HOUTF LOUTF (RAM)	2
Returns			

Changes in CPU Registers and Flags:



● : Not Affected
× : Undefined
↕ : Result

Specifications:

1 word = 10 bits

ROM (Words):	35
RAM (Digits):	9
Stack (Digits):	4
No. of cycles:	22
Reentrant:	No
Relocatable:	No
Interrupt OK?:	Yes

Description:

1. Function Details

(1) Arguments details

HTIMEU(RAM): Store upper data of high pulse width
HTIMEL(RAM): Store lower data of high pulse width
LTIMEU(RAM): Store upper data of low pulse width
LTIMEL(RAM): Store lower data of low pulse width

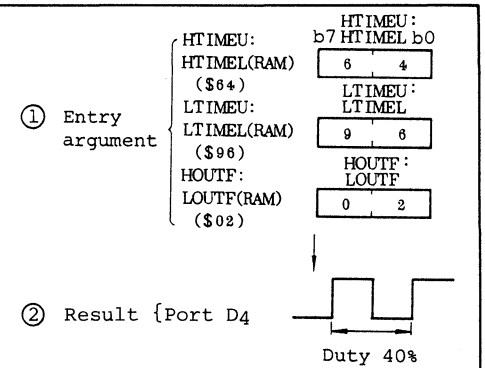


Fig. 3.7. Example of DUOUT Execution

Specifications Notes:

Program Module Name:	MCU: HMCS402C/ HMCS404C/HMCS408C	Label:
Output Pulse		DUOUT

Description:

HOUTF(RAM): Store flag indicating what output is performed; low
LOUTF(RAM) consecutive output, high consecutive output, or pulse output.

Table 3.4. Flag Functions

HOUTF	LOUTF	Function
1	0	Output high consecutively from port D_4
0	0	Output pulse from port D_4
0	1	Output low consecutively from port D_4

(2) Program module DUOUT calls neither program modules nor subroutines.

2. User Notes

(1) Initialize timer B before use.

(2) IE bit should be initialized for interrupts when using timer B interrupt.

Program Module Name: Output Pulse	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: DUOUT

Description:

3. RAM Allocation

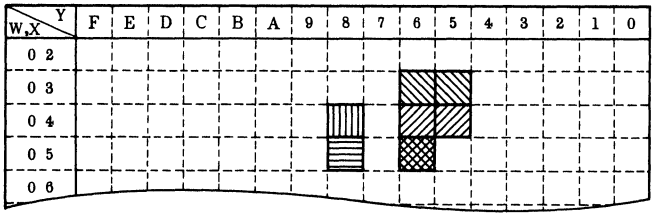


Fig. 3.8. RAM Allocation

Label	RAM	Function
HOUTF	b3 b0 MD($0, $048)	Flag specifying mode selection for pulses outputted from port D_4
LOUTF	b3 b0 MD($0, $058)	
HTIMEU:HTIMEL	b7 b0 MD($036, $035)	Store upper and lower data of high output width
LTIMEU:LTIMEL	b7 b0 MD($046, $045)	Store upper and lower data of low output width
HLFLG	b3 b0 MD($0, $056)	Flag specifying high, low or pulse from port D_4

Program Module Name:	MCU: HMCS402C/ HMCS404C/HMCS408C	Label:
Output Pulse		DUOUT

Description:

4. Sample Application

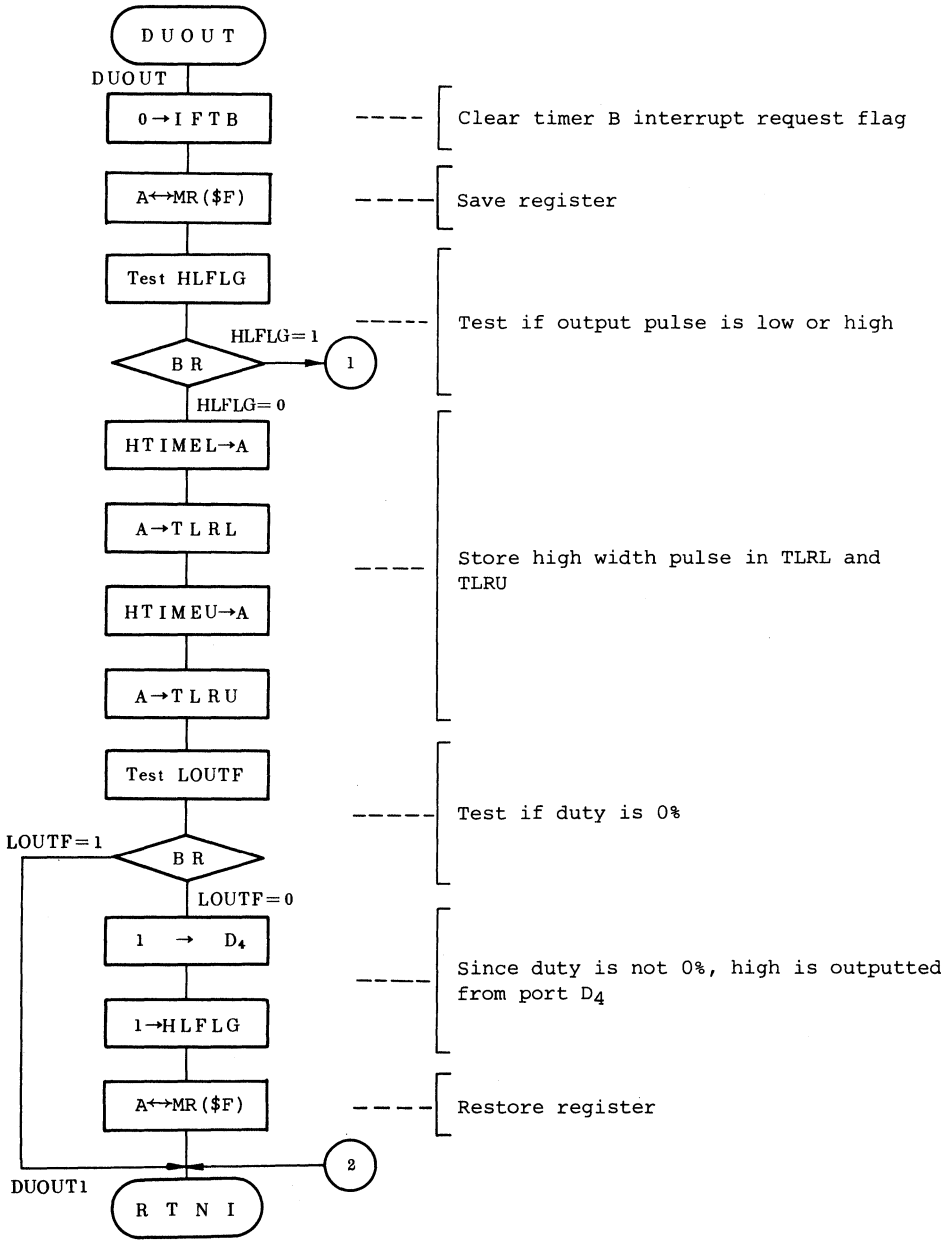
```
        ⋮
LMID    $B, $009    ····· Initialize timer mode register B
REMD    1, $002     ····· Reset timer B interrupt mask
REMD    0, $002     ····· Reset timer B interrupt request flag
SEMD    0, $000     ····· Set interrupt enable flag
LAMD    WORK        }
LYA                 } ····· Store duty from user program in
                            subroutine DUSET entry argument
CALL    DUSET       ····· Call DUSET and initialize high output
                            width and low output width in DUOUT
                            entry argument
        ⋮
```

5. Basic Operation

Determines the previous output at every timer interrupt, and changes output port, high to low, low to high. The width of high or low is stored in TLRL and TLRU.

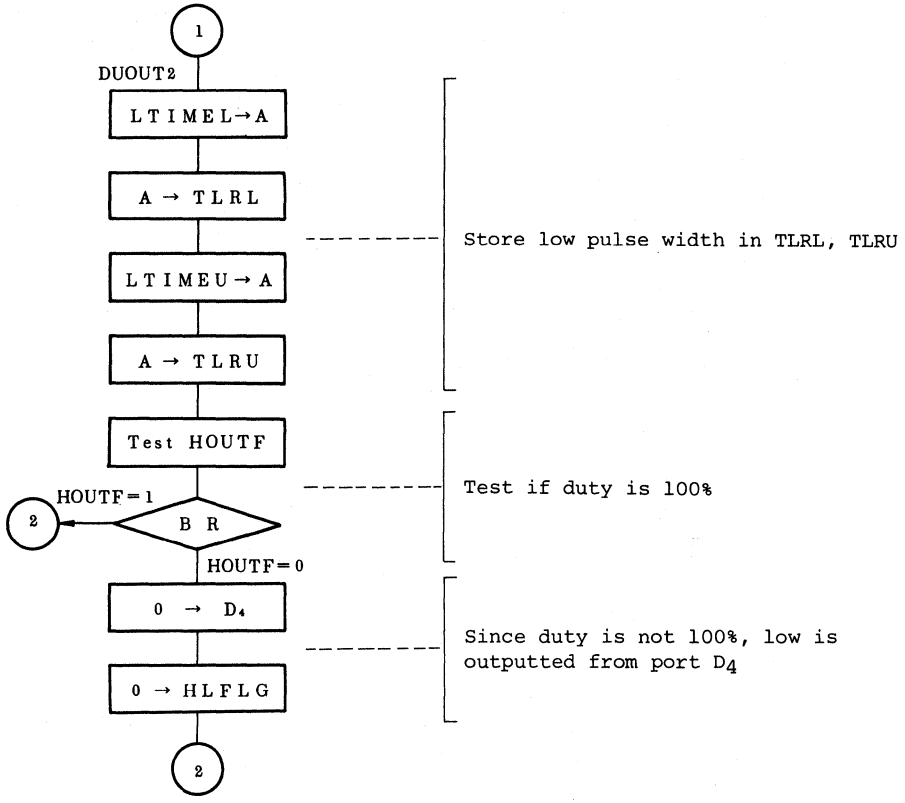
Program Module Name: Output Pulse	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: DUOUT

Flowchart:



Program Module Name: Output Pulse	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: DUOUT

Flowchart:



3.4 SUBROUTINES

This application example calls no subroutines.

3.5 PROGRAM LISTING

```
ST-NO   OBJECT   ADRS   SOURCE STATEMENTS

00001   010      0000           LLEN     132
00002                           TITLE    DUTY CONTROL OF PULSE OUTPUT
00003                  *
00004                  ****   RAM ALLOCATION   **********************************
00005                  *
00006                  WORK     EQU      $038          WORK AREA FOR DUTY DATA
00007                  HOUTF    EQU      $0.$048       UPPER DIGIT OF PULSE STATUS FLAG
00008                  LOUTF    EQU      $0.$058       LOWER DIGIT OF PULSE STATUS FLAG
00009                  HTIMEU   EQU      $036          UPPER DIGIT OF HIGH PERIOD OF PULSE
00010                  HTIMEL   EQU      $035          LOWER DIGIT OF HIGH PERIOD OF PULSE
00011                  LTIMEU   EQU      $046          UPPER DIGIT OF LOW PERIOD OF PULSE
00012                  LTIMEL   EQU      $045          LOWER DIGIT OF LOW PERIOD OF PULSE
00013                  HLFLG    EQU      $0.$056       PULSE OUTPUT SELECTION FLAG
00014                  XDCNT    EQU      $3            BCD COUNTER ADDR(X)
00015                  YDCNT    EQU      $A            BCD COUNTER LSD ADDR(Y)
00016                  BADCN    EQU      $E            BCD COUNTER FOR ADDR(Y)
00017                  *
00018                  ****   SYMBOL DEFINITIONS   ****************************
00019                  *
00020                  IE       EQU      0.$000        INTERRUPT ENABLE FLAG
00021                  IFTB     EQU      0.$002        IF OF TIMER-B
00022                  IMTB     EQU      1.$002        IM OF TIMER-B
00023                  TMB      EQU      $009          TIMER MODE REG B
00024                  TLRL     EQU      $00A          TIMER LOAD REG LOWER
00025                  TLRU     EQU      $00B          TIMER LOAD REG UPPER
00026                  ************************************************************
00027                  *                                                          *
00028                  *               VECTOR ADDRESSES                           *
00029                  *                                                          *
00030                  ************************************************************
00031                  *
00032                           ORG      $0000
00033                  *
00034   150 010  0000           JMPL     DUMN          RESET
00035   150 010  0002           JMPL     DUMN          INT0
00036   150 010  0004           JMPL     DUMN          INT1
00037   150 010  0006           JMPL     DUMN          TIMER-A
00038   150 05C  0008           JMPL     DUOUT         TIMER-B
00039                  ************************************************************
00040                  *                                                          *
00041                  *               MAIN PROGRAMN : DUMN                       *
00042                  *                                                          *
00043                  ************************************************************
00044                  *
00045                           ORG      $0010
00046                  *
00047   0F0      0010  DUMN     LWI      $0            INITIALIZE W REGISTER
00048   1AB 009  0011           LMID     $B.TMB        INITIALIZE PRESCALER 1/32
00049   189 002  0013           REMD     IMTB
00050   184 000  0015           SEMD     IE            ENABLE INTERRUPTS
00051   190 038  0017  DUMN1    LAMD     WORK          LOAD ENTRY ARGUMENT OF DUSET
00052   0D8      0019           LYA
00053   160 030  001A           CALL     DUSET         DEFINE DUTY RATE OF PULSE
00054   05C      001C           IY                     DUTY+10% -> ENTRY ARGUMENT
00055   078      001D           YNEI     11            DUTY =100% ?
00056   320      001E           BRS      DUMN2
00057   210      001F           LYI      $0            STORE 0% DUTY
```

```
00058  0AF      0020  DUMN2   LAY                    EXECUTE 0.7MS SOFTWARE TIMER
00059  194 038  0021          LMAD    WORK
00060  1A0 03A  0023          LMID    $0,$03A
00061  1A0 03B  0025          LMID    $0,$03B
00062  1A0 03C  0027          LMID    $0,$03C
00063  1A0 03D  0029          LMID    $0,$03D
00064  160 081  002B  DUMN3   CALL    DECNT
00065  06F      002D          TC
00066  317      002E          BRS     DUMN1
00067  32B      002F          BRS     DUMN3
00068                *************************************************************
00069                *                                                           *
00070                *          NAME : DUSET (SET DUTY)                          *
00071                *                                                           *
00072                *************************************************************
00073                *                                                           *
00074                *         ENTRY : Y REGISTER (DUTY DATA)                    *
00075                *       RETRUNS : HTIMEU (UPPER HIGH PERIOD OF PULSE)       *
00076                *                 HTIMEL (LOWER HIGH PERIOD OF PULSE)       *
00077                *                 LTIMEU (UPPER LOW PERIOD OF PULSE)        *
00078                *                 LTIMEL (LOWER LOW PERIOD OF PULSE)        *
00079                *                 HOUTF (UPPER PULSE STATUS FLAG)           *
00080                *                 LOUTF (LOWER PULSE STATUS FLAG)           *
00081                *                                                           *
00082                *************************************************************
00083  070      0030  DUSET   YNEI    $0             TEST IF DUTY =0% ?
00084  338      0031          BRS     DUSET1
00085  188 048  0032          REMD    HOUTF          DEFINE FLAG TO OUTPUT LOW
00086  184 058  0034          SEMD    LOUTF
00087  170 03E  0036          BRL     DUSET2
00088  07A      0038  DUSET1  YNEI    10             TEST IF DUTY =100% ?
00089  347      0039          BRS     DUSET4
00090  184 048  003A          SEMD    HOUTF          DEFINE FLAG TO OUTPUT HIGH
00091  188 058  003C          REMD    LOUTF
00092  1A8 036  003E  DUSET2  LMID    8,HTIMEU       SET 50% DUTY RATE
00093  1A3 035  0040          LMID    3,HTIMEL
00094  1A8 046  0042          LMID    8,LTIMEU
00095  1A3 045  0044          LMID    3,LTIMEL
00096  010      0046  DUSET3  RTN
00097  0AF      0047  DUSET4  LAY                    SET HIGH PERIOD OF PULSE
00098  20F      0048          LBI     $F
00099  1B0      0049          P       $0             PATTERN
00100  194 035  004A          LMAD    HTIMEL
00101  048      004C          LAB
00102  194 036  004D          LMAD    HTIMEU
00103  0AF      004F          LAY                    SET LOW PERIOD OF PULSE
00104  20F      0050          LBI     $F
00105  1B1      0051          P       $1             PATTERN
00106  194 045  0052          LMAD    LTIMEL
00107  048      0054          LAB
00108  194 046  0055          LMAD    LTIMEU
00109  188 048  0057          REMD    HOUTF          DEFINE FLAG TO OUTPUT PULSE
00110  188 058  0059          REMD    LOUTF
00111  346      005B          BRS     DUSET3
00112                *************************************************************
00113                *                                                           *
00114                *          NAME : DUOUT (OUTPUT PULSE)                      *
```

```
00115                   *                                                     *
00116                   *******************************************************
00117                   *                                                     *
00118                   *        ENTRY : HTIMEU (UPPER HIGH PERIOD OF PULSE)  *
00119                   *                HTIMEL (LOWER HIGH PERIOD OF PULSE)  *
00120                   *                LTIMEU (UPPER LOW PERIOD OF PULSE)   *
00121                   *                LTIMEL (LOWER LOW PERIOD OF PULSE)   *
00122                   *                HOUTF (UPPER PULSE STATUS FLAG)      *
00123                   *                LOUTF (LOWER PULSE STATUS FLAG)      *
00124                   *      RETURNS : NOTHING                              *
00125                   *                                                     *
00126                   *******************************************************
00127   188 002   005C  DUOUT     REMD     IFTB          CLEAR INTERRUPT REQUEST BIT
00128   2FF       005E             XMRA     $F            SAVE X REGISTER
00129   18C 056   005F             TMD      HLFLG         HIGH OR LOW OUTPUT ?
00130   372       0061             BRS      DUOUT2        BRANCH IF LOW OUTPUT
00131   190 035   0062             LAMD     HTIMEL        STORE HIGH PERIOD OF PULSE
00132   194 00A   0064             LMAD     TLRL
00133   190 036   0066             LAMD     HTIMEU
00134   194 00B   0068             LMAD     TLRU
00135   18C 058   006A             TMD      LOUTF         DUTY =0% ?
00136   370       006C             BRS      DUOUT1
00137   2E4       006D             SEDD     $4            OUTPUT HIGH PULSE
00138   184 056   006E             SEMD     HLFLG
00139   2FF       0070  DUOUT1     XMRA     $F            RESTORE X REGISTER
00140   011       0071             RTNI
00141   190 045   0072  DUOUT2     LAMD     LTIMEL        STORE LOW PERIOD OF PULSE
00142   194 00A   0074             LMAD     TLRL
00143   190 046   0076             LAMD     LTIMEU
00144   194 00B   0078             LMAD     TLRU
00145   18C 048   007A             TMD      HOUTF         DUTY =100% ?
00146   370       007C             BRS      DUOUT1
00147   264       007D             REDD     $4            OUTPUT LOW PULSE
00148   188 056   007E             REMD     HLFLG
00149   370       0080             BRS      DUOUT1
00150                   *******************************************************
00151                   *                                                     *
00152                   *         NAME : DECNT (4-DIGIT BCD COUNTER)          *
00153                   *                                                     *
00154                   *******************************************************
00155                   *                                                     *
00156                   *        ENTRY :  NOTHING                             *
00157                   *      RETURNS :  MD($03D-$03A)(4-DIGIT BCD COUNTER)  *
00158                   *                 CA FLAG (CA=0:TRUE.CA=1:OVERFLOW)   *
00159                   *                                                     *
00160                   *******************************************************
00161   223       0081  DECNT     LXI      XDCNT         LOAD ADDR(X)
00162   21A       0082             LYI      YDCNT         LOAD LSD ADDR(Y)
00163   0EF       0083             SEC                    SET CARRY FLAG
00164   230       0084  DECNT1     LAI      $0            CLEAR A
00165   018       0085             AMC                    INCREMENT BCD COUNTER
00166   0A6       0086             DAA                    CONVERT INTO BCD DATA
00167   050       0087             LMAIY                  STORE BCD COUNTER AND BCD DATA
00168   07E       0088             YNEI     BADCN         REG(Y) = / BCD COUNTER ?
00169   384       0089             BRS      DECNT1        LOOP UNTIL REG(Y) = BCD COUNTER
00170   010       008A             RTN
00171                   *******************************************************
```

```
00172                   *                                                     *
00173                   *       DATA TABLE                                    *
00174                   *                                                     *
00175                   *******************************************************
00176                   *
00177                           ORG     $0F1
00178                   *
00179   1E7     00F1            DC      $1E7            10      HIGH PERIOD OF PULSE
00180   1CE     00F2            DC      $1CE            20
00181   1B5     00F3            DC      $1B5            30
00182   19C     00F4            DC      $19C            40
00183   183     00F5            DC      $183            50
00184   16A     00F6            DC      $16A            60
00185   151     00F7            DC      $151            70
00186   138     00F8            DC      $138            80
00187   11F     00F9            DC      $11F            90
00188                   *
00189                           ORG     $1F1
00190                   *
00191   11F     01F1            DC      $11F            90      LOW PERIOD OF PULSE
00192   138     01F2            DC      $138            80
00193   151     01F3            DC      $151            70
00194   16A     01F4            DC      $16A            60
00195   183     01F5            DC      $183            50
00196   19C     01F6            DC      $19C            40
00197   1B5     01F7            DC      $1B5            30
00198   1CE     01F8            DC      $1CE            20
00199   1E7     01F9            DC      $1E7            10
00200                   *
00201                           END
```

4.1 HARDWARE DESCRIPTION

4.1.1 Function

Measures the input cycle of a pulse to determine pulse width in the range from 100 μs to 256 μs stores result as a binary coded decimal (BCD) number.

4.1.2 Microcomputer Operation

The HMCS404C uses the eight-bit auto reload type timer and event counter to fetch values in the timer and event counter on the falling and rising edges of the $\overline{INT_1}$ pin, using the difference between these values to measure the pulse width.

4.1.3 Circuit Diagram

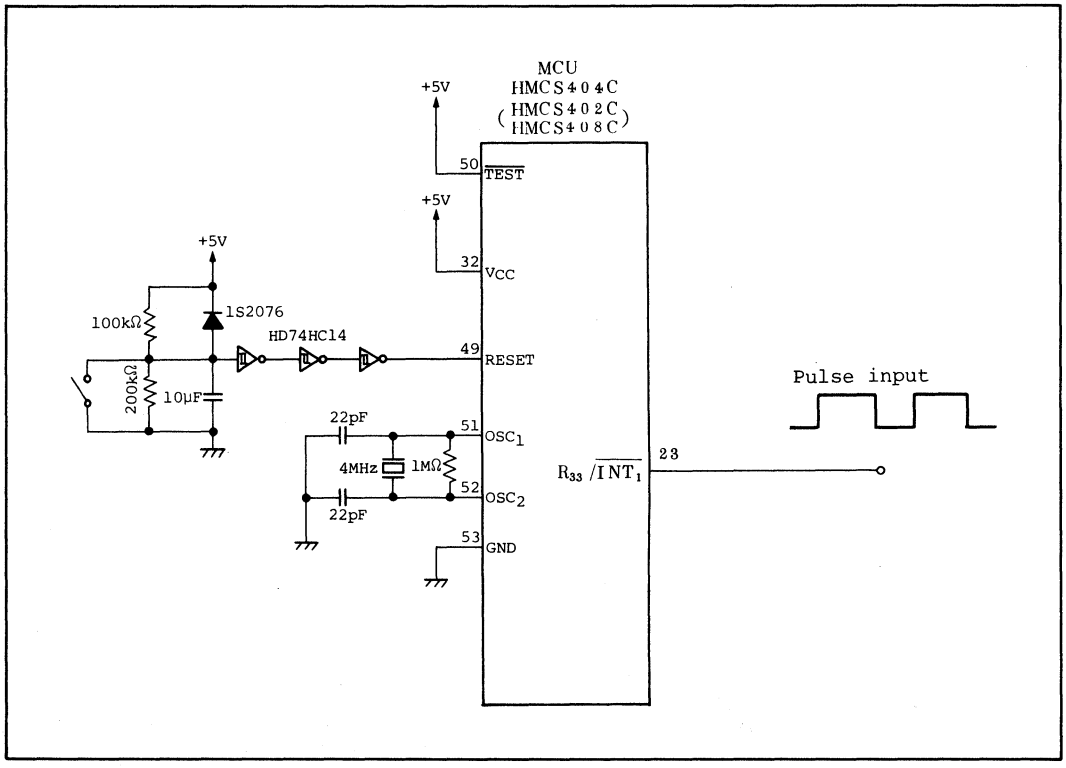


Fig. 4.1. Pulse Width Measurement Circuit

4.1.4 Pin Functions

Pin functions for pulse width measurement are shown in Table 4.1.

Table 4.1. Pin Functions

Pin Name (HMCS404C)	Input/ Output	Active level (High or Low)	Function
$R_{33}/\overline{INT_1}$	Input	Low	Detects falling edge of input signal and executes interrupt routine

4.1.5 Hardware Operation

Fig. 4.2 shows pulse width measurement. Since system clock cycle is 2 µs, E clock cycle is 1 µs. In Fig. 4.2, pulse width W is 6 µs.

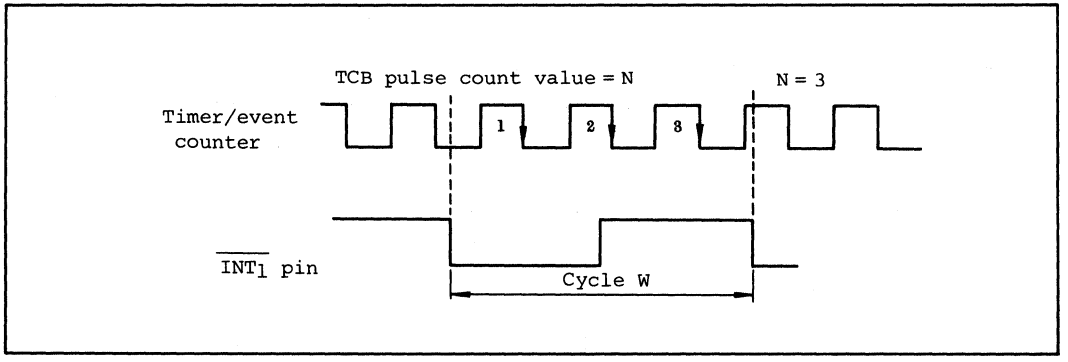


Fig. 4.2. Measure Pulse Width

4.2.1 Program Module Configuration

The program module configuration for pulse width measurement and BCD conversion is shown in Fig. 4.3.

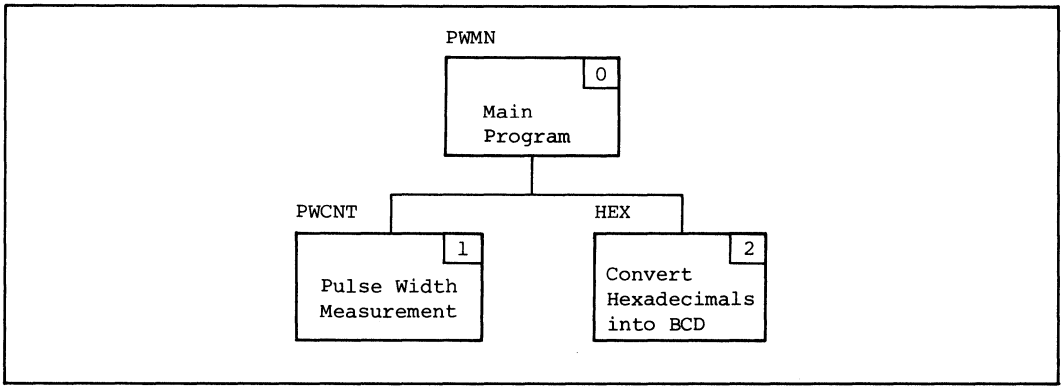


Fig. 4.3. Program Module Configuration

4.2.2 Program Module Functions

Program module functions are summarized in Table 4.2.

Table 4.2. Program Module Functions

No.	Program Module Name	Label	Function
0	Main Program	PWMN	Measures pulse width as a BCD number
1	Measure Pulse Width	PWCNT	Obtains pulse width as a 2-byte hexadecimal number
2	Convert Hexadecimals into BCD	HEX	Converts 2-byte hexadecimal number into BCD number. Refer to HEX in HMCS400 Series Application Note (Software Edition) for details

4.2.3 Program Module Process Flow (Main Program)

The flowchart in Fig. 4.4 is an example of pulse width measurement performed by the program module in Fig. 4.3.

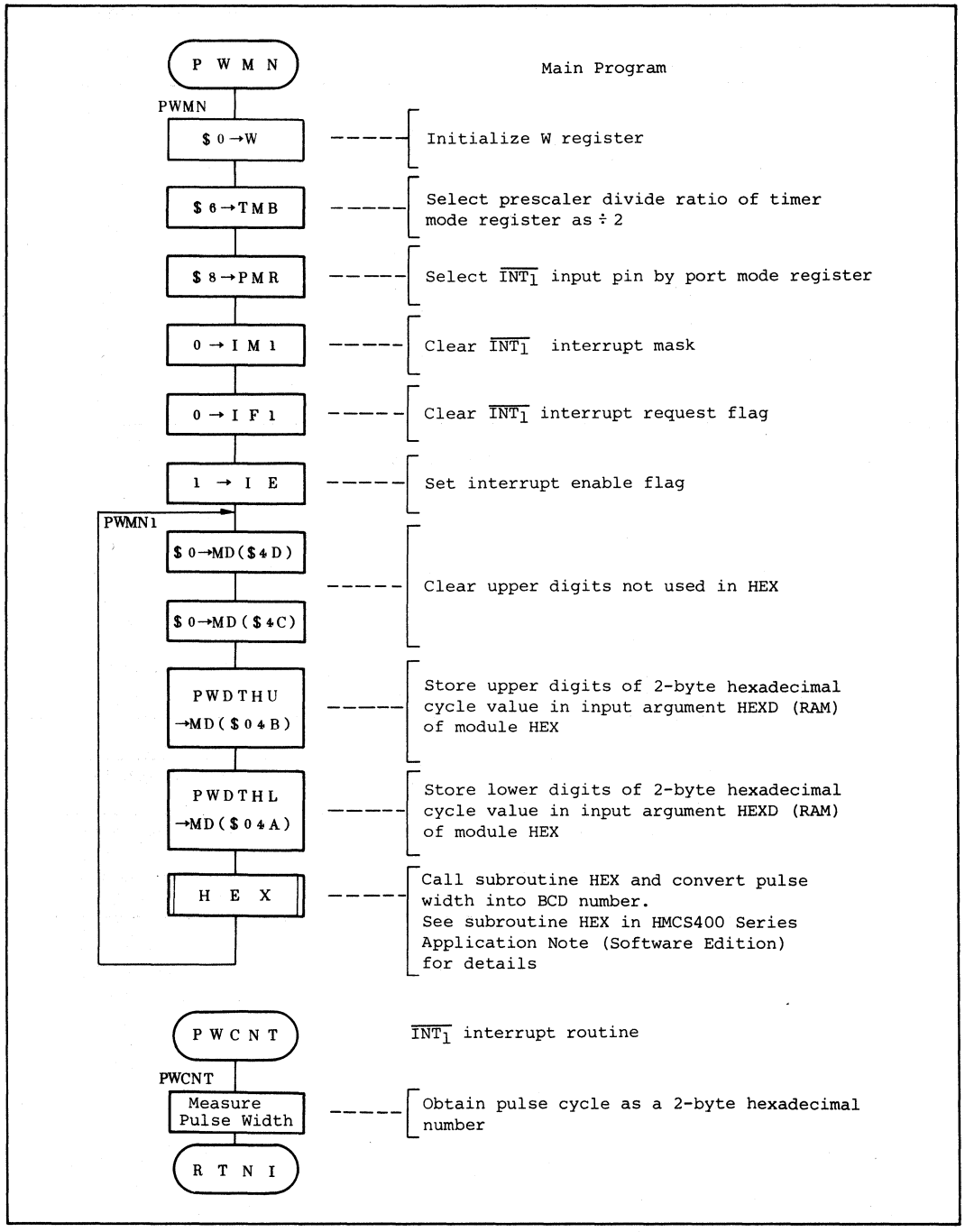


Fig. 4.4. Program Module Flowchart

Program Module Name:	MCU:	Label:
MEASURE PULSE WIDTH	HMCS402C/ HMCS404C/HMCS408C	PWCNT

Function:

Obtains pulse cycle as a 2-byte hexadecimal number, and stores result in PWDTHU, PWDTHL (RAM).

Arguments:

Contents		1 digit = 4 bits Storage Location	No. of Digits
Entry			
Returns	Period (upper)	PWDTHU (RAM)	1
	Period (lower)	PWDTHL (RAM)	1

Changes in CPU Registers and Flags:

A	B
●	●

X	Y
●	●

SPX	SPY
●	●

W
●

CA	ST
●	●

● : Not Affected
× : Undefined
↕ : Result

Specifications:

1 word = 10 bits

ROM (Words):	34
RAM (Digits):	6
Stack (Digits):	0
No. of cycles:	36
Reentrant:	No
Relocatable:	No
Interrupt OK?:	No

Description:

1. Function Details

(1) Argument details

PWDTHU, PWDTHL
(RAM): Contains pulse width as a 2-byte hexadecimal number.

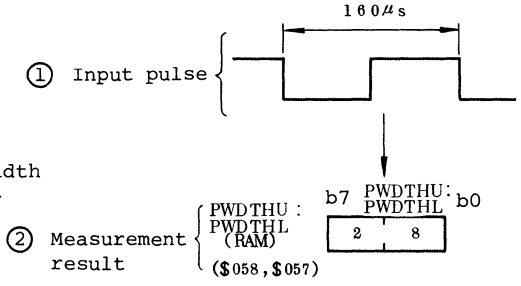


Fig. 4.5. Example of PWCNT Execution

Specifications Notes:

Program Module Name: MEASURE PULSE WIDTH	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: PWCNT

Description:

(2) Example of PWCNT execution is shown in Fig. 4.5. If pulse, whose high period of pulse is 160 µs, is input as shown in part ① of Fig. 4.5, measurement result is stored in PWDTHU, PWDTHL(RAM) as a hexadecimal number.

(3) PWCNT calls neither program modules nor subroutines.

2. User Notes

(1) Only period of pulses between 100 and 256 µs can be correctly measured.

(2) Bit IE is set to enable $\overline{INT_1}$ interrupt.

(3) Since a 2 µs system clock is employed, an oscillator frequency of 4 MHz is used to execute measurement of pulse cycles.

3. RAM Allocation

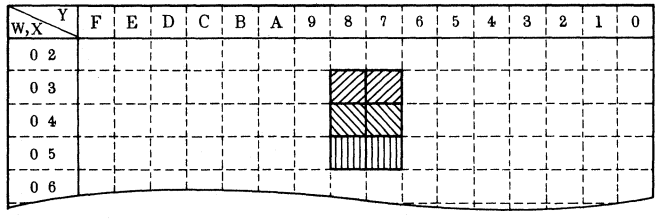


Fig. 4.6. RAM Allocation

Label	RAM	Description
TCBNWU:TCBNWL	b7 b0 MD(\$038, \$037)	Stores timer/event counter value at the second falling edge
TCBODU:TCBODL	b7 b0 MD(\$048, \$047)	Stores timer/event counter value at the first falling edge
PWDTHU:PWDTHL	b7 b0 MD(\$058, \$057)	Stores difference (cycle) in timer/event counter at the first and second falling edges

Program Module Name: MEASURE PULSE WIDTH	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: PWCNT

Description:

4. Sample Application

```
        :
LMID    $8, PMR     ..... Initialize port mode register as INT1,
                          input pin
REMD    IF1         ..... Reset external interrupt request flag
        :
```

5. Basic Operation

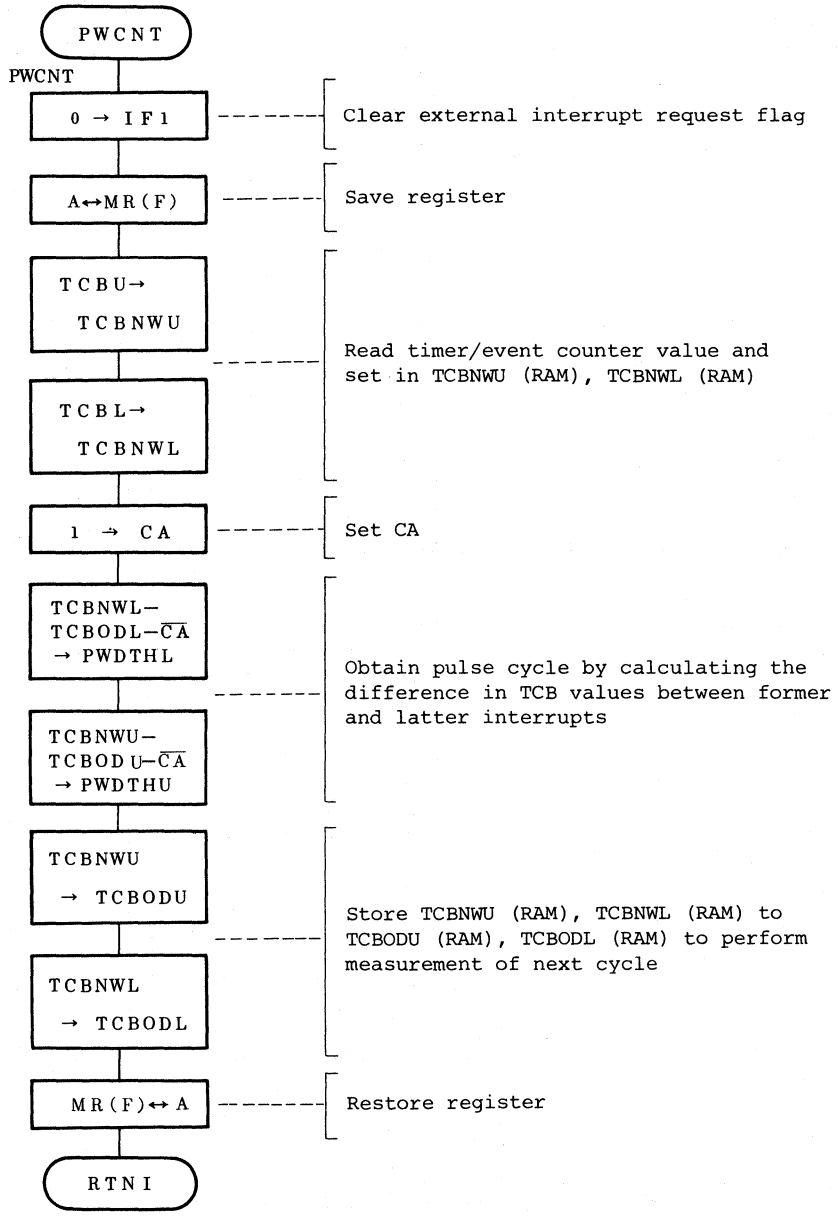
Reads the timer/event counter value when the external interrupt request flag is set on the $\overline{INT_1}$ pin input falling edge.

In the same manner, read the timer/event counter value at the next falling edge.

Pulse cycle is measured by calculating the difference between the two values.

Program Module Name: MEASURE PULSE WIDTH	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: PWCNT

Flowchart:



4.4 SUBROUTINE DESCRIPTION

This application example calls no subroutines.

4.5 PROGRAM LISTING

```
ST-NO   OBJECT   ADRS   SOURCE STATEMENTS

00001   1E7      0000           LLEN     132
00002                           TITLE    PULSE WIDTH MEASUREMENT
00003                  *
00004                  ****   RAM ALLOCATION   ****************************************
00005                  *
00006                  PWDTHU   EQU      $058          UPPER DIGIT OF PULSE WIDTH
00007                  PWDTHL   EQU      $057          LOWER DIGIT OF PULSE WIDTH
00008                  TCBNWU   EQU      $038          TIMER CNTR B NEW UPPER
00009                  TCBNWL   EQU      $037          TIMER CNTR B NEW LOWER
00010                  TCBODU   EQU      $048          TIMER CNTR B OLD UPPER
00011                  TCBODL   EQU      $047          TIMER CNTR B OLD LOWER
00012                  XHEXD    EQU      $4            HEXADECIMAL AND BCD DATA ADDR(X)
00013                  YHHEX    EQU      $A            2-BYTE HEXADECIMAL DATA LSD ADDR(Y)
00014                  YHDEC    EQU      $2            5-DIGIT BCD DATA LSD ADDR(Y)
00015                  BAHEX    EQU      $E            2-BYTE HEXADECIMAL DATA MSD ADDR(Y)+1
00016                  BADEC    EQU      $7            5-DIGIT BCD DATA MSD ADDR(Y)+1
00017                  *
00018                  ****   SYMBOL  DEFINITIONS   ************************************
00019                  *
00020                  TMB      EQU      $009          TIMER MODE REGISTER B
00021                  PMR      EQU      $004          PORT MODE REGISTER
00022                  IM1      EQU      $1,$001       INTERRUPT MODE FLAG
00023                  IF1      EQU      $0,$001       INTERRUPT REQUEST FLAG
00024                  IE       EQU      $0,$000       INTERRUPT ENABLE FLAG
00025                  TCBU     EQU      $00B          TIMER CNTR B UPPER
00026                  TCBL     EQU      $00A          TIMER CNTR B LOWER
00027                  ************************************************************
00028                  *                                                          *
00029                  *              VECTOR ADDRESSES                            *
00030                  *                                                          *
00031                  ************************************************************
00032                  *
00033                           ORG      $0000
00034                  *
00035   150 010  0000           JMPL     PWMN          RESET
00036   150 010  0002           JMPL     PWMN          INT0
00037   150 02A  0004           JMPL     PWCNT         INT1
00038   150 010  0006           JMPL     PWMN          TIMER-A
00039   150 010  0008           JMPL     PWMN          TIMER-B
00040                  ************************************************************
00041                  *                                                          *
00042                  *              MAIN PROGRAM : PWMN                         *
00043                  *                                                          *
00044                  ************************************************************
00045                  *
00046                           ORG      $0010
00047                  *
00048   0F0      0010  PWMN     LWI      $0            INITIALIZE W REGISTER
00049   1A6 009  0011           LMID     $6,TMB        SELECT PRESCALER AS 1/2
00050   1A8 004  0013           LMID     $8,PMR        SELECT INT1 PIN
00051   189 001  0015           REMD     IM1           ENABLE INT1 INTERRUPT
00052   188 001  0017           REMD     IF1
00053   184 000  0019           SEMD     IE            ENABLE INTERRUPT
00054   1A0 04D  001B  PWMN1    LMID     $0,$04D       LOAD ENTRY ARGUMENT
00055   1A0 04C  001D           LMID     $0,$04C
00056   190 058  001F           LAMD     PWDTHU
00057   194 04B  0021           LMAD     $04B
```

```
00058   190 057  0023            LAMD     PWDTHL
00059   194 04A  0025            LMAD     $04A
00060   160 04C  0027            CALL     HEX             CONVERT HEX DATA INTO BCD DATA
00061   31B      0029            BRS      PWMN1
00062                    ********************************************************************
00063                    *                                                                  *
00064                    *          NAME : PWCNT (MEASURE PULSE WIDTH)                      *
00065                    *                                                                  *
00066                    ********************************************************************
00067                    *                                                                  *
00068                    *         ENTRY : NOTHING                                          *
00069                    *       RETURNS : PWDTHU (UPPER PULSE WIDTH)                       *
00070                    *                 PWDTHL (LOWER PULSE WIDTH)                       *
00071                    *                                                                  *
00072                    ********************************************************************
00073   188 001  002A    PWCNT    REMD     IF1             CLEAR INTERRUPT REQUEST FLAG
00074   2FF      002C             XMRA     $F              SAVE REGISTER
00075   190 00B  002D             LAMD     TCBU
00076   194 038  002F             LMAD     TCBNWU          STORE TIMER/EVENT COUNTER
00077   190 00A  0031             LAMD     TCBL
00078   194 037  0033             LMAD     TCBNWL
00079   0EF      0035             SEC                      SET CARRY FLAG
00080   190 047  0036             LAMD     TCBODL          CALCULATE PULSE WIDTH
00081   198 037  0038             SMCD     TCBNWL
00082   194 057  003A             LMAD     PWDTHL
00083   190 048  003C             LAMD     TCBODU
00084   198 038  003E             SMCD     TCBNWU
00085   194 058  0040             LMAD     PWDTHU
00086   190 038  0042             LAMD     TCBNWU          TCBNEW->TCBOLD
00087   194 048  0044             LMAD     TCBODU
00088   190 037  0046             LAMD     TCBNWL
00089   194 047  0048             LMAD     TCBODL
00090   2FF      004A             XMRA     $F              RESTORE REGISTER
00091   011      004B             RTNI
00092                    ********************************************************
00093                    *                                                      *
00094                    *         NAME : HEX (CONVERT 2-BYTE HEXADCIMALS INTO  *
00095                    *                  5-DIGIT BCD)                        *
00096                    ********************************************************
00097                    *                                                      *
00098                    *        ENTRY : MD($04D-$04A) (2-BYTE HEXADECIMALS)   *
00099                    *      RETURNS : MD($046-$042) (5-DIGIT BCD)           *
00100                    *                                                      *
00101                    ********************************************************
00102   224      004C    HEX      LXI      XHEXD           LOAD HEXADECIMAL ADDR(X)
00103   212      004D             LYI      YHDEC           LOAD BCD DATA ADDR(Y)
00104   290      004E    HEX1     LMIIY    $0              CLEAR BCD DATA ADDR(Y)
00105   077      004F             YNEI     BADEC
00106   34E      0050             BRS      HEX1
00107   20F      0051             LBI      $F              LOAD SHIFT COUNTER
00108   21A      0052    HEX2     LYI      YHHEX           LOAD HEXADECIMAL DATA ADDR(Y)
00109   090      0053    HEX3     LAM                      SHIFT HEXADECIMAL DATA 1-BIT LEFT
00110   0A1      0054             ROTL
00111   050      0055             LMAIY
00112   07E      0056             YNEI     BAHEX
00113   353      0057             BRS      HEX3
00114   212      0058             LYI      YHDEC           LOAD BCD DATA LSD ADDR(Y)
```

```
00115  090    0059  HEX4   LAM                    BCD DATA AREA *2+CA->A
00116  0A1    005A         ROTL
00117  0A6    005B         DAA                    CONVERT INTO BCD DATA ?
00118  050    005C         LMAIY                  LOAD DECIMAL DATA
00119  077    005D         YNEI     BADEC         TEST IF CONVERSION IS COMPLETED
00120  359    005E         BRS      HEX4
00121  0CF    005F         DB                     DECREMENT SHIFT COUNTER
00122  352    0060         BRS      HEX2          LOOP UNTIL SHIFT COUNTER = $F
00123  010    0061         RTN
00124                *
00125                       END
```

SECTION 5. INPUT PULSE COUNT

5.1 HARDWARE DESCRIPTION

5.1.1 Function

Counts input pulses up to 255 pulses; the count value is returned as a Hexadecimal number.

5.1.2. Microcomputer Operation

Inputs pulses through the $\overline{INT_1}$ Pin of the HMCS404C and performs pulse count by counting up timer B timer/event counter (hereinafter, TCB).

5.1.3 Circuit Diagram

Input pulse measurement circuit is shown in Fig. 5.1.

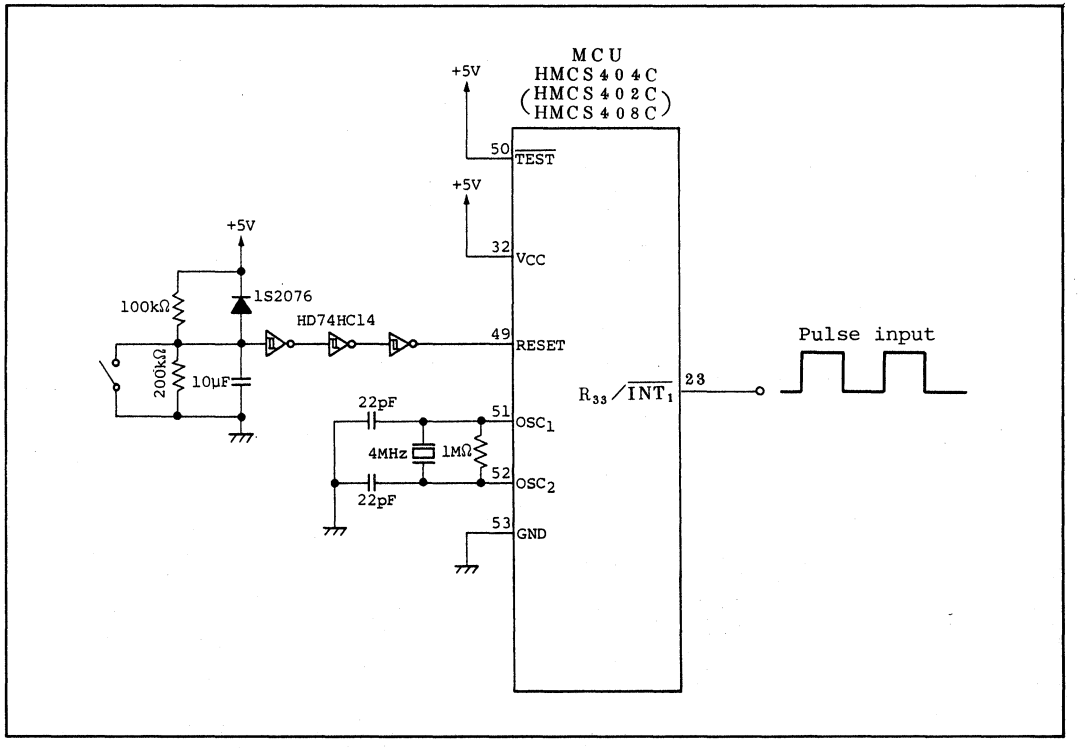


Fig. 5.1. Input Pulse Measurement Circuit

5.1.4 Pin Functions

Pin functions of HMCS404C for counting pulses is shown in Table 5.1.

Table 5.1. Pin Functions

Pin Name (HMCS404C)	Input/ Output	Active Level (High or Low)	Function
$R_{33}/\overline{INT_1}$	Input	Low	Inputs pulse event

5.1.5 Hardware Operation

Fig. 5.2.shows input pulse count using $\overline{INT_1}$ pin of the HMCS404C. To set start/end timing for counting input pulses, the procedure below must be performed in the main program.

① Set flag in STRTF(RAM)

② Clear flag is STRTF(RAM)

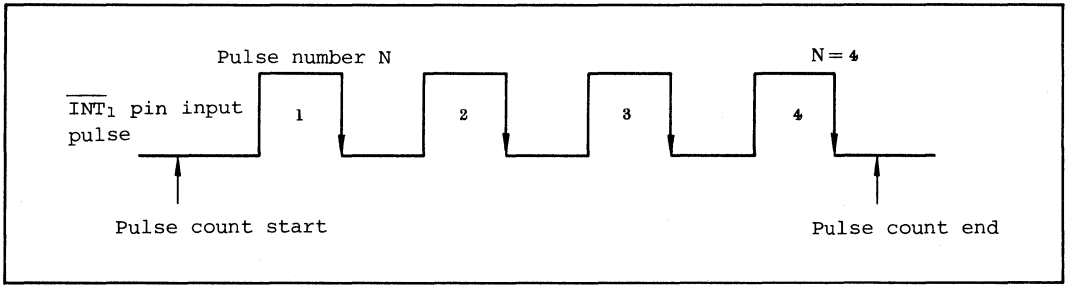


Fig. 5.2. Input Pulse Count

5.2.1 Program Module Configuration

The program module configuration for input pulse count is shown in Fig. 5.3.

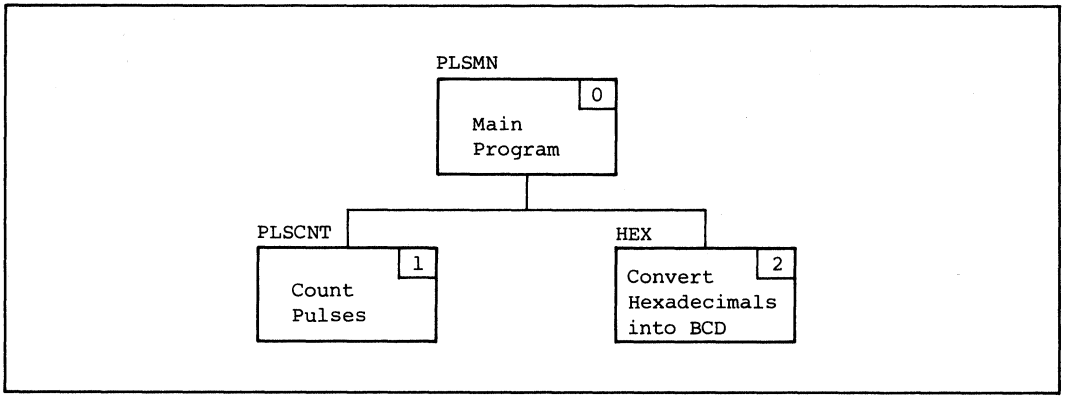


Fig. 5.3. Program Module Configuration

5.2.2 Program Module Functions

Program module functions are summarized in Table 5.2.

Table 5.2. Program Module Functions

No.	Program Module Name	Label	Function
0	Main Program	PLSMN	Counts input pulses as a BCD numbers
1	Count Pulses	PLSCNT	Obtains input pulse number by TCB value
2	Convert Hexadecimals into BCD	HEX	Converts 1-byte hexadecimal number into BCD number. Refer to HEX in HMCS400 Series Application Note (Software Edition) for details

5.2.3 Program Module Process Flow (Main Program)

The flowchart in Fig. 5.4 is an example of counting input pulses, performed by the program module in Fig. 5.3.

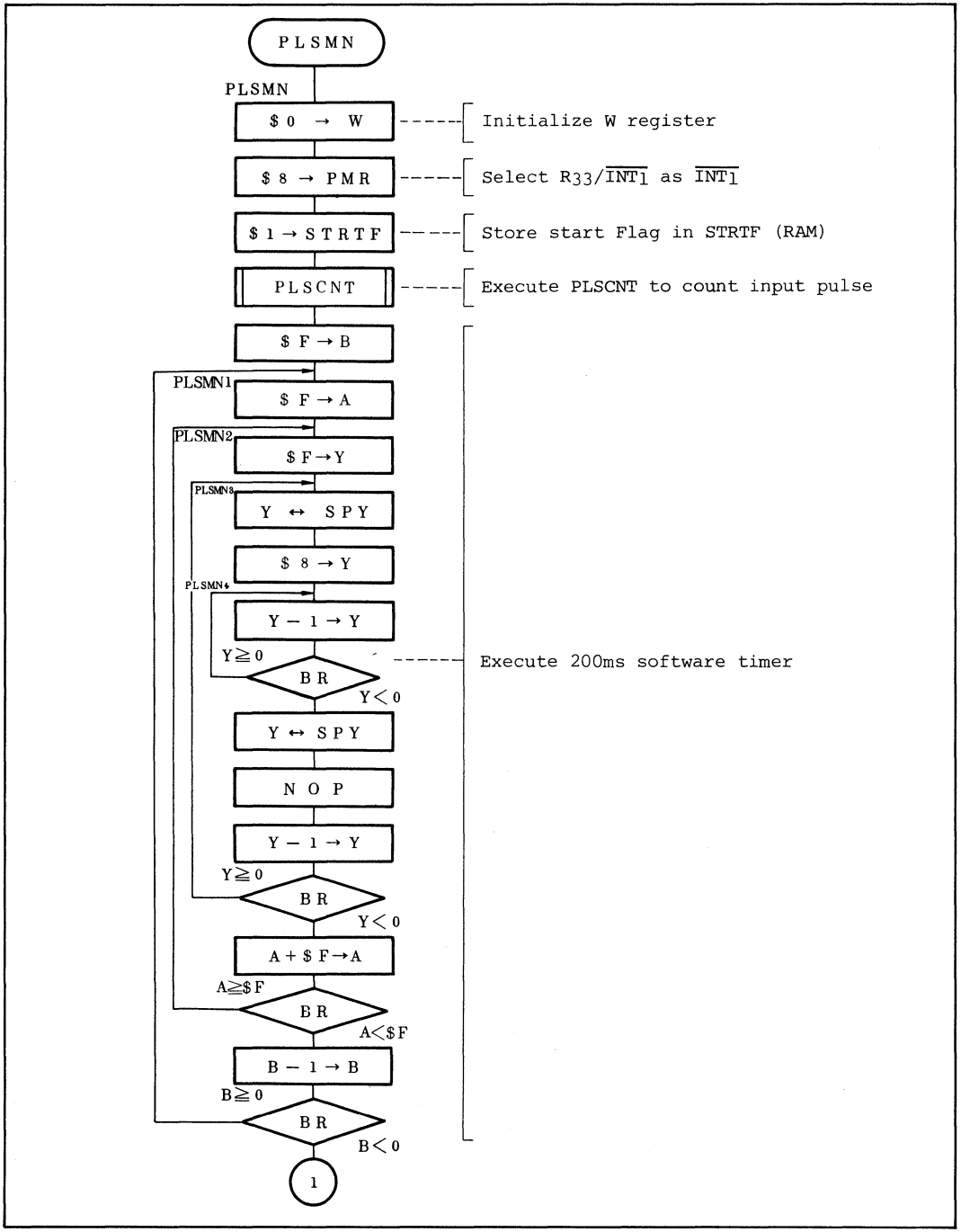


Fig. 5.4. Program Module Flowchart

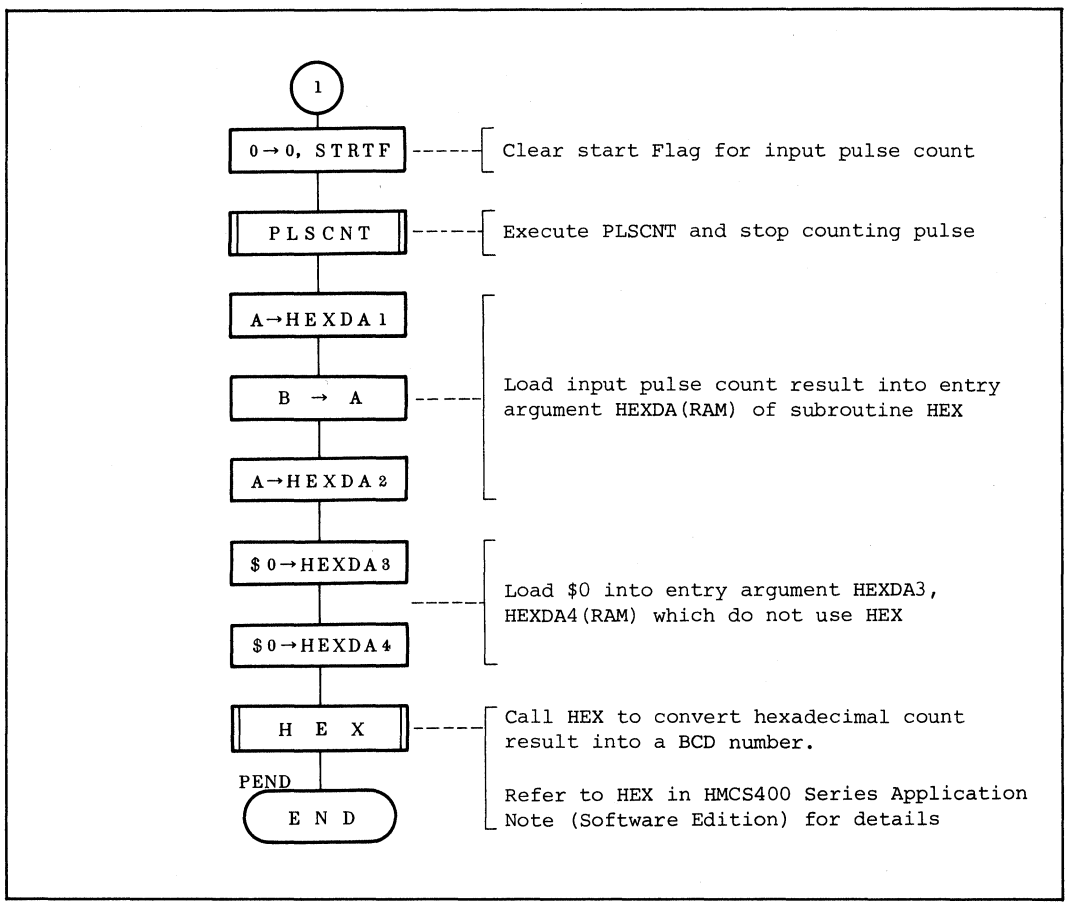


Fig. 5.4. Program Module Flowchart (cont)

Program Module Name: COUNT PULSES	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: PLSCNT

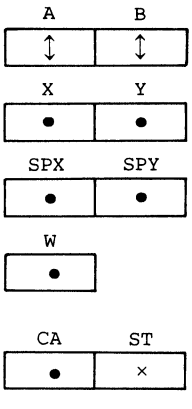
Function:

Counts pulses input from $\overline{INT_1}$ pin, and loads count result into Accumulator, and B register.

Arguments:

	Contents	1 digit = 4 bits Storage Location	No. of Digits
Entry	Start/stop request flag	STRTF (RAM)	1
Returns	Set TCB value	A, B	2

Changes in CPU Registers and Flags:



● : Not Affected
× : Undefined
↕ : Result

Specifications:

1 word = 10 bits
ROM (Words): 16
RAM (Digits): 1
Stack (Digits): 0
No. of cycles: 13
Reentrant: No
Relocatable: No
Interrupt OK?: No

Description:

1. Function Details

(1) Argument details

STRTF(RAM): Holds flag indicating whether input pulse count will start or stop. Table 5.3 shows flag functions.

A.B: Contains input pulse count result as a 1-byte hexadecimal number.

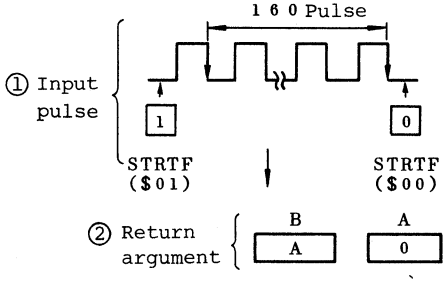


Fig. 5.5. Example of PLSCNT Execution

Specifications Notes:

Program Module Name: COUNT PULSES	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: PLSCNT

Description:

Table 5.3. Flag Function

Label	Bit 0	Function
STRTF	0	Ends pulse count
	1	Start pulse count

(2) Example of PLSCNT execution is shown in Fig. 5.5.

If 160 pulse is inputted by $\overline{INT_1}$ pin as shown in port ① of Fig. 5.5, loads measurement result into Accumulator, B register as shown in port ② of Fig. 5.5.

(3) PLSCNT calls neither the program modules nor subroutines.

2. User Notes

The following procedure must be performed before PLSCNT execution.

(1) Counts input pulse up to 255 pulses by using TCB.

(2) Sets $R_{33}/\overline{INT_1}$ as $\overline{INT_1}$ before using module PLSCNT.

(3) Sets STRTF to indicate the start and end of pulse counting.

3. RAM Allocation

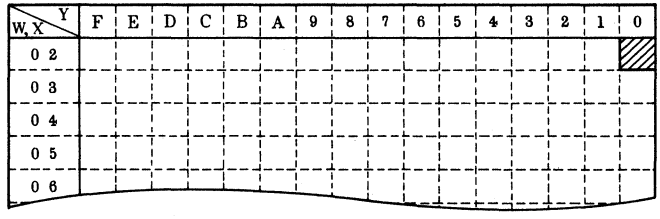


Fig. 5.6. RAM Allocation

Label	RAM	Description
STRTF	b3 b0  MD(\$020)	Flag indicating whether input pulse will start or stop

Program Module Name: COUNT PULSES	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: PLSCNT

Description:

4. Sample Application

```
WORK1   EQU     $030    } ..... Allocate RAM area for pulse count result
WORK2   EQU     $031    } ..... in Hexadecimal
          :
        LMID    $8, PMR ..... Select R33/INT1 as INT1
        LMID    $1, STRTF ..... Store Flag indicating start of pulse count

        CALL    PLSCNT  ..... Call PLSCNT and start pulse count

          :
        REMD    0, STRTF ..... Clear flag indicating end of pulse count

        CALL    PLSCNT  ..... Call PLSCNT and end pulse count

        LMAD    WORK1   } ..... Store Hexadecimal pulse count in return
        LAB             } ..... argument into RAM area
        LMAD    WORK2   }
          :
```

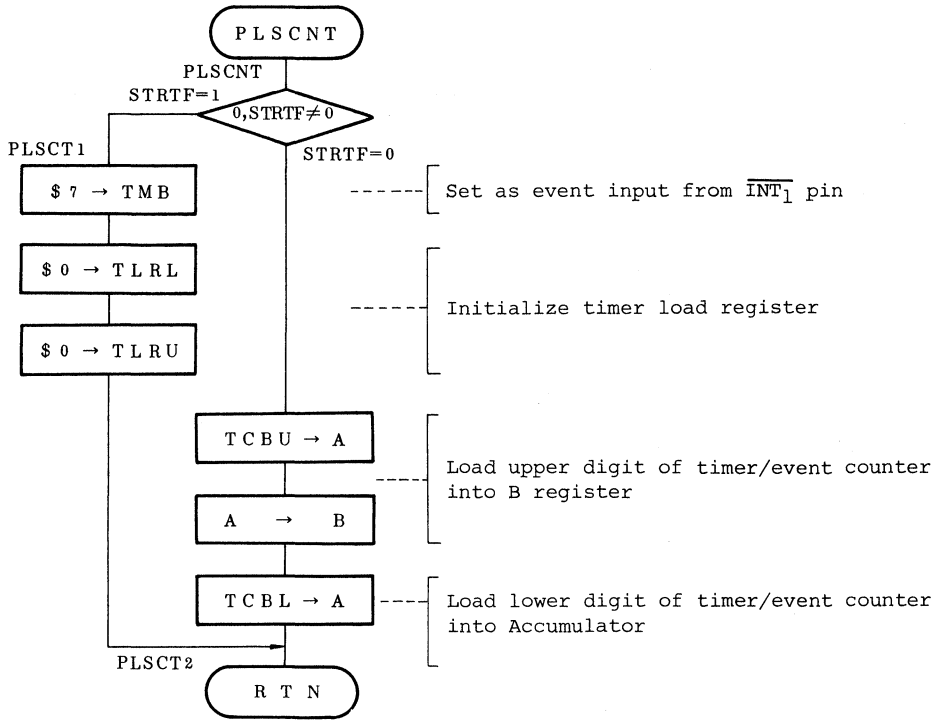
5. Basic Operation

(1) TCB (timer/event counter) is initialized by writing $00 into TCR (timer/load register).

(2) TCB counts up so that at the point of ending pulse count, reading TCB will enable determination of the pulse count.

Program Module Name: COUNT PULSES	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: PLSCNT

Flowchart:



5.4 SUBROUTINE DESCRIPTION

This application example calls no subroutines.

5.5 PROGRAM LISTING

```
ST-NO   OBJECT    ADRS   SOURCE STATEMENTS

00001   010       0000           LLEN     132
00002                            TITLE    INPUT PULSE COUNT
00003                  *
00004                  ****   RAM ALLOCATION   ****************************************
00005                  *
00006                  STRTF    EQU      $020          START/STOP REQUEST FLAG
00007                  HEXDA1   EQU      $04A          HEXADECIMAL DATA (1-DIGIT)
00008                  HEXDA2   EQU      $04B          HEXADECIMAL DATA (2-DIGIT)
00009                  HEXDA3   EQU      $04C          HEXADECIMAL DATA (3-DIGIT)
00010                  HEXDA4   EQU      $04D          HEXADECIMAL DATA (4-DIGIT)
00011                  XHEXD    EQU      $4            HEXADECMAL AND BCD DATA ADDR(X)
00012                  YHHEX    EQU      $A            2-BYTE HEXADECIMAL DATA LSD ADDR(Y)
00013                  YHDEC    EQU      $2            5-DIGIT BCD DATA LSD ADDR(Y)
00014                  BAHEX    EQU      $E            2-BYTE HEXADECIMAL DATA MSD ADDR(Y)+1
00015                  BADEC    EQU      $7            5-DIGIT BCD DATA MSD ADDR(Y)+1
00016                  *
00017                  ****   SYMBOL DEFINITIONS   ************************************
00018                  *
00019                  PMR      EQU      $004          PORT MODE REGISTER
00020                  TMB      EQU      $009          TIMER MODE REGISTER
00021                  TCBL     EQU      $00A          TIMER/EVENT COUNTER B (LOWER)
00022                  TCBU     EQU      $00B          TIMER/EVENT COUNTER B (UPPER)
00023                  TLRL     EQU      $00A          TIMER LOAD REGISTER (LOWER)
00024                  TLRU     EQU      $00B          TIMER LOAD REGISTER (UPPER)
00025                  ********************************************************************
00026                  *                                                                  *
00027                  *              VECTOR ADDRESSES                                    *
00028                  *                                                                  *
00029                  ********************************************************************
00030                  *
00031                           ORG      $0000
00032                  *
00033   150 010   0000         JMPL     PLSMN         RESET
00034   150 010   0002         JMPL     PLSMN         INT0
00035   150 010   0004         JMPL     PLSMN         INT1
00036   150 010   0006         JMPL     PLSMN         TIMER-A
00037   150 010   0008         JMPL     PLSMN         TIMER-B
00038                  *
00039                           ORG      $000C
00040                  *
00041   150 010   000C         JMPL     PLSMN         SERIAL
00042                  ********************************************************************
00043                  *                                                                  *
00044                  *              MAIN PROGURAM : PLSMN                               *
00045                  *                                                                  *
00046                  ********************************************************************
00047                  *
00048                           ORG      $0010
00049                  *
00050   0F0       0010 PLSMN    LWI      $0            INITIALIZE W REGISTER
00051   1A8 004   0011         LMID     $8,PMR        SELECT INT1
00052   1A1 020   0013         LMID     $1,STRTF      SET START FLAG
00053   160 036   0015         CALL     PLSCNT        START TO COUNT PULSE
00054   20F       0017         LBI      15            EXECUTE 200MS SOFTWARE TIMER
00055   23F       0018 PLSMN1  LAI      15
00056   21F       0019 PLSMN2  LYI      15
00057   002       001A PLSMN3  XSPY
```

```
00058  218       001B            LYI     8
00059  0DF       001C  PLSMN4    DY
00060  31C       001D            BRS     PLSMN4
00061  002       001E            XSPY
00062  000       001F            NOP
00063  0DF       0020            DY
00064  31A       0021            BRS     PLSMN3
00065  28F       0022            AI      15
00066  319       0023            BRS     PLSMN2
00067  0CF       0024            DB
00068  318       0025            BRS     PLSMN1
00069  188 020   0026            REMD    0.STRTF         CLEAR START FLAG
00070  160 036   0028            CALL    PLSCNT          STOP COUNTING PULSE
00071  194 04A   002A            LMAD    HEXDA1          LOAD HEX ENTRY ARGUMENT
00072  048       002C            LAB
00073  194 04B   002D            LMAD    HEXDA2          B REGISTER ---> HEX.DATA AREA 2
00074  1A0 04C   002F            LMID    $0.HEXDA3       CLEAR UNUSED HEX.DATA AREA 3
00075  1A0 04D   0031            LMID    $0.HEXDA4       CLEAR UNUSED HEX.DATA AREA 4
00076  160 047   0033            CALL    HEX
00077  335       0035  PEND      BRS     PEND
00078                  ********************************************************************
00079                  *                                                                  *
00080                  *       NAME : PLSCNT (COUNT PULSE)                                *
00081                  *                                                                  *
00082                  ********************************************************************
00083                  *                                                                  *
00084                  *      ENTRY : STRTF (START/STOP REQUEST FLAG)                     *
00085                  *    RETURNS : ACCUMULATOR A,B REGISTER (2'S COMPLEMENT OF TCB)*
00086                  *                                                                  *
00087                  ********************************************************************
00088  18C 020   0036  PLSCNT    TMD     0.STRTF         TEST START FLAG
00089  340       0038            BRS     PLSCT1
00090  190 00B   0039            LAMD    TCBU            LOAD UPPER DIGIT OF TIMER/EVENT COUNTER
00091  0C8       003B            LBA                     -INTO B REGISTER
00092  190 00A   003C            LAMD    TCBL            LOAD LOWER DIGIT OF TIMER/EVENT COUNTER INTO A
00093  150 046   003E            JMPL    PLSCT2
00094  1A7 009   0040  PLSCT1    LMID    $7.TMB          SET EVENT INPUT
00095  1A0 00A   0042            LMID    $0.TLRL         CLEAR TIMER LOAD REGISTER
00096  1A0 00B   0044            LMID    $0.TLRU
00097  010       0046  PLSCT2    RTN
00098                  ********************************************************************
00099                  *                                                                  *
00100                  *       NAME : HEX (CONVERT 2-BYTE HEXADECIMAL INTO 5-DIGIT BCD)*
00101                  *                                                                  *
00102                  ********************************************************************
00103                  *                                                                  *
00104                  *      ENTRY : MD($04D-$04A) (2-BYTE HEXADECIMAL DATA)             *
00105                  *    RETURNS : MD($046-$042) (5-DIGIT BCD DATA)                    *
00106                  *                                                                  *
00107                  ********************************************************************
00108  224       0047  HEX       LXI     XHEXD           LOAD HEXADECIMAL ADDR(X)
00109  212       0048            LYI     YHDEC           LOAD BCD DATA ADDR(Y)
00110  290       0049  HEX1      LMIIY   $0              CLEAR BCD DATA ADDR(Y)
00111  077       004A            YNEI    BADEC
00112  349       004B            BRS     HEX1
00113  20F       004C            LBI     $F              LOAD SHIFT COUNTER
00114  21A       004D  HEX2      LYI     YHHEX           LOAD HEXADECIMAL DATA ADDR(Y)
```

```
00115  090    004E  HEX3   LAM                    SHIFT HEXADECIMAL DATA 1-BIT LEFT
00116  0A1    004F         ROTL
00117  050    0050         LMAIY
00118  07E    0051         YNEI     BAHEX
00119  34E    0052         BRS      HEX3
00120  212    0053         LYI      YHDEC         LOAD BCD DATA LSD ADDR(Y)
00121  090    0054  HEX4   LAM                    BCD DATA AREA *2+CA->ACCA
00122  0A1    0055         ROTL
00123  0A6    0056         DAA                    CONVERT INTO BCD DATA ?
00124  050    0057         LMAIY                  LOAD DECIMAL DATA
00125  077    0058         YNEI     BADEC         TEST IF CONVERSION IS COMPLETED
00126  354    0059         BRS      HEX4
00127  0CF    005A         DB                     DECREMENT SHIFT COUNTER
00128  34D    005B         BRS      HEX2          LOOP UNTIL SHIFT COUNTER = $F
00129  010    005C         RTN
00130                *
00131                       END
```

SECTION 6. KEY MATRIX (8 × 4)

6.1 HARDWARE DESCRIPTION

6.1.1 Function

Performs key scan of 8 × 4 key matrix, invalidating simultaneous depression of more than 2 keys by software, and converting valid key data into ASCII characters (A - Z or 1 - 6).

6.1.2 Microcomputer Operation

The HMCS404C uses timer B to execute timer/event counter every 8 ms. Key scan is performed by an output strobe signal through port D during the interrupt routine and key scan data is fetched through port R.

6.1.3 Peripheral Devices

8 × 4 Key matrix : Keys to be depressed.

6.1.4 Circuit Diagram

Key scan control circuit is shown in Fig. 6.1.

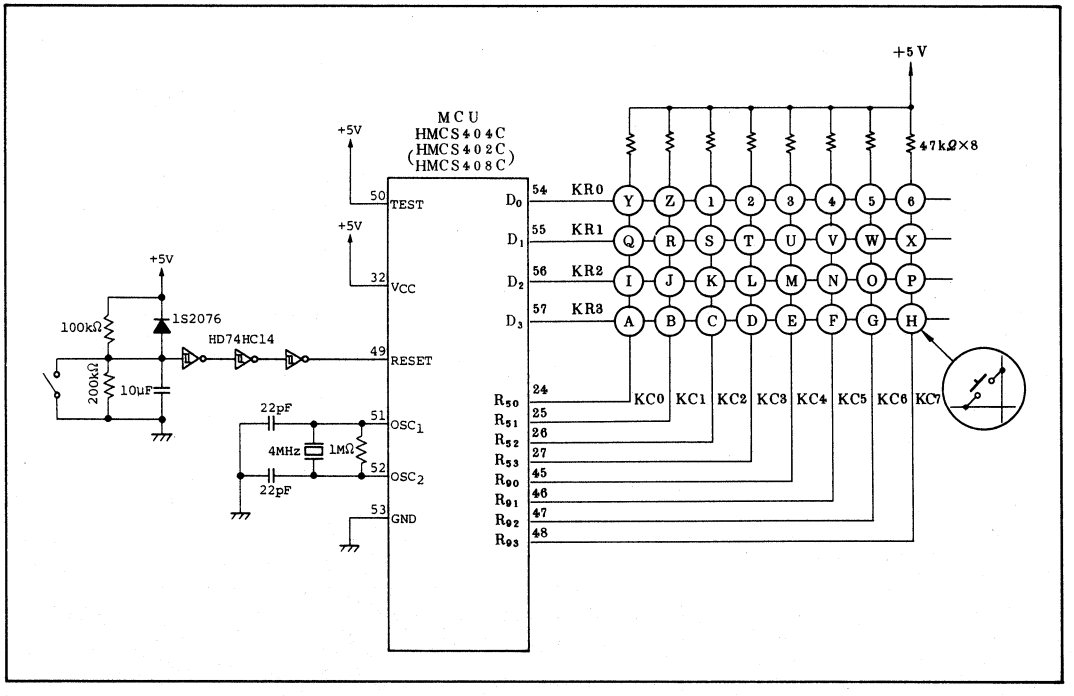


Fig. 6.1. Key Scan Control Circuit

6.1.5 Pin Functions

Pin functions at the interface between the HMCS404C and the key matrix are shown in Table 6.1.

Table 6.1. Pin Functions

Pin Name (HMCS404C)	Input/ Output	Active Level (High or Low)	Function	Pin Name (Key matrix)
D_3	Output	Low	Outputs strobe signal	KR_3
D_2	Output	Low		KR_2
D_1	Output	Low		KR_1
D_0	Output	Low		KR_0
R_{50}	Input	—	Inputs key data	KC_0
R_{51}	Input	—		KC_1
R_{52}	Input	—		KC_2
R_{53}	Input	—		KC_3
R_{90}	Input	—		KC_4
R_{91}	Input	—		KC_5
R_{92}	Input	—		KC_6
R_{93}	Input	—		KC_7

6.1.6 Hardware Operation

The timing chart for key scan is shown in Fig. 6.2.

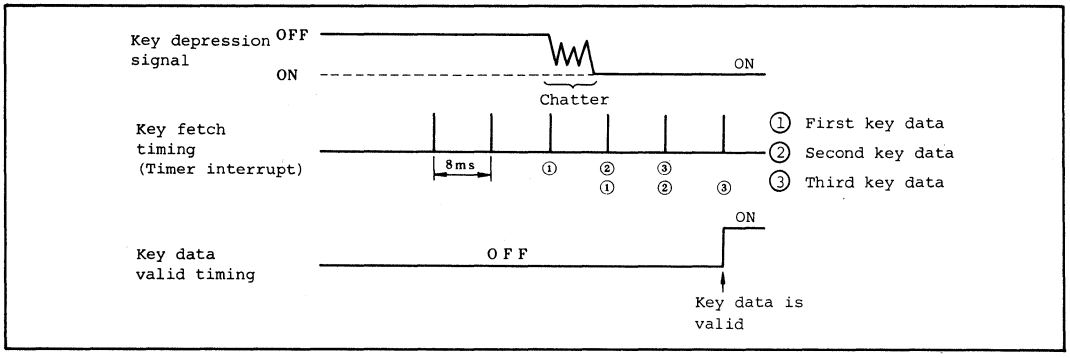


Fig. 6.2. Chatter Prevention Timing

Key depression signal is checked every 8 ms. If key data is the same 3 consecutive times, it is considered valid, and invalid otherwise.

6.2 SOFTWARE DESCRIPTION

6.2.1 Program Module Configuration

The program module configuration for key scan of 8 × 4 key matrix is shown in Fig. 6.3.

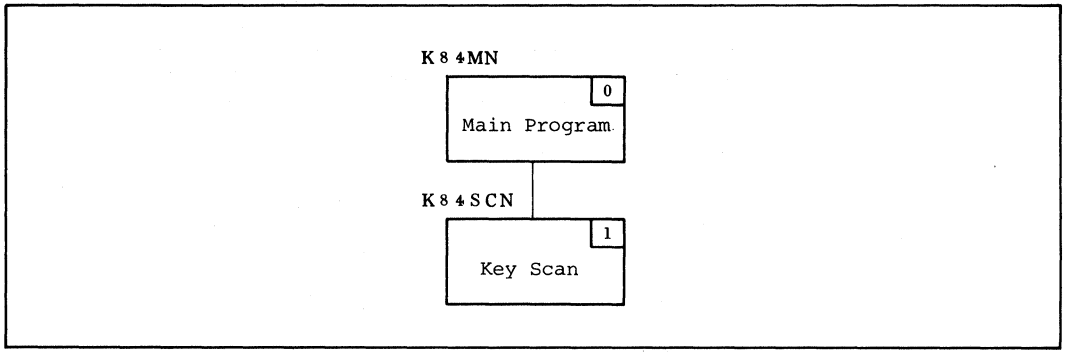


Fig. 6.3. Program Module Configuration

6.2.2 Program Module Functions

Program module functions are summarized in Table 6.2.

Table 6.2. Program Module Functions

No.	Program Module Name	Label	Function
0	Main Program	K84MN	Performs key scan of 8×4 key matrix and converts key data into ASCII
1	Key Scan	K84SCN	Performs key scan of 8×4 key matrix

6.2.3 Program Module Process Flow (Main Program)

The Flowchart in Fig. 6.4 is an example of a key scan of the 8 × 4 key matrix performed by the program module in Fig. 6.3.

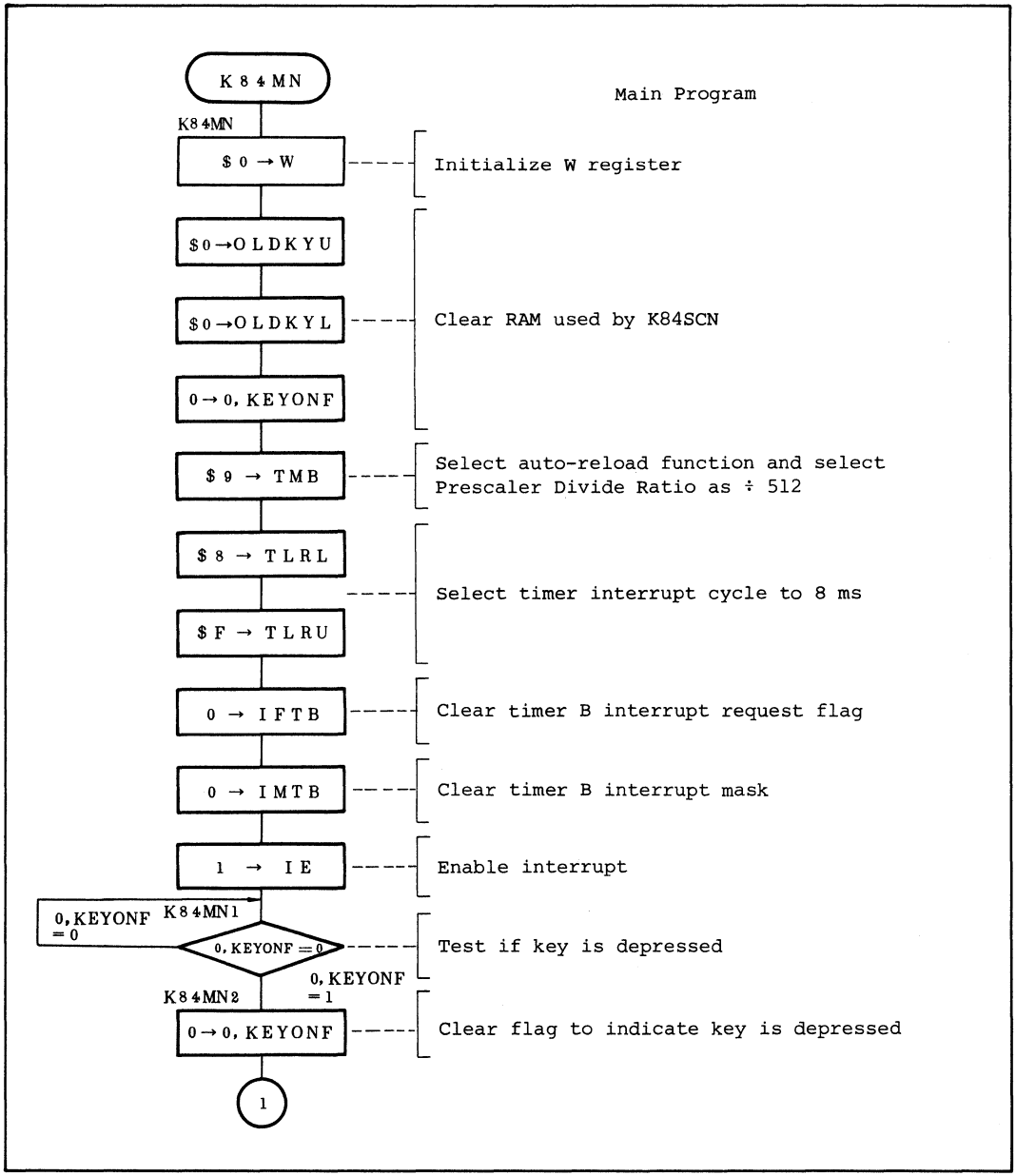


Fig. 6.4. Program Module Sample Application

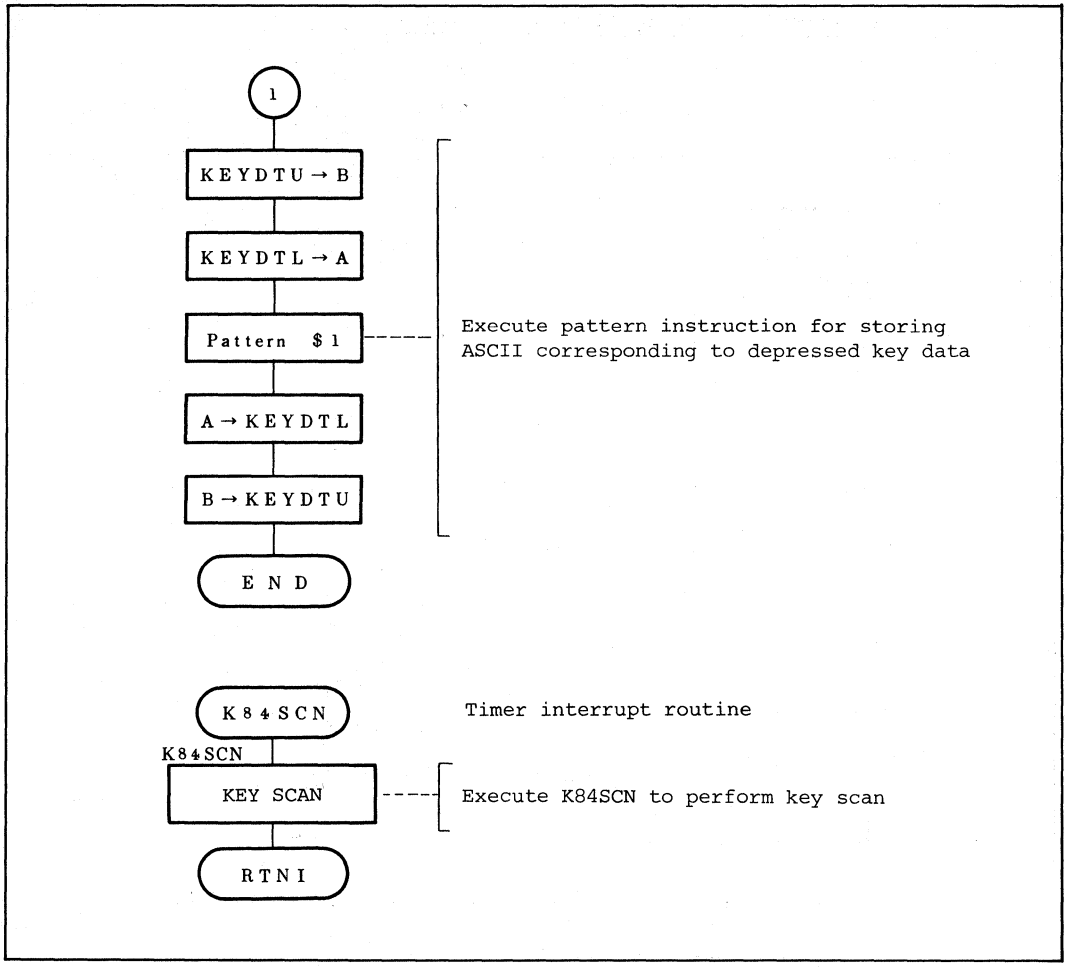


Fig. 6.4. Program Module Sample Application (Cont)

Program Module Name: KEY SCAN	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: K84SCN

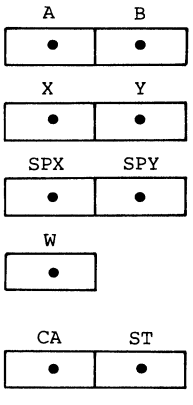
Function:

Performs key scan of 8×4 key matrix to store key data in KEYDTU(RAM) and KEYDTL(RAM).

Arguments:

Contents		1 digit = 4 bits Storage Location	No. of Digits
Entry	—	—	—
Returns	Key data	KEYDTU	1
		KEYDTL	1
	Key data valid/ invalid flag	KEYONF	1

Changes in CPU Registers and Flags:



● : Not Affected
× : Undefined
↕ : Result

Specifications:

1 word = 10 bits
ROM (Words): 141
RAM (Digits): 13
Stack (Digits): 0
No. of cycles: 1870
Reentrant: No
Relocatable: No
Interrupt OK?: No

Description:

1. Function Details

(1) Argument details

KEYDTU(RAM): Stores upper digit of key data

KEYDTL(RAM): Stores lower digit of key data

KEYONF(RAM): Contains Flag indicating whether or not key data is valid.

Specifications Notes:

The number of cycles indicate is that necessary to validate key data.

Program Module Name: KEY SCAN	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: K84SCN

Description:

Table 6.3. Flag Function

Label	bit 0	Function
KEYONF	0	Indicates key data is invalid
	1	Indicates key data is valid

(2) Example of K84SCN execution is shown in Fig. 6.5. If a key is pressed as shown in part ① of Fig. 6.5 , key data is stored in KEYDTU(RAM) and KEYDTL(RAM).

(3) K84SCN calls neither the program modules nor subroutines.

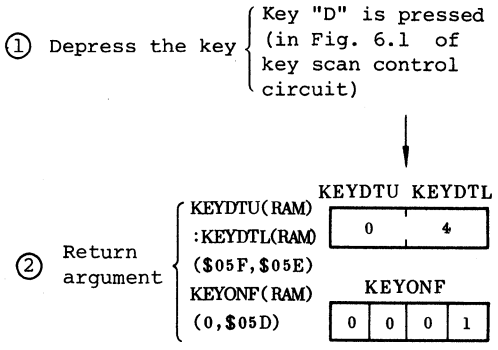


Fig. 6.5. Example of K84SCN Execution

2. User Notes

The following procedure must be performed before K84SCN execution.

(1) Initializes timer control/ status register B.

(2) Sets IE to enable timer B interrupt.

3. RAM Allocation

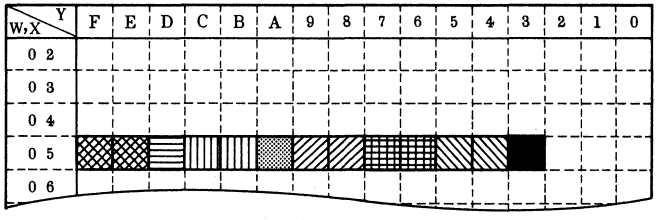
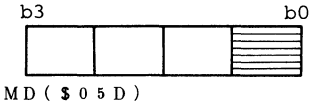
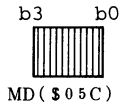
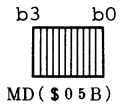
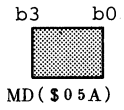
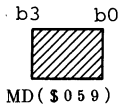
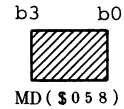
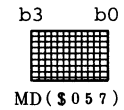
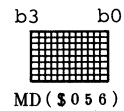


Fig. 6.6. RAM Allocation

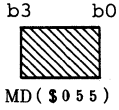
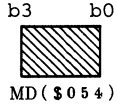
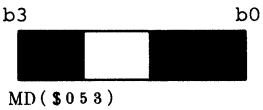
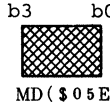
Program Module Name: KEY SCAN	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: K84SCN

Description:

Label	RAM	Description
KEYONF		Flag indicating whether or not key data is valid
KEYNMU		Upper digit for storing key number
KEYNML		Lower digit for storing key number
TOTLKY		Stores total number of pressed keys in the present key scan
NEWKYU		Upper digit for storing current key data input
NEWKYL		Lower digit for storing current key data input
OLDKYU		Upper digit for storing previous key data input
OLDKYL		Lower digit for storing previous key data input

Program Module Name: KEY SCAN	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: K84SCN

Description:

Label	RAM	Description
PRDATU		Stores depressed key data (R9)
PRDATL		Stores depressed key data (R7)
CHATFL		Bit 0, 1: Stores counter for counting number of key scan data comparison Bit 3: Stores flag for indicating whether chatter elimination has been completed
KEYDTU		Upper digit for storing defined key number by key scan
KEYDTL		Lower digit for storing defined key number by key scan

Program Module Name: KEY SCAN	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: K84SCN

Description:

4. Sample Application

```
                :
                LMID    0, OLDKYU   } ..... Clear RAM to be used
                LMID    0, OLDKYL   }
                REMD    KEYONF      }
                LMID    $9, TMB     }
                LMID    $8, TLRL    }  ..... Select timer B interrupt cycle to 8 ms
                LMID    $F, TLRU    }        and enable timer B interrupt
                REMD    IMTB        }
                SEMD    IE          }
K84MN1          TMD     KEYONF      }
                BR      K84MN1      }  ..... Test if key is depressed
                REMD    KEYONF         ..... Clear key depressed flag
                :
```

5. Basic Operation

(1) Key scan is executed every 8 ms interrupt. At the beginning of K84SCN, key data valid/invalid flag KEYONF(RAM) is checked to determine whether or not previous valid key data has been processed.

(2) Strobe signal (=low) is output through bits 0-3 of port D, and key scan data is fetched through port R.

(3) Key scan data fetched in (2) is tested whether or not it is $FF.

(a) If $FF, no key is depressed and key scan for next column is executed.

(b) If not $FF, some key is depressed and what row of depressed key is tested.

(i) Accumulator, containing key scan data, is shifted 1 bit right 8 times. CA is determined. If CA is 0 , it means a key is depressed.

(ii) Key data is numbered from 1 to 32, based on position in 8×4 key matrix. Key data is stored in KEYNMU(RAM) and KEYNML(RAM).

(iii) TOTLKY(RAM) is incremented every time a key is depressed to check for chatter. If TOTLKY(RAM) $\leqq$ 1, key data is stored in NEWKYU(RAM) and NEWKYL(RAM). If TOTLKY(RAM) > 1, key scan is completed since it indicates two keys are pressed at the same time.

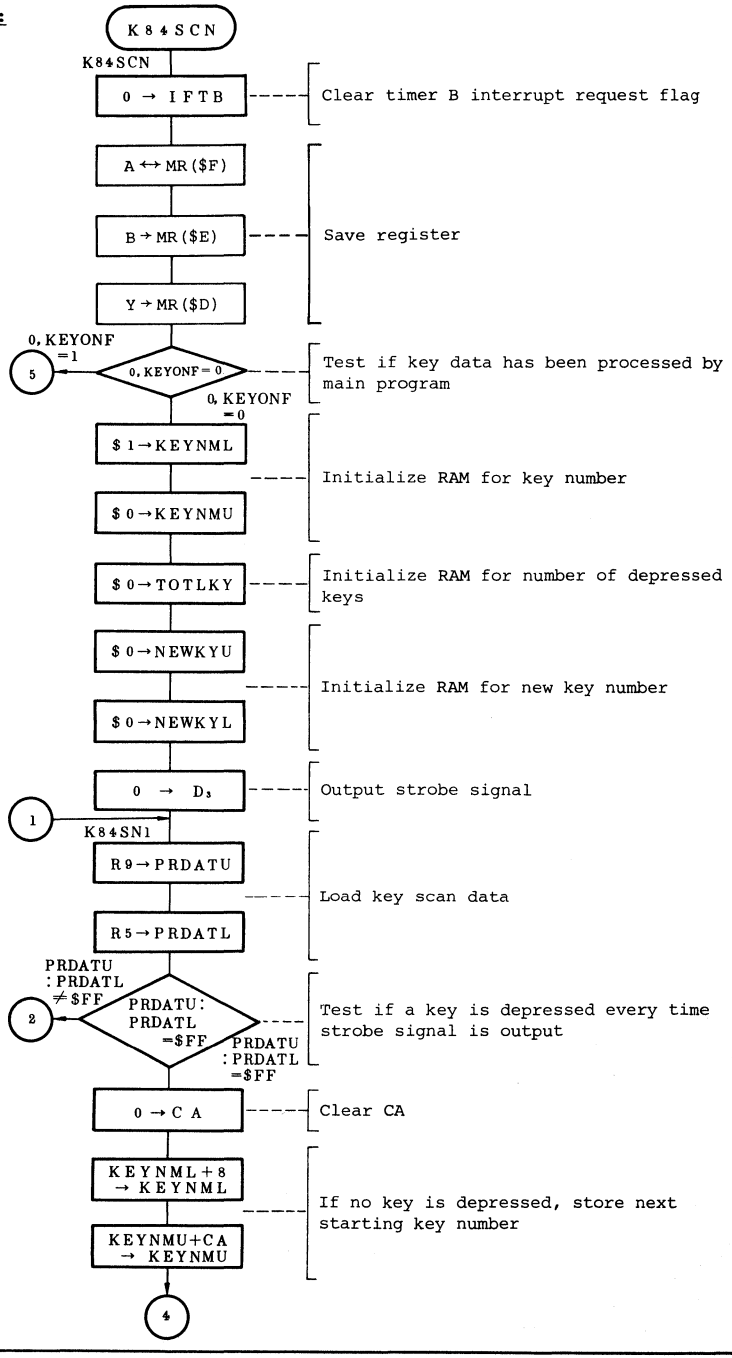
Program Module Name: KEY SCAN	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: K84SCN

Description:

(4) Key data (NEWKYU(RAM) and NEWKYL(RAM)) obtained in (3) is compared with previous key data (OLDKYU(RAM) and OLDKYL(RAM)). If they are the same, chatter counter (CHATFL(RAM)) is counted up. When chatter counter becomes 3, key data is valid. If key data is valid, MSB of CHATFL(RAM) is set to 1 to indicate that key data is valid. CHATFL(RAM) includes both a counter and a flag. CHATFL(RAM) is cleared, when (NEWKYU(RAM) and NEWKYL(RAM)) data differs from (OLDKYU(RAM) and OLDKYL(RAM)) data or no key is depressed.

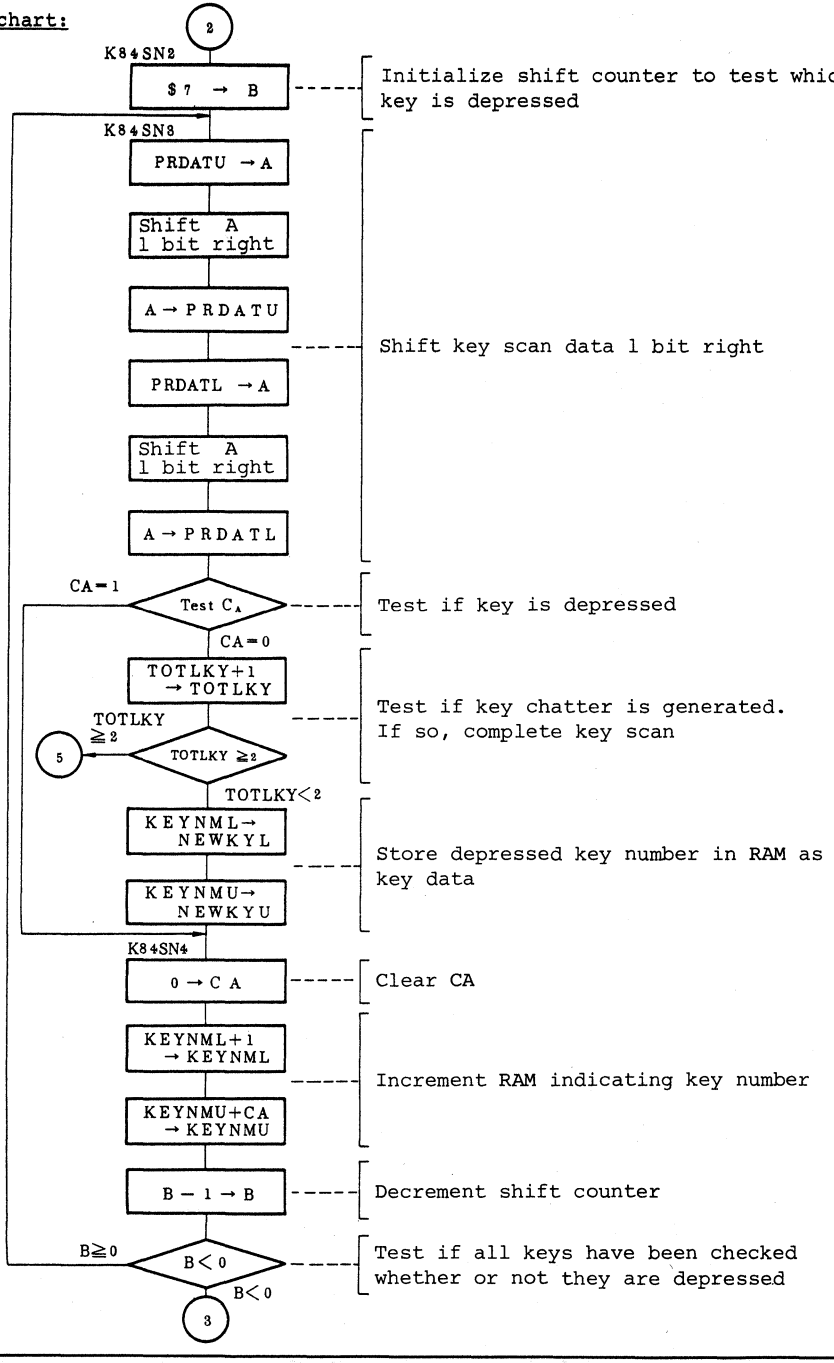
Program Module Name: KEY SCAN	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: K84SCN

Flowchart:



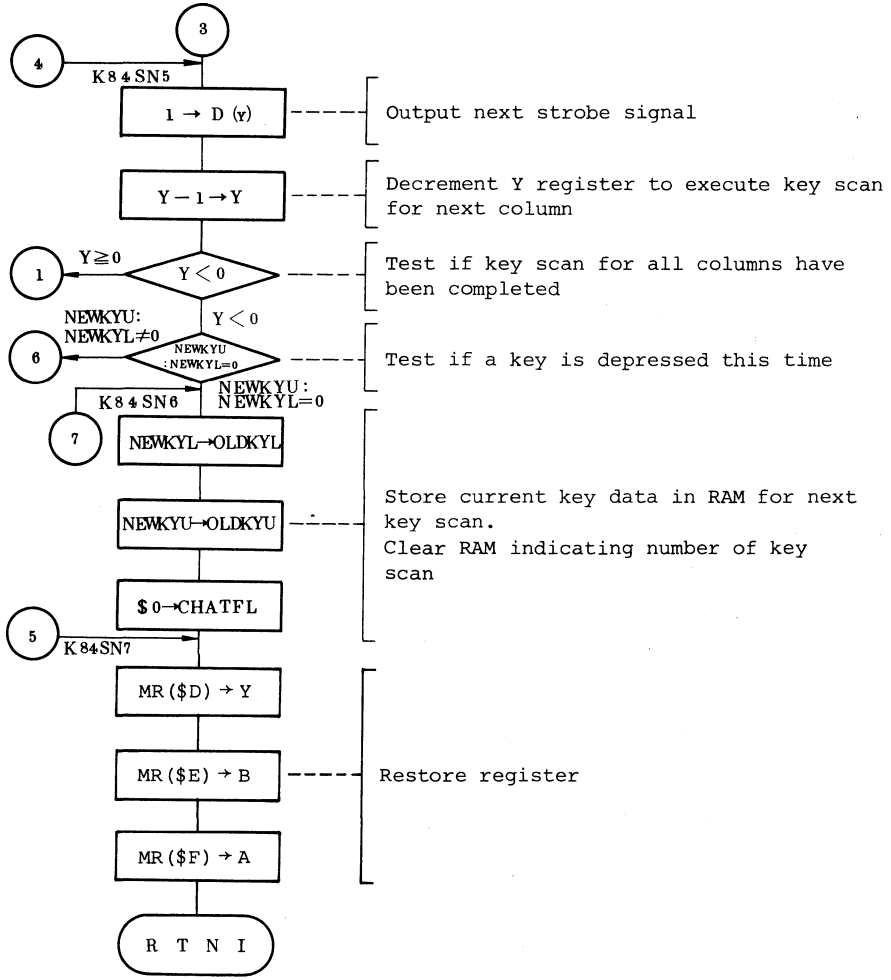
Program Module Name: KEY SCAN	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: K84SCN

Flowchart:



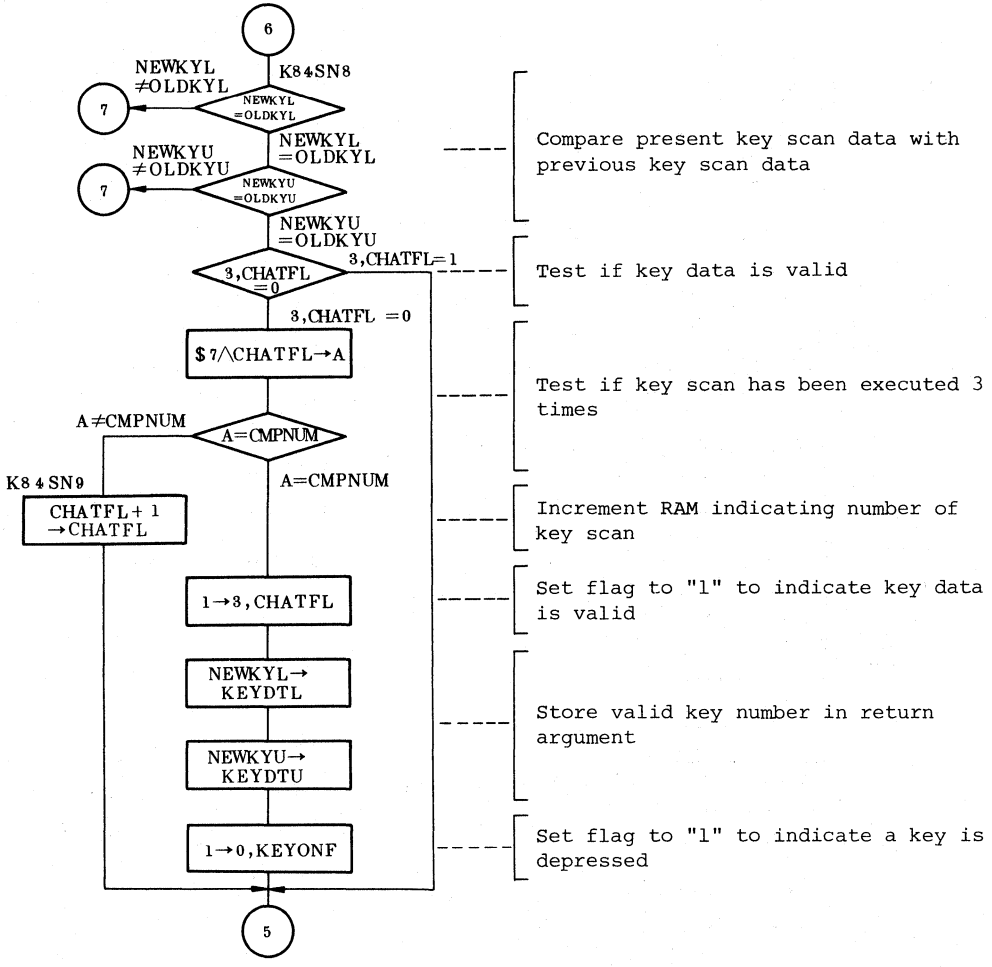
Program Module Name: KEY SCAN	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: K84SCN

Flowchart:



Program Module Name: KEY SCAN	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: K84SCN

Flowchart:



6.4 SUBROUTINES

This application example calls no subroutines.

6.5 PROGRAM LISTING

```
ST-NO   OBJECT   ADRS   SOURCE STATEMENTS

00001   010      0000           LLEN     132
00002                  *              TITLE   KEY MATRIX (8*4)
00003                  ****   RAM ALLOCATION   **********************************
00004                  *
00005                  KEYDTU   EQU      $05F          UPPER KEY DATA
00006                  KEYDTL   EQU      $05E          LOWER KEY DATA
00007                  KEYONF   EQU      0,$05D        KEY ON FLAG
00008                  KEYNMU   EQU      $05C          UPPER KEY NUMBER
00009                  KEYNML   EQU      $05B          LOWER KEY NUMBER
00010                  TOTLKY   EQU      $05A          TOTAL KEY NUMBER
00011                  NEWKYU   EQU      $059          UPPER NEW KEY DATA
00012                  NEWKYL   EQU      $058          LOWER NEW KEY DATA
00013                  OLDKYU   EQU      $057          UPPER OLD KEY DATA
00014                  OLDKYL   EQU      $056          LOWER OLD KEY DATA
00015                  PRDATU   EQU      $055          R9 PORT DATA
00016                  PRDATL   EQU      $054          R5 PORT DATA
00017                  CHATFL   EQU      $053          CHATTER CTR AND END FLAG
00018                  *
00019                  ****   SYMBOL DEFINITIONS   ****************************
00020                  *
00021                  TMB      EQU      $009          TIMER MODE REGISTER B
00022                  IFTB     EQU      0,$002        IF OF TIMER B
00023                  IMTB     EQU      1,$002        IM OF TIMER B
00024                  IE       EQU      0,$000        INTERRUPT ENABLE FLAG
00025                  TLRU     EQU      $00B          TLR UPPER
00026                  TLRL     EQU      $00A          TLR LOWER
00027                  CMPNUM   EQU      $2            CHATTER NUMBER
00028                  ************************************************************
00029                  *                                                          *
00030                  *                VECTOR ADDRESSES                          *
00031                  *                                                          *
00032                  ************************************************************
00033                  *
00034                           ORG      $0000
00035                  *
00036   150 010  0000           JMPL     K84MN         RESET
00037   150 010  0002           JMPL     K84MN         INT0
00038   150 010  0004           JMPL     K84MN         INT1
00039   150 010  0006           JMPL     K84MN         TIMER-A
00040   150 036  0008           JMPL     K84SCN        TIMER-B
00041   000      000A           NOP
00042   000      000B           NOP
00043   150 010  000C           JMPL     K84MN         SERIAL
00044                  ************************************************************
00045                  *                                                          *
00046                  *              MAIN PROGRAM : K84MN                        *
00047                  *                                                          *
00048                  ************************************************************
00049                  *
00050                           ORG      $0010
00051                  *
00052   0F0      0010  K84MN    LWI      $0            INITIALIZE W REGISTER
00053   1A0 057  0011           LMID     0,OLDKYU      INITIALIZE RAM
00054   1A0 056  0013           LMID     0,OLDKYL
00055   188 05D  0015           REMD     KEYONF
00056   1A9 009  0017           LMID     $9,TMB        INITIALIZE TMB
00057   1A8 00A  0019           LMID     $8,TLRL       INITIALIZE TLR
```

```
00058   1AF 00B  001B            LMID     $F,TLRU
00059   188 002  001D            REMD     IFTB            INITIALIZE IFTB
00060   189 002  001F            REMD     IMTB            INITIALIZE IMTB
00061   184 000  0021            SEMD     IE              ENABLE INTERRUPT
00062   18C 05D  0023  K84MN1    TMD      KEYONF          TEST IF KEY DEPRESSED
00063   327      0025            BR       K84MN2
00064   323      0026            BR       K84MN1
00065   188 05D  0027  K84MN2    REMD     KEYONF          CLEAR KEY ON FLAG
00066   190 05F  0029            LAMD     KEYDTU          LOAD KEY DATA
00067   0C8      002B            LBA
00068   190 05E  002C            LAMD     KEYDTL
00069   1B1      002E            P        $1              STORE ASCII CORRESPOND TO KEY DATA
00070   194 05E  002F            LMAD     KEYDTL
00071   048      0031            LAB
00072   194 05F  0032            LMAD     KEYDTU
00073   150 034  0034  PEND      JMPL     PEND            END OF PROGRAM
00074                   ********************************************************
00075                   *                                                      *
00076                   *          NAME : K84SCN (KEY SCAN)                    *
00077                   *                                                      *
00078                   ********************************************************
00079                   *                                                      *
00080                   *         ENTRY : NOTHING                              *
00081                   *       RETURNS : KEYDTU (UPPER KEY DATA)              *
00082                   *                 KEYDTL (LOWER KEY DATA)              *
00083                   *                 KEYONF (INDICATES KEY ON)            *
00084                   *                                                      *
00085                   ********************************************************
00086   188 002  0036  K84SCN    REMD     IFTB            CLEAR TIMER B INTERRUPT REQUEST BIT
00087   2FF      0038            XMRA     $F              SAVE REGISTER
00088   048      0039            LAB
00089   2FE      003A            XMRA     $E
00090   0AF      003B            LAY
00091   2FD      003C            XMRA     $D
00092   18C 05D  003D            TMD      KEYONF          TEST IF KEY DATA PROCESSED IM MAIN PROGRAM
00093   3A2      003F            BR       K84SN7
00094   1A1 05B  0040            LMID     $1,KEYNML       INITIALIZE KEY NUMBER (L)
00095   1A0 05C  0042            LMID     $0,KEYNMU       INITIALIZE KEY NUMBER (U)
00096   1A0 05A  0044            LMID     $0,TOTLKY       INITIALIZE TOTAL KEY NUMBER
00097   1A0 059  0046            LMID     $0,NEWKYU       INITIALIZE CURRENT KEY(U)
00098   1A0 058  0048            LMID     $0,NEWKYL       INITIALIZE CURRENT KEY(L)
00099   213      004A            LYI      $3              OUTPUT STROBE SIGNAL
00100   064      004B  K84SN1    RED
00101   259      004C            LAR      $9              LOAD R9 PORT DATA(L)
00102   194 055  004D            LMAD     PRDATU
00103   255      004F            LAR      $5              LOAD R5 PORT DATA(U)
00104   194 054  0050            LMAD     PRDATL
00105   12F 054  0052            INEMD    $F,PRDATL       TEST IF KEY IS PRESSED
00106   365      0054            BRS      K84SN2
00107   12F 055  0055            INEMD    $F,PRDATU
00108   365      0057            BRS      K84SN2
00109   0EC      0058            REC                      CLEAR CARRY
00110   238      0059            LAI      $8              STORE NEXT KEY NUMBER
00111   118 05B  005A            AMCD     KEYNML
00112   194 05B  005C            LMAD     KEYNML
00113   230      005E            LAI      $0
00114   118 05C  005F            AMCD     KEYNMU
```

```
00115   194 05C  0061                LMAD    KEYNMU
00116   150 08F  0063                JMPL    K84SN5
00117   207      0065    K84SN2      LBI     $7          INITIALIZE SHIFT COUNTER
00118   190 055  0066    K84SN3      LAMD    PRDATU      TEST IF KEY IS PRESSED
00119   0A0      0068                RDTR
00120   194 055  0069                LMAD    PRDATU
00121   190 054  006B                LAMD    PRDATL
00122   0A0      006D                RDTR
00123   194 054  006E                LMAD    PRDATL
00124   06F      0070                TC                  TEST CARRY
00125   382      0071                BRS     K84SN4
00126   231      0072                LAI     $1          INCREMENT TOTAL KEY NUMBER
00127   108 05A  0073                AMD     TOTLKY
00128   194 05A  0075                LMAD    TOTLKY
00129   132 05A  0077                ILEMD   $2.TOTLKY   CHECK KEY CHATTER GENERATION
00130   3A2      0079                BRS     K84SN7
00131   190 05B  007A                LAMD    KEYNML      STORE KEY DATA IN NEWKEY (L)
00132   194 058  007C                LMAD    NEWKYL
00133   190 05C  007E                LAMD    KEYNMU      STORE KEY DATA IN NEWKEY (U)
00134   194 059  0080                LMAD    NEWKYU
00135   0EC      0082    K84SN4      REC                 CLEAR CARRY
00136   231      0083                LAI     $1          INCREMENT KEY NUMBER
00137   118 05B  0084                AMCD    KEYNML
00138   194 05B  0086                LMAD    KEYNML
00139   230      0088                LAI     $0
00140   118 05C  0089                AMCD    KEYNMU
00141   194 05C  008B                LMAD    KEYNMU
00142   0CF      008D                DB                  DECREMENT SHIFT COUNTER
00143   366      008E                BRS     K84SN3      TEST IF 8 BITS SHIFTED
00144   0E4      008F    K84SN5      SED                 SET DPORT
00145   0DF      0090                DY                  OUTPUT NEXT STROBE SIGNAL
00146   34B      0091                BRS     K84SN1      TEST IF SCAN ALL COMPLETED
00147   120 059  0092                INEMD   0.NEWKYU
00148   3A8      0094                BRS     K84SN8      TEST IF NEW KEY IS PRESSED
00149   120 058  0095                INEMD   0.NEWKYL
00150   3A8      0097                BRS     K84SN8
00151   190 058  0098    K84SN6      LAMD    NEWKYL      STORE NEWKEY IN OLDKEY (L)
00152   194 056  009A                LMAD    OLDKYL
00153   190 059  009C                LAMD    NEWKYU      STORE NEWKEY IN OLDKEY (U)
00154   194 057  009E                LMAD    OLDKYU
00155   1A0 053  00A0                LMID    $0.CHATFL   CLEAR CHATTER COUNTER
00156   27D      00A2    K84SN7      LAMR    $D          RESTORE REGISTER
00157   0D8      00A3                LYA
00158   27E      00A4                LAMR    $E
00159   0C8      00A5                LBA
00160   27F      00A6                LAMR    $F
00161   011      00A7                RTNI
00162   190 058  00A8    K84SN8      LAMD    NEWKYL
00163   104 056  00AA                ANEMD   OLDKYL
00164   398      00AC                BRS     K84SN6      NEWKEY = OLDKEY ? (L)
00165   190 059  00AD                LAMD    NEWKYU
00166   104 057  00AF                ANEMD   OLDKYU
00167   398      00B1                BRS     K84SN6      NEWKEY = OLDKEY ? (U)
00168   18F 053  00B2                TMD     3.CHATFL    TEST IF KEY DATA IS VALID
00169   3A2      00B4                BRS     K84SN7
00170   237      00B5                LAI     $7          TEST IF KEY SCAN EXECUTED 3 TIMES
00171   19C 053  00B6                ANMD    CHATFL
```

```
00172   2B2       00B8            ALEI      CMPNUM
00173   3C8       00B9            BRS       K84SN9
00174   187 053   00BA            SEMD      3,CHATFL        SET CHATTER FLAG
00175   190 058   00BC            LAMD      NEWKYL          STORE NEWKEY IN KEYDATA (L)
00176   194 05E   00BE            LMAD      KEYDTL
00177   190 059   00C0            LAMD      NEWKYU          STORE NEWKEY IN KEYDATA (U)
00178   194 05F   00C2            LMAD      KEYDTU
00179   184 05D   00C4            SEMD      KEYONF          SET KEY DATA VALID FLAG
00180   150 0A2   00C6            JMPL      K84SN7
00181   231       00C8  K84SN9    LAI       $1              INCREMENT CHATTER COUNTER
00182   108 053   00C9            AMD       CHATFL
00183   194 053   00CB            LMAD      CHATFL
00184   150 0A2   00CD            JMPL      K84SN7
00185                   ************************************************************
00186                   *                                                          *
00187                   *       DATA TABLE                                         *
00188                   *                                                          *
00189                   ************************************************************
00190                   *
00191                             ORG       $0101
00192                   *
00193   141       0101            DC        $141
00194   142       0102            DC        $142
00195   143       0103            DC        $143
00196   144       0104            DC        $144
00197   145       0105            DC        $145
00198   146       0106            DC        $146
00199   147       0107            DC        $147
00200   148       0108            DC        $148
00201   149       0109            DC        $149
00202   14A       010A            DC        $14A
00203   14B       010B            DC        $14B
00204   14C       010C            DC        $14C
00205   14D       010D            DC        $14D
00206   14E       010E            DC        $14E
00207   14F       010F            DC        $14F
00208   150       0110            DC        $150
00209   151       0111            DC        $151
00210   152       0112            DC        $152
00211   153       0113            DC        $153
00212   154       0114            DC        $154
00213   155       0115            DC        $155
00214   156       0116            DC        $156
00215   157       0117            DC        $157
00216   158       0118            DC        $158
00217   159       0119            DC        $159
00218   15A       011A            DC        $15A
00219   131       011B            DC        $131
00220   132       011C            DC        $132
00221   133       011D            DC        $133
00222   134       011E            DC        $134
00223   135       011F            DC        $135
00224   136       0120            DC        $136
00225                   *
00226                             END
```

7.1 HARDWARE DESCRIPTION

7.1.1 Function

Displays "76543210" on fluorescent display tube by control signals from the HMCS404C.

7.1.2 Microcomputer Operation

HMCS404C utilizes a timer/event counter every 7 ms with timer B to output segment data through port R and digit data through port D.

7.1.3 Peripheral Devices

Fluorescent Display is driven dynamically at a frame frequency of 100 Hz and duty rate of 1/8.

7.1.4 Circuit Diagram

8-segment × 8-digit fluorescent display control circuit is shown in Fig. 7.1.

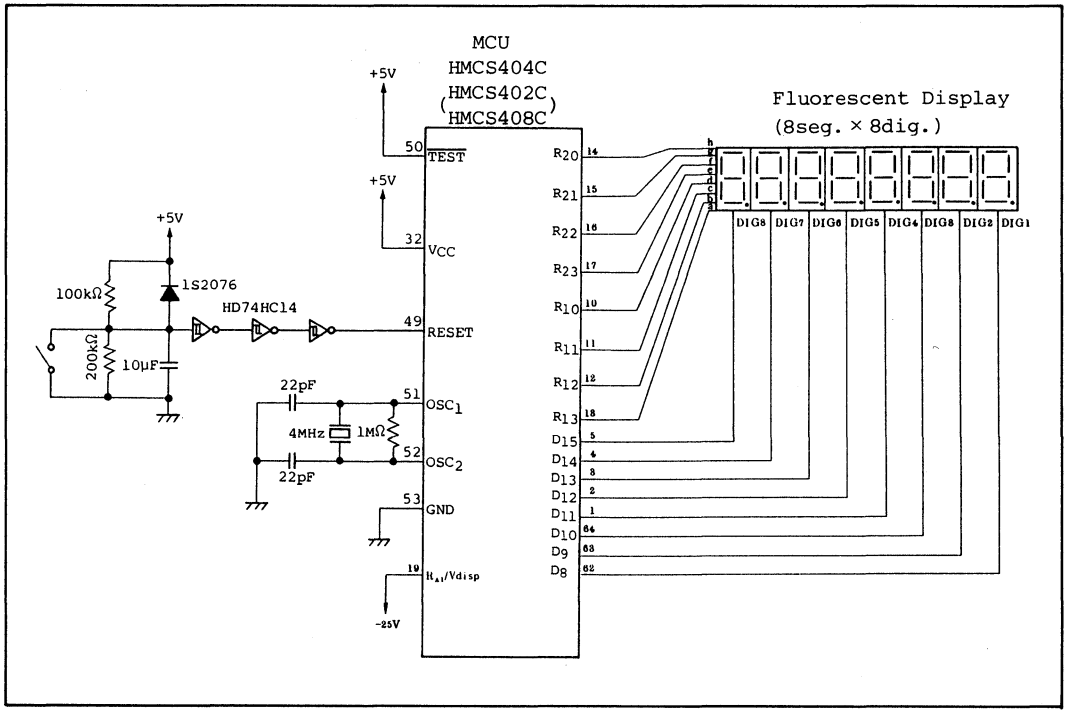


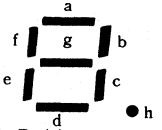
Fig. 7.1. Control for 8-segment × 8-digit Fluorescent Display Tube

7.1.5 Pin Functions

Pin functions at the interface between the HMCS404C and fluorescent display tube are shown in Table 7.1.

Table 7.1. Pin Functions

Pin Name (HMCS404C)	Input/ Output	Active Level (High or Low)	Function	Pin Name (Fluorescent display)
D_8	Output	High	Outputs digit data to fluorescent display tube	DIG1
D_9	Output	High		DIG2
D_{10}	Output	High		DIG3
D_{11}	Output	High		DIG4
D_{12}	Output	High		DIG5
D_{13}	Output	High		DIG6
D_{14}	Output	High		DIG7
D_{15}	Output	High		DIG8
R_{10}	Output	High	Outputs segment data to fluorescent display tube. "a-h" in Pin Name (Fluorescent display) correspond to segment pattern below	d
R_{11}	Output	High		c
R_{12}	Output	High		b
R_{13}	Output	High		a
R_{20}	Output	High		h
R_{21}	Output	High		g
R_{22}	Output	High		f
R_{23}	Output	High		e



Segment Pattern

7.1.6 Hardware Operation

Fluorescent display tube timing at frame frequency 100 Hz, duty 1/8 is shown in Fig. 7.2.

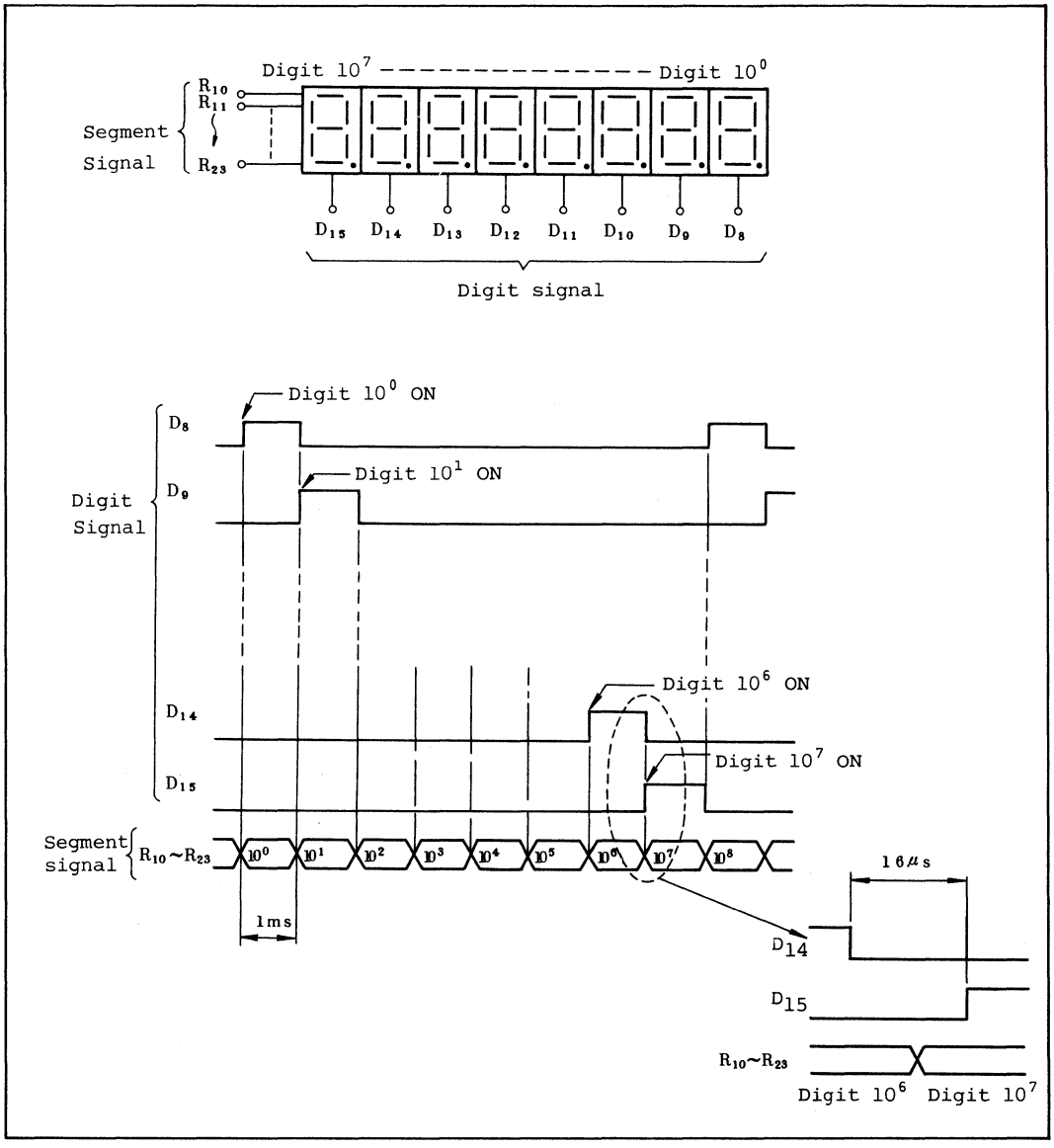


Fig. 7.2. Timing of Segment Signal and Digit Signal

7.2 SOFTWARE DESCRIPTION

7.2.1 Program Module Configuration

Program module configuration for display on 8-segment×8-digit fluorescent display tube is shown in Fig. 7.3.

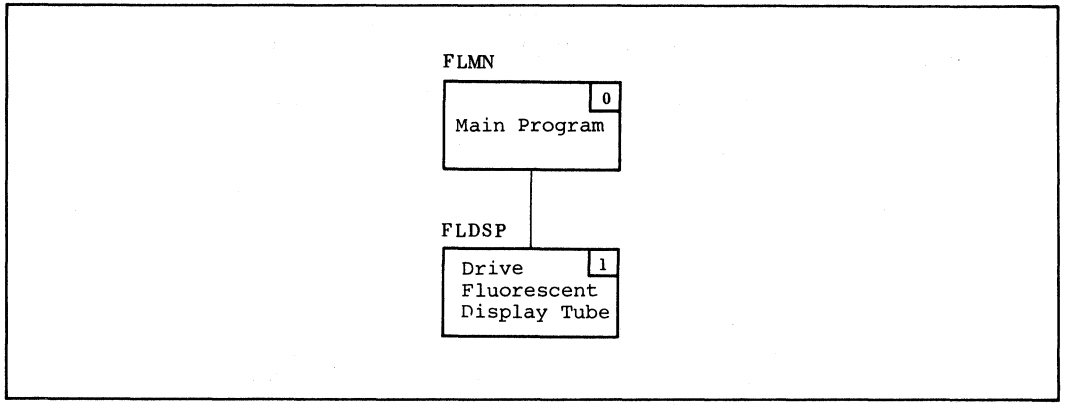


Fig. 7.3. Program Module Configuration

7.2.2 Program Module Functions

Program module functions are summarized in Table 7.2.

Table 7.2. Program Module Functions

No.	Program Module Name	Label	Function
0	Main Program	FLMN	Demonstrates display on fluorescent display tube
1	Drive Fluorescent Display Tube	FLDSP	Displays digits on fluorescent display tube using dynamic drive

7.2.3 Program Module Process Flow (Main Program)

The flowchart in Fig. 7.4. is an example of the 8-segment × 8-digit. Fluorescent display tube display performed by the program modules in Fig. 7.3. display example is shown in Fig. 7.5.

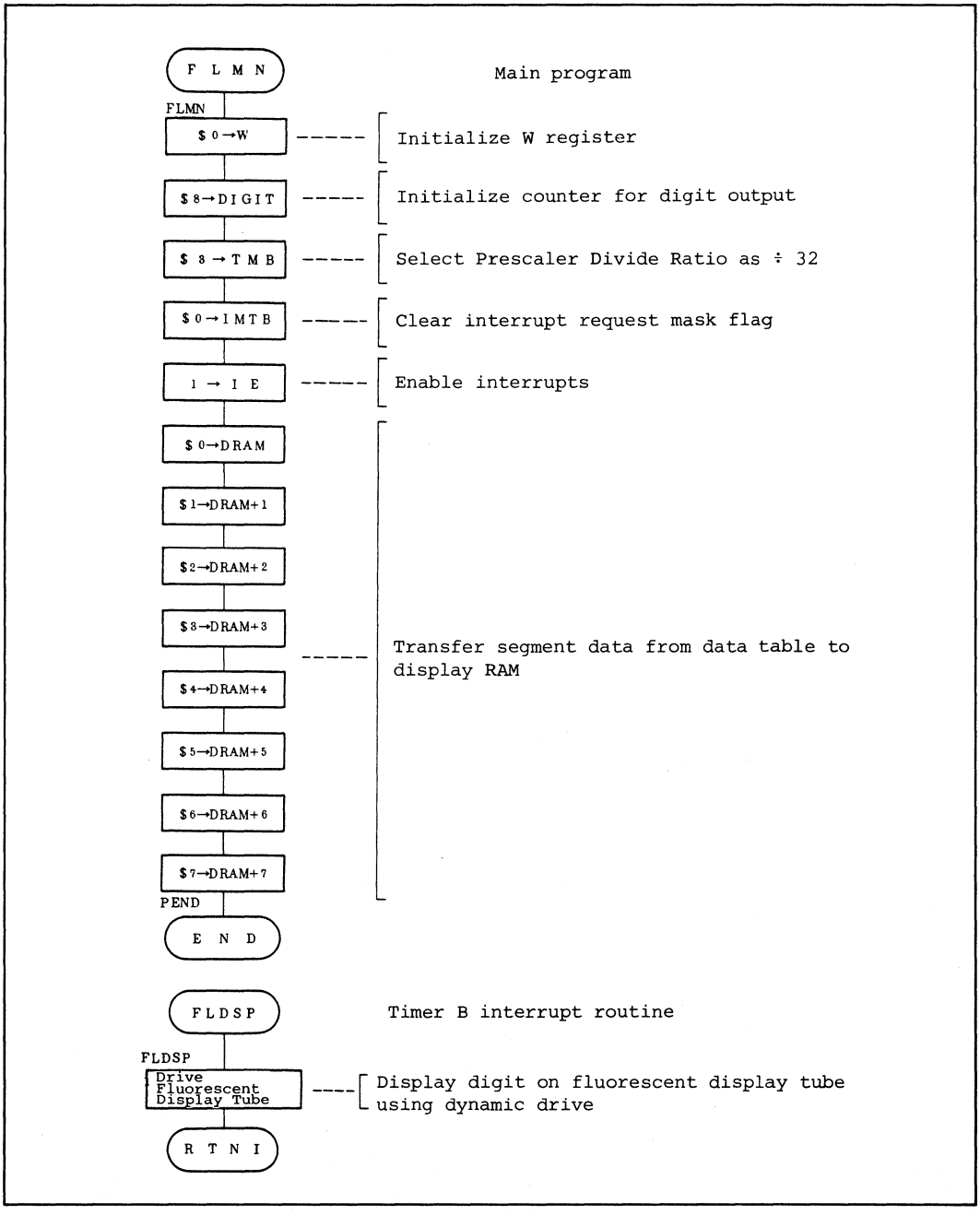


Fig. 7.4. Program Module Flowchart

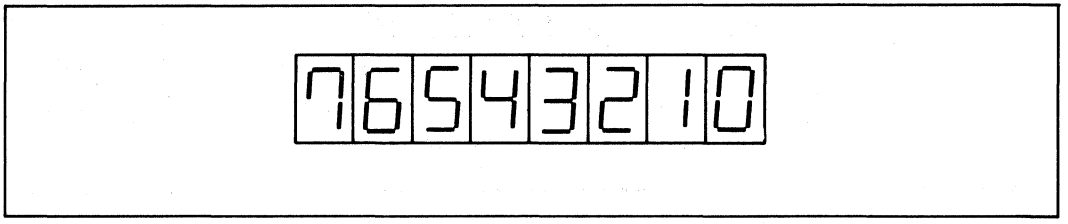


Fig. 7.5. Example of 8-segment × 8-digit Fluorescent Display Tube Display

Program Module Name:
Drive Fluorescent Display Tube

MCU: HMCS402C/
HMCS404C/HMCS408C

Label: FLDSP

Function: Drives dynamically fluorescent display tube consisting of 8-segment × 8-digit.

Arguments:

	Contents	1 digit = 4 bits Storage Location	No. of Digits
Entry	Display data	DRAM (RAM)	8
Re-turns			

Changes in CPU Registers and Flags:

A	B
•	•

X	Y
•	•

SPX	SPY
•	•

W
•

CA	ST
•	•

• : Not Affected
× : Undefined
↕ : Result

Specifications:

1 word = 10 bits

ROM (Words): 38
RAM (Digits): 9
Stack (Digits): 0
No. of cycles: 41
Reentrant: No
Relocatable: No
Interrupt OK?: No

Description:

1. Function Details

(1) Argument details
DRAM(RAM): Holds display data for each digit.

(2) Fig. 7.6 shows an example of program module FLDSP execution. If entry argument is held as shown in part ① of Fig. 7.6 fluorescent displays as shown in part ② of Fig. 7.6.

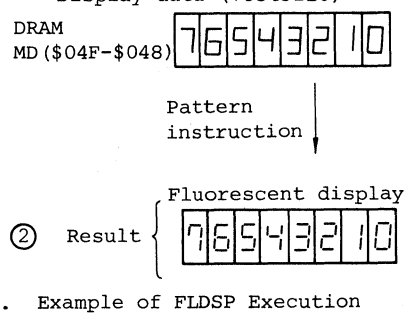


Fig. 7.6. Example of FLDSP Execution

Specifications Notes:

Program Module Name: Drive Fluorescent Display Tube	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: FLDSP

Description:

Table 7.3. Segment Data and Corresponding Display

Segment data	Display	Segment data	Display
$3F	0	$66	4
$06	1	$6D	5
$5B	2	$7D	6
$4F	3	$07	7

(3) Table 7.3 shows relation between segment data by pattern instruction and actual display.

(4) FLDSP calls neither program modules nor subroutines.

2. User Notes

(1) Data shown in Table 7.3 is allocated as a data table.

(2) Timer B is initialized before program execution.

(3) IFTB is cleared to enable timer 2 interrupt.

(4) Ports R_1 and R_2 are used to output display data through use of the P instruction in module FLDSP.

(5) High voltage ports are used to drive the fluorescent display tube.

3. RAM Allocation

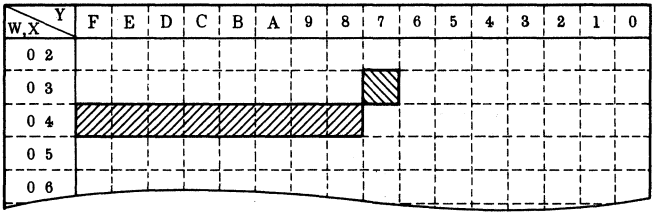


Fig. 7.7. RAM Allocation

Program Module Name: Drive Fluorescent Display Tube	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: FLDSP

Description:

Label	RAM	Description
DRAM	b31 … b0 MD($04F~$048)	Stores display data for pattern
DIGIT	b3 b0 MD($087)	Stores Y register pointer to Port D and counter indicating which digit is being displayed

4. Sample Application

```
        :
LMID    $8, DIGIT ..... Initialize RAM indicating Port D pointer
                        and digit output counter

LMID    $3, TMB   ..... Select timer mode register
REMD    IMTB      ..... Clear interrupt request mask
SEMD    IE        ..... Enable interrupt

Store display data in display RAM

        :
ORG     $1F0
DC      $23F
DC      $206
DC      $25B
DC      $24F
DC      $266          ---- Segment data
DC      $26D
DC      $27D
DC      $207
```

Program Module Name:	MCU:	Label:
Drive Fluorescent Display Tube	HMCS402C/ HMCS404C/HMCS408C	FLDSP

Description:

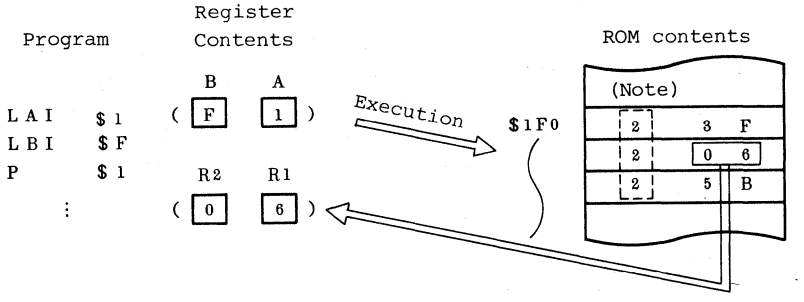
5. Basic Operation

(1) Previous digit display is turned off and next digit is displayed at every timer interrupt.

(2) DRAM(RAM) is used as counter for display digits and Y register pointer to Port D.

(3) Increments Y register for pointer to Port D and counter for display digit.

(4) Data stored at the address indicated by Accumulator, B register and operand of the pattern instruction is transferred to Port R_1 register and Port R_2 register, using the table look-up function of the pattern generation instruction.



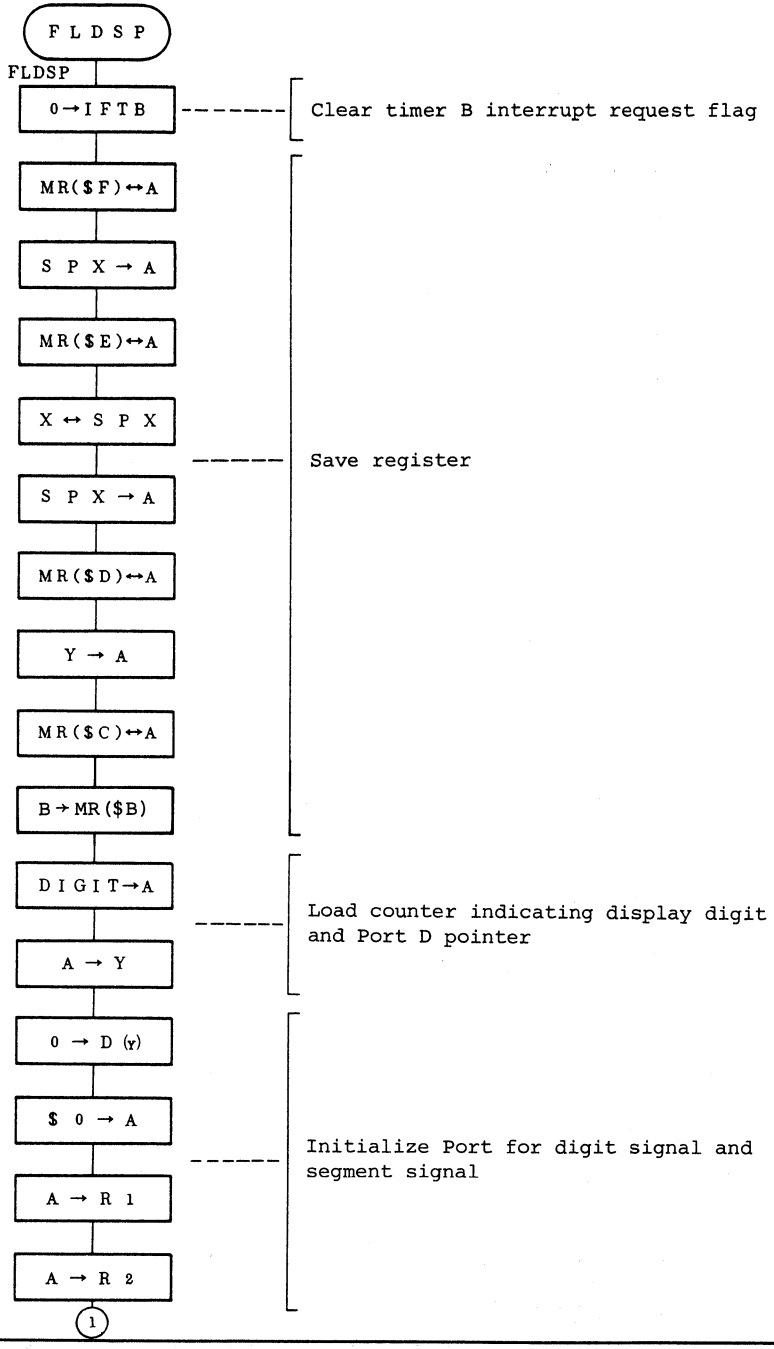
After executing the P instruction in the above program sequence, \$06 is contained in Port R_1 register and Port R_2 register. (\$06 is stored in lower 8 bits of word located at \$1F1.)

Note: If dotted area (bit 8) is \$2 as shown above, ROM data is transferred to Port R_1 register and Port R_2 register after executing the P instruction.

(5) Lower 8 bits of word stores segment data.

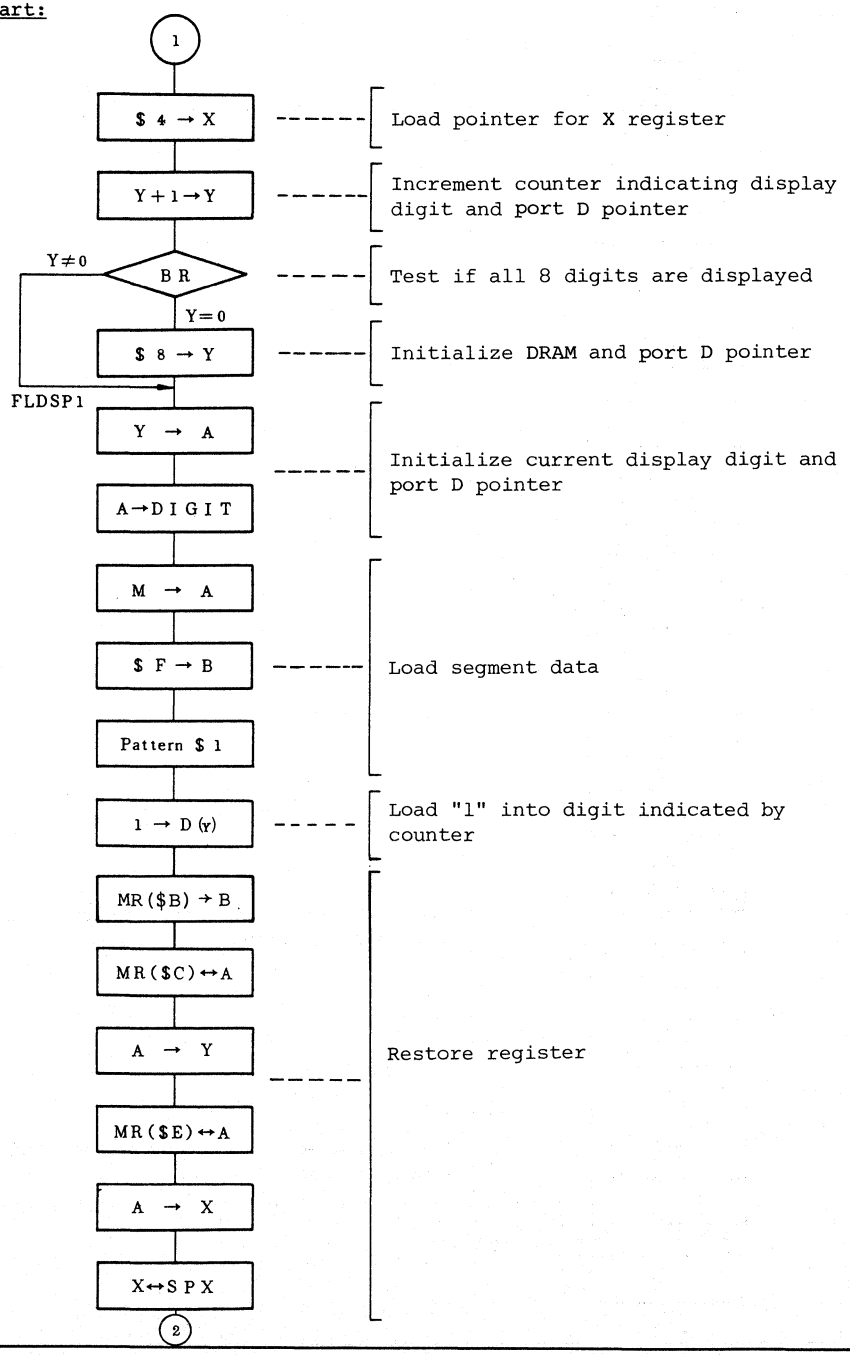
Program Module Name: Drive Fluorescent Display Tube	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: FLDSP

Flowchart:



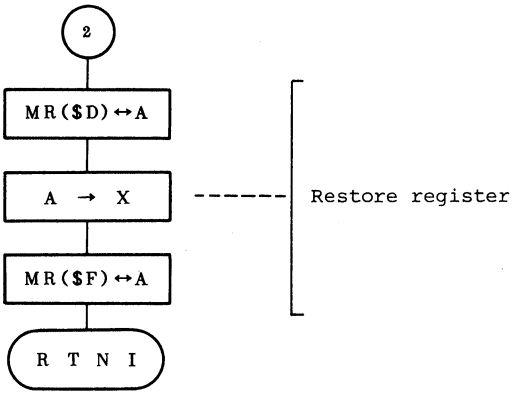
Program Module Name: Drive Fluorescent Display Tube	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: FLDSP

Flowchart:



Program Module Name:	MCU:	Label:
Drive Fluorescent Display Tube	HMCS402C/ HMCS404C/HMCS408C	FLDSP

Flowchart:



7.4 SUBROUTINE DESCRIPTION

This application example calls no subroutines.

7.5 PROGRAM LISTING

```
ST-NO   OBJECT   ADRS   SOURCE STATEMENTS

00001   136      0000           LLEN     132
00002                           TITLE    FLUORESCENT DISPLAY CONTROL
00003                  *
00004                  ****   RAM ALLOCATION   ********************************
00005                  *
00006                  DIGIT    EQU      $037          DIGIT DATA COUNTER
00007                  DRAM     EQU      $048          MOVE SEGMENT DATA TO DISPLAY RAM
00008                  *
00009                  ****   SYMBOL DEFINITIONS   ****************************
00010                  *
00011                  TMB      EQU      $009          TIMER MODE REGISTER B
00012                  IFTB     EQU      0.$002        IF OF TIMER B
00013                  IMTB     EQU      1.$002        IM OF TIMER B
00014                  IE       EQU      0.$000        INTERRUPT ENABLE FLAG
00015                  ************************************************************
00016                  *                                                          *
00017                  *               VECTOR ADDRESSES                           *
00018                  *                                                          *
00019                  ************************************************************
00020                  *
00021                           ORG      $0000
00022                  *
00023   150 010  0000           JMPL     FLMN          RESET
00024   150 010  0002           JMPL     FLMN          INT0
00025   150 010  0004           JMPL     FLMN          INT1
00026   150 010  0006           JMPL     FLMN          TIMER-A
00027   150 02A  0008           JMPL     FLDSP         TIMER-B
00028                  ************************************************************
00029                  *                                                          *
00030                  *               MAIN PROGRAM : FLMN                        *
00031                  *                                                          *
00032                  ************************************************************
00033                  *
00034                           ORG      $0010
00035                  *
00036   0F0      0010  FLMN     LWI      $0            INITIALIZE W REGISTER
00037   1A8 037  0011           LMID     $8.DIGIT      INITIALIZE DIGIT DATA COUNTER
00038   1A3 009  0013           LMID     $3.TMB        SELECT PRESCALER TO 1/32
00039   189 002  0015           REMD     IMTB          CLEAR INTERRUPT REQUEST FLAG
00040   184 000  0017           SEMD     IE            ENABLE INTERRUPTS
00041   1A0 048  0019           LMID     $0.DRAM       MOVE SEGMENT DATA TO DISPLAY RAM
00042   1A1 049  001B           LMID     $1.DRAM+1
00043   1A2 04A  001D           LMID     $2.DRAM+2
00044   1A3 04B  001F           LMID     $3.DRAM+3
00045   1A4 04C  0021           LMID     $4.DRAM+4
00046   1A5 04D  0023           LMID     $5.DRAM+5
00047   1A6 04E  0025           LMID     $6.DRAM+6
00048   1A7 04F  0027           LMID     $7.DRAM+7
00049   329      0029  PEND     BR       PEND          END OF PROGRAM
00050                  ************************************************************
00051                  *                                                          *
00052                  *         NAME : FLDSP (DRIVE FLOURSCENT DISPLAY TUBE)     *
00053                  *                                                          *
00054                  ************************************************************
00055                  *                                                          *
00056                  *        ENTRY :  DRAM (DISPLAY DATA)                      *
00057                  *      RETURNS :  NOTHING                                  *
```

```
00058                    *                                                     *
00059                    *******************************************************
00060   188 002  002A    FLDSP    REMD     IFTB          CLEAR TIMER B INTERRUPT REQUEST FLAG
00061   2FF      002C             XMRA     $F            SAVE REGISTER
00062   068      002D             LASPX
00063   2FE      002E             XMRA     $E
00064   001      002F             XSPX
00065   068      0030             LASPX
00066   2FD      0031             XMRA     $D
00067   0AF      0032             LAY
00068   2FC      0033             XMRA     $C
00069   048      0034             LAB
00070   2FB      0035             XMRA     $B
00071   190 037  0036             LAMD     DIGIT         LOAD DIGIT DATA COUNTER
00072   0D8      0038             LYA
00073   064      0039             RED                    INITIALIZE PORT
00074   230      003A             LAI      $0
00075   2D1      003B             LRA      $1
00076   2D2      003C             LRA      $2
00077   224      003D             LXI      $4            LOAD POINTER FOR X REGISTER
00078   05C      003E             IY                     INCREMENT DISPLAY COUNTER & D PORT POINTER
00079   353      003F             BR       FLDSP2        TEST IF 8 DIGITS ARE DISPLAYED
00080   218      0040             LYI      $8
00081   0AF      0041    FLDSP1   LAY                    INITIALIZE CURRENT DISPLAY COUNTER & D PORT POINTER
00082   194 037  0042             LMAD     DIGIT
00083   090      0044             LAM
00084   20F      0045             LBI      $F            OUTPUT DISPLAY DATA
00085   1B1      0046             P        $1
00086   0E4      0047             SED                    LOAD 1 INTO DIGIT INDICATED BY COUNTER
00087   2FB      0048             XMRA     $B            RESTORE REGISTERS
00088   0C8      0049             LBA
00089   2FC      004A             XMRA     $C
00090   0D8      004B             LYA
00091   2FE      004C             XMRA     $E
00092   0E8      004D             LXA
00093   001      004E             XSPX
00094   2FD      004F             XMRA     $D
00095   0E8      0050             LXA
00096   2FF      0051             XMRA     $F
00097   011      0052             RTNI
00098   341      0053    FLDSP2   BRS      FLDSP1
00099                    *******************************************************
00100                    *                                                     *
00101                    *       DATA TABLE                                    *
00102                    *                                                     *
00103                    *******************************************************
00104                    *
00105                             ORG      $1F0
00106                    *
00107   23F      01F0             DC       $23F          0       SEGMENT DATA
00108   206      01F1             DC       $206          1
00109   25B      01F2             DC       $25B          2
00110   24F      01F3             DC       $24F          3
00111   266      01F4             DC       $266          4
00112   26D      01F5             DC       $26D          5
00113   27D      01F6             DC       $27D          6
00114   207      01F7             DC       $207          7
```

```
00115                   *
00116                           END
```

SECTION 8. STEPPING MOTOR CONTROL

8.1 HARDWARE DESCRIPTION

8.1.1 Function

(1) Drives stepping motor using the HMCS404C.

(2) Uses 4-phase 2 exciting stepping motor.

(3) 1 to 255 steps can be selected.

(4) Controls slue pulse rate (slow-up, operating and slow-down) when 12 steps or more are selected.

(5) Selects clockwise (normal) slue and counterclockwise (reverse) slue with stepping motor. Operates backlash when reverse slue is selected.

8.1.2 Microcomputer Operation

(1) Executes interrupt routine using 8-bit autoreload timer/event counter contained in the HMCS404C (hereinafter, timer B).

(2) Drives stepping motor by outputting pulses from port B by interrupt routine.

(3) Generates slue-up and slue-down pulse rate by changing the timer load register (hereinafter TLR) value.

8.1.3 Peripheral Devices

8.1.4 Circuit Diagram

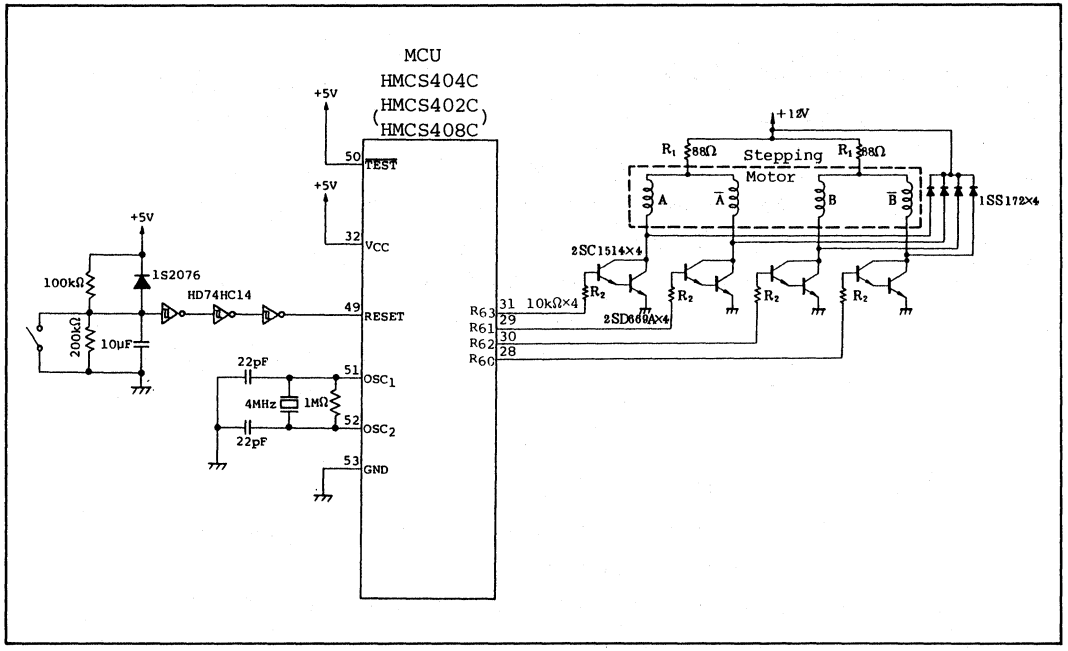


Fig. 8.1. Stepping Motor Control

8.1.5 Pin Functions

Pin functions at the interface between the HMCS404C and stepping motor are shown in Table 8.1.

Table 8.1. Pin Functions

Pin Name HMCS404C	Input/ Output	Active level (High or Low)	Function	Pin Name (Motor)
R_{60}	Output	—	Connects stepping motor	$\overline{B}$
R_{61}	Output	—		$\overline{A}$
R_{62}	Output	—		B
R_{63}	Output	—		A

8.1.6 Hardware Operation

The stepping motor supplies pulses at the HMCS404C I/O port as shown in Fig. 8.2.

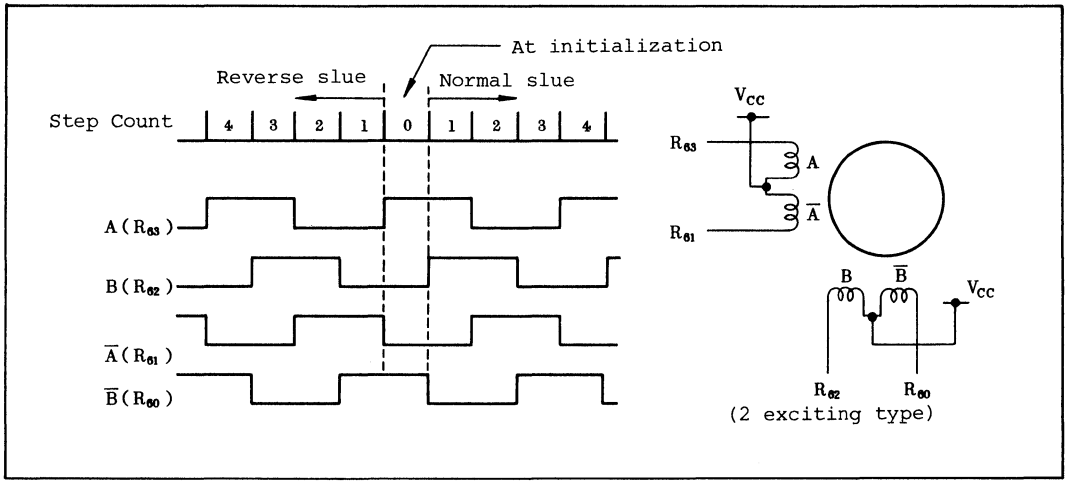


Fig. 8.2. Outline of Stepping Motor Operation

Slue-up and slue-down pulse rate are supplied to the stepping motor every four steps as shown in Fig. 8.3.

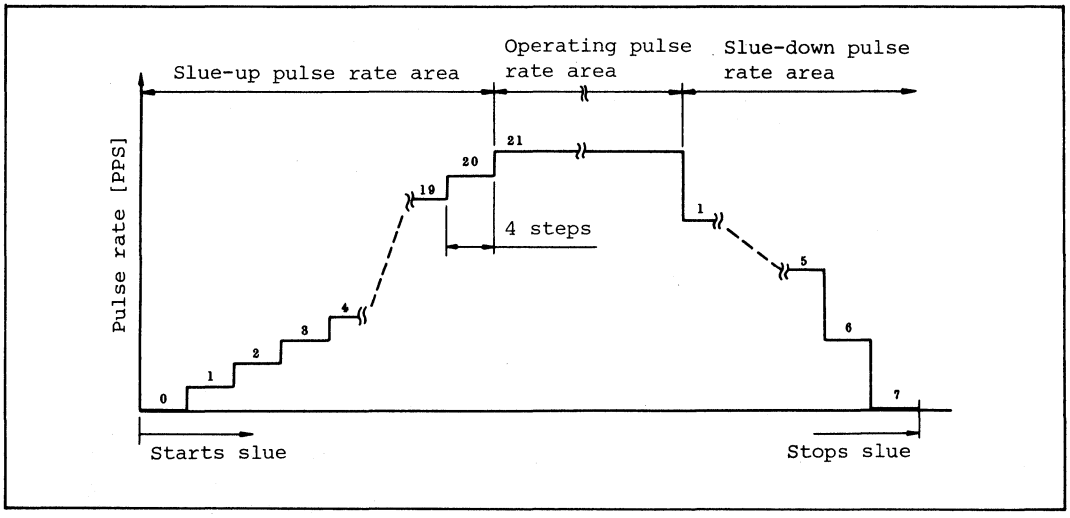


Fig. 8.3. Outputs to the Stepping Motor

(1) The pulse rate changes every 4 steps for slue-up and slue-down.

(2) In case of slue-up area to the operating pulse area, the pulse rate changes in 21 steps to gradually increase the slue speed.

(3) In case of slue-down area to stop, the pulse rate changes in 7 steps to gradually reduce the slue speed.

(4) When reverse slue is selected, an additional one-step rotation, followed by a normal one-step rotation, is executed.

8.2 SOFTWARE DESCRIPTION

8.2.1 Program Module Configuration

The program module configuration for stepping motor control is shown in Fig. 8.4.

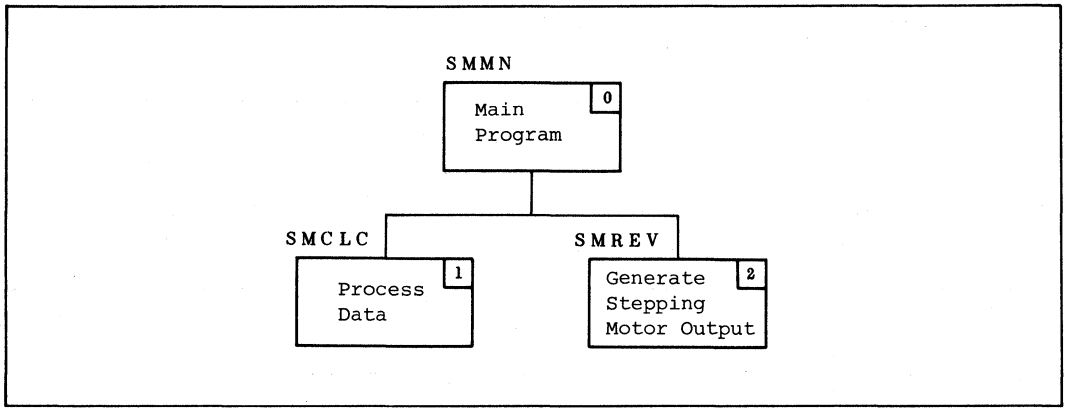


Fig. 8.4. Program Module Configuration

8.2.2 Program Module Functions

Program module functions are summarized in Table 8.2.

Table 8.2. Program Module Functions

No.	Program Module Name	Label	Function
0	Main Program	SMMN	Rotates stepping motor
1	Process Data	SMCLC	Calculates output data for slue-up, operating, slue-down and backlash by supplying total step count
2	Generate Stepping Motor Output	SMREV	Supplies pulses to the stepping motor

8.2.3 Program Module Process Flow (Main Program)

The flowchart in Fig. 8.5 is an example of the stepping motor rotation performed by the program module in Fig. 8.4. When the program module of Fig. 8.5 is executed, the stepping motor makes 201 reverse slue, then one normal slue.

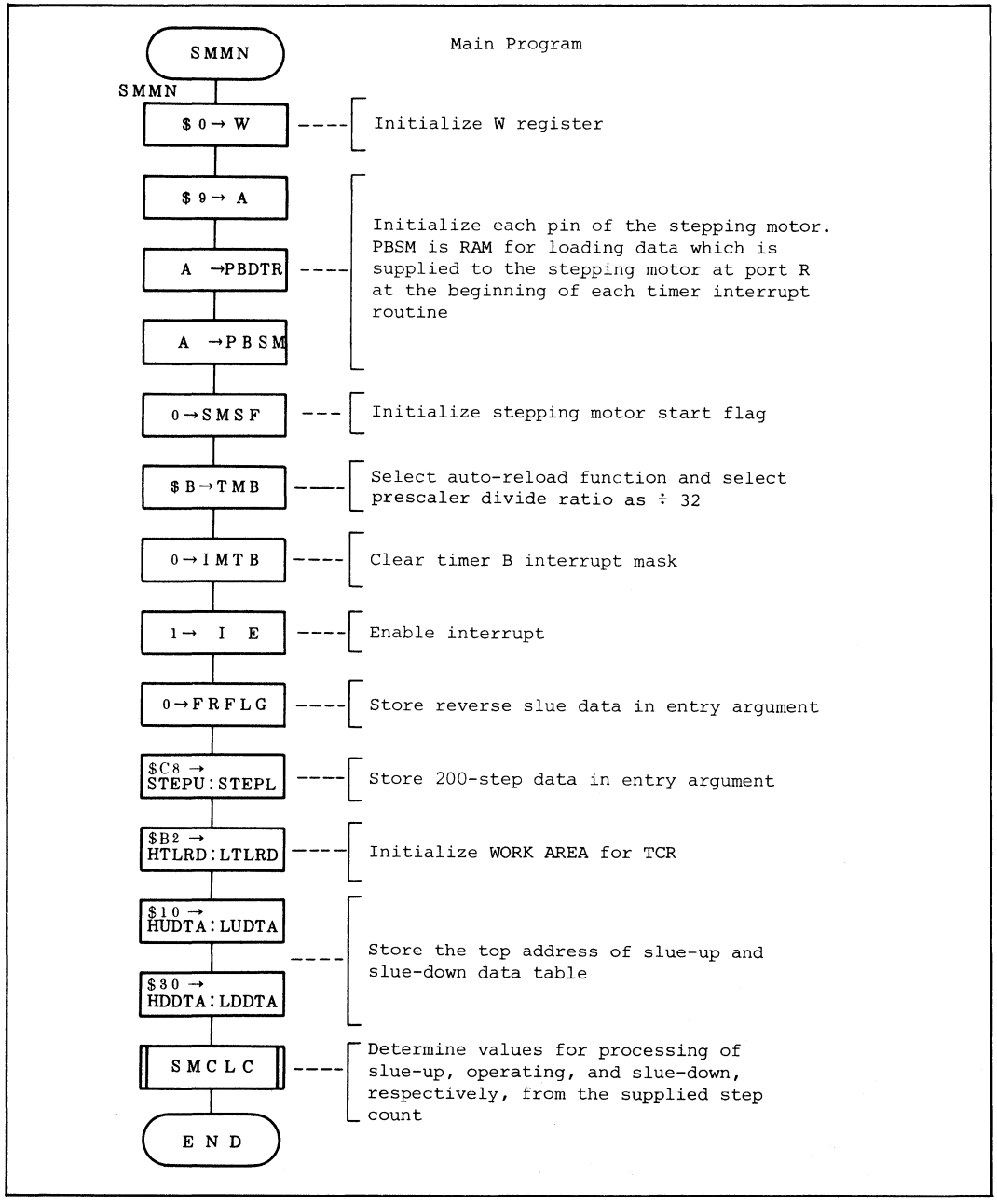


Fig. 8.5. Program Module Sample Application

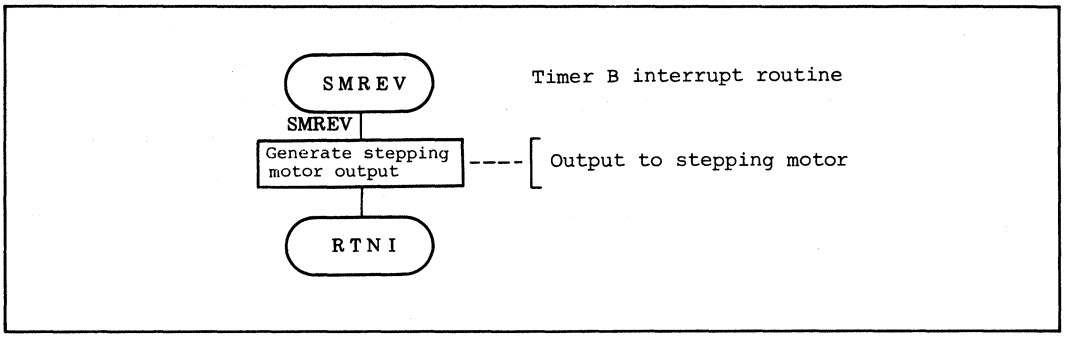


Fig. 8.5. Program Module Sample Application (Cont)

Program Module Name: Process Data	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: SMCLC

Function:

Generates data for outputting to the stepping motor by the timer routine. Sets data for slue-up, operating, and slue-down after determining the backlash requirement.

Arguments:

Contents		Storage Location (1 digit = 4 bits)	No. of Digits
Entry	Normal/ Reverse slue flag	FRFLG (RAM)	1
	Total Step Count	STEPU (RAM)	1
		STEPL (RAM)	1
Returns	Slue-up data	HSUP (RAM)	1
		LSUP (RAM)	1
	Hold data	HSHOLD (RAM)	1
		LSHOLD (RAM)	1
	Slue-down data	SDWN (RAM)	1
	Remainder of step	STEPE (RAM)	1
	Normal/ Reverse slue flag	FRFLG (RAM)	1
	Slue start flag	SMSF (RAM)	1

Changes in CPU Registers and Flags:

A	B
×	×

X	Y
×	×

SPX	SPY
×	×

W
●

CA	ST
×	×

● : Not Affected
× : Undefined
↕ : Result

Specifications:

1 word = 10 bits

ROM (Words): 250

RAM (Digits): 27

Stack (Digits): 4

No. of cycles: 1026

Reentrant: No

Relocatable: No

Interrupt OK?: No

Description:

1. Function Details

(1) Argument details

FRFLG: SMSF(RAM) : Holds flag indicating slue direction and start of stepping motor rotation. Flag functions are shown in Table 8.3.
STEPU: STEPL(RAM): Holds total step count.
HSUP: LSUP(RAM) : Contains slue-up data.
LSHOLD: HSHOLD(RAM): Contains operating data.
SDWN (RAM): Contains slue-down data.
STEPE (RAM): Contains remainder of total step count divided by 4.

Specifications Notes:

Program Module Name: Process Data	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: SMCLC

Description:

Table 8.3. Flag Functions

Label	Bit/Label SMSF	FRFLG	Function
FLG1	-	0	Rotates stepping motor clockwise (normal)
	-	1	Rotates stepping motor counterclockwise (reverse)
	0	-	Stops slue
	1	-	Starts slue

(2) Fig. 8.6 shows an example of program module SMCLC execution.
If entry arguments are held as shown in part ① of Fig. 8.6, the remainders of slue-up, operating and slue-down are contained as shown in part ② of Fig. 8.6.

(3) SMCLC calls subroutines shown in Table 8.4.

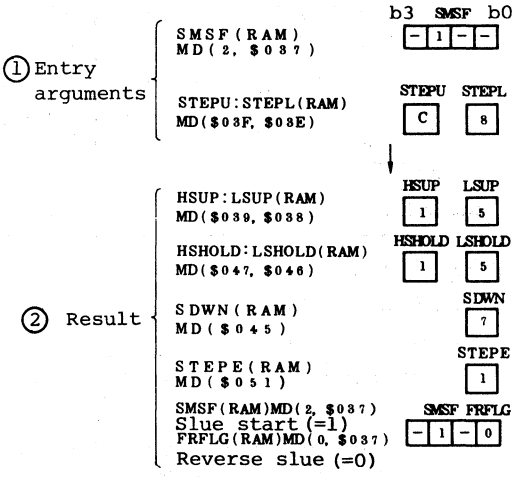


Fig. 8.6. Example of SMCLC Execution

Program Module Name: Process Data	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: SMCLC

Description:

Table 8.4. Subroutines Called in SMCLC

Program Module/ Subroutine Name	Label	Function
Shifting 8-bit Binary Data	SHR	Shifts 8-bit binary data in RAM to right
Adding 8-bit Binary Data	ADD	Performs addition of 8-bit binary data in RAM, and stores result in RAM
Subtracting 8-bit Binary Data	SUB	Performs subtraction of 8-bit binary data in RAM, and stores result in RAM
Comparing 8-bit Binary Data	CMP	Determines larger than/smaller than relationship (>, =,<) of 8-bit binary data of 2 group, and loads "$0" "$1" or "$2" into B register as a result
Calculate Normal/Reverse Data	SMFR	Tests whether the slue is normal or reverse and sets the next data to the stepping motor

2. User Notes

When the slue is reverse, the maximum number of steps is $FE. When $FF is set, an exact slue data is not set.

3. RAM Allocation

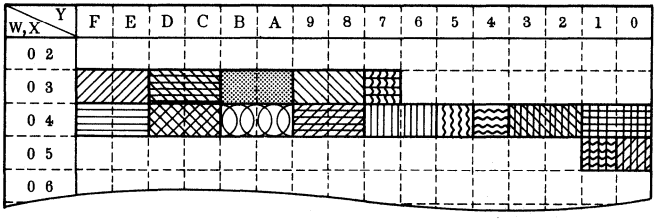
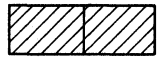
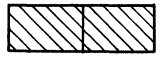
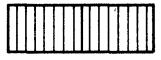
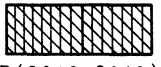


Fig. 8.7. RAM Allocation

Specifications Notes: "No. of cycles" in "Specification" represents the number of cycles required to execute data in the sample application.

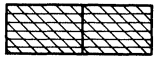
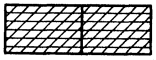
Program Module Name: Process Data	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: SMCLC

Description:

Label	RAM	Description
STEPU:STEPL	b7 b0  MD($03F, $03E)	Stores total step count
HSUP:LSUP	b7 b0  MD($039,$038)	Stores one-fourth of the step count due to slue-up
HSHOLD:LSHOLD	b7 b0  MD($047,$046)	Stores one-fourth of the step count due to operating
SDWN	b3 b0  MD($045)	Stores one-fourth of the step count due to slue-down
SDWNW	b3 b0  MD($044)	WORK AREA for SDWN
SMSF	b3 b0  MD(2,$037)	Stores flag indicating start slue for the stepping motor
STEPE	b3 b0  MD($051)	Stores remainder of a total step divided by 4
HSTEPW:LSTEPW	b7 b0  MD($043,$042)	Stores WORK AREA for STEP

Program Module Name: Process Data	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: SMCLC

Description:

Label	RAM	Description
SCNTR	b3 b0  MD($050)	Stores 4-step counter
HAUG:LAUG	b7 b0  MD($03D,$03C)	Holds 8-bit binary augend. After execution, contain addition result
HADD:LADD	b7 b0  MD($049,$048)	Holds 8-bit binary addend
HSHR:LSHR	b7 b0  MD($04F,$04E)	Holds 8-bit binary data to be shifted to right
HMIN:LMIN	b7 b0  MD($041,$040)	Holds 8-bit binary minuend. After execution, contain subtraction result
HSUB:LSUB	b7 b0  MD($04D,$04C)	Holds 8-bit binary subtrahend
HCMD:LCMD	b7 b0  MD($03B,$03A)	Holds the first 8-bit binary value
HCMT:LCMT	b7 b0  MD($04B,$04A)	Holds the second 8-bit binary value

Program Module Name: Process Data	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: SMCLC

Description:

4. Sample Application

```
WORKU   EQU     $055    }
WORKL   EQU     $054    }  ..... Reserve memory byte for total step count
          :
        SEMD    SMSF    ..... Set normal/reverse flag to normal
                              rotation
        LAMD    WORKU   }
        LMAD    STEPU   }  ..... Load the total step count into entry
        LAMD    WORKL   }        argument
        LMAD    STEPL   }

        CALL    SMCLC   ..... Call SMCLC and process the data
          :
```

Program Module Name: Process Data	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: SMCLC

Description:

5. Basic Operation

(1) Tests whether slue is normal or reverse. If it is reverse, 1 is added to the total step count for backlush processing.

(2) Slue-up and slue-down operations are executed every 4 steps. The total step count is multiplied by one-fourth.
If the result is 2 or less, slue-up and slue-down are not executed.

(3) If one-fourth the step count is 3 or more, values of slue-up data, operating data, and slue-down data are determined for slue-up and slue-down processing.

(4) Slue-up of 21 steps and slue-down of 7 steps are executed.

(5) Slue-up data, operating data, and slue-down data are shown in (a) to (c), by one-fourth the step count "n" obtained in (3) and (4).

(a) At n = 3 to 29, n = 2+4m (m = 1 to 6)

slue-up data = one-fourth the total step count - (m+2)
operating data = 0
slue-down data = m

(b) At n = 3 to 29, n < 2+4m (m = 1 to 6)

slue-up data = one-fourth the total step count - (m+1)
operating data = 0
slue-down data = m

(c) At n ≧ 30

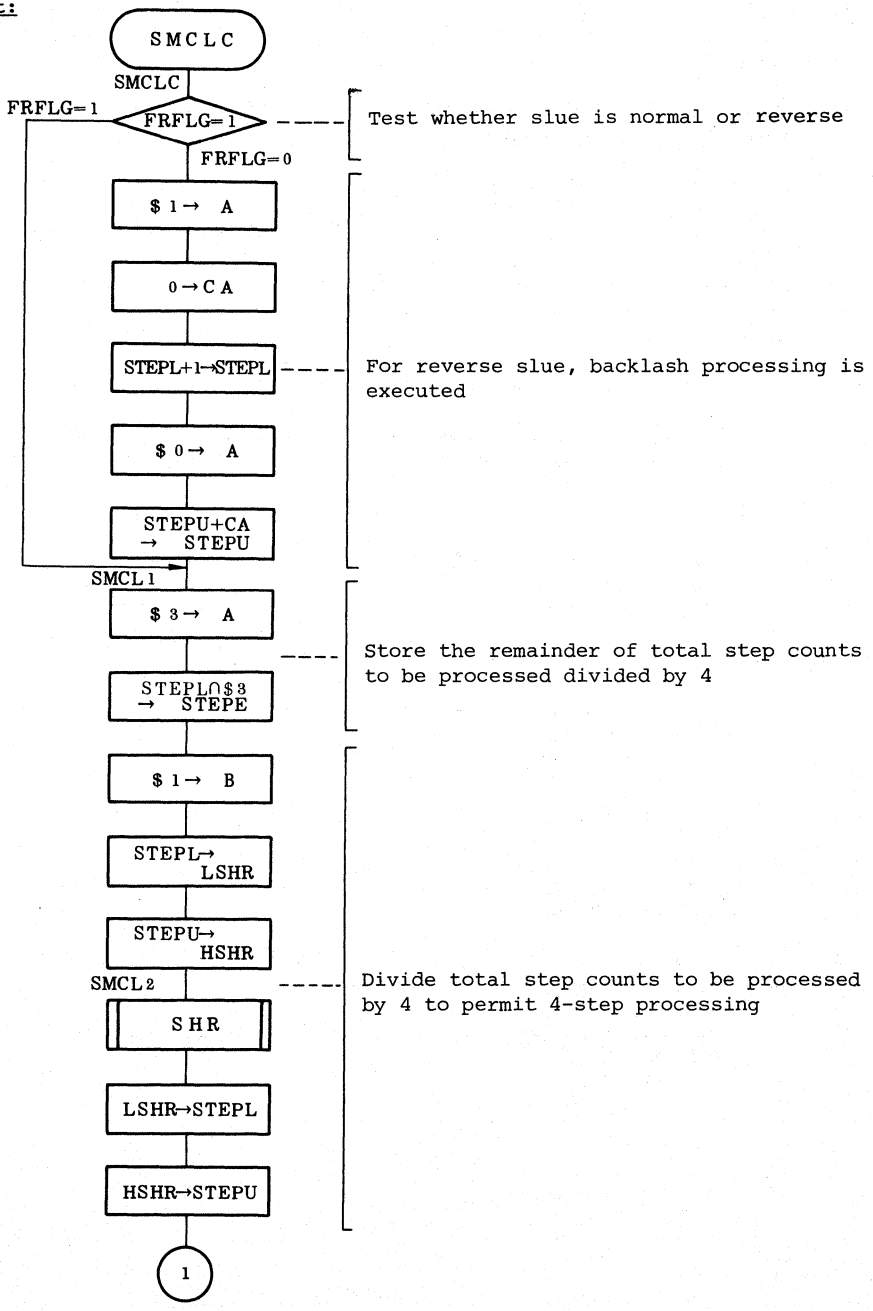
slue-up data = 21
operating data = one-fourth the total step count - 29
slue-down data = 7

(6) If one-fourth the step count is 0 , STEPE(RAM), the remainder of one-fourth the step count only is output.

(7) If one-fourth the step count is n(1 to 2), a step count of "n×4+STEPE (RAM)" is output.

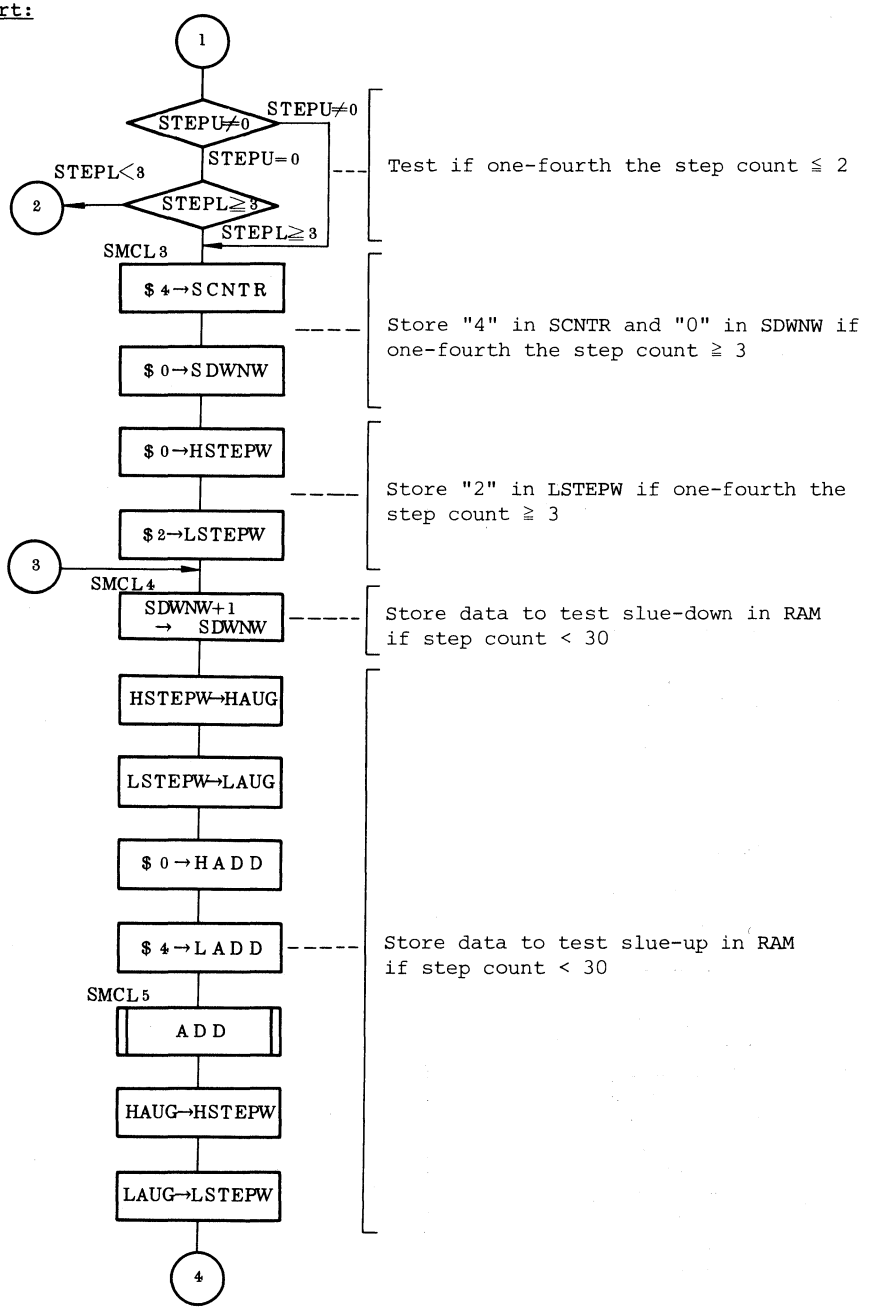
Program Module Name: Process Data	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: SMCLC

Flowchart:



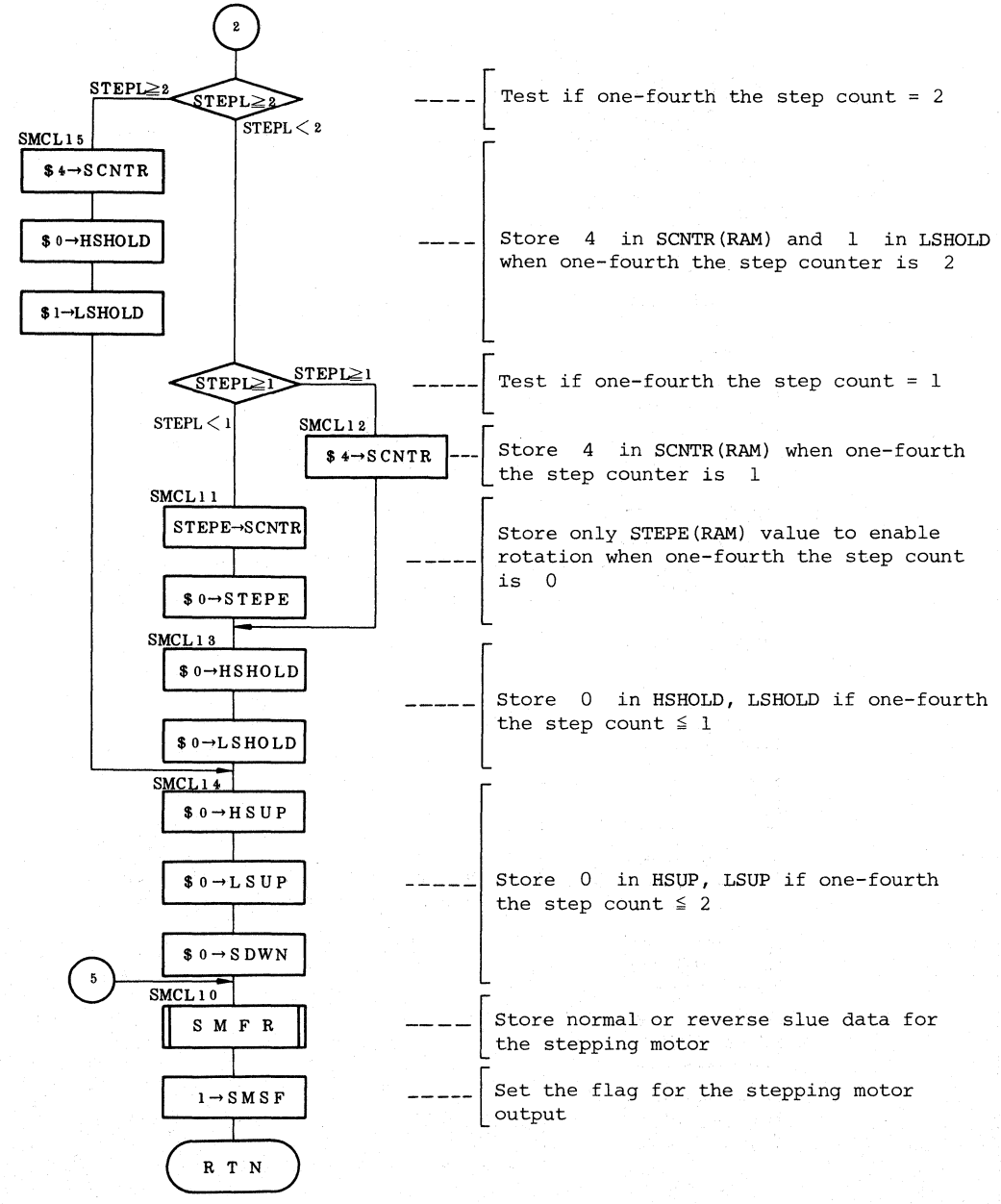
Program Module Name: Process Data	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: SMCLC

Flowchart:



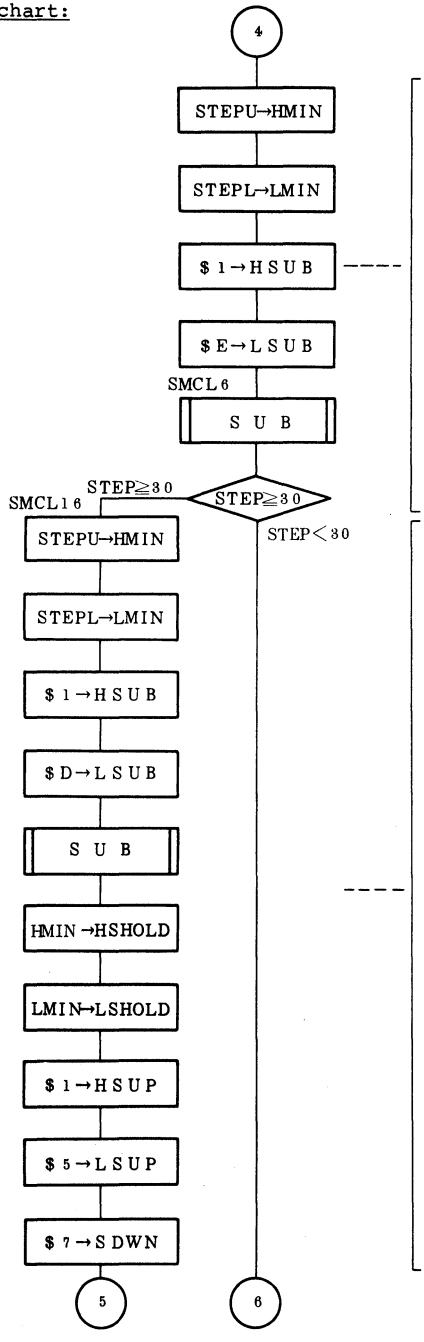
Program Module Name: Process Data	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: SMCLC

Flowchart:



Program Module Name: Process Data	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: SMCLC

Flowchart:

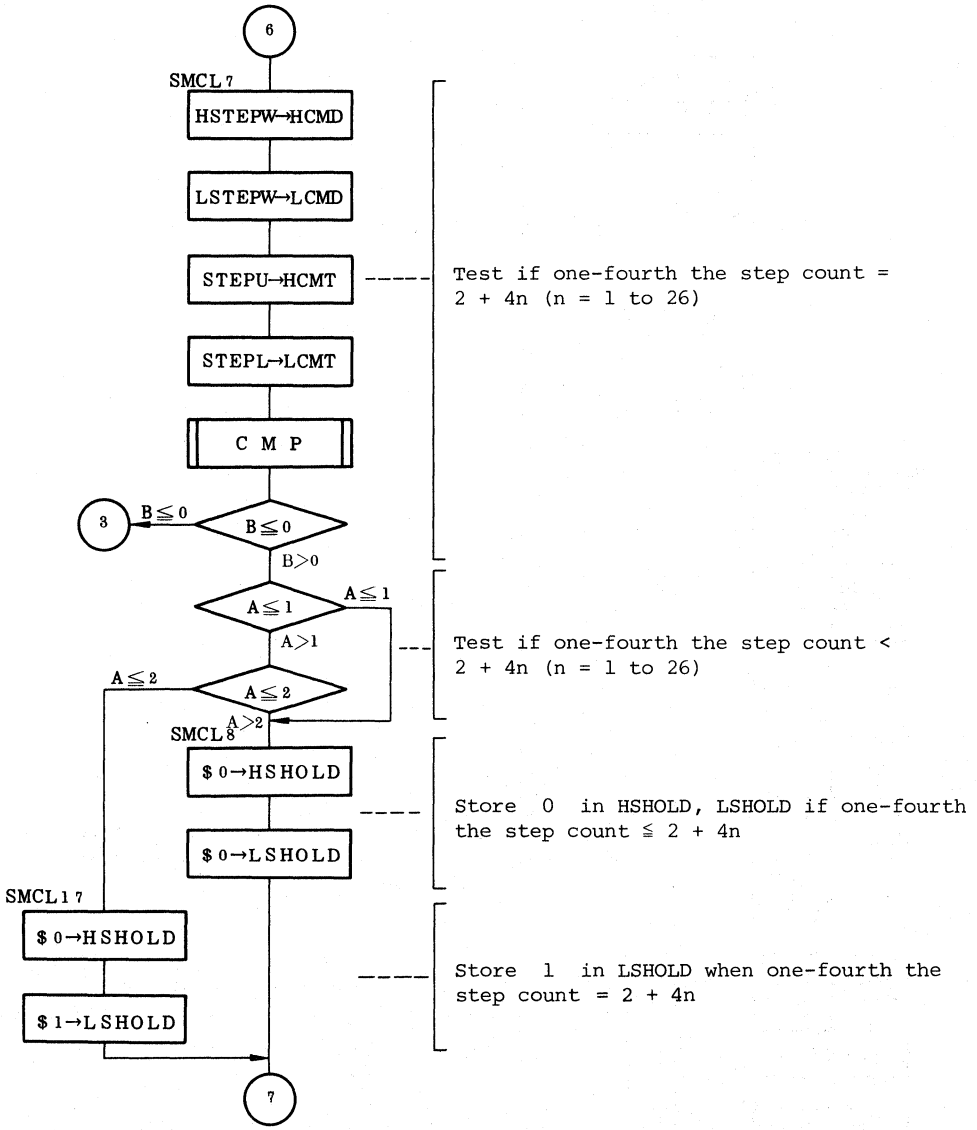


---- Test if one-fourth the step counter = 30

---- Store values of operating, slue-up, and slue-down if one-fourth the step counter ≧ 30

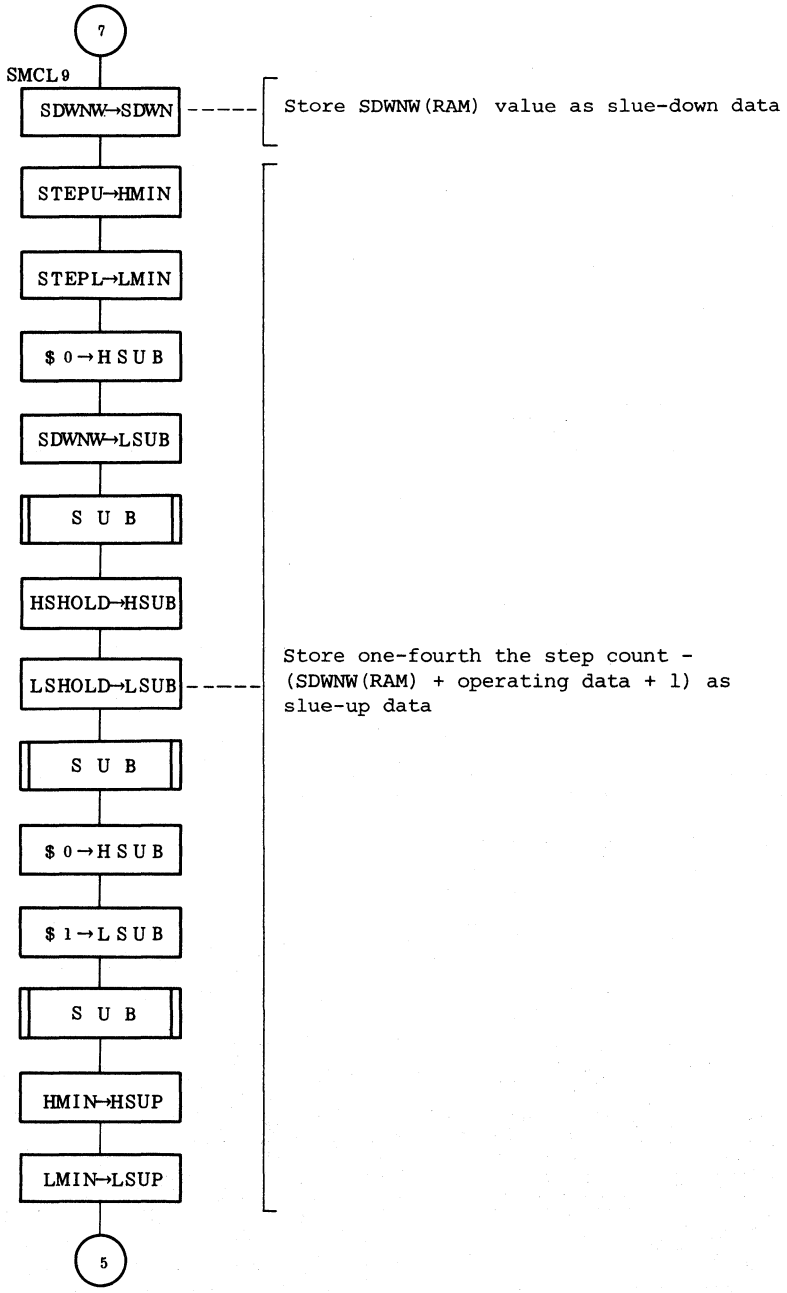
Program Module Name: Process Data	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: SMCLC

Flowchart:



Program Module Name: Process Data	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: SMCLC

Flowchart:



Program Module Name:

Generate Stepping Motor Output

MCU: HMCS402C/ HMCS404C/HMCS408C

Label: SMREV

Function:

Outputs pulse to the stepping motor.

Arguments:

Contents		1 digit = 4 bits Storage Location	No. of Digits
Entry	Slue-up data	HSUP(RAM)	1
		LSUP(RAM)	1
	Operating data	HSHOLD(RAM)	1
		LSHOLD(RAM)	1
	Slue-down data	SDWN(RAM)	1
	Remainder of the step	STEPE(RAM)	1
	Normal/ reverse slue flag	FRFLG(RAM)	1
	Slue start flag	SMSF(RAM)	1
Returns	Slue start flag	SMSF(RAM)	1

Changes in CPU Registers and Flags:

A	B
×	×

X	Y
×	×

SPX	SPY
×	×

W
●

CA	ST
×	×

● : Not Affected
× : Undefined
↕ : Result

Specifications:

1 word = 10 bits

ROM (Words): 205

RAM (Digits): 23

Stack (Digits): 4

No. of cycles: 83

Reentrant: No

Relocatable: No

Interrupt OK?: No

Description:

1. Function Details

(1) Argument details

FRFLG: SMSF(RAM) : Holds flag indicating rotation direction and rotation start of the stepping motor. Flag functions are shown in Table 8.5.
STEPU: STEPL(RAM) : Holds total slue step count.
HSUP : LSUP(RAM) : Holds slue-up data.
HSHOLD: LSHOLD(RAM): Holds operating data.
SDWN(RAM) : Holds slue-down data.
STEPE(RAM) : Holds remainder of the total step count divided by 4.

Specifications Notes:

Program Module Name:	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: SMREV
Generate Stepping Motor Output		

Description:

Table 8.5. Flag Functions

Label	Bit/Label SMSF	FRFLG	Function
FLG1	-	0	Rotates stepping motor clockwise (normal)
	-	1	Rotates stepping motor counterclockwise (reverse)
	0	-	Stops slue
	1	-	Starts slue

(2) Fig. 8.8 shows an example of SMREV execution. If the entry argument is held as shown in part ① of Fig. 8.8 outputs pulse to the stepping motor. Then, the slue start flag SMSF(RAM) is contained as shown in part ② of Fig. 8.8.

(3) SMREV calls other routines shown in Table 8.6.

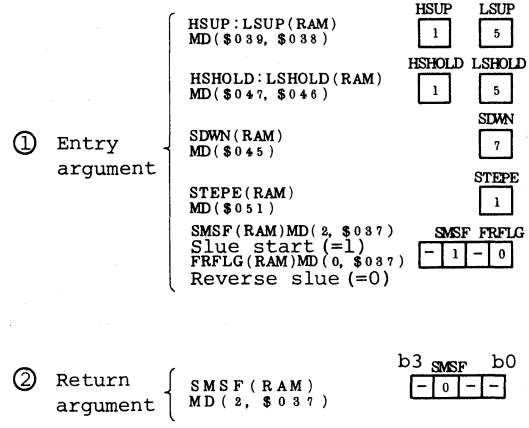


Fig. 8.8. Example of SMREV Execution

Table 8.6. Program Modules and Subroutines Used in SMREV

Program Module/ Subroutine Name	Label	Function
Load normal/ Reverse slue data	SMFR	Tests whether slue is normal or reverse and stores next data in stepping motor
Subtracting 8-bit binary data	SUB	Performs subtraction of 8-bit binary data in RAM, and stores result in RAM
Adding 8-bit binary data	ADD	Performs addition of 8-bit binary data in RAM, and stores result in RAM

Program Module Name:	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: SMREV
Generate Stepping Motor Output		

Description:

2. User Notes

   (1) Initializes timer B.

   (2) Clears bit IE to enable timer interrupt.

3. RAM Allocation

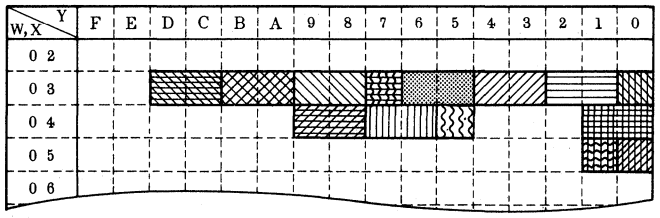
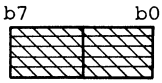
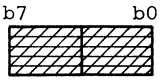
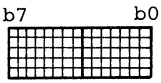
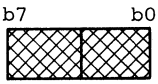
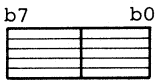


Fig. 8.9. RAM Allocation

Label	RAM	Description
HSUP:LSUP	b7 b0 MD($039,$038)	Stores one-fourth of the step count due to slue-up
HSHOLD:LSHOLD	b7 b0 MD($047,$046)	Stores one-fourth of the step count due to operating
SDWN	b3 b0 MD($045)	Stores one-fourth of the step count due to slue-down
FRFLG	b3 b0 MD(0,$037)	Stores flag indicating normal rotation of the stepping motor
SMSF	b3 b0 MD(2,$037)	Stores flag indicating start slue for the stepping motor

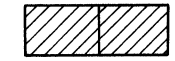
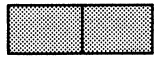
Program Module Name:	MCU:	Label:
Generate Stepping Motor Output	HMCS402C/ HMCS404C/HMCS408C	SMREV

Description:

Label	RAM	Description
SCNTR	 MD($050)	Stores 4-step counter
STEPE	 MD($051)	Stores remainder of a total step divided by 4
PBSM	 MD($030)	Stores data output to stepping motor
HAUG:LAUG	 MD($03D,$03C)	Holds 8-bit binary augend. After execution, contain addition result
HADD:LADD	 MD($049,$048)	Holds 8-bit binary addend
HMIN:LMIN	 MD($041, $040)	Holds 8-bit binary minuend. After execution contain subtraction result
HSUB:LSUB	 MD($03B,$03A)	Holds 8-bit binary subtrahend
HTLRD:LTLRD	 MD($032, $031)	Stores WORK AREA for timer load register

Program Module Name:	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: SMREV
Generate Stepping Motor Output		

Description:

Label	RAM	Description
HDDTA:LDDTA	b7 b0  MD($034,$033)	Stores address for slue-down data
HUDTA:LUDTA	b7 b0  MD($036,$035)	Stores address for slue-up data

4. Sample Application

```
WORKU   EQU     $055    }
WORKL   EQU     $054    }  ..... Reserve memory byte for the total step
                            counter
        :
        LAI     $9      }
        LMAD    PBSM    }  ..... Initialize the stepping motor output pin
        LRA     PBDTR   }
        REMD    SMSF    ..... Set rotation stop
        LMID    $2, LTLRD }
        LMID    $B, HTLRD }
        LMID    $2, TLRL  }  ..... Initialize output timing to the stepping
        LMID    $B, TLRU  }        motor
        LMID    $B, TMB   }
        SEMD    IE      ..... Enable interrupt
        REMD    FRFLG   }
        LAMD    WORKU   }
        LMAD    STEPU   }  ..... Load entry argument
        LAMD    WORKL   }
        LMAD    STEPL   }
        CALL    SMCLC   ..... Calculate entry argument
        :
```

Program Module Name:	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: SMREV
Generate Stepping Motor Output		

Description:

5. Basic Operation

(1) Program module SMREV is called by the timer routine.

(2) Outputs pulses one step to stepping motor every timer interrupt.

(3) At the beginning of timer interrupt, timer timing (slue-up, operating and slue-down slue) which is held in advance and output to the stepping motor.

(4) Following (3), data to be output at the next timer interrupt is loaded.

(5) Only the timer timing is held and no output is provided to the stepping motor, when the rotation start flag SMSF(RAM) is cleared.

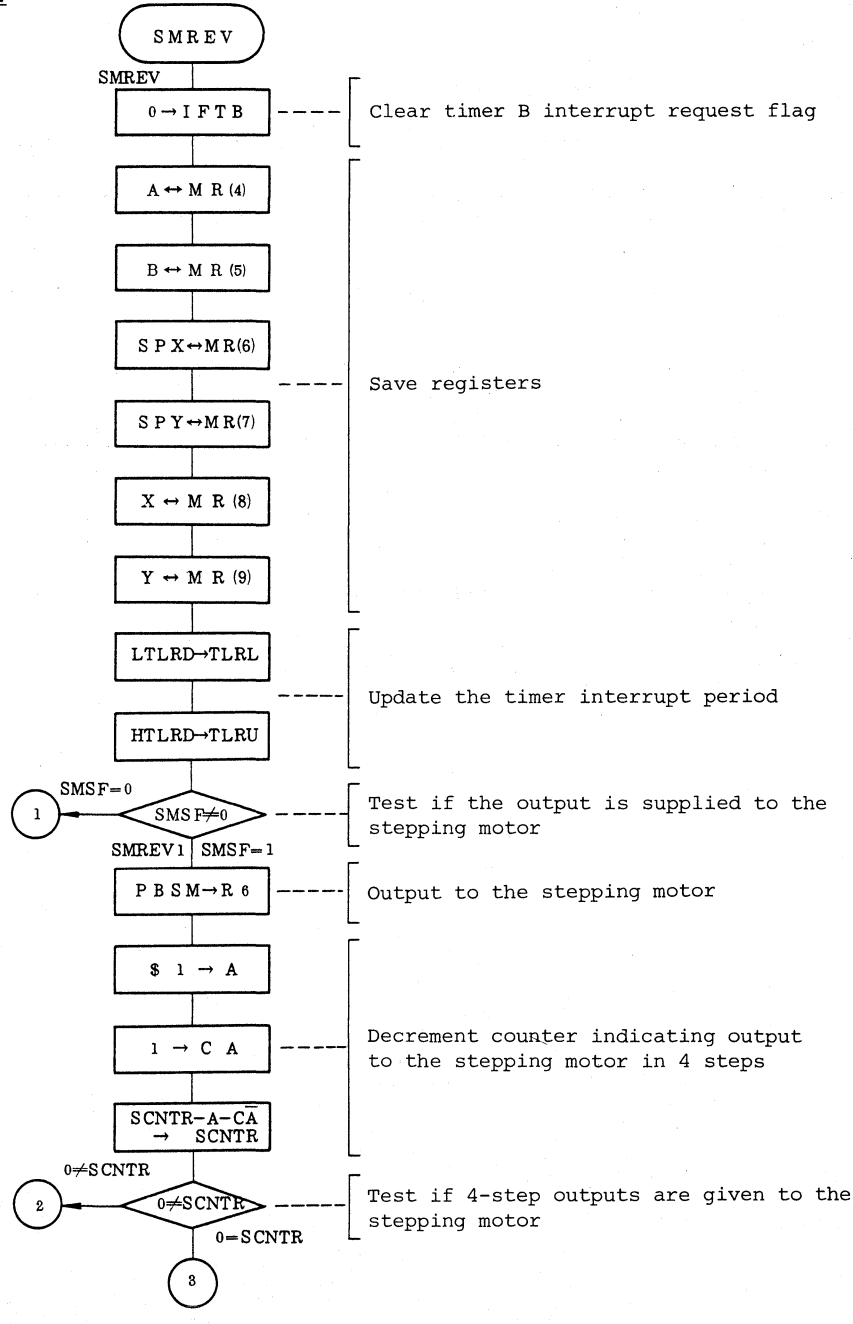
(6) Slue-up or slue-down output data is supplied to the stepping motor every 4-step.

(7) To implement (6), a test is performed to determine if the 4-step output is supplied within the same timing. Then, outputs are supplied in the sequence of slue-up, operating and slue-down.
Normal/reverse slue is tested and if it is reverse, output for backlash is provided.

(8) Timing data for slue-up and slue-down should be set in the data table in advance.

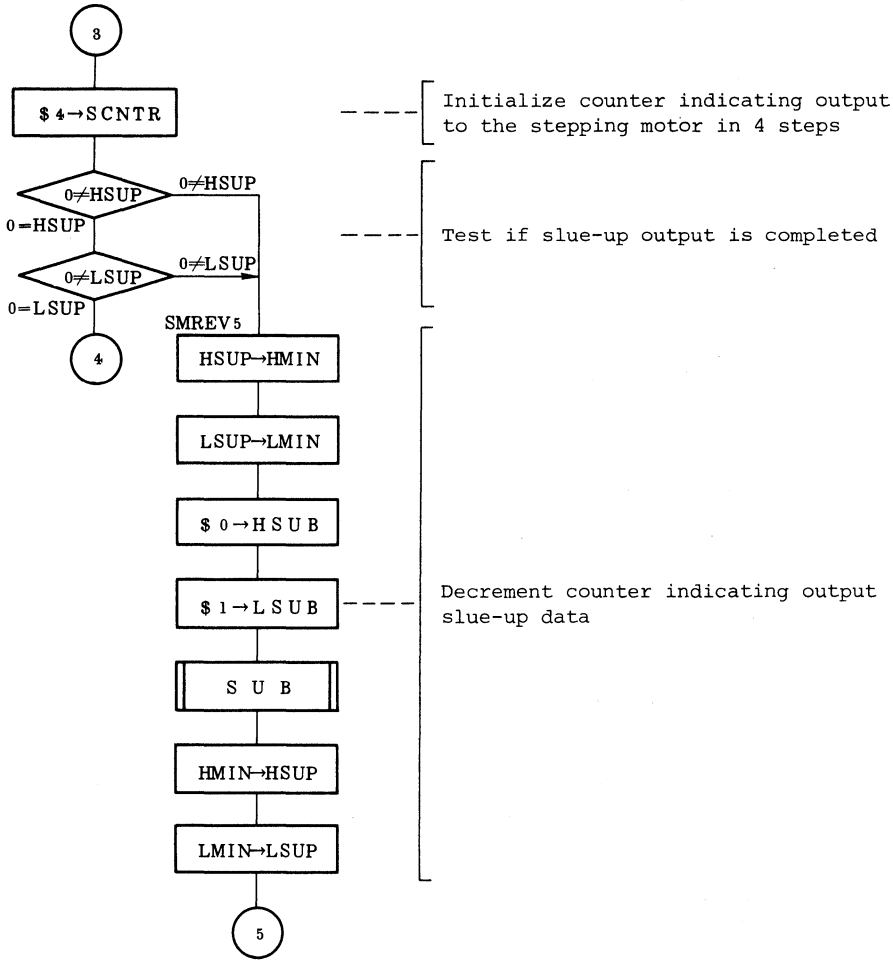
Program Module Name:	MCU:	Label:
Generate Stepping Motor Output	HMCS402C/ HMCS404C/HMCS408C	SMREV

Flowchart:



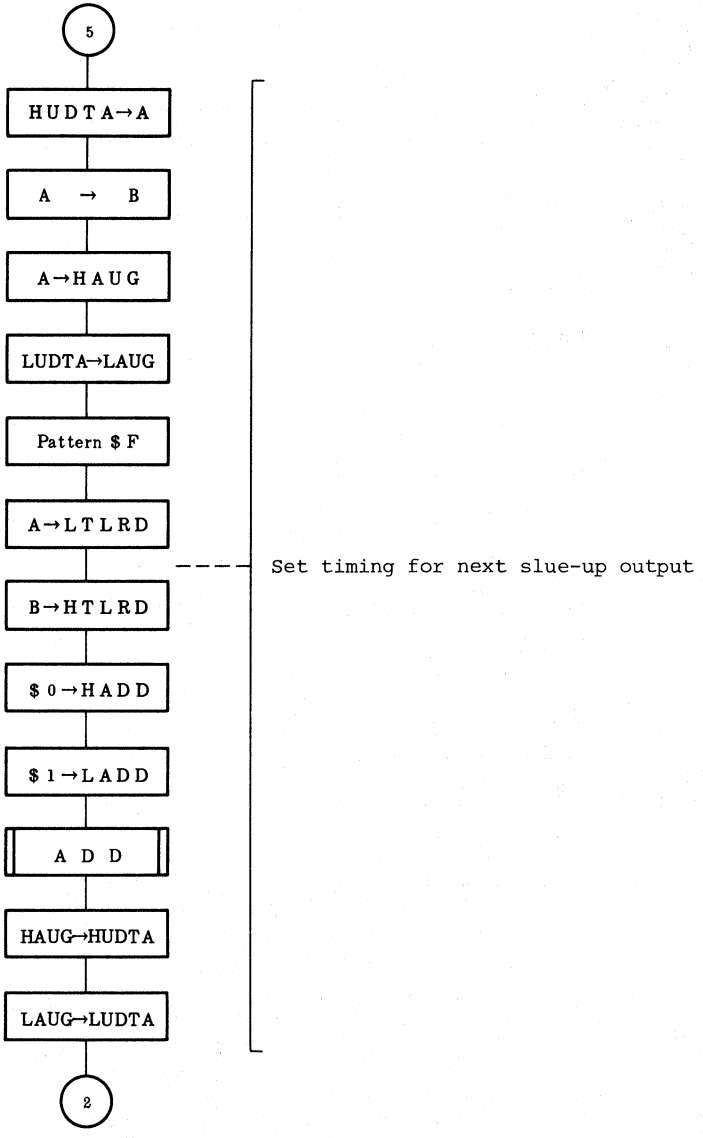
Program Module Name:	MCU:	Label:
Generate Stepping Motor Output	HMCS402C/ HMCS404C/HMCS408C	SMREV

Flowchart:



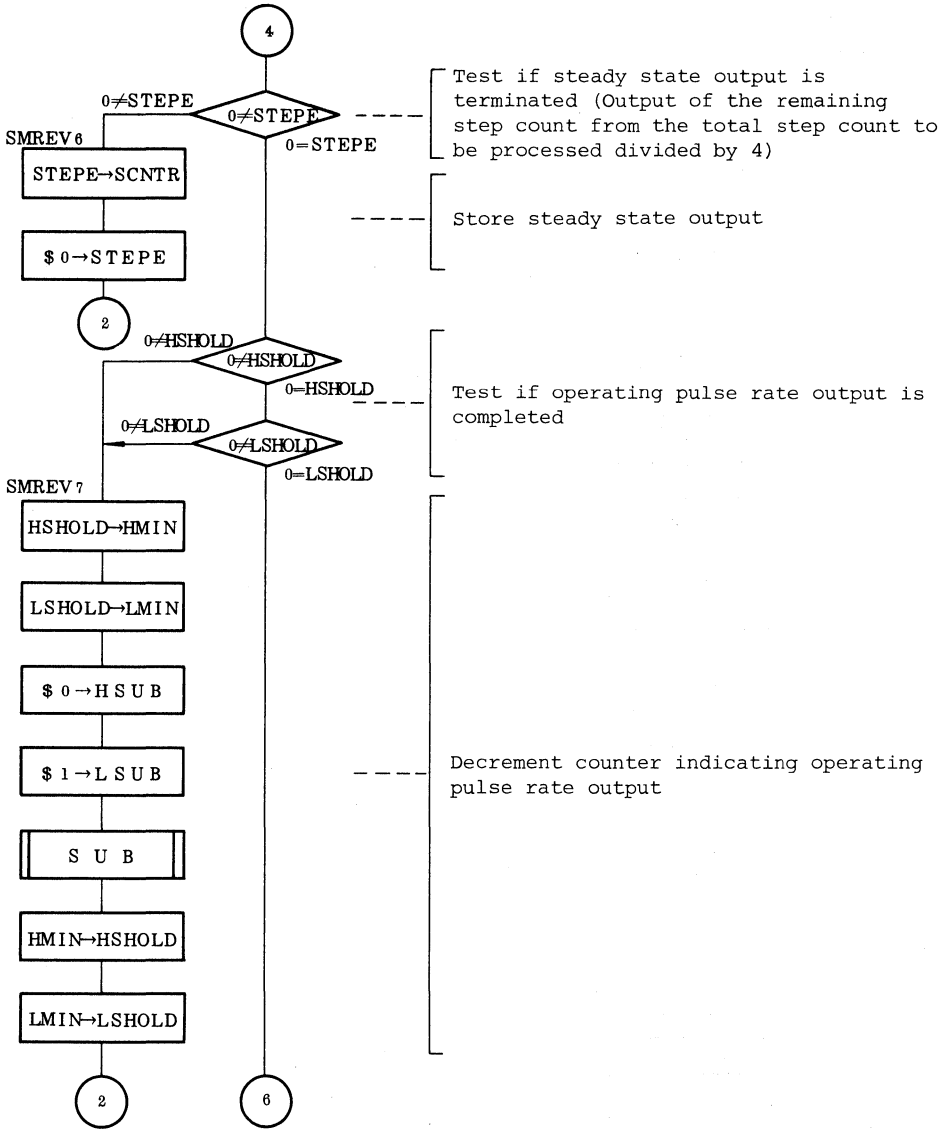
Program Module Name:	MCU:	Label:
Generate Stepping Motor Output	HMCS402C/ HMCS404C/HMCS408C	SMREV

Flowchart:



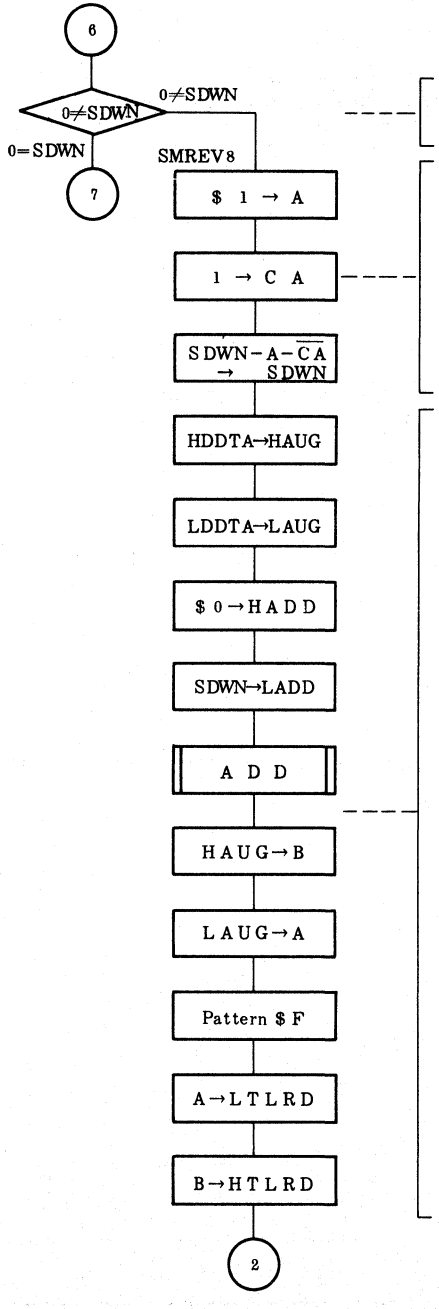
Program Module Name: Generate Stepping Motor Output	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: SMREV

Flowchart:



Program Module Name: Generate Stepping Motor Output	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: SMREV

Flowchart:



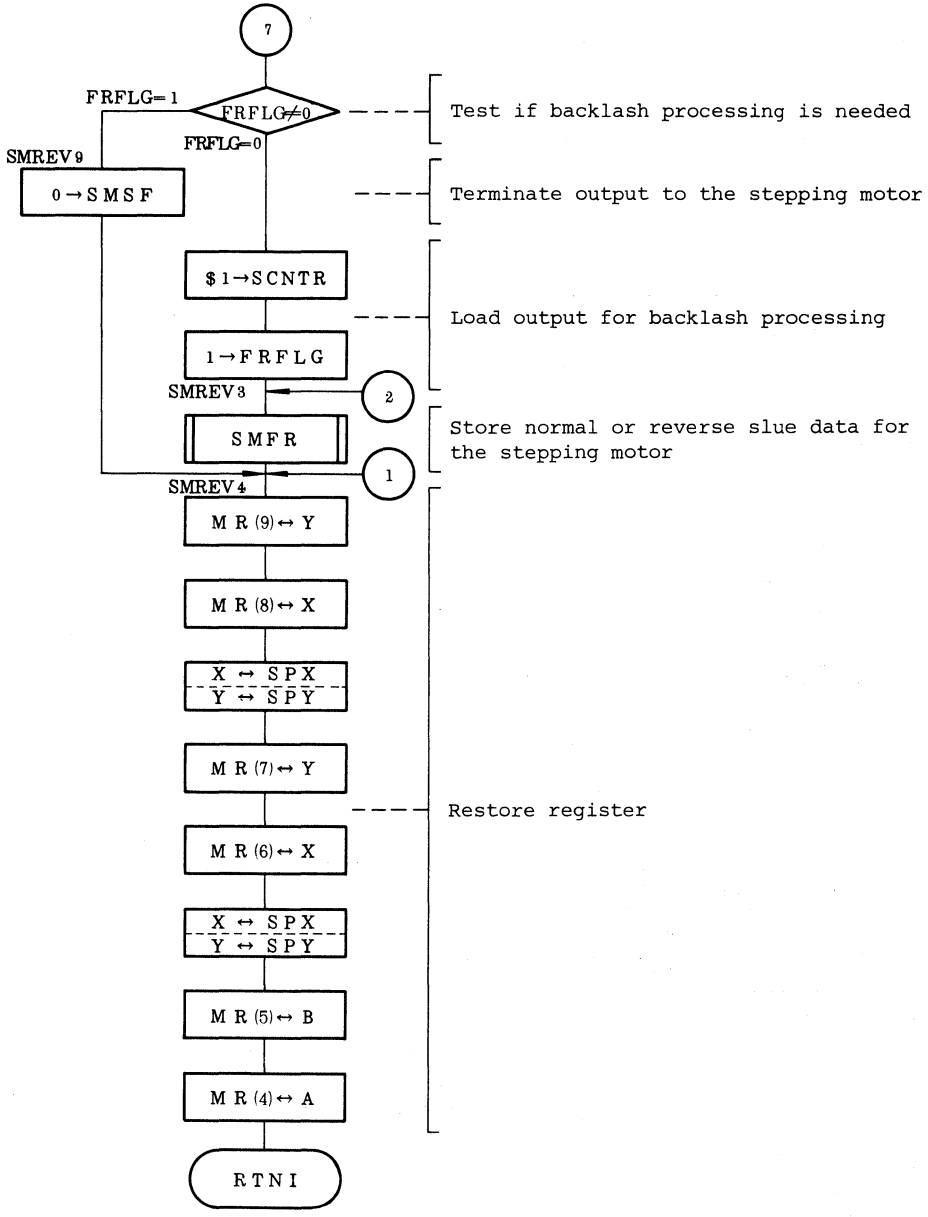
Test if slue-down output is completed

Decrement counter indicating slue-down output

Initialize timing for the next slue-down output

Program Module Name:	MCU:	Label:
Generate Stepping Motor Output	HMCS402C/ HMCS404C/HMCS408C	SMREV

Flowchart:



8.4 SUBROUTINE DESCRIPTION

Subroutine Name:	MCU:	Label:
Calculate Normal/Reverse Rotation Data	HMCS402C/ HMCS404C/HMCS408C	SMFR

Function:

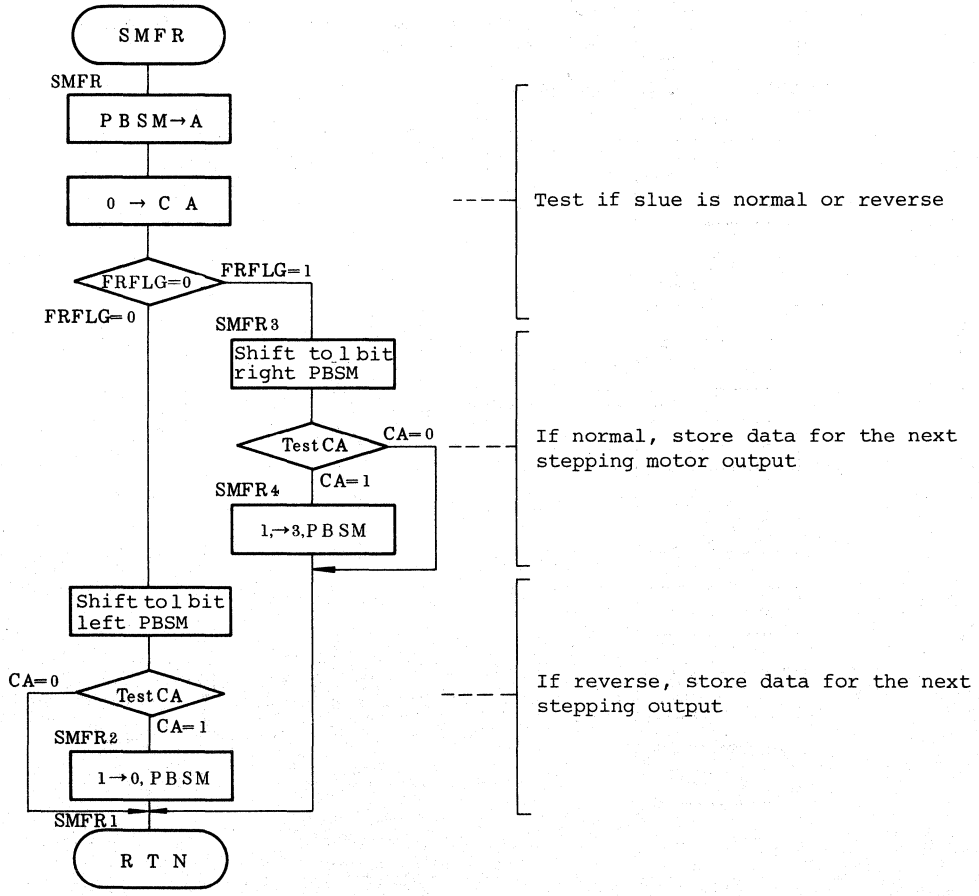
Tests whether slue flag (FRFLG(RAM)) is normal or reverse, stores data for the next stepping motor output in PBSM(RAM).

Basic Operation:

Tests whether slue is normal or reverse, and then performs shift.
As a result, output data for one-step normal/reverse slue is obtained.

Program Module Using This Subroutine: SMCLC, SMREV

Flowchart:



8.5 PROGRAM LISTING

```
ST-NO   OBJECT   ADRS   SOURCE STATEMENTS

00001   207      0000           LLEN     132
00002                           TITLE    STEPPING MOTOR CONTROL
00003                  *
00004                  ****   RAM ALLOCATION   ****************************************
00005                  *
00006                  PBSM     EQU      $030          WORK AREA FOR PBDTR
00007                  LTLRD    EQU      $031          LOWER WORK AREA FOR TLR
00008                  HTLRD    EQU      $032          UPPER WORK AREA FOR TLR
00009                  LDDTA    EQU      $033          LOWER WORK AREA FOR TLR DOWN
00010                  HDDTA    EQU      $034          UPPER WORK AREA FOR TLR DOWN
00011                  LUDTA    EQU      $035          LOWER WORK AREA FOR TLR UP
00012                  HUDTA    EQU      $036          UPPER WORK AREA FOR TLR UP
00013                  FRFLG    EQU      0.$037        FORWARD OR REVERSE FLAG
00014                  SMSF     EQU      2.$037        STEPPING MOTOR START FLAG
00015                  LSUP     EQU      $038          LOWER SLUE-UP NUMBER
00016                  HSUP     EQU      $039          UPPER SLUE-UP NUMBER
00017                  LCMD     EQU      $03A          LOWER FIRST VALUE
00018                  HCMD     EQU      $03B          UPPER FIRST VALUE
00019                  LAUG     EQU      $03C          LOWER AUGEND
00020                  HAUG     EQU      $03D          UPPER AUGEND
00021                  STEPL    EQU      $03E          LOWER STEP NUMBER
00022                  STEPU    EQU      $03F          UPPER STEP NUMBER
00023                  LMIN     EQU      $040          LOWER MINUEND
00024                  HMIN     EQU      $041          UPPER MINUEND
00025                  LSTEPW   EQU      $042          LOWER WORK FOR SUP
00026                  HSTEPW   EQU      $043          UPPER WORK FOR SUP
00027                  SDWNW    EQU      $044          WORK FOR SDWN
00028                  SDWN     EQU      $045          SLUE DOWN NUMBER
00029                  LSHOLD   EQU      $046          LOWER OPERATING NUMBER
00030                  HSHOLD   EQU      $047          UPPER OPERATING NUMBER
00031                  LADD     EQU      $048          LOWER ADDEND
00032                  HADD     EQU      $049          UPPER ADDEND
00033                  LCMT     EQU      $04A          LOWER SECOND VALUE NUMBER
00034                  HCMT     EQU      $04B          UPPER SECOND VALUE NUMBER
00035                  LSUB     EQU      $04C          LOWER SUBTRAHEND
00036                  HSUB     EQU      $04D          UPPER SUBTRAHEND
00037                  LSHR     EQU      $04E          LOWER 8-BIT BINARY
00038                  HSHR     EQU      $04F          UPPER 8-BIT BINARY
00039                  SCNTR    EQU      $050          4 STEPS COUNTER
00040                  STEPE    EQU      $051          STEP-EXTRA NUMBER
00041                  XSFT     EQU      $4            8-BIT BINARY DATA ADDR(X)
00042                  YSFT     EQU      $F            8-BIT BINARY DATA ADDR(Y)
00043                  BASFT    EQU      $D            8-BIT BINARY DATA BRANCH ADDR(Y)
00044                  XCMD     EQU      $3            FIRST VALUE ADDR(X)
00045                  XCMT     EQU      $4            SECOND VALUE ADDR(Y)
00046                  YCMT     EQU      $B            START ADDR(Y) OF FIRST & SECOND VALUE
00047                  BACMT    EQU      $9            BRANCH ADDR(Y) OF FIRST & SECOND VALUE
00048                  *
00049                  ****   SYMBOL DEFINITIONS   ************************************
00050                  *
00051                  TMB      EQU      $009          TIMER MODE REGISTER B
00052                  IFTB     EQU      0.$002        TIMER B INTERRUPT REQUEST BIT
00053                  IMTB     EQU      1.$002        TIMER B INTERRUPT MUSK BIT
00054                  IE       EQU      0.$000        INTERRUPT ENABLE BIT
00055                  TLRL     EQU      $00A          LOWER TIMER LOAD REGISTER
00056                  TLRU     EQU      $00B          UPPER TIMER LOAD REGISTER
00057                  PBDTR    EQU      $6            R(6) PORT TO STEPPING MOTOR
```

```
00058                   ****************************************************************
00059                   *                                                              *
00060                   *                VECTOR ADDRESSES                              *
00061                   *                                                              *
00062                   ****************************************************************
00063                   *
00064                           ORG     $0000
00065                   *
00066   150 010  0000           JMPL    SMMN            RESET
00067   150 010  0002           JMPL    SMMN            INT0
00068   150 010  0004           JMPL    SMMN            INT1
00069   150 010  0006           JMPL    SMMN            TIMER-A
00070   150 12D  0008           JMPL    SMREV           TIMER-B
00071   000      000A           NOP
00072   000      000B           NOP
00073   150 010  000C           JMPL    SMMN            SERIAL
00074                   ****************************************************************
00075                   *                                                              *
00076                   *               MAIN PROGRAM : SMMN                            *
00077                   *                                                              *
00078                   ****************************************************************
00079                   *
00080                           ORG     $0010
00081                   *
00082   0F0      0010   SMMN    LWI     $0              INITIALIZE W REGISTER
00083   239      0011           LAI     $9              INITIALIZE WORK AREA FOR PBDTR
00084   2D6      0012           LRA     PBDTR           INITIALIZE PBDTR
00085   180 030  0013           XMAD    PBSM
00086   18A 037  0015           REMD    SMSF            INITIALIZE SMSF
00087   1AB 009  0017           LMID    $B.TMB          INITIALIZE TMB
00088   189 002  0019           REMD    IMTB            INITIALIZE IMTB
00089   184 000  001B           SEMD    IE              ENABLE INTERRUPT
00090   188 037  001D           REMD    FRFLG           LOAD ARGUMENT
00091   1AC 03F  001F           LMID    $C.STEPU        STORE ARGUMENT OF 200 STEP
00092   1AB 03E  0021           LMID    $B.STEPL
00093   1AB 032  0023           LMID    $B.HTLRD        INITIALIZE WORK AREA FOR TLR
00094   1A2 031  0025           LMID    $2.LTLRD
00095   1A1 036  0027           LMID    $1.HUDTA        LOAD SLUE-UP ADDRESS
00096   1A0 035  0029           LMID    $0.LUDTA
00097   1A3 034  002B           LMID    $3.HDDTA        LOAD SLUE-DOWN ADDRESS
00098   1A0 033  002D           LMID    $0.LDDTA
00099   160 032  002F           CALL    SMCLC           CALCULATE SLUE DATA
00100   331      0031   PEND    BR      PEND            END OF PROGRAM
00101                   ****************************************************************
00102                   *                                                              *
00103                   *         NAME : SMCLC (PROCESS DATA)                          *
00104                   *                                                              *
00105                   ****************************************************************
00106                   *                                                              *
00107                   *        ENTRY : STEPU (UPPER NUMBER OF STEPS)                 *
00108                   *                STEPL (LOWER NUMBER OF STEPS)                 *
00109                   *                FRFLG (1:FORWARD SLUE,0:REVERSE SLUE)         *
00110                   *      RETURNS : HSUP   (UPPER SLOW-UP DATA)                   *
00111                   *                LSUP   (LOWER SLOW-UP DATA)                   *
00112                   *                HSHOLD (UPPER OPERATING DATA)                 *
00113                   *                LSHOLD (LOWER OPERATING DATA)                 *
00114                   *                SDWN  (SLOW-DOWN DATA)                        *
```

```
00115                    *              STEPE (REMAINDER OF STEP)                    *
00116                    *              FRFLG (FORWARD/REVERSE SLUE FLAG)            *
00117                    *              SMSF  (SLUE START FLAG)                      *
00118                    *                                                           *
00119                    *************************************************************
00120  18C 037  0032  SMCLC    TMD     FRFLG          TEST IF FORWARD OR REVERSE SLUE
00121  340      0034           BR      SMCL1
00122  231      0035           LAI     $1             IF REVERSE INCREMENT STEP
00123  0EC      0036           REC
00124  118 03E  0037           AMCD    STEPL
00125  194 03E  0039           LMAD    STEPL
00126  230      003B           LAI     $0
00127  118 03F  003C           AMCD    STEPU
00128  194 03F  003E           LMAD    STEPU
00129  233      0040  SMCL1    LAI     3              STORE 1/4 TIME OF STEP
00130  19C 03E  0041           ANMD    STEPL
00131  194 051  0043           LMAD    STEPE          SET STEP-EXTRA DATA
00132  201      0045           LBI     1              SET SHIFT COUNTER
00133  190 03E  0046           LAMD    STEPL          SET LOWER STEP
00134  194 04E  0048           LMAD    LSHR
00135  190 03F  004A           LAMD    STEPU          SET UPPER STEP
00136  194 04F  004C           LMAD    HSHR
00137  34F      004E           BRS     SMCL2
00138  160 216  004F  SMCL2    CALL    SHR            CALCULATE 1/4 TIME OF STEP
00139  190 04E  0051           LAMD    LSHR           SET 1/4 TIME OF STEP
00140  194 03E  0053           LMAD    STEPL
00141  190 04F  0055           LAMD    HSHR
00142  194 03F  0057           LMAD    STEPU
00143  120 03F  0059           INEMD   0.STEPU
00144  367      005B           BRS     SMCL3          BRANCH IF HSTEP=/0
00145  133 03E  005C           ILEMD   3.STEPL        LOWER STEP DATA>=3?
00146  367      005E           BRS     SMCL3          BRANCH IF STEP>=3
00147  132 03E  005F           ILEMD   2.STEPL        LOWER STEP DATA>=2?
00148  170 101  0061           BRS     SMCL15         BRANCH IF STEP=2
00149  131 03E  0063           ILEMD   1.STEPL        LOWER STEP DATA>=1?
00150  3F3      0065           BRS     SMCL12         BRANCH IF STEP=1
00151  3EB      0066           BRS     SMCL11         BRANCH IF STEP=0
00152  1A4 050  0067  SMCL3    LMID    4.SCNTR        LOAD 4 INTO SCNTR
00153  1A0 044  0069           LMID    0.SDWNW        CLEAR WORK AREA FOR SLUE-DOWN
00154  1A0 043  006B           LMID    0.HSTEPW
00155  1A2 042  006D           LMID    2.LSTEPW
00156  190 044  006F  SMCL4    LAMD    SDWNW          STORE SLUE-DOWN DATA
00157  281      0071           AI      1              INCREMENT SLUE-DOWN WORK
00158  194 044  0072           LMAD    SDWNW
00159  190 043  0074           LAMD    HSTEPW
00160  194 03D  0076           LMAD    HAUG           LOAD DATA FOR JUDGING
00161  190 042  0078           LAMD    LSTEPW
00162  194 03C  007A           LMAD    LAUG
00163  1A0 049  007C           LMID    0.HADD
00164  1A4 048  007E           LMID    4.LADD
00165  381      0080           BRS     SMCL5
00166  160 233  0081  SMCL5    CALL    ADD
00167  190 03D  0083           LAMD    HAUG
00168  194 043  0085           LMAD    HSTEPW
00169  190 03C  0087           LAMD    LAUG
00170  194 042  0089           LMAD    LSTEPW
00171  190 03F  008B           LAMD    STEPU
```

```
00172   194 041   008D            LMAD    HMIN          1/4 STEP = 30 ?
00173   190 03E   008F            LAMD    STEPL
00174   194 040   0091            LMAD    LMIN
00175   1A1 04D   0093            LMID    1.HSUB
00176   1AE 04C   0095            LMID    $E.LSUB
00177   398       0097            BRS     SMCL6
00178   160 241   0098    SMCL6   CALL    SUB
00179   170 109   009A            BRS     SMCL16        BRANCH IF 1/4 STEP > 30
00180   190 043   009C    SMCL7   LAMD    HSTEPW        1/4 STEP = 2+4N (N=1-26) ?
00181   194 03B   009E            LMAD    HCMD
00182   190 042   00A0            LAMD    LSTEPW
00183   194 03A   00A2            LMAD    LCMD
00184   190 03F   00A4            LAMD    STEPU
00185   194 04B   00A6            LMAD    HCMT
00186   190 03E   00A8            LAMD    STEPL
00187   194 04A   00AA            LMAD    LCMT
00188   160 221   00AC            CALL    CMP
00189   048       00AE            LAB
00190   2B0       00AF            ALEI    0
00191   36F       00B0            BRS     SMCL4         BRANCH IF STEPW < 1/4 STEP
00192   2B1       00B1            ALEI    1
00193   3B6       00B2            BRS     SMCL8         BRANCH IF STEPW > 1/4 STEP
00194   2B2       00B3            ALEI    2
00195   170 127   00B4            BRS     SMCL17
00196   1A0 047   00B6    SMCL8   LMID    0.HSHOLD      DEFINE OPERATING DATA AS 0
00197   1A0 046   00B8            LMID    0.LSHOLD
00198   190 044   00BA    SMCL9   LAMD    SDWNW         LOAD SLUE-DOWN DATA
00199   194 045   00BC            LMAD    SDWN          SLUE-UP DATA = 1/4 STEP - (SDWNW+OPERATING DATA+1)
00200   190 03F   00BE            LAMD    STEPU
00201   194 041   00C0            LMAD    HMIN
00202   190 03E   00C2            LAMD    STEPL
00203   194 040   00C4            LMAD    LMIN
00204   1A0 04D   00C6            LMID    0.HSUB
00205   190 044   00C8            LAMD    SDWNW
00206   194 04C   00CA            LMAD    LSUB
00207   160 241   00CC            CALL    SUB           (STEP-SDWN)
00208   190 047   00CE            LAMD    HSHOLD
00209   194 04D   00D0            LMAD    HSUB
00210   190 046   00D2            LAMD    LSHOLD
00211   194 04C   00D4            LMAD    LSUB
00212   160 241   00D6            CALL    SUB           (STEP-SDWNW-SHOLD)
00213   1A0 04D   00D8            LMID    0.HSUB
00214   1A1 04C   00DA            LMID    1.LSUB
00215   160 241   00DC            CALL    SUB           (STEP-SDWNW-SHOLD-1)
00216   190 041   00DE            LAMD    HMIN
00217   194 039   00E0            LMAD    HSUP
00218   190 040   00E2            LAMD    LMIN
00219   194 038   00E4            LMAD    LSUP
00220   160 1FB   00E6    SMCL10  CALL    SMFR          SET FORWARD OR REVERSE DATA
00221   186 037   00E8            SEMD    SMSF          START STEPPING MOTOR
00222   010       00EA            RTN
00223   190 051   00EB    SMCL11  LAMD    STEPE
00224   194 050   00ED            LMAD    SCNTR         DEFINE STEPE AS OPERATING NUMBER
00225   1A0 051   00EF            LMID    0.STEPE
00226   150 0F5   00F1            JMPL    SMCL13
00227   1A4 050   00F3    SMCL12  LMID    4.SCNTR       LOAD 4 INTO SCNTR
00228   1A0 047   00F5    SMCL13  LMID    0.HSHOLD      LOAD 0 INTO OPERATING DATA
```

```
00229  1A0 046  00F7            LMID    0,LSHOLD
00230  1A0 039  00F9  SMCL14    LMID    0,HSUP          SLUE-UP DATA = 0
00231  1A0 038  00FB            LMID    0,LSUP
00232  1A0 045  00FD            LMID    0,SDWN          SLUE-DOWN DATA = 0WN
00233  150 0E6  00FF            JMPL    SMCL10
00234  1A4 050  0101  SMCL15    LMID    4,SCNTR
00235  1A0 047  0103            LMID    0,HSHOLD        LOAD 4 INTO SCNTR
00236  1A1 046  0105            LMID    1,LSHOLD        DEFINE OPERATING DATA AS 1
00237  150 0F9  0107            JMPL    SMCL14
00238  190 03F  0109  SMCL16    LAMD    STEPU           LOAD FETCH DATA
00239  194 041  010B            LMAD    HMIN
00240  190 03E  010D            LAMD    STEPL
00241  194 040  010F            LMAD    LMIN
00242  1A1 04D  0111            LMID    1,HSUB
00243  1AD 04C  0113            LMID    $D,LSUB
00244  160 241  0115            CALL    SUB             STEP - 29
00245  190 041  0117            LAMD    HMIN
00246  194 047  0119            LMAD    HSHOLD
00247  190 040  011B            LAMD    LMIN
00248  194 046  011D            LMAD    LSHOLD
00249  1A1 039  011F            LMID    1,HSUP          SET SLUE UP DATA
00250  1A5 038  0121            LMID    5,LSUP
00251  1A7 045  0123            LMID    7,SDWN          SET SLUE DOWN DATA
00252  150 0E6  0125            JMPL    SMCL10
00253  1A0 047  0127  SMCL17    LMID    0,HSHOLD        DEFINE OPERATING DATA AS 1
00254  1A1 046  0129            LMID    1,LSHOLD
00255  150 0BA  012B            JMPL    SMCL9
00256                  ********************************************************************
00257                  *                                                                  *
00258                  *          NAME : SMREV (GENERATE STEPPING MOTOR OUTPUT)           *
00259                  *                                                                  *
00260                  ********************************************************************
00261                  *                                                                  *
00262                  *         ENTRY : HSUP (UPPER SLOW-UP DATA)                        *
00263                  *                 LSUP (LOWER SLOW-UP DATA)                        *
00264                  *                 HSHOLD (UPPER OPERATING DATA)                    *
00265                  *                 LSHOLD (LOWER OPERATING DATA)                    *
00266                  *                 SDWN (SLOW-DOWN DATA)                            *
00267                  *                 STEPE (REMAINDER OF STEP)                        *
00268                  *                 FRFLG (FORWARD / REVERSE SLUE FLAG)              *
00269                  *                 SMSF (SLUE START FLAG)                           *
00270                  *       RETURNS : SMSF (SLUE START FLAG)                           *
00271                  *                                                                  *
00272                  ********************************************************************
00273  188 002  012D  SMREV     REMD    IFTB            CLEAR TIMER-B INTERRUPT REQUEST BIT
00274  2F4      012F            XMRA    4               SAVE A
00275  048      0130            LAB
00276  2F5      0131            XMRA    5               SAVE B
00277  068      0132            LASPX
00278  2F6      0133            XMRA    6               SAVE SPX
00279  058      0134            LASPY
00280  2F7      0135            XMRA    7               SAVE SPY
00281  001      0136            XSPX
00282  068      0137            LASPX
00283  2F8      0138            XMRA    8               SAVE X
00284  0AF      0139            LAY
00285  2F9      013A            XMRA    9               SAVE Y
```

```
00286   190 031   013B            LAMD    LTLRD
00287   194 00A   013D            LMAD    TLRL        REINITIALIZE TIMER INTERRUPTS
00288   190 032   013F            LAMD    HTLRD
00289   194 00B   0141            LMAD    TLRU
00290   18E 037   0143            TMD     SMSF        TEST IF DRIVE MOTOR
00291   347       0145            BRS     SMREV1      BRANCH IF SMSF = 1
00292   371       0146            BRS     SMREV4      BRANCH IF SMSF = 0
00293   190 030   0147   SMREV1   LAMD    PBSM        DRIVE STEPPING MOTOR
00294   2D6       0149            LRA     PBDTR
00295   231       014A            LAI     1
00296   0EF       014B            SEC
00297   198 050   014C            SMCD    SCNTR       DECREMENT EVERY 4 STEPS
00298   194 050   014E            LMAD    SCNTR
00299   120 050   0150            INEMD   0,SCNTR     OUTPUT 4 STEPS ?
00300   36E       0152            BRS     SMREV2      BRANCH IF SCNTR =/ 0
00301   1A4 050   0153            LMID    4,SCNTR     INITIALIZE COUNTER EVERY 4 STEPS
00302   120 039   0155            INEMD   0,HSUP
00303   37F       0157            BRS     SMREV5      SLUE-UP COMPLETE
00304   120 038   0158            INEMD   0,LSUP
00305   37F       015A            BRS     SMREV5
00306   120 051   015B            INEMD   0,STEPE
00307   3B4       015D            BRS     SMREV6
00308   120 047   015E            INEMD   0,HSHOLD
00309   3BC       0160            BRS     SMREV7      TEST IF OPERATING COMPLETED
00310   120 046   0161            INEMD   0,LSHOLD
00311   3BC       0163            BRS     SMREV7      TEST IF SLUE-DOWN COMPLETED
00312   120 045   0164            INEMD   0,SDWN
00313   3D4       0166            BRS     SMREV8
00314   18C 037   0167            TMD     FRFLG
00315   3F7       0169            BRS     SMREV9
00316   1A1 050   016A            LMID    1,SCNTR
00317   184 037   016C            SEMD    FRFLG
00318   36F       016E   SMREV2   BRS     SMREV3
00319   160 1FB   016F   SMREV3   CALL    SMFR        SET FORWARD OR REVERSE DATA
00320   2F9       0171   SMREV4   XMRA    9           RESTORE REGISTERS
00321   0D8       0172            LYA
00322   2F8       0173            XMRA    8
00323   0E8       0174            LXA
00324   003       0175            XSPXY
00325   2F7       0176            XMRA    7
00326   0D8       0177            LYA
00327   2F6       0178            XMRA    6
00328   0E8       0179            LXA
00329   003       017A            XSPXY
00330   2F5       017B            XMRA    5
00331   0C8       017C            LBA
00332   2F4       017D            XMRA    4
00333   011       017E            RTNI
00334   190 039   017F   SMREV5   LAMD    HSUP        DECREMENT SLUE-UP COUNTER
00335   194 041   0181            LMAD    HMIN
00336   190 038   0183            LAMD    LSUP
00337   194 040   0185            LMAD    LMIN
00338   1A0 04D   0187            LMID    0,HSUB
00339   1A1 04C   0189            LMID    1,LSUB
00340   160 241   018B            CALL    SUB         SUP - 1
00341   190 041   018D            LAMD    HMIN
00342   194 039   018F            LMAD    HSUP
```

```
00343   190 040   0191              LAMD    LMIN
00344   194 038   0193              LMAD    LSUP
00345   190 036   0195              LAMD    HUDTA          SET TIMING FOR NEXT SLUE-UP
00346   0C8       0197              LBA
00347   194 03D   0198              LMAD    HAUG
00348   190 035   019A              LAMD    LUDTA
00349   194 03C   019C              LMAD    LAUG
00350   1BF       019E              P       $F
00351   194 031   019F              LMAD    LTLRD          STORE LOWER WORK FOR TLR
00352   048       01A1              LAB
00353   194 032   01A2              LMAD    HTLRD
00354   1A0 049   01A4              LMID    0.HADD
00355   1A1 048   01A6              LMID    1.LADD
00356   160 233   01A8              CALL    ADD            INCREMENT UDTA
00357   190 03D   01AA              LAMD    HAUG
00358   194 036   01AC              LMAD    HUDTA
00359   190 03C   01AE              LAMD    LAUG
00360   194 035   01B0              LMAD    LUDTA
00361   150 16E   01B2              JMPL    SMREV2
00362   190 051   01B4    SMREV6    LAMD    STEPE          SET FOR OPERATING
00363   194 050   01B6              LMAD    SCNTR
00364   1A0 051   01B8              LMID    0.STEPE
00365   150 16E   01BA              JMPL    SMREV2
00366   190 047   01BC    SMREV7    LAMD    HSHOLD
00367   194 041   01BE              LMAD    HMIN
00368   190 046   01C0              LAMD    LSHOLD
00369   194 040   01C2              LMAD    LMIN
00370   1A0 04D   01C4              LMID    0.HSUB
00371   1A1 04C   01C6              LMID    1.LSUB
00372   160 241   01C8              CALL    SUB            DECREMENT OPERATING COUNTER
00373   190 041   01CA              LAMD    HMIN
00374   194 047   01CC              LMAD    HSHOLD
00375   190 040   01CE              LAMD    LMIN
00376   194 046   01D0              LMAD    LSHOLD
00377   150 16E   01D2              JMPL    SMREV2
00378   231       01D4    SMREV8    LAI     1
00379   0EF       01D5              SEC
00380   198 045   01D6              SMCD    SDWN           DECREMENT SLUE-DOWN COUNTER
00381   194 045   01D8              LMAD    SDWN
00382   190 034   01DA              LAMD    HDDTA          SET TIMING FOR NEXT SLUE-DOWN
00383   194 03D   01DC              LMAD    HAUG
00384   190 033   01DE              LAMD    LDDTA
00385   194 03C   01E0              LMAD    LAUG
00386   1A0 049   01E2              LMID    0.HADD
00387   190 045   01E4              LAMD    SDWN
00388   194 048   01E6              LMAD    LADD
00389   160 233   01E8              CALL    ADD            DDTA + SDWN
00390   190 03D   01EA              LAMD    HAUG
00391   0C8       01EC              LBA
00392   190 03C   01ED              LAMD    LAUG
00393   1BF       01EF              P       $F
00394   194 031   01F0              LMAD    LTLRD
00395   048       01F2              LAB
00396   194 032   01F3              LMAD    HTLRD
00397   150 16E   01F5              JMPL    SMREV2
00398   18A 037   01F7    SMREV9    REMD    SMSF           STOP DRIVING STEPPING MOTOR
00399   150 171   01F9              JMPL    SMREV4
```

```
00400                   ********************************************************************
00401                   *                                                                  *
00402                   *       NAME : SMFR (CALCULATE NORMAL/INVERSE ROTATION             *
00403                   *                                   DATA)                          *
00404                   ********************************************************************
00405   18C 037  01FB   SMFR    TMD     FRFLG           NORMAL OR INVERSE ROTATION ?
00406   190 030  01FD           LAMD    PBSM
00407   30B      01FF           BRS     SMFR3           BRANCH IF FRFLG=1
00408   0EC      0200           REC
00409   0A1      0201           ROTL                    IF INVERSE, SET DATA FOR NEXT OUTPUT
00410   180 030  0202           XMAD    PBSM
00411   06F      0204           TC
00412   307      0205           BRS     SMFR2
00413   010      0206   SMFR1   RTN
00414   184 030  0207   SMFR2   SEMD    $0,PBSM
00415   150 206  0209           JMPL    SMFR1
00416   0EC      020B   SMFR3   REC                     IF NORMAL, SET DATA FOR NEXT OUTPUT
00417   0A0      020C           ROTR
00418   180 030  020D           XMAD    PBSM
00419   06F      020F           TC
00420   312      0210           BRS     SMFR4
00421   306      0211           BRS     SMFR1
00422   187 030  0212   SMFR4   SEMD    $3,PBSM
00423   150 206  0214           JMPL    SMFR1
00424                   ********************************************************************
00425                   *                                                                  *
00426                   *       NAME : SHR (SHIFT 8-BIT DATA)                              *
00427                   *                                                                  *
00428                   ********************************************************************
00429   224      0216   SHR     LXI     XSFT            LOAD ADDR(X)
00430   21F      0217   SHR1    LYI     YSFT            LOAD ADDR(Y)
00431   0EC      0218           REC
00432   090      0219   SHR2    LAM                     LOAD BINARY DATA
00433   0A0      021A           ROTR                    ROTATE BINARY DATA
00434   0D0      021B           LMADY                   STORE SHIFT DATA AND DECREMENT ADDR(Y)
00435   07D      021C           YNEI    BASFT
00436   319      021D           BRS     SHR2
00437   0CF      021E           DB                      DECREMENT SHIFT COUNTER
00438   317      021F           BRS     SHR1            BRANCH UNTIL SHIFT COUNTER = $0
00439   010      0220           RTN
00440                   ********************************************************************
00441                   *                                                                  *
00442                   *       NAME : CMP (COMPARE 8-BIT BINARY DATA)                     *
00443                   *                                                                  *
00444                   ********************************************************************
00445   223      0221   CMP     LXI     XCMD            LOAD FIRST VALUE ADDR(SPX)
00446   001      0222           XSPX
00447   224      0223           LXI     XCMT            LOAD SECOND VALUE NUMBER ADDR(X)
00448   21B      0224           LYI     YCMT            LOAD SECOND VALUE NUMBER ADDR(Y)
00449   200      0225           LBI     $0              CLEAR B
00450   091      0226   CMP1    LAMX                    LOAD SECOND VALUE
00451   014      0227           ALEM                    DETERMINE RELATION
00452   32A      0228           BRS     CMP2            BRANCH IF A <= M
00453   332      0229           BRS     CMP4            BRANCH IF A > M
00454   004      022A   CMP2    ANEM                    TEST IF A = M
00455   331      022B           BRS     CMP3            BRANCE IF A = M
00456   001      022C           XSPX                    LOAD SECOND VALUE ADDR(X)
```

```
00457  0DF      022D            DY                      DECREMENT ADDR(Y)
00458  079      022E            YNEI    BACMT           TEST LOOP COUNTER
00459  326      022F            BRS     CMP1
00460  04C      0230            IB                      INCREMENT B (RESULT)
00461  04C      0231    CMP3    IB                      INCREMENT B (RESULT)
00462  010      0232    CMP4    RTN
00463                    ********************************************************************
00464                    *                                                                  *
00465                    *       NAME : ADD (ADD 8-BIT BINARY DATA)                         *
00466                    *                                                                  *
00467                    ********************************************************************
00468  0EC      0233    ADD     REC                     CLEAR CARRY FLAG
00469  190 048  0234            LAMD    LADD            LOAD LOWER ADDEND
00470  118 03C  0236            AMCD    LAUG            ADD LOWER ADDEND TO LOWER AUGEND
00471  194 03C  0238            LMAD    LAUG            STORE LOWER ADDITION RESULT
00472  190 049  023A            LAMD    HADD            LOAD UPPER ADDEND
00473  118 03D  023C            AMCD    HAUG            ADD UPPER ADDEND TO UPPER AUGEND
00474  194 03D  023E            LMAD    HAUG            STORE UPPER ADDITION RESULT
00475  010      0240            RTN
00476                    ********************************************************************
00477                    *                                                                  *
00478                    *       NAME : SUB (SUBTRACT 8-BIT BINARY DATA)                    *
00479                    *                                                                  *
00480                    ********************************************************************
00481  0EF      0241    SUB     SEC                     SET CARRY FLAG
00482  190 04C  0242            LAMD    LSUB            LOAD LOWER SUBTRAHEND
00483  198 040  0244            SMCD    LMIN            SUBTRACT LOWER SUBTRAHEND FROM LOWER MINUEND
00484  194 040  0246            LMAD    LMIN            STORE LOWER SUBTRACTION RESULT
00485  190 04D  0248            LAMD    HSUB            LOAD UPPER SUBTRAHEND
00486  198 041  024A            SMCD    HMIN            SUBTRACT UPPER SUBTRAHEND FROM UPPER MINUEND
00487  194 041  024C            LMAD    HMIN            STORE UPPER SUBTRACTION RESULT
00488  010      024E            RTN
00489                    ********************************************************************
00490                    *                                                                  *
00491                    *       SLUE-UP DATA TABLE                                         *
00492                    *                                                                  *
00493                    ********************************************************************
00494                    *
00495                            ORG     $0F10
00496                    *
00497  1B7      0F10            DC      $1B7            215PPS
00498  1BC      0F11            DC      $1BC            230PPS
00499  1C0      0F12            DC      $1C0            245PPS
00500  1C4      0F13            DC      $1C4            260PPS
00501  1C7      0F14            DC      $1C7            275PPS
00502  1CA      0F15            DC      $1CA            290PPS
00503  1CD      0F16            DC      $1CD            305PPS
00504  1CF      0F17            DC      $1CF            320PPS
00505  1D1      0F18            DC      $1D1            335PPS
00506  1D3      0F19            DC      $1D3            350PPS
00507  1D5      0F1A            DC      $1D5            365PPS
00508  1D7      0F1B            DC      $1D7            380PPS
00509  1D8      0F1C            DC      $1D8            395PPS
00510  1DA      0F1D            DC      $1DA            410PPS
00511  1DB      0F1E            DC      $1DB            425PPS
00512  1DC      0F1F            DC      $1DC            440PPS
00513  1DE      0F20            DC      $1DE            455PPS
```

```
00514   1DF     0F21            DC      $1DF            470PPS
00515   1E0     0F22            DC      $1E0            485PPS
00516   1E1     0F23            DC      $1E1            500PPS
00517   1E2     0F24            DC      $1E2            515PPS
00518                   ************************************************************
00519                   *                                                          *
00520                   *              SLUE-DOWN DATA TABLE                        *
00521                   *                                                          *
00522                   ************************************************************
00523                   *
00524                           ORG     $0F30
00525                   *
00526   1B2     0F30            DC      $1B2            200PPS
00527   1C0     0F31            DC      $1C0            245PPS
00528   1CA     0F32            DC      $1CA            290PPS
00529   1D1     0F33            DC      $1D1            335PPS
00530   1D7     0F34            DC      $1D7            380PPS
00531   1DB     0F35            DC      $1DB            425PPS
00532   1DF     0F36            DC      $1DF            470PPS
00533                           END
```

SECTION 9. USE OF COMMERCIAL KEYBOARDS

9.1 HARDWARE DESCRIPTION

9.1.1 Function

Receives key data from a standard ASCII keyboard.

9.1.2 Microcomputer Operation

The HMCS404C accesses data from an ASCII keyboard using a First In-First Out roll buffer. Port R is selected to perform parallel handshaking between the $\overline{INT_1}$ pin and port R. Input data is read at the falling edge of the $\overline{STROBE}$ signal and data is written to the roll buffer by input strobe interrupt.

9.1.3 Peripheral Devices

ASCII keyboard: Outputs ASCII codes and $\overline{STROBE}$ signal.

9.1.4 Circuit Diagram

The interface circuit for reading data from an ASCII keyboard is shown in Fig. 9.1.

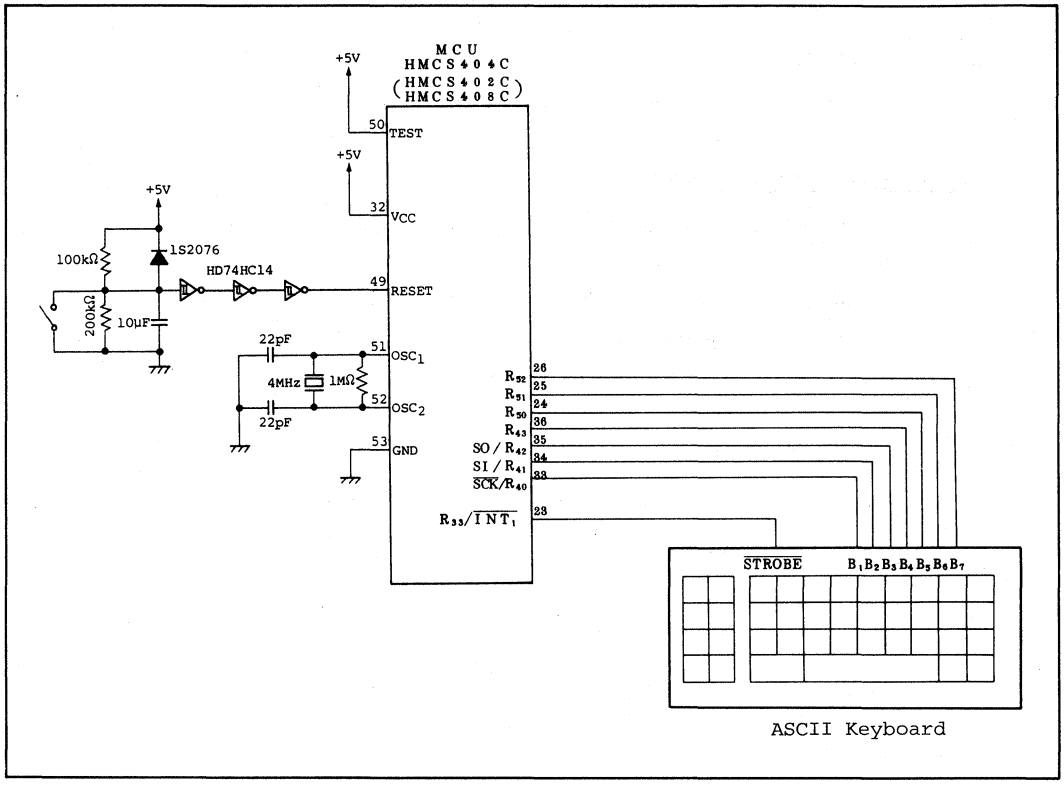


Fig. 9.1. Reading Data from ASCII Keyboard

9.1.5 Pin Functions

Pin functions at the interface between the HMCS404C and ASCII keyboard are shown in Table 9.1.

Table 9.1. Pin Functions

Pin Name (HMCS404C)	Input/ Output	Active Level (High or Low)	Function	Pin Name (Key-board)
$R_{33}/\overline{INT_1}$	Input	Low	$\overline{STROBE}$ signal	$\overline{STROBE}$
R_{40}	Input	——	Key data input signal	B_1
R_{41}	Input	——		B_2
R_{42}	Input	——		B_3
R_{43}	Input	——		B_4
R_{50}	Input	——		B_5
R_{51}	Input	——		B_6
R_{52}	Input	——		B_7

9.1.6 Hardware Operation

The timing chart for the ASCII keyboard is shown in Fig. 9.2. If a key in ASCII keyboard is depressed, data and $\overline{STROBE}$ signal are output as shown in Fig. 9.2.

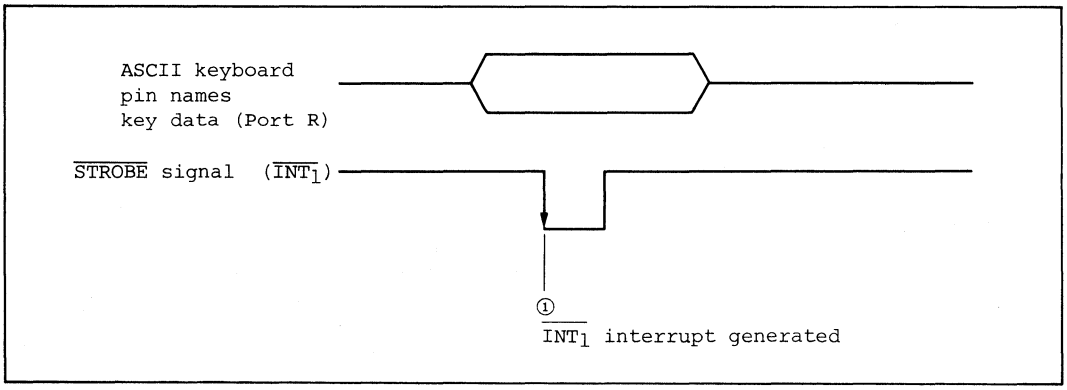


Fig. 9.2. ASCII Keyboard Timing Chart

9.2 SOFTWARE DESCRIPTION

9.2.1 Program Module Configuration

The program module configuration for reading key data from ASCII keyboard is shown in Fig. 9.3.

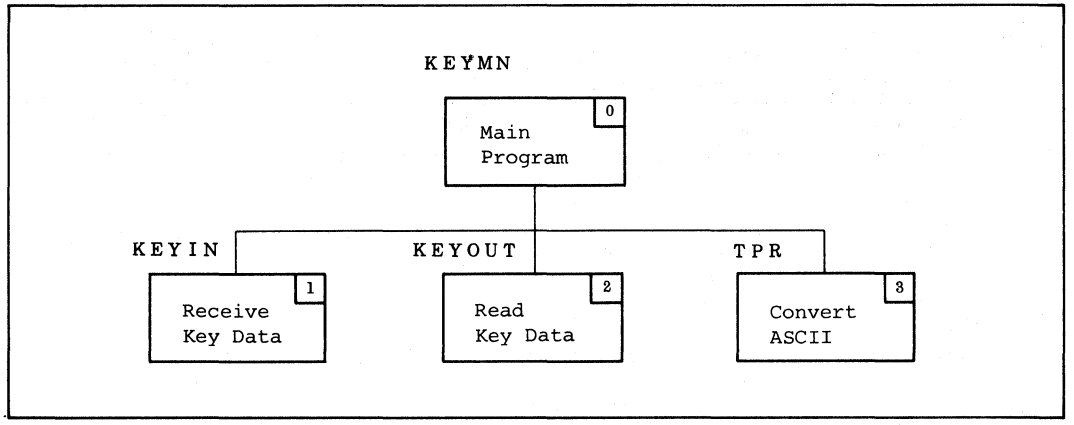


Fig. 9.3. Program Module Configuration

9.2.2 Program Module Functions

Program module functions are summarized in Table 9.2.

Table 9.2. Program Module Functions

No.	Program Module Name	Label	Function
0	Main Program	KEYMN	Receives key data from ASCII keyboard
1	Receive Key Data	KEYIN	Receives key data and writes then to roll buffer
2	Read Key Data	KEYOUT	Reads data in roll buffer
3	Convert ASCII	TPR	Converts ASCII lower case into upper case. (Refer to TPR in HMCS400 Series Application Note (Software Edition) for details)

9.2.3 Program Module Process Flow (Main Program)

The flowchart in Fig. 9.4 is an example of key data input from ASCII keyboard performed by the program module in Fig. 9.4.

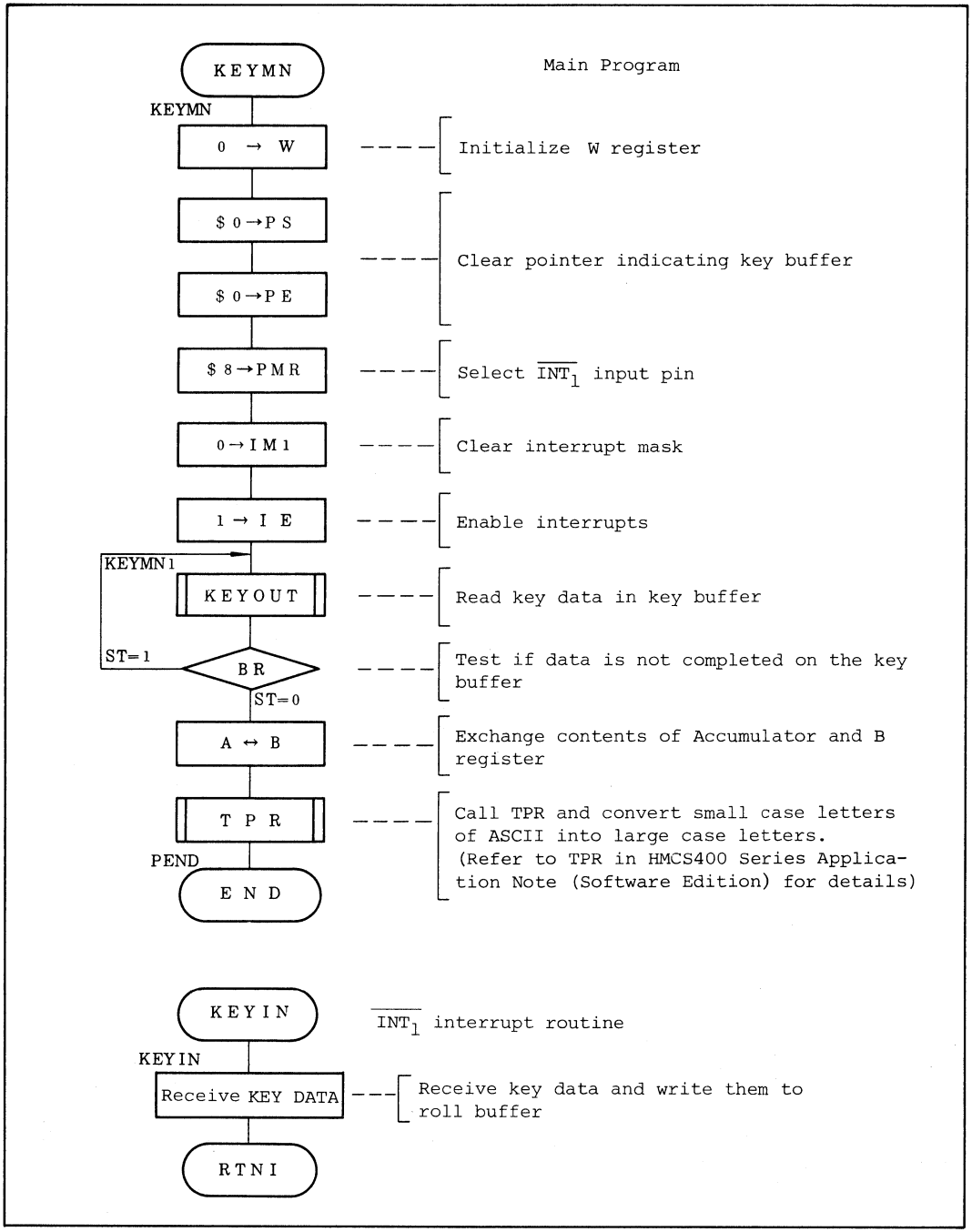


Fig. 9.4. Program Module Flowchart

Program Module Name:
Receive Key Data

MCU: HMCS402C/
HMCS404C/HMCS408C

Label: KEYIN

Function:

Receives key data from ASCII key board and writes them to roll buffer.

Arguments:

None

Changes in CPU Registers and Flags:

A	B
×	●

X	Y
×	×

SPX	SPY
×	●

W
●

CA	ST
×	×

● : Not Affected
× : Undefined
↕ : Result

Specifications:

1 word = 10 bits
ROM (Words): 35
RAM (Digits): 2
Stack (Digits): 0
No. of cycles: 36
Reentrant: No
Relocatable: No
Interrupt OK?: No

Description:

1. Function Details

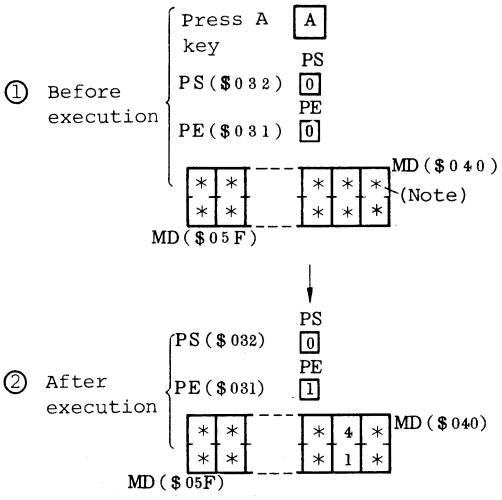
(1) KEYIN has no arguments.

(2) Example of KEYIN execution is shown in Fig. 9.5. If A in ASCII keyboard is pressed as shown in part ① of Fig. 9.5. key data is written to key buffer as shown in part ② of Fig. 9.5.

Specifications Notes:

Program Module Name: Receive Key Data	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: KEYIN

Description:



Note: **: hexadecimal

Fig. 9.5. Example of Program Module KEYIN Execution

(3) KEYIN calls neither the program modules nor subroutines.

2. User Notes

(1) Both KEYIN and KEYOUT must use the same roll buffer.

(2) The following procedure must be performed before KEYIN execution.

(a) Selects Port Mode register as $\overline{INT_1}$.

(b) Clears $\overline{INT_1}$ interrupt mask bit.

(c) Sets bit IE to enable $\overline{INT_1}$ interrupt.

3. RAM Allocation

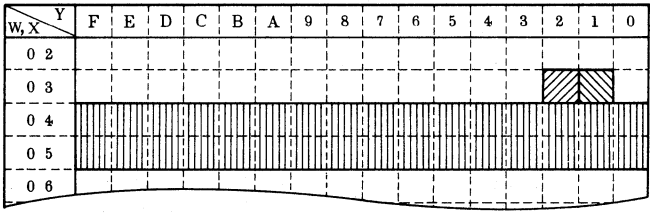
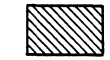
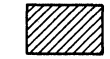
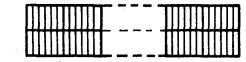


Fig. 9.6. RAM Allocation

Program Module Name:	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: KEYIN
Receive Key Data		

Description:

Label	RAM	Description
PE	b3 b0  MD($031)	Ending pointer indicating key data is set to end of key buffer
PS	b3 b0  MD($032)	Starting pointer indicating start address of unprocessed key data in key buffer
—	MD($04F) MD($040)  MD($05F) MD($050)	Key buffer to which 16 bytes key data will be set

4. Sample Application

```
        ⋮
LMID    $0, PS  }
LMID    $0, PE  }  ..... Clear RAM to be used
LMID    $8, PMR  ..... Initialize INT1
REMD    IM1      ..... Clear INT1 interrupt mask
SEMD    IE       ..... Enable interrupts
        ⋮
```

5. Basic Operation

(1) Input/output to/from roll buffer.

(a) Calls program module KEYIN at every $\overline{INT_1}$ interrupt and stores key data in key buffer. Then, calls program module KEYOUT in main program and fetches key data from key buffer.

(b) Clears starting pointer PS(RAM) and ending pointer PE(RAM) and stores key data in key buffer starting address.

Program Module Name:	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: KEYIN
Receive Key Data		

Description:

(c) Program module KEYIN stores 1 byte of key data in 16-byte buffer area pointed by PE(RAM), and increments PE(RAM).

(d) Program module KEYOUT fetches 1 byte from 16-byte buffer area pointed by PS(RAM) and increments PS(RAM).

(e) PS(RAM) and PE(RAM) become "0" if they are incremented till 15 bytes because the buffer area is 16-byte long.

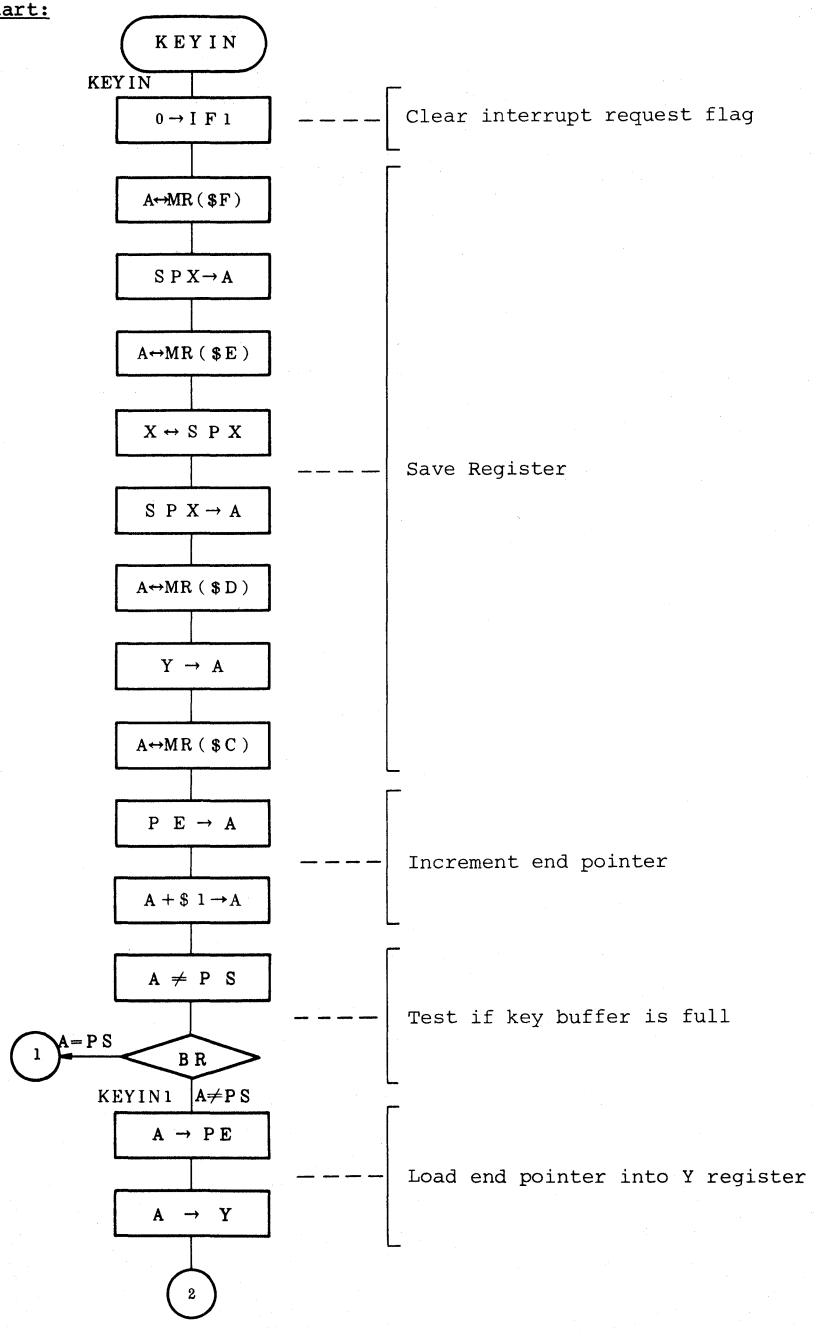
(2) Input to key buffer

(a) Program module KEYIN loads PE(RAM) into Accumulator and increments Accumulator. Then, compares Accumulator content with PS(RAM). If (Accumulator)=PS(RAM), key data is not stored in key buffer.
If (Accumulator) $\neq$ (PS), key data is stored in key buffer and PE(RAM) is incremented.

(b) Key buffer can be used up to 15 bytes.

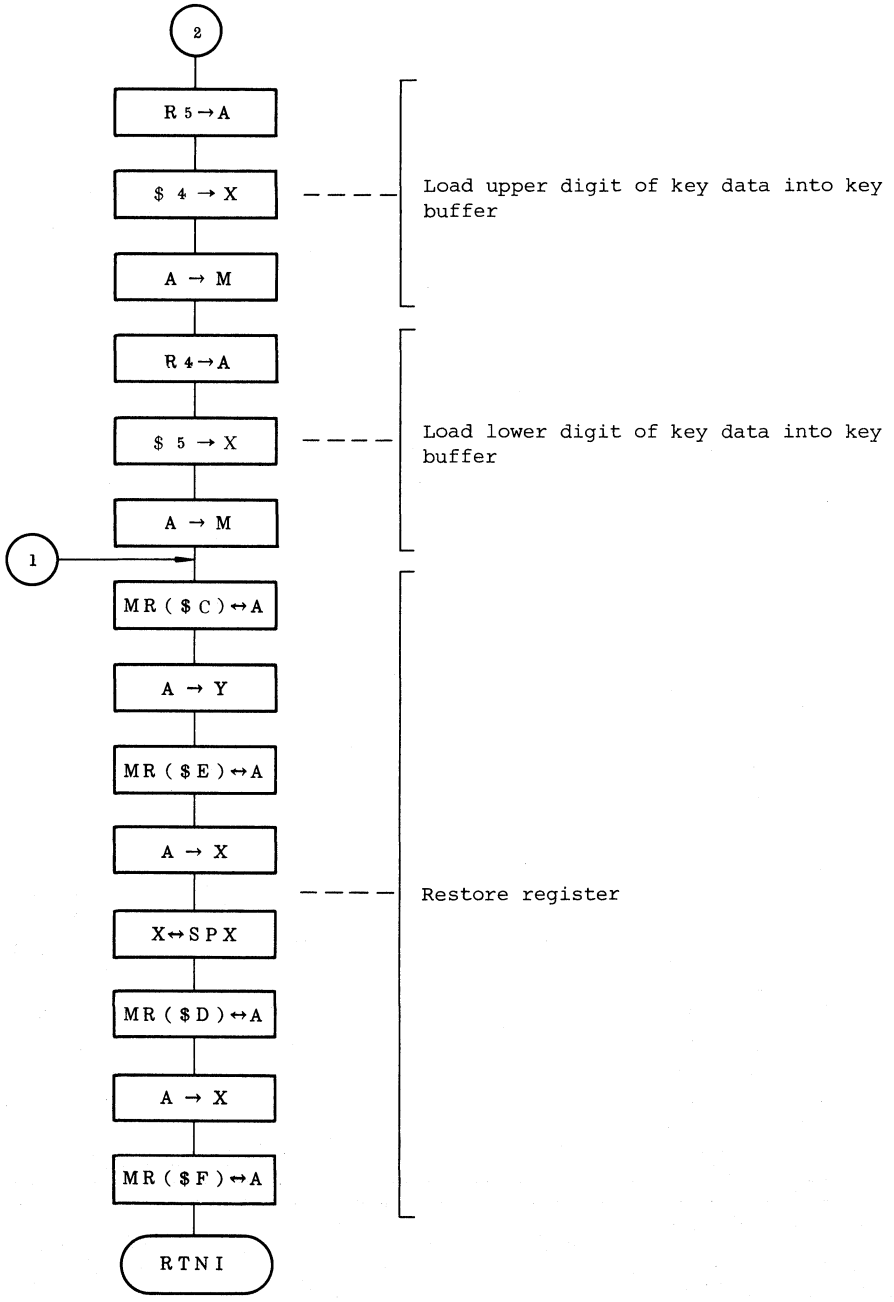
Program Module Name: Receive Key Data	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: KEYIN

Flowchart:



Program Module Name:	MCU:	Label:
Receive Key Data	HMCS402C/ HMCS404C/HMCS408C	KEYIN

Flowchart:



Program Module Name: Read Key Data	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: KEYOUT

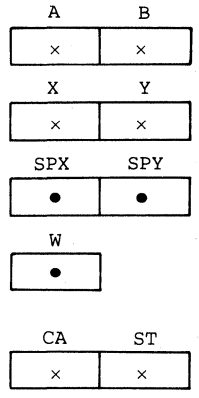
Function:

Reads key data from key buffer.

Arguments:

Contents		1 digit = 4 bits Storage Location	No. of Digits
Entry			
Returns	Unprocessed key data	A, B	2
	Unprocessed key data existence	ST (STATUS)	1 bit

Changes in CPU Registers and Flags:



● : Not Affected
× : Undefined
↕ : Result

Specifications:

1 word = 10 bits

ROM (Words): 15

RAM (Digits): 2

Stack (Digits): 4

No. of cycles: 17

Reentrant: No

Relocatable: No

Interrupt OK?: Yes

Description:

1. Function Details

(1) Argument details

A : Loads upper digit of unprocessed key data into key buffer.

B : Loads lower digit of unprocessed key data into key buffer.

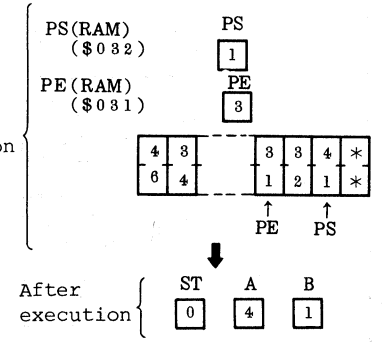


Fig. 9.7. Example of Program Module KEYOUT Execution

Specifications Notes:

Program Module Name: Read Key Data	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: KEYOUT

Description:

ST: Indicates existence of unprocessed key data in key buffer.

ST=0: Unprocessed data is in key buffer.

ST=1: No unprocessed data is in key buffer.

(2) Example of program module KEYOUT execution is shown in Fig. 9.7. After indicating program module KEYOUT, load unprocessed key data into A, B RAM.

(3) KEYOUT calls neither the program module nor subroutine.

2. User Notes

Both KEYIN and KEYOUT must be executed in pair.

3. RAM Allocation

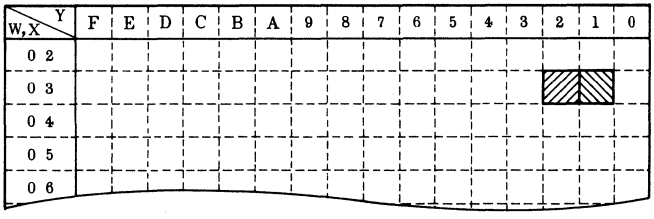


Fig. 9.8. RAM Allocation

Label	RAM	Description
PE	b3 b0 MD($031)	Ending pointer indicating key data is set to end of key buffer
PS	b3 b0 MD($032)	Starting pointer indicating start address of unprocessed key data in key buffer

Program Module Name: Read Key Data	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: KEYOUT

Description:

4. Sample Application

```
                    ⋮
                LMID    0, PS   }
                LMID    0, PE   }  ..... Clears RAM to be used
                LMID    8, PMR  ..... Initialize INT1
                REMD    IM1     ..... Clears INT1 interrupt mask
                SEMD    IE      ..... Enable interrupt
KEMN1           CALL    KEYOUT  }
                BR      KEYMN1  }  ..... Call program module KEYOUT
                LYA             }
                XSPY            }
                LAB             }
                LYA             }  ..... Store unprocessed data in return
                LASPY           }        argument into RAM area
                LBA             }
                LAY             }
                    ⋮
```

5. Basic Operation

(1) Input/output to/from key buffer.
See (1) of 5. Basic operation in program module KEYIN for details.

(2) Output from key buffer.

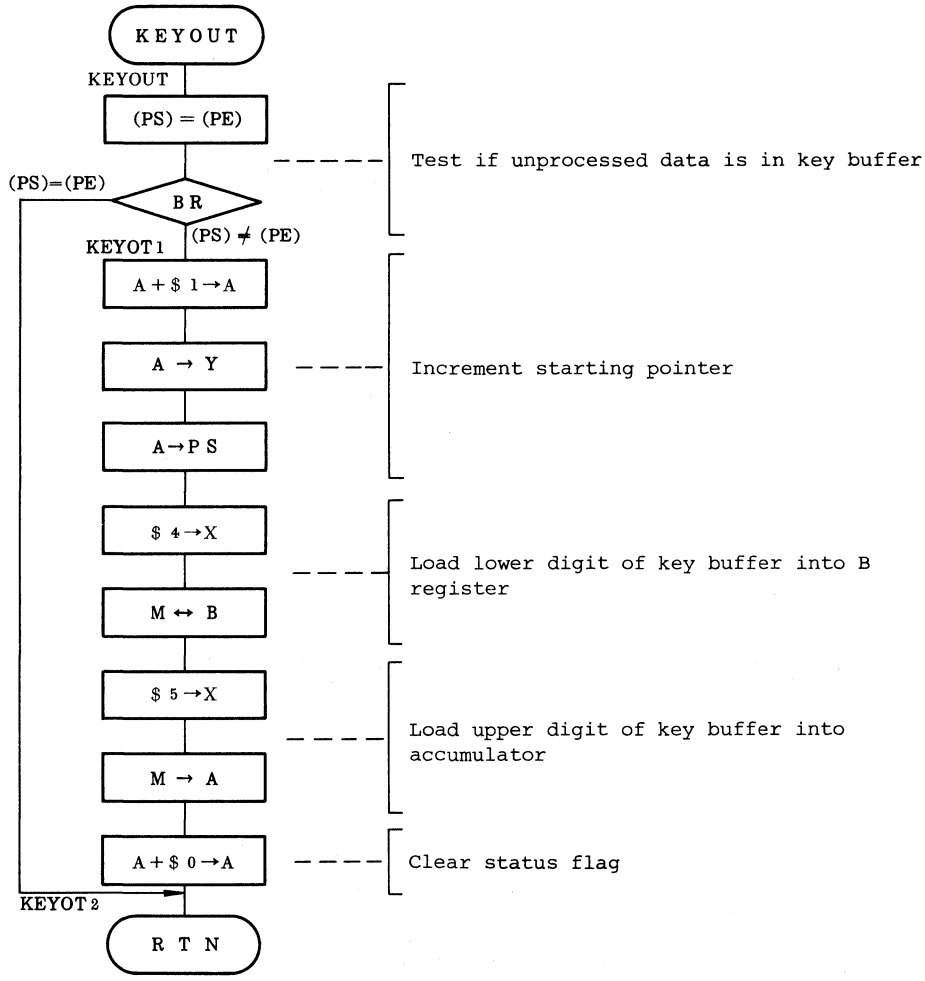
(a) Program module KEYOUT tests if valves in starting pointer PS(RAM) and values in ending pointer PE(RAM) are equal.

(b) In case of PS(RAM)=PE(RAM), no key data is in key buffer and bit C is set.

(c) In case of PS(RAM)≠PE(RAM), key data is fetched from key buffer pointed by PS(RAM), and PS(RAM) is incremented.

Program Module Name: Read Key Data	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: KEYOUT

Flowchart:



9.4 SUBROUTINE DESCRIPTION

This application example calls no subroutines.

9.5 PROGRAM LISTING

```
ST-NO   OBJECT    ADRS   SOURCE STATEMENTS

00001   1DF       0000           LLEN     132
00002                            TITLE    UTILIZING COMMERCIAL KEYBOARDS
00003                  *
00004                  ****   RAM ALLOCATION   ********************************
00005                  *
00006                  PE       EQU      $031          ROLL BUFFER END POINTER
00007                  PS       EQU      $032          ROLL BUFFER START POINTER
00008                  *
00009                  ****   SYMBOL DEFINITIONS   ***************************
00010                  *
00011                  PMR      EQU      $004          PORT MODE REGISTER
00012                  IF1      EQU      0.$001        INTERRUPT REQUEST FLAG
00013                  IM1      EQU      1.$001        INTERRUPT MODE FLAG
00014                  IE       EQU      0.$000        INTERRUPT ENABLE
00015                  ********************************************************
00016                  *                                                      *
00017                  *               VECTOR ADDRESSES                       *
00018                  *                                                      *
00019                  ********************************************************
00020                  *
00021                           ORG      $0000
00022                  *
00023   150 010   0000         JMPL     KEYMN         RESET
00024   150 010   0002         JMPL     KEYMN         INT0
00025   150 028   0004         JMPL     KEYIN         INT1
00026   150 010   0006         JMPL     KEYMN         TIMER-A
00027   150 010   0008         JMPL     KEYMN         TIMER-B
00028                  ********************************************************
00029                  *                                                      *
00030                  *               MAIN PROGRAM : KEYMN                   *
00031                  *                                                      *
00032                  ********************************************************
00033                  *
00034                           ORG      $0010
00035                  *
00036   0F0       0010 KEYMN    LWI      $0            INITIALIZE W REGISTER
00037   1A0 032   0011         LMID     0.PS          CLEAR RAM
00038   1A0 031   0013         LMID     0.PE
00039   1A8 004   0015         LMID     8.PMR         SELECT INT1
00040   189 001   0017         REMD     IM1           CLEAR INTERRUPT MASK
00041   184 000   0019         SEMD     IE            ENABLE INTERRUPTS
00042   160 04B   001B KEYMN1  CALL     KEYOUT        READ BUFFER DATA
00043   31B       001D         BR       KEYMN1        TEST IF READ
00044   0D8       001E         LYA                    EXCHANGE A AND B
00045   002       001F         XSPY
00046   048       0020         LAB
00047   0D8       0021         LYA
00048   058       0022         LASPY
00049   0C8       0023         LBA
00050   0AF       0024         LAY
00051   160 05B   0025         CALL     TPR           CONVERT ASCII LOWERCASE INTO UPPERCASE
00052   327       0027 PEND     BR       PEND
00053                  ********************************************************
00054                  *                                                      *
00055                  *         NAME : KEYIN (RECEIVE KEY DATA)              *
00056                  *                                                      *
00057                  ********************************************************
00058                  *                                                      *
00059                  *        ENTRY : NOTHING                               *
00060                  *      RETURNS : NOTHING                               *
00061                  *                                                      *
00062                  ********************************************************
00063   188 001   0028 KEYIN    REMD     IF1           CLEAR INTERRUPT REQUEST FLAG
00064   2FF       002A         XMRA     $F            SAVE REGISTER
00065   068       002B         LASPX
00066   2FE       002C         XMRA     $E
00067   001       002D         XSPX
00068   068       002E         LASPX
00069   2FD       002F         XMRA     $D
00070   0AF       0030         LAY
00071   2FC       0031         XMRA     $C
00072   190 031   0032         LAMD     PE            INCREMENT END POINTER
00073   281       0034         AI       $1
00074   104 032   0035         ANEMD    PS            TEST IF KEY BUFFER IS FULL
00075   339       0037         BR       KEYIN1
```

```
00076   342         0038            BR      KEYIN2
00077   194 031     0039  KEYIN1    LMAD    PE              LOAD END POINTER INTO Y REGISTER
00078   0D8         003B            LYA
00079   255         003C            LAR     $5              LOAD UPPER DIGIT OF KEY DATA INTO KEY BUFFER
00080   224         003D            LXI     $4
00081   094         003E            LMA
00082   254         003F            LAR     $4              LOAD LOWER DIGIT OF KEY DATA INTO KEY BUFFER
00083   225         0040            LXI     $5
00084   094         0041            LMA
00085   2FC         0042  KEYIN2    XMRA    $C              RESTORE REGISTERS
00086   0D8         0043            LYA
00087   2FE         0044            XMRA    $E
00088   0E8         0045            LXA
00089   001         0046            XSPX
00090   2FD         0047            XMRA    $D
00091   0E8         0048            LXA
00092   2FF         0049            XMRA    $F
00093   011         004A            RTNI
00094                       ************************************************************
00095                       *                                                          *
00096                       *          NAME : KEYOUT (READ KEY DATA)                   *
00097                       *                                                          *
00098                       ************************************************************
00099                       *                                                          *
00100                       *         ENTRY : NOTHING                                  *
00101                       *       RETURNS : ACCUMULATOR B REGISTER (KEY DATA)        *
00102                       *                 STATUS (ST=0;TRUE,ST=1;FALSE)            *
00103                       *                                                          *
00104                       ************************************************************
00105   190 032     004B  KEYOUT    LAMD    PS              START POINTER = END POINTER ?
00106   104 031     004D            ANEMD   PE
00107   351         004F            BR      KEYOT1
00108   35A         0050            BR      KEYOT2
00109   281         0051  KEYOT1    AI      $1              INCREMENT START POINTER
00110   0D8         0052            LYA
00111   194 032     0053            LMAD    PS
00112   224         0055            LXI     $4              LOAD LOWER DIGIT OF KEY BUFFER INTO B REG
00113   0C0         0056            XMB
00114   225         0057            LXI     $5              LOAD UPPER DIGIT OF KEY BUFFER INTO ACCA
00115   090         0058            LAM
00116   280         0059            AI      $0              CLEAR STATUS FLAG
00117   010         005A  KEYOT2    RTN
00118                       ************************************************************
00119                       *                                                          *
00120                       *          NAME : TPR (CONVERT ASCII LOWERCASE INTO        *
00121                       *                      UPPERCASE)                          *
00122                       *                                                          *
00123                       ************************************************************
00124                       *                                                          *
00125                       *         ENTRY : A (UPPER 4 BITS OF ASCII LOWERCASE)      *
00126                       *                 B (LOWER 4 BITS OF ASCII LOWERCASE)      *
00127                       *       RETURNS : A (UPPER 4 BITS OF ASCII UPPERCASE)      *
00128                       *                 B (LOWER 4 BITS OF ASCII UPPERCASE)      *
00129                       *                                                          *
00130                       ************************************************************
00131   0D8         005B  TPR       LYA                     LOAD UPPER 4 BITS OF ASCII LOWERCASE
00132   048         005C            LAB                     LOAD LOWER 4 BITS OF ASCII LOWERCASE
00133   076         005D            YNEI    $6              UPPER 4 BITS OF ASCII LOWERCASE=$6 ?
00134   364         005E            BRS     TPR3            BRANCH IF UPPER 4 BITS OF ASCII LOWERCASE =/$6
00135   2B0         005F            ALEI    $0              LOWER 4 BITS OF ASCII LOWERCASE =<$0 ?
00136   363         0060            BRS     TPR2            BRANCH IF LOWER 4 BITS OF ASCII LOWERCASE =<$0
00137   0AF         0061  TPR1      LAY                     LOAD A FROM Y
00138   28E         0062            AI      $E              CONVERT LOWERCASE INTO UPPERCASE
00139   010         0063  TPR2      RTN
00140   077         0064  TPR3      YNEI    $7              UPPER ASCII =$7 ?
00141   363         0065            BRS     TPR2            BRANCH IF UPPER 4 BITS OF ASCII LOWERCASE
00142   2BA         0066            ALEI    $A              LOWER 4 BITS OF ASCII LOWERCASE =<$A ?
00143   361         0067            BRS     TPR1            BRANCH IF A = < $A
00144   363         0068            BRS     TPR2            BRANCH IF A > $A
00145                       *
00146                               END
```

10.1 HARDWARE DESCRIPTION

10.1.1 Function

(1) Receives ASCII sent from master as clock synchronous serial data using the HMCS404C, converts the received data from ASCII lowercase into uppercase and sends it to the master system.

(2) Converts ASCII lowercase into uppercase, if lowercase is received.

(3) Uses protocol in which data is sent from master system first.

10.1.2 Microcomputer Application

(1) Transfers data to/from master system using clocked synchronous serial communication interface (hereinafter, SCI).

(2) Transfers data by sending master receives request to master system and receiving transfer clock from master system using port D.

(3) Outputs port D_0 slave receives request signal and informs data transfer to slave system.

10.1.3 Circuit Diagram

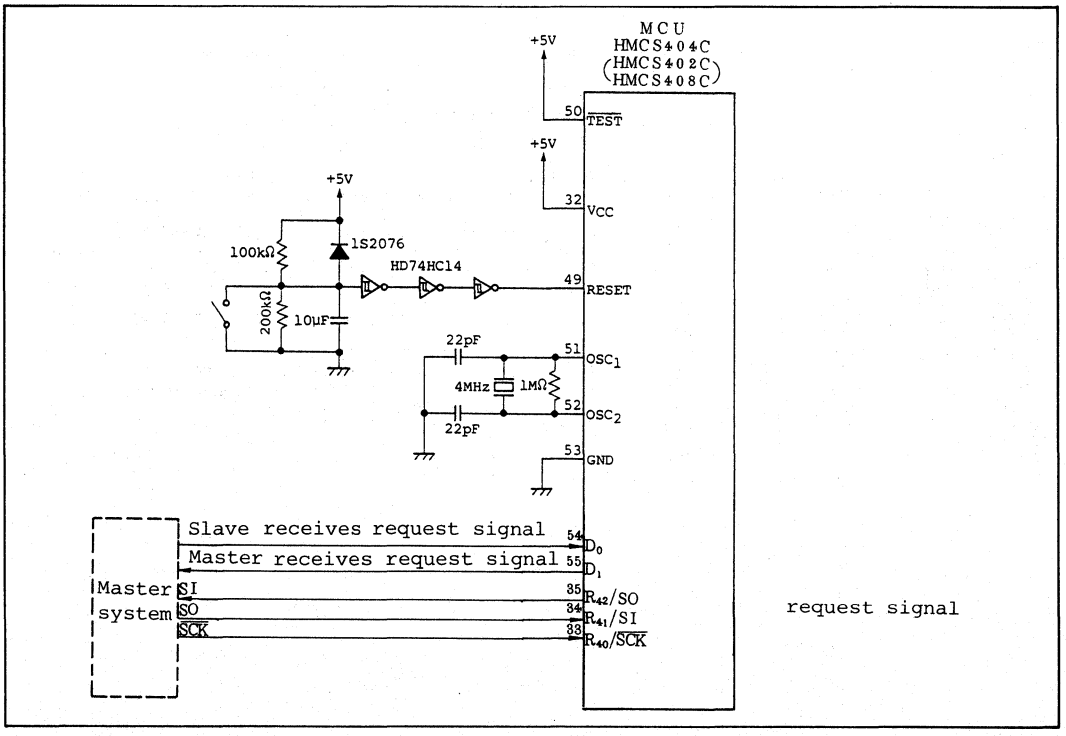


Fig. 10.1. SCI Serial Communication

10.1.4 Pin Functions

Pin functions at the interface between the HMCS404C SCI pins and master system pins.

Table 10.1. Pin Functions

Pin Name (HMSC404C)	Input/ Output	Active Level (High or Low)	Function	Pin Name (Master / System)
$R_{40}/\overline{SCK}$	Input	Low	Inputs transfer clock when receiving/sending serial data	Serial clock
R_{41}/SI	Input	—	Inputs serial data	Serial data output
R_{42}/SO	Output	—	Outputs serial data	Serial data input
D_1	Output	Low	Requests transfer clock to output to master system	Master receives request signal
D_0	Input	Low	Inputs transfer clock request from master system and output clock if low	Slave receives request signal

10.1.5 Hardware Operation

SCI timing chart is shown in Fig. 10.2.

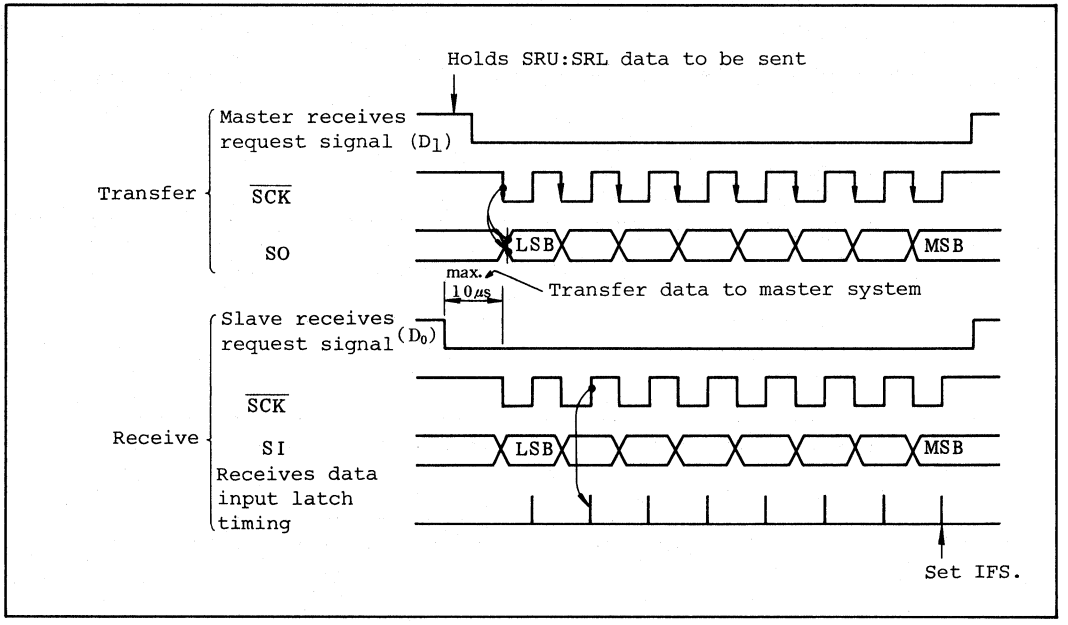


Fig. 10.2. SCI Timing Chart

10.2 SOFTWARE DESCRIPTION

10.2.1 Program Module Configuration

The program module configuration for SCI communication with master system is shown in Fig. 10.3.

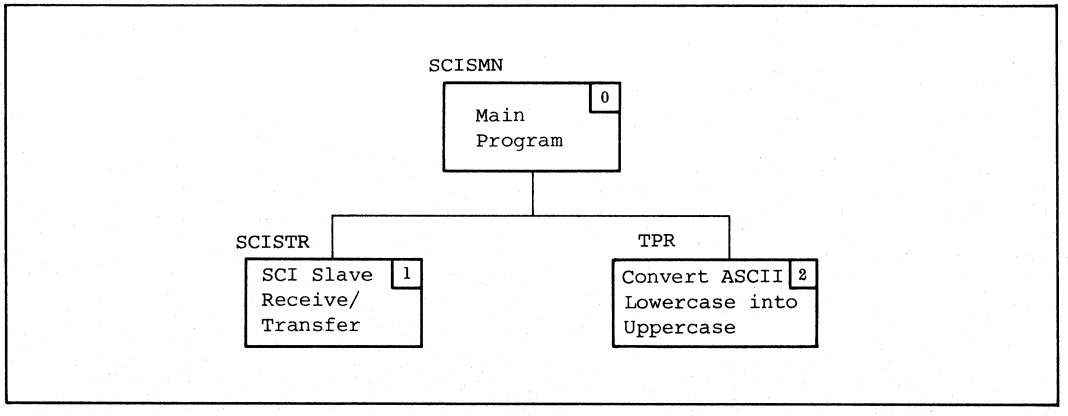


Fig. 10.3. Program Module Configuration

10.2.2 Program Module Functions

Program module functions are summarized in Table 10.2.

Table 10.2. Program Module Functions

No.	Program Module Name	Label	Function
0	Main Program	SCISMN	Communicates with master system using clocked synchronous interface SCI
1	SCI Slave Receive/ Transfer	SCISTR	Receives data from master system using external clock
2	Convert ASCII Lowercase into Uppercase	TPR	Converts ASCII lowercase into uppercase. (Refer to TPR in HMCS400 Series Application Note (Software Edition) for details)

10.2.3 Program Module Application (Main Program)

Flowchart in Fig. 10.3 is an example of receiving ASCII from master system, converting ASCII lowercase into uppercase and sending it to master system, performed by the program module in Fig. 10.3.

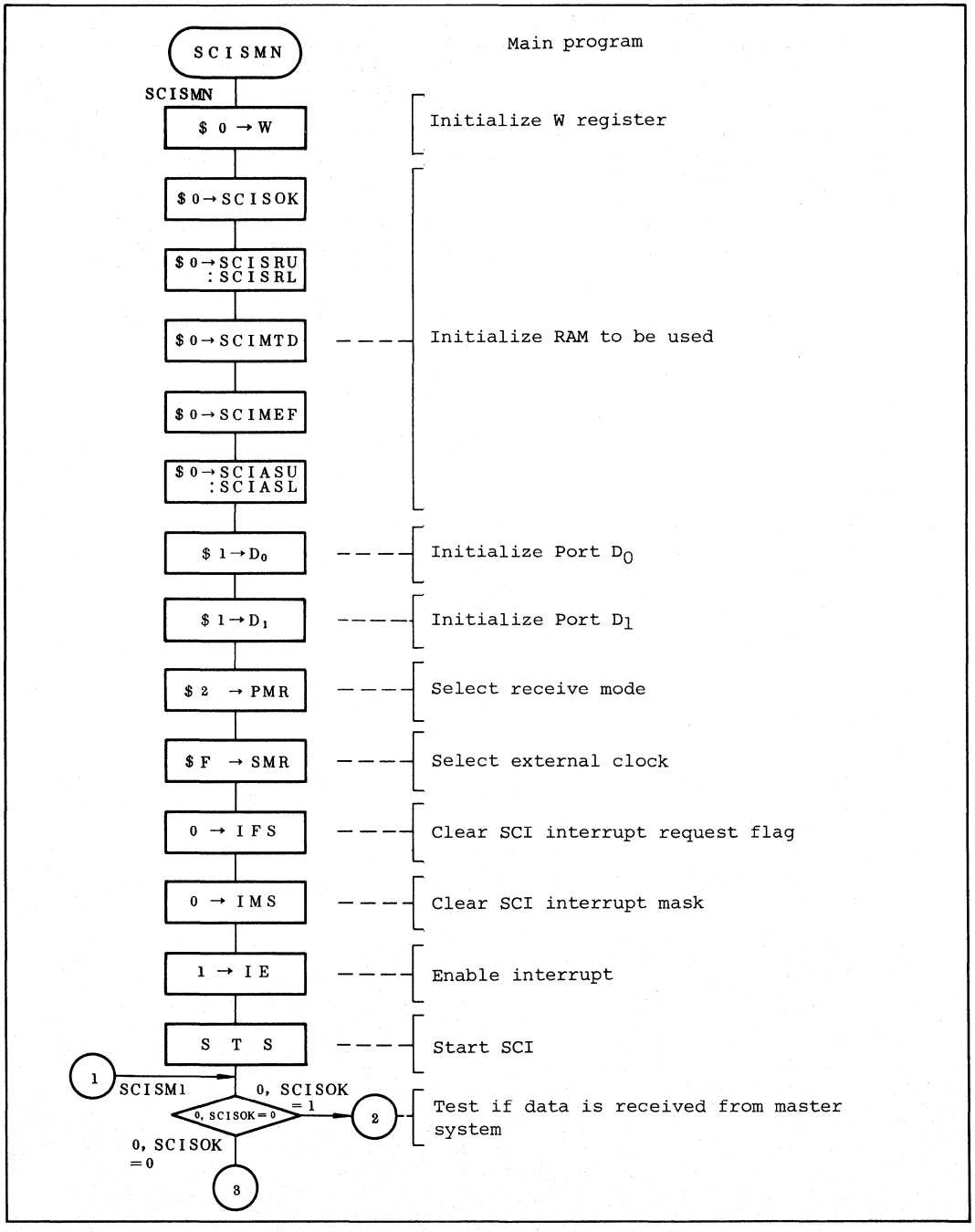


Fig. 10.4. Program Module Flowchart

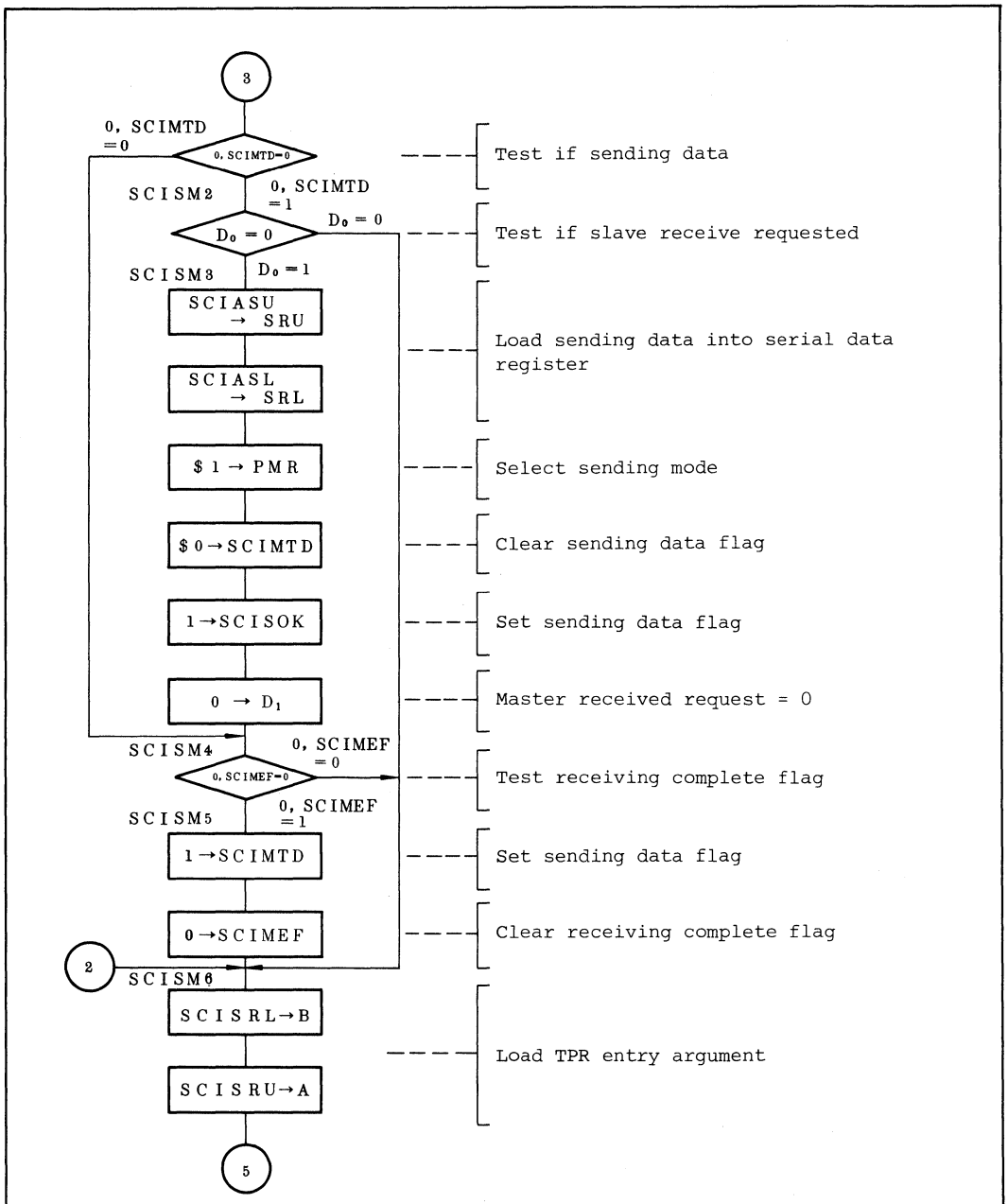


Fig. 10.4. Program Module Flowchart (Cont)

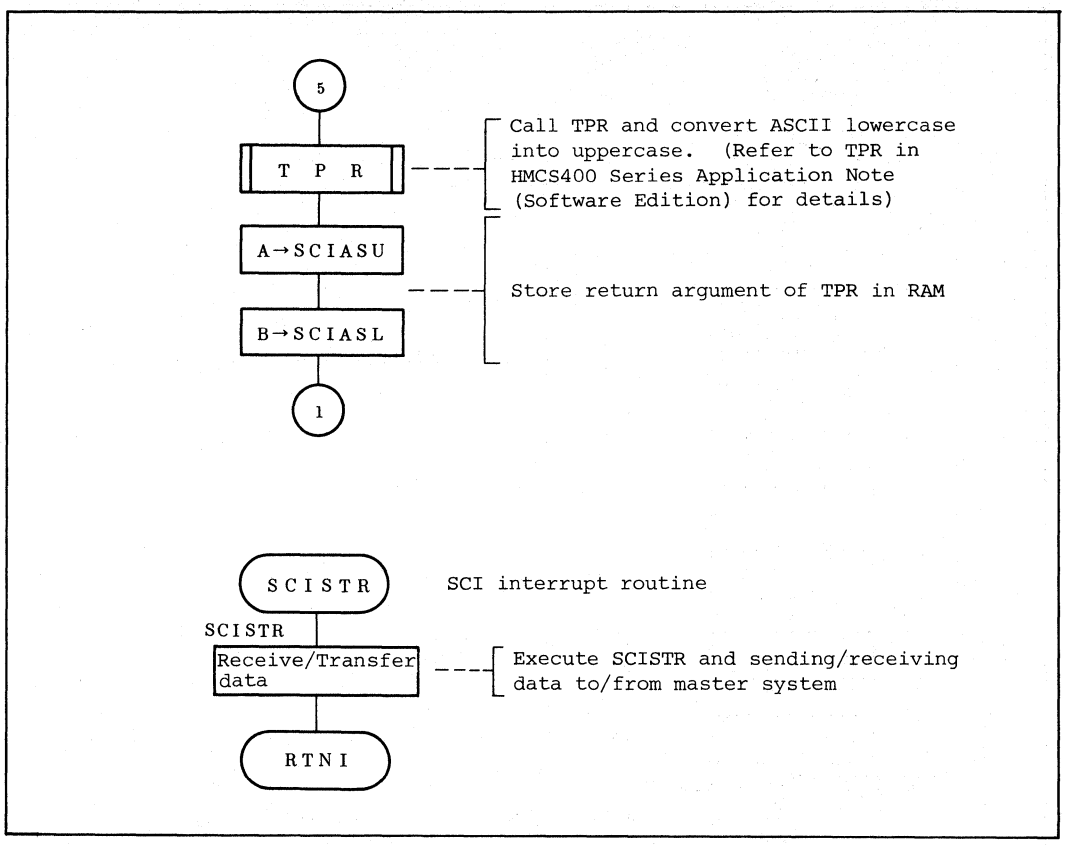


Fig. 10.4. Program Module Flowchart (Cont)

Program Module Name: SCI SLAVE TRANSFER/RECEIVE	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: SCISTR

Function:

(1) Input serial clock and receives data from master system.

(2) Permits outputting when slave system cannot output serial clock.

Arguments:

	Contents	1 digit = 4 bits Storage Location	No. of Digits
Entry	——	——	——
Returns	Received data	SCISRV (RAM)	1
		SCISRL (RAM)	1
	End of received data	SCIMEF (RAM)	1

Changes in CPU Registers and Flags:

A	B
×	●

X	Y
●	●

SPX	SPY
●	●

W
●

CA	ST
●	×

● : Not Affected
× : Undefined
↕ : Result

Specifications:

1 word = 10 bits

ROM (Words): 25

RAM (Digits): 4

Stack (Digits): 0

No. of cycles: 20

Reentrant: No

Relocatable: No

Interrupt OK?: Yes

Description:

1. Function Details

(1) Argument details

SCISRV (RAM): Contains data
SCISRL sent from master system.

SCIMEF (RAM): Indicates existence of received data.

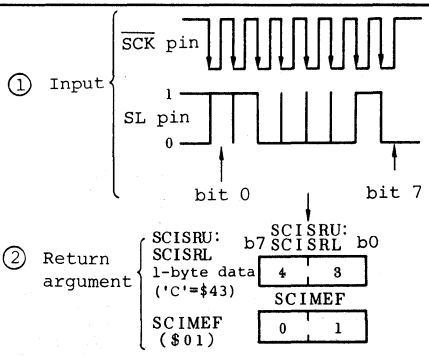


Fig. 10.5. Example of SCISRD Execution

Specifications Notes: (1) "No. of cycles" in "SPECIFICATIONS" represents the number of cycles are needed when having no wait time for receiving data.
(2) Reset interrupt request flag with SOFTWARE in interrupt routine.

Program Module Name: SCI SLAVE TRANSFER/RECEIVE	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: SCISTR

Description:

SCIMEF (RAM)=1: Data is received from master system.

SCIMEF (RAM)=0: No data is received from master system.

(2) SCISTR execution stores contents of SCI data register in SCISRU(RAM) and SCISRL(RAM).

(3) SCISTR calls neither program modules nor subroutines.

2. User Notes

(1) When program module SCISTR is used, resetting system (in case of power on reset, supplying power) should be performed from master system.

(2) Program module SCISTR should be called before master system begins to send data.

3. RAM Allocation

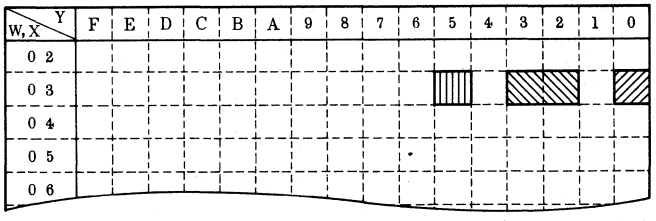


Fig. 10.6. RAM Allocation

Label	RAM	Description
SCISOK	b3 b0 MD($030)	Flag indicating if data is received from master system
SCISRU:SCISRL	b7 b0 MD($033,$032)	Store data sent from master system
SCIMEF	b3 b0 MD($035)	Flag indicating existence of receives data

Program Module Name: SCI SLAVE TRANSFER/RECEIVE	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: SCISTR

Description:

4. Sample Application

```
                ⋮
          SEDD    $1            ..... Set Port D1 to High
          LMID    $2, PMR       ..... Select received mode
          LMID    $F, SMR       ..... Load clock source into external clock
          REMD    IFS           ..... Clear SCI interrupt request flag
          REMD    IMS           ..... Clear SCI interrupt mask
          SEMD    IE            ..... Enable interrupt
          TMD     $0, SCIMEF    ┐
          BR      LOOP1         ├ ..... Test if receiving completed
          BR      LOOP2         ┘
LOOP1     LMID    $0, SCIMEF    ..... Clear flag indicating receive
LOOP2     LAMD    SCISRL        ┐       completion
          LBA                   ├ ..... Load receives data into Accumulator
          LAMD    SCISRU        ┘       and B register
                ⋮
```

5. Basic Operation

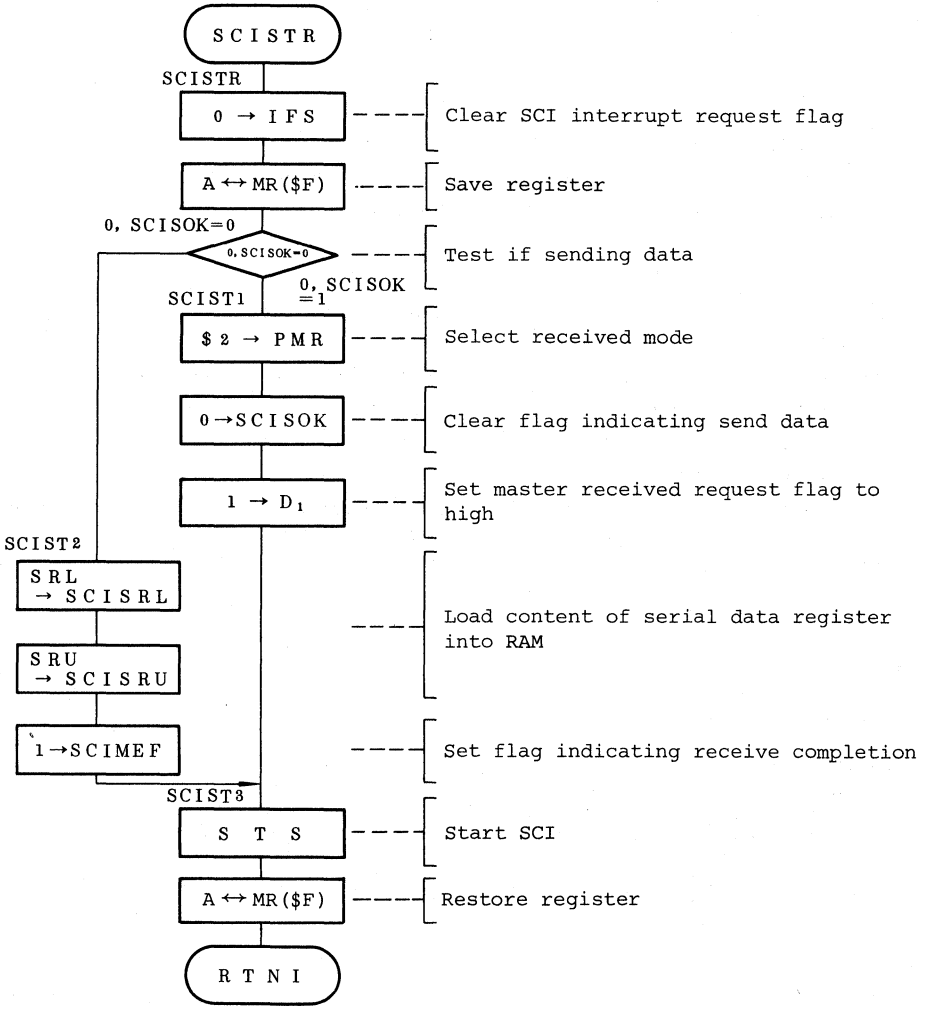
(1) Receives serial data using SCI interrupt routine.

(2) In case of receiving complete, read receives data and set flag indicating receive completion.
In case of sending complete, select receives mode and clear flag indicating send data. Set master receives request to high.

(3) Serial interface movement is started by STS instruction.

Program Module Name: SCI SLAVE TRANSFER/RECEIVE	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: SCISTR

Flowchart:



10.4 SUBROUTINE DESCRIPTION

This application example calls no subroutines.

10.5 PROGRAM LISTING

```
ST-NO   OBJECT   ADRS   SOURCE STATEMENTS

00001   363      0000           LLEN     132
00002                           TITLE    CLOCKED SYNCHRONOUS SCI (EXTERNAL CLOCK)
00003                  *
00004                  ****   RAM ALLOCATION   ****************************************
00005                  *
00006                  SCISOK   EQU      $030          FLAG FOR SENDING DATA
00007                  SCISRU   EQU      $032          UPPER RECEIVED DATA
00008                  SCISRL   EQU      $033          LOWER RECEIVED DATA
00009                  SCIMTD   EQU      $034          SENDING DATA FLAG
00010                  SCIMEF   EQU      $035          RECEIVING DATA COMPLETE FALG
00011                  SCIASU   EQU      $036          UPPER OUTPUT DATA
00012                  SCIASL   EQU      $037          LOWER OUTPUT DATA
00013                  *
00014                  ****   SYMBOL DEFINITIONS   ************************************
00015                  *
00016                  PMR      EQU      $004          PORT MODE REGISTER
00017                  SMR      EQU      $005          SERIAL MODE REGISTER
00018                  SRL      EQU      $006          LOWER SERIAL DATA REGISTER
00019                  SRU      EQU      $007          UPPER SERIAL DATA REGISTER
00020                  IMS      EQU      1,$003        INTERRUPT MASK OF SERIAL
00021                  IFS      EQU      0,$003        INTERRUPT REQUEST FLAG
00022                  IE       EQU      0,$000        INTERRUPT ENABLE FLAG
00023                  ************************************************************
00024                  *                                                          *
00025                  *               VECTOR ADDRESSES                           *
00026                  *                                                          *
00027                  ************************************************************
00028                  *
00029                           ORG      $0000
00030                  *
00031   150 010  0000           JMPL     SCISMN        RESET
00032   150 010  0002           JMPL     SCISMN        INT0
00033   150 010  0004           JMPL     SCISMN        INT1
00034   150 010  0006           JMPL     SCISMN        TIMER-A
00035   150 010  0008           JMPL     SCISMN        TIMER-B
00036   000      000A           NOP
00037   000      000B           NOP
00038   150 058  000C           JMPL     SCISTR        SERIAL
00039                  ************************************************************
00040                  *                                                          *
00041                  *               MAIN PROGURAM : SCISMN                     *
00042                  *                                                          *
00043                  ************************************************************
00044                  *
00045                           ORG      $0010
00046                  *
00047   0F0      0010   SCISMN   LWI      $0            INITIALIZE W REGISTER
00048   1A0 030  0011           LMID     $0,SCISOK     INITIALIZE RAM
00049   1A0 032  0013           LMID     $0,SCISRU
00050   1A0 033  0015           LMID     $0,SCISRL
00051   1A0 034  0017           LMID     $0,SCIMTD
00052   1A0 035  0019           LMID     $0,SCIMEF
00053   1A0 036  001B           LMID     $0,SCIASU
00054   1A0 037  001D           LMID     $0,SCIASL
00055   2E0      001F           SEDD     $0            INITIALIZE D PORT
00056   2E1      0020           SEDD     $1
00057   1A2 004  0021           LMID     $2,PMR        INITIALIZE PMR
00058   1AF 005  0023           LMID     $F,SMR        INITIALIZE SMR
00059   188 003  0025           REMD     IFS           INITIALIZE IFS
00060   189 003  0027           REMD     IMS           INITIALIZE IMS
00061   184 000  0029           SEMD     IE            ENABLE INTERRUPT
00062   148      002B           STS                    SCI START
00063   18C 030  002C   SCISM1  TMD      $0,SCISOK     TEST IF SENDING DATA
00064   34D      002E           BR       SCISM6
00065   18C 034  002F           TMD      $0,SCIMTD     TEST IF SCIMTD
00066   333      0031           BR       SCISM2
00067   345      0032           BR       SCISM4
00068   2A0      0033   SCISM2  TDD      $0            TEST IF SLAVE RECEIVE REQUESTED
00069   336      0034           BR       SCISM3
00070   34D      0035           BR       SCISM6
00071   190 036  0036   SCISM3  LAMD     SCIASU        LOAD SENDING DATA INTO SERIAL DATA REGISTER
00072   194 007  0038           LMAD     SRU
00073   190 037  003A           LAMD     SCIASL
00074   194 006  003C           LMAD     SRL
```

```
00075   1A1 004   003E              LMID     $1,PMR          SELECT SENDING MODE
00076   1A0 034   0040              LMID     $0,SCIMTD       CLEAR SENDING DATA FLAG
00077   1A1 030   0042              LMID     $1,SCISOK       SET SENDING DATA FLAG
00078   261       0044              REDD     $1              CLEAR MASTER RECEIVE REQUEST
00079   18C 035   0045   SCISM4     TMD      $0,SCIMEF       TEST RECEIVING COMPLETE FLAG
00080   349       0047              BR       SCISM5
00081   34D       0048              BR       SCISM6
00082   1A1 034   0049   SCISM5     LMID     $1,SCIMTD       SET SENDING DATA FLAG
00083   1A0 035   004B              LMID     $0,SCIMEF       CLEAR RECEIVING COMPLETE FLAG
00084   190 033   004D   SCISM6     LAMD     SCISRL          LOAD TPR ENTRY ARGUMENT
00085   0C8       004F              LBA
00086   190 032   0050              LAMD     SCISRU
00087   160 076   0052              CALL     TPR             CONVERT LOWERCASE INTO UPPERCASE
00088   194 036   0054              LMAD     SCIASU          STORE TPR RETURN ARGUMENT IN RAM
00089   048       0056              LAB
00090   194 037   0057              LMAD     SCIASL
00091   150 02C   0059              JMPL     SCISM1
00092                     ********************************************************************
00093                     *                                                                  *
00094                     *       NAME : SCISTR                                              *
00095                     *                                                                  *
00096                     ********************************************************************
00097                     *                                                                  *
00098                     *      ENTRY : NOTHING                                             *
00099                     *    RETURNS : SCISRU (UPPER RECEIVED DATA)                        *
00100                     *              SCIRDU (LOWER RECEIVED DATA)                        *
00101                     *              SCIMEF (SCIMEF=1;TRUE,SCIMEF=0;FALSE)               *
00102                     *                                                                  *
00103                     ********************************************************************
00104   188 003   005B   SCISTR    REMD     IFS             CLEAR SERIAL INTERRUPT REQUEST FLAG
00105   2FF       005D              XMRA     $F              SAVE A REGISTER
00106   18C 030   005E              TMD      $0,SCISOK       TEST IF SENDING DATA
00107   362       0060              BR       SCIST1
00108   369       0061              BR       SCIST2
00109   1A2 004   0062   SCIST1    LMID     $2,PMR          SELECT RECEIVING MODE
00110   1A0 030   0064              LMID     $0,SCISOK       CLEAR SENDING DATA FLAG
00111   2E1       0066              SEDD     $1              SET MASTER RECEIVE REQUEST FLAG
00112   150 073   0067              JMPL     SCIST3
00113   190 006   0069   SCIST2    LAMD     SRL             STORE RECEIVED DATA IN RAM
00114   194 033   006B              LMAD     SCISRL
00115   190 007   006D              LAMD     SRU
00116   194 032   006F              LMAD     SCISRU
00117   1A1 035   0071              LMID     $1,SCIMEF       SET RECEIVING COMPLETE FLAG
00118   148       0073   SCIST3    STS                      SCI START
00119   2FF       0074              XMRA     $F              RESTORE A REGISTER
00120   011       0075              RTNI
00121                     ********************************************************************
00122                     *                                                                  *
00123                     *       NAME : TPR (COVERT ASCII LOWERCASE INTO UPPERCASE)         *
00124                     *                                                                  *
00125                     ********************************************************************
00126   0D8       0076   TPR       LYA                      LOAD UPPER 4 BIT OF ASCII LOWERCASE
00127   048       0077              LAB                      LOAD LOWER 4 BIT OF ASCII LOWERCASE
00128   076       0078              YNEI     $6              UPPER 4 BIT OF ASCII LOWERCASE = $6 ?
00129   37F       0079              BRS      TPR3            BRANCH IF UPPER 4 BIT OF ASCII LOWERCASE
00130   2B0       007A              ALEI     $0              LOWER 4 BIT OF ASCII LOWERCASE =< $0 ?
00131   37E       007B              BRS      TPR2            BRANCH IF LOWER 4 BIT OF ASCII LOWERCASE
00132   0AF       007C   TPR1      LAY                      LOAD A FROM Y
00133   28E       007D              AI       $E              CONVERT LOWERCASE INTO UPPERCASE
00134   010       007E   TPR2      RTN
00135   077       007F   TPR3      YNEI     $7              UPPER ASCII = $7 ?
00136   37E       0080              BRS      TPR2            BRANCH IF UPPER 4 BIT OF ASCII LOWERCASE
00137   2BA       0081              ALEI     $A              LOWER 4 BIT OF ASCII LOWERCASE =< $A ?
00138   37C       0082              BRS      TPR1            BRANCH IF A =< $A
00139   37E       0083              BRS      TPR2            BRANCH IF A > $A
00140                     *
00141                               END
```

SECTION 11. CLOCK SYNCHRONOUS SCI (INTERNAL CLOCK)

11.1 HARDWARE DESCRIPTION

11.1.1 Function

(1) Transfers clock synchronous serial data to slave system, and receives data from slave.

(2) Interfaces transfer clock output pin (hereinafter, $\overline{SCK}$), serial data input pin (hereinafter, SI) and serial data output pin (hereinafter, SO) of the HMCS404C to master pins with each pin of slave system.

(3) Handshakes using transfer/receive control signal to transfer or receive data.

11.1.2 Microcomputer Applications

(1) Transfers data to/from slave system using clock synchronous serial communication interface (hereinafter, SCI).

(2) Inputs master receives request signal port D_1 from slave system. Outputs transfer rate clock to slave system considering input state and receives data.

(3) Outputs slave receives request signal from port D_0 to inform data transfer to slave system.

11.1.3 Circuit Diagram

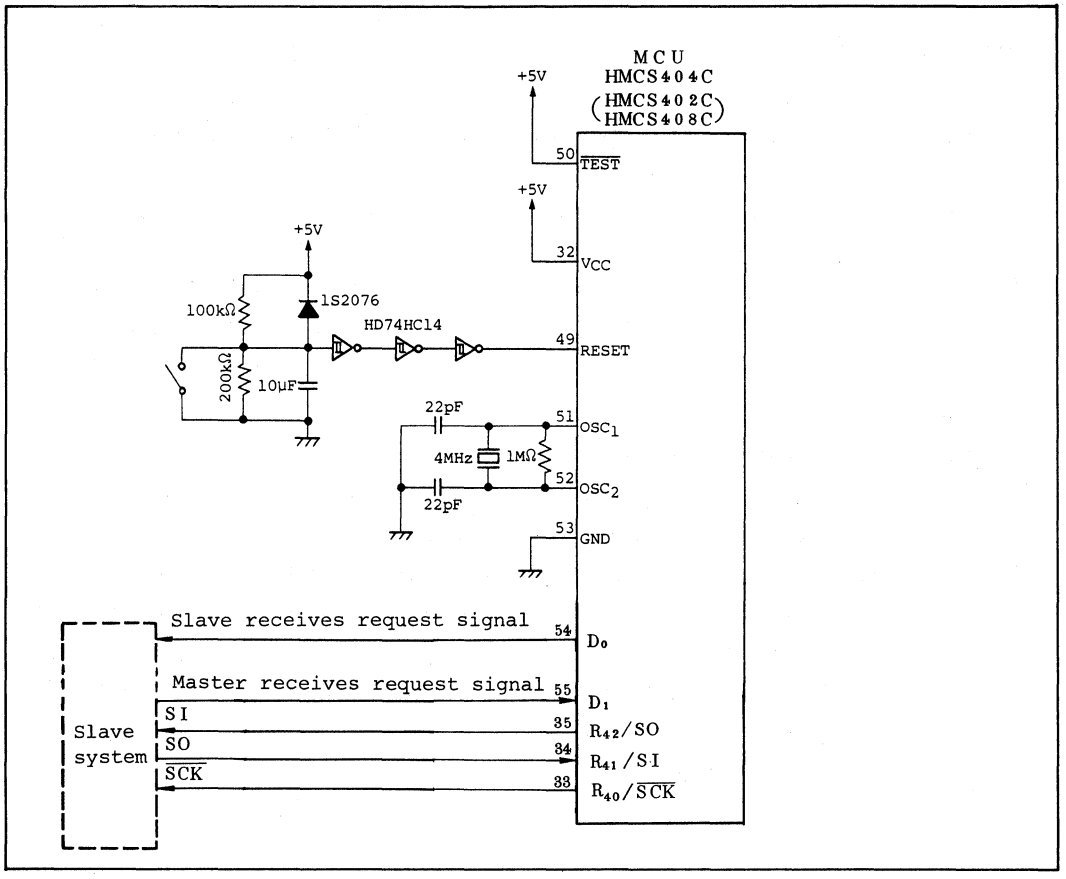


Fig. 11.1. Clock Synchronous SCI Circuit

11.1.4 Pin Functions

Pin functions at the interface between the HMCS404C SCI pins and slave system are shown in Table 11.1.

Table 11.1. Pin Functions

Pin Name (HMCS404C)	Input/ Output	Active Level (High or Low)	Function	Pin Name (Slave system)
$\overline{SCK}$	Output	Low	Outputs transfer clock	Serial clock
SI	Input	——	Receives data	Serial data output
SO	Output	——	Transfers data	Serial data input
D_1	Input	Low	Inputs transfer clock request from slave. Outputs clock if low	Master receives request signal
D_0	Output	Low	Informs transfer start to slave	Slave receives request signal

11.1.5 Hardware Operation

SCI timing chart is shown in **Fig. 11.2.**

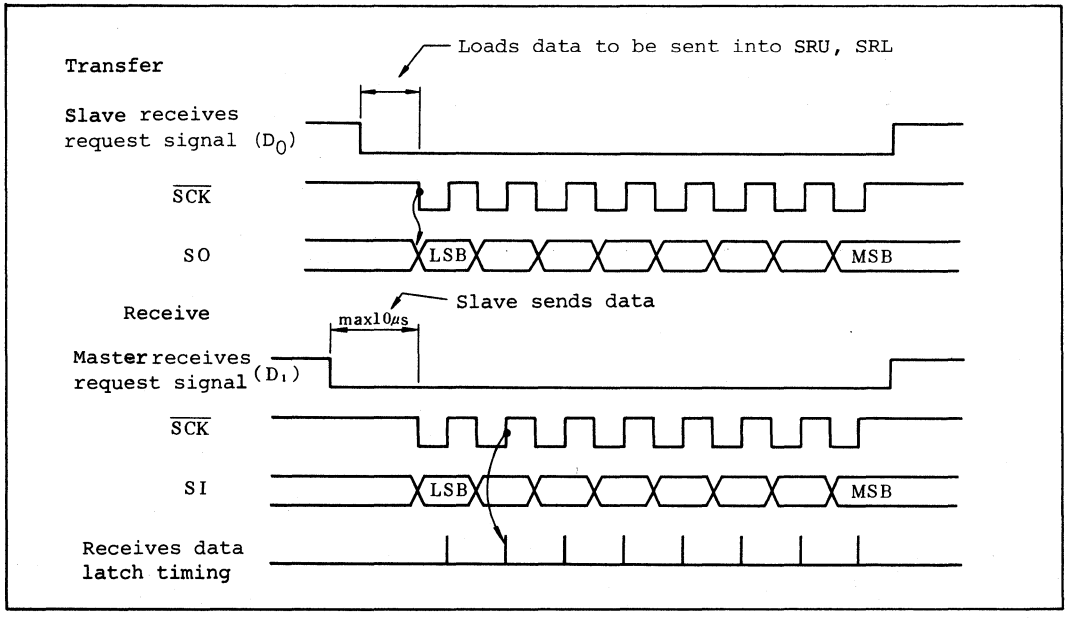


Fig. 11.2. SCI Timing Chart

11.2.1 Program Module Configuration

The program module configuration for serial communication with slave system is shown in Fig. 11.3.

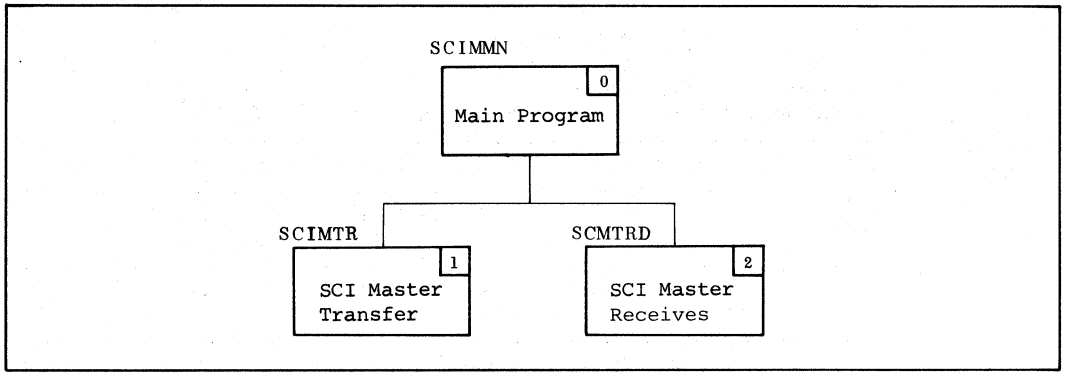


Fig. 11.3. Program Module Configuration

11.2.2 Program Module Functions

Program module functions are summarized in Table 11.2.

Table 11.2. Program Module Functions

No.	Program Module Name	Label	Function
0	Main Program	SCIMMN	Sends data to slave system and receive it from slave system without change by using the HMCS404C SCI
1	SCI Master Transfer	SCIMTR	Sends serial data to slave system using internal clock
2	SCI Master Receive	SCMTRD	Receives data from slave system using internal clock

11.2.3 Program Module Sample Application (Main Program)

The flowchart in **Fig. 11.4** is an example of sending serial data to slave system and receiving it from slave system, performed by the program modules in **Fig. 11.3.**

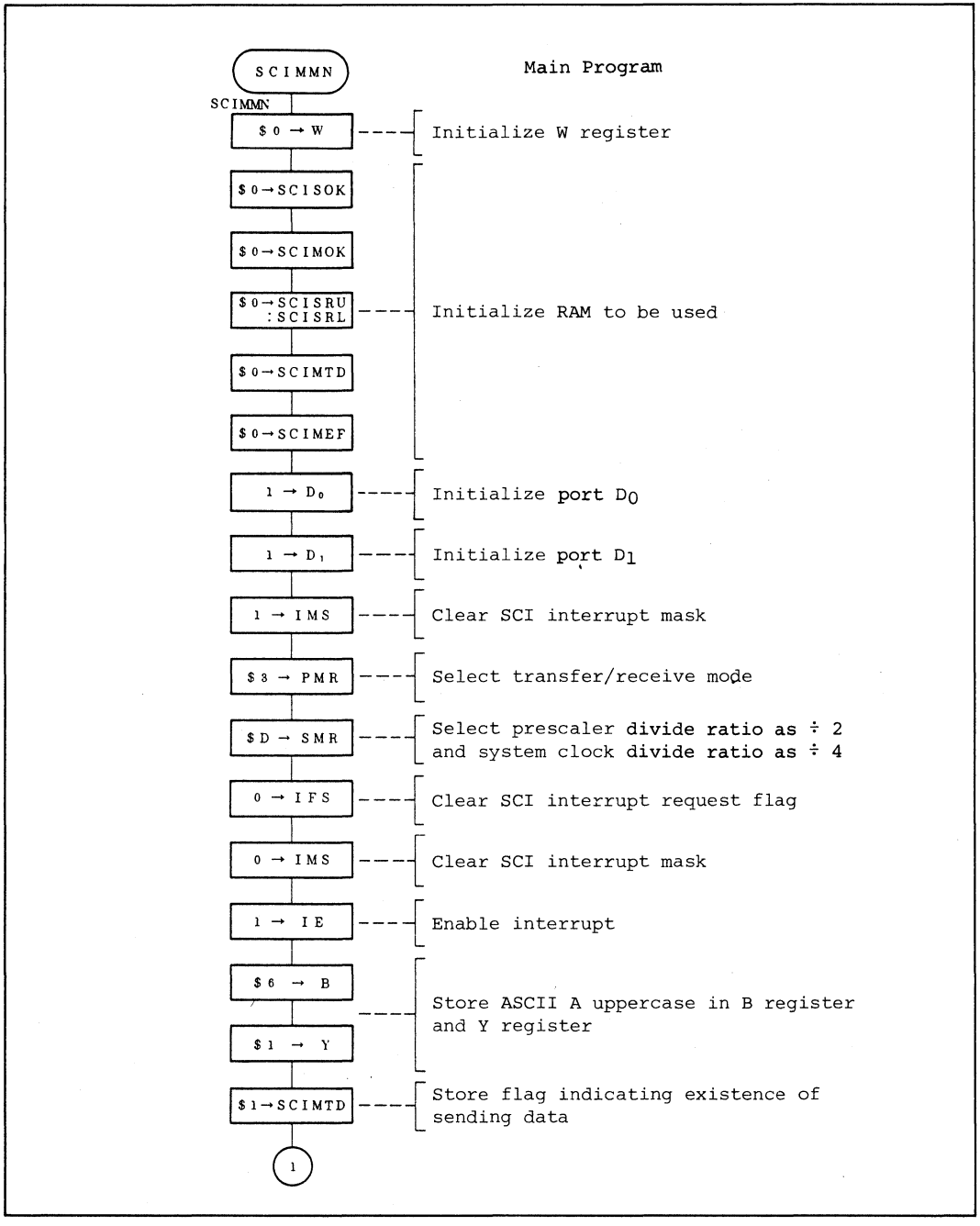


Fig. 11.4. Program Module Flowchart

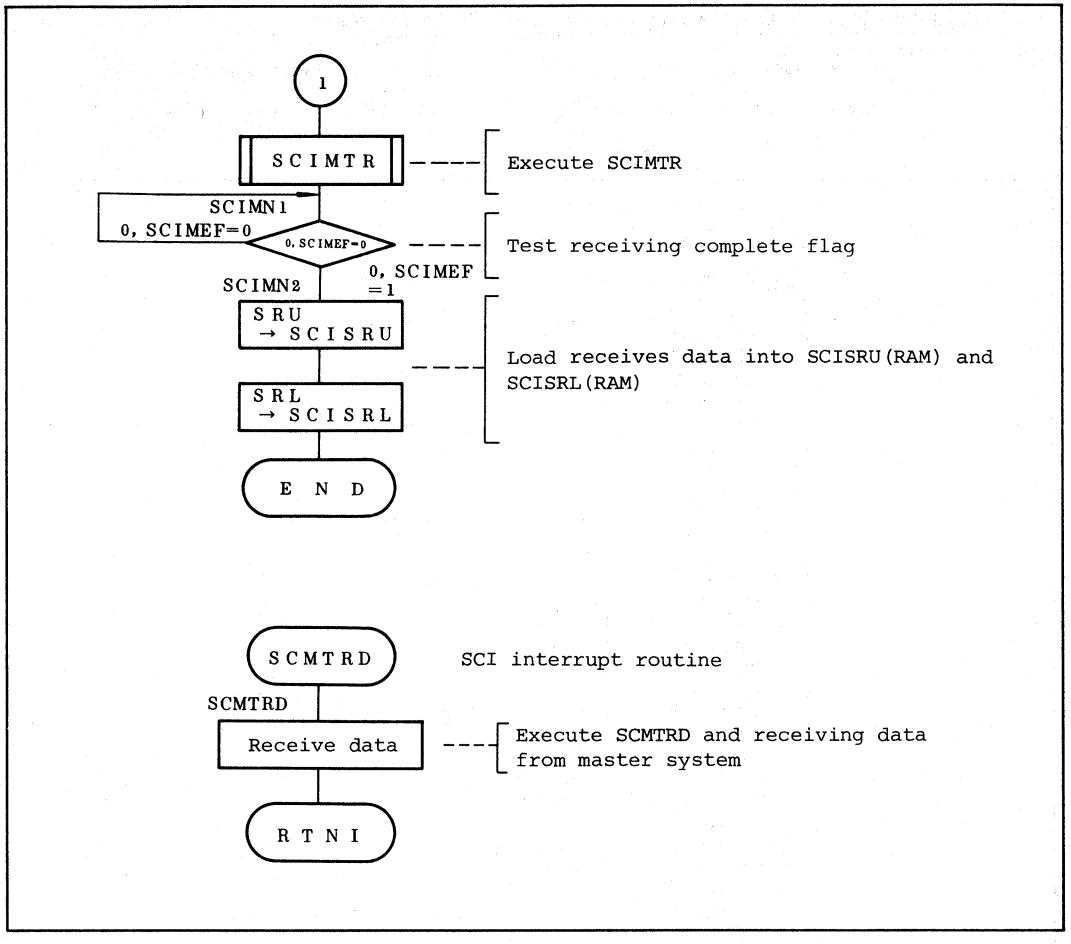


Fig. 11.4. Program Module Flowchart (cont.)

Program Module Name:	MCU:	Label:
SCI MASTER TRANSFER	HMCS402C/ HMCS404C/HMCS408C	SCIMTR

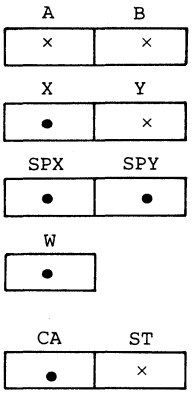
Function:

Sends data in B and Y register to slave system using internal clock.

Arguments:

Contents		1 digit = 4 bits Storage Location	No. of Digits
Entry	Data to be sent	B	1
	Data to be sent	Y	1
Returns	——	——	——

Changes in CPU Registers and Flags:



● : Not Affected
× : Undefined
↕ : Result

Specifications:

1 word = 10 bits
ROM (Words): 41
RAM (Digits): 3
Stack (Digits): 0
No. of cycles: 43
Reentrant: No
Relocatable: No
Interrupt OK?: Yes

Description:

1. Function Details
   (1) Argument details
   B register, Y register (RAM):
   Holds data to be sent to slave system.
   (2) Program module SCIMTR execution loads entry argument content into SRU, SRL and sends it to slave system.
   (3) SCIMTR calls neither program modules nor subroutines.

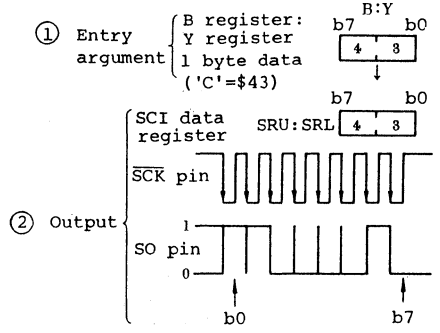


Fig. 11.5. Example of SCIMTR Execution

Specifications Notes:

Program Module Name: SCI MASTER TRANSFER	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: SCIMTR

Description:

2. User Notes

(1) Sets slave receives request signal to low and sets slave receiving state before program module SCIMTR execution.

(2) When using program module SCIMTR, resetting system (in case of power on reset, activating power) is performed master reset firstly, then reset slave system secondly.

3. RAM Allocation

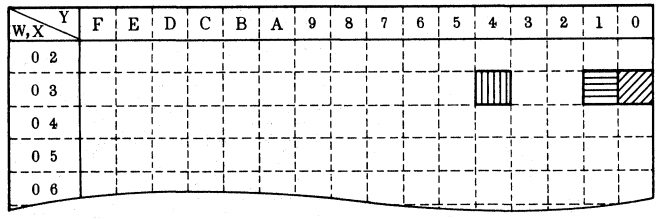


Fig. 11.6. RAM Allocation

Label	R A M	Description
S C I S O K	b3 b0 MD($030)	Flag indicating if sending data
S C I M O K	b3 b0 MD($031)	Flag indicating if receiving data
S C I M T D	b3 b0 MD($034)	Flag indicating existence of transfer data

Program Module Name:	SCI MASTER TRANSFER	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: SCIMTR

Description:

4. Sample Application

```
        ⋮
    SEDD    $0          } ..... Set port D0 and D1 high
    SEDD    $1          }
    LMID    $8, PMR     ..... Select transfer/receive mode
    LMID    $D, SMR     ..... Select internal clock source
    LBI     $6          }
    LYI     $1          } ..... Load data to be send into entry
                                argument
    CALL    SCIMTR      ..... Subroutine call program module SCIMTR
        ⋮
```

5. Basic Operation

(1) When slave receives request signal is output, master receives request signal may output from slave system with the same timing. Then, 10 μs software timer is executed and master receives request signal is tested after slave receives request signal is output.

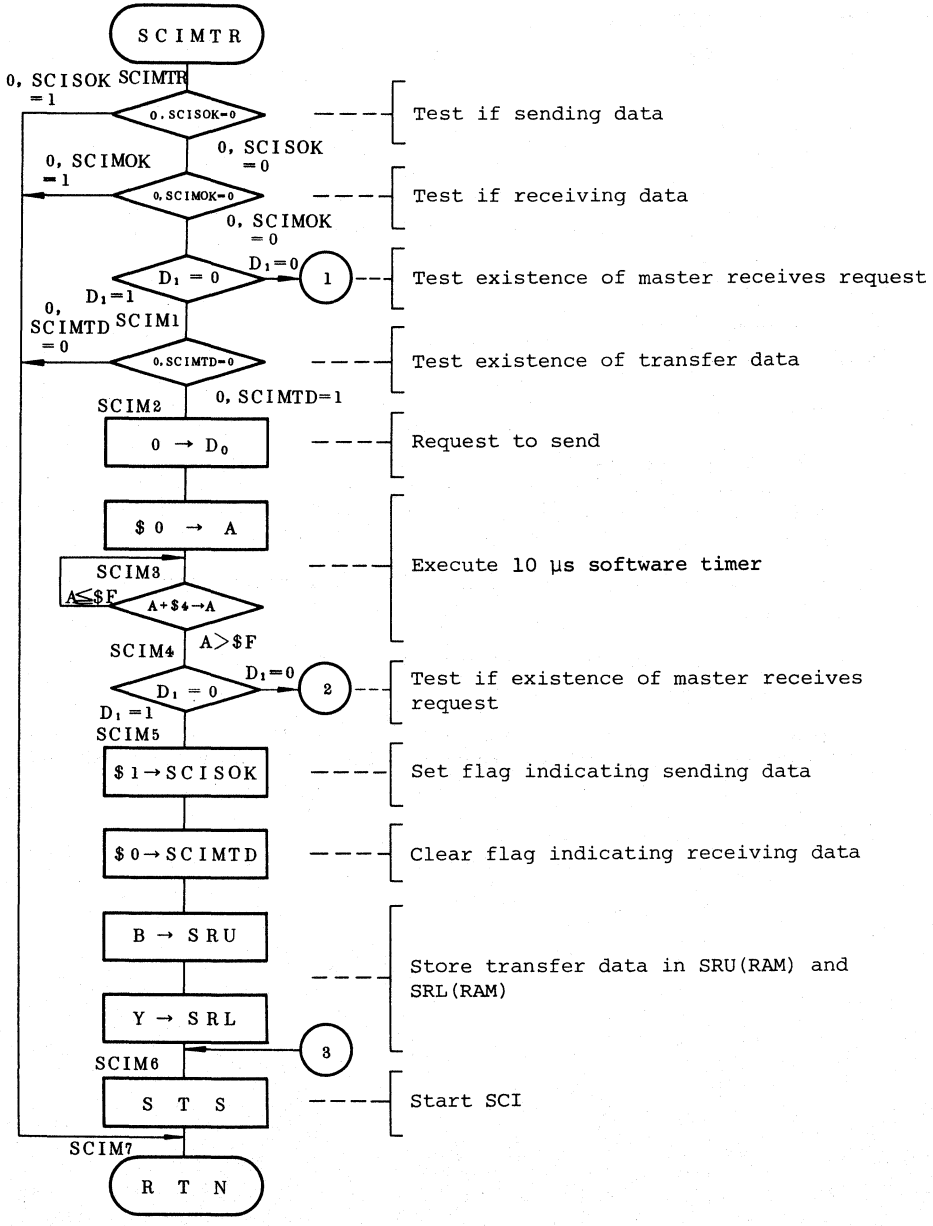
(2) Transfers receives serial data if master receives request signal is output.

(3) Goes to the next step after SCI interrupt request flag is set and transfer/receive completes.

(4) When serial data is received, retains transfer request in output state until main program processes receives data so that next data can not be received.

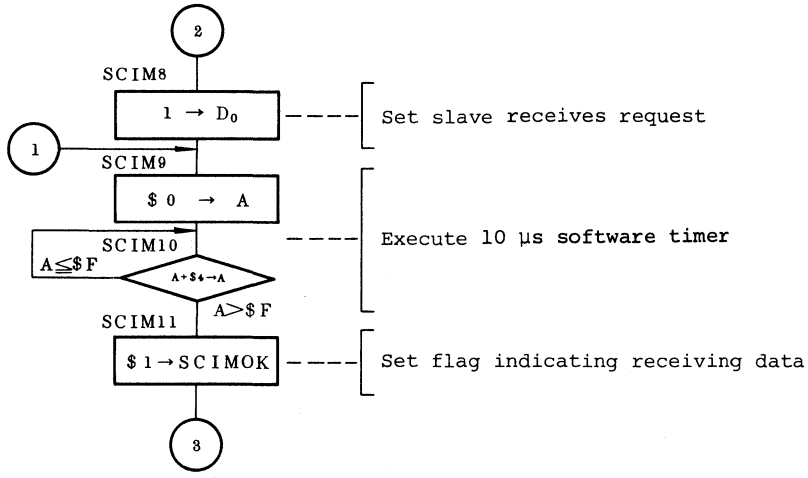
Program Module Name: SCI MASTER TRANSFER	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: SCIMTR

Flowchart:



Program Module Name: SCI MASTER TRANSFER	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: SCIMTR

Flowchart:



Program Module Name:	MCU:	Label:
SCI MASTER RECEIVE	HMCS402C/ HMCS404C/HMCS408C	SCMTRD

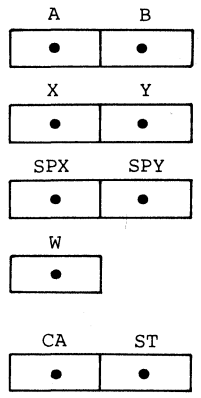
Function:

1. Outputs serial clock and receives data from slave system.
2. Permits outputting when slave system cannot output serial clock.

Arguments:

Contents		1 digit = 4 bits Storage Location	No. of Digits
Entry	—	—	—
Returns	Receives data	SCISRU (RAM)	1
		SCISRL (RAM)	1
	End of receives data	SCIMEF (RAM)	1

Changes in CPU Registers and Flags:



● : Not Affected
× : Undefined
↕ : Result

Specifications:

1 word = 10 bits

ROM (Words): 24
RAM (Digits): 5
Stack (Digits): 0
No. of cycles: 21
Reentrant: No
Relocatable: No
Interrupt OK?: Yes

Description:

1. Function Details

(1) Argument details
SCISRU,SCISRL(RAM):
Contains data received from slave.
SCIMEF(RAM): Indicates existence of receives data.
SCIMEF=0: No data is received from slave system.
SCIMEF=1: Data is received from slave system.

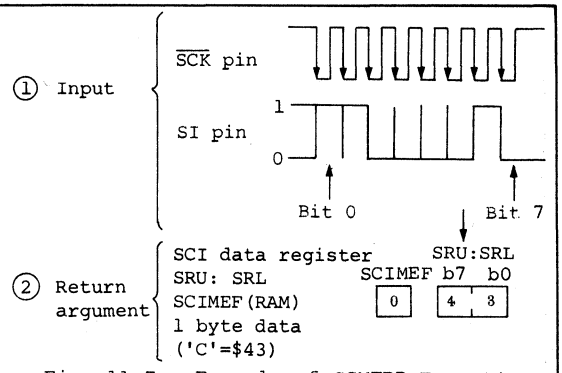


Fig. 11.7. Example of SCMTRD Execution

Specifications Notes:

1. "No. of cycles" in "Specifications" represents the number of cycles needed when having no wait time for receiving data.
2. Reset interrupt requests flag with software in interrupt routine.

Program Module Name: SCI MASTER RECEIVE	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: SCMTRD

Description:

(2) Program module SCMTRD execution contains SRU,SRL contents in return argument accumulator.

(3) SCMTRD calls neither program modules nor subroutines.

2. User Notes

(1) When program module SCMTRD is executed, set Port D_1 and Port D_2 to 1.

(2) Goes to transfer possible state in slave system before program module SCMTRD execution.

3. RAM Allocation

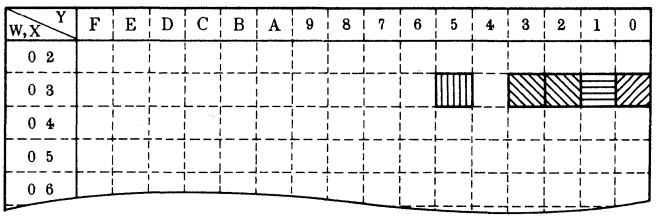


Fig. 11.8. RAM Allocation

Label	RAM	Description
SCISOK	b3 b0 MD($030)	Flag indicating if sending data
SCIMOK	b3 b0 MD($031)	Flag indicating if receiving data
SCISRU:SCISRL	b7 b0 MD($033,$032)	Stores data sent from slave system
SCIMEF	b3 b0 MD($035)	Flag indicating existence of receives data

Program Module Name: SCI MASTER RECEIVE	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: SCMTRD

Description:

4. Sample Application

```
                ⋮
            SEDD    $0          } ····· Set port D0 and port D1 high
            SEDD    $1          }
            LMID    $8, PMR     ····· Select transfer/receive mode
            LMID    $D, SMR     ····· Select transfer clock Ratio
            REMD    IFS         ····· Clear SCI interrupt request flag
            REMD    IMS         ····· Clear SCI interrupt mask
            SEMD    IE          ····· Enable interrupt
LOOP1       TMD     $0, SCIMEF  }
            BR      LOOP2       } ····· Test if receiving flag
            BR      LOOP1       }
LOOP2       LAMD    SRU         }
            LMAD    SCISRU      } ····· Load receives data into accumulator
            LAMD    SRL         }       A and B register
            LMAD    SCISRL      }
                ⋮
```

5. Basic Operation

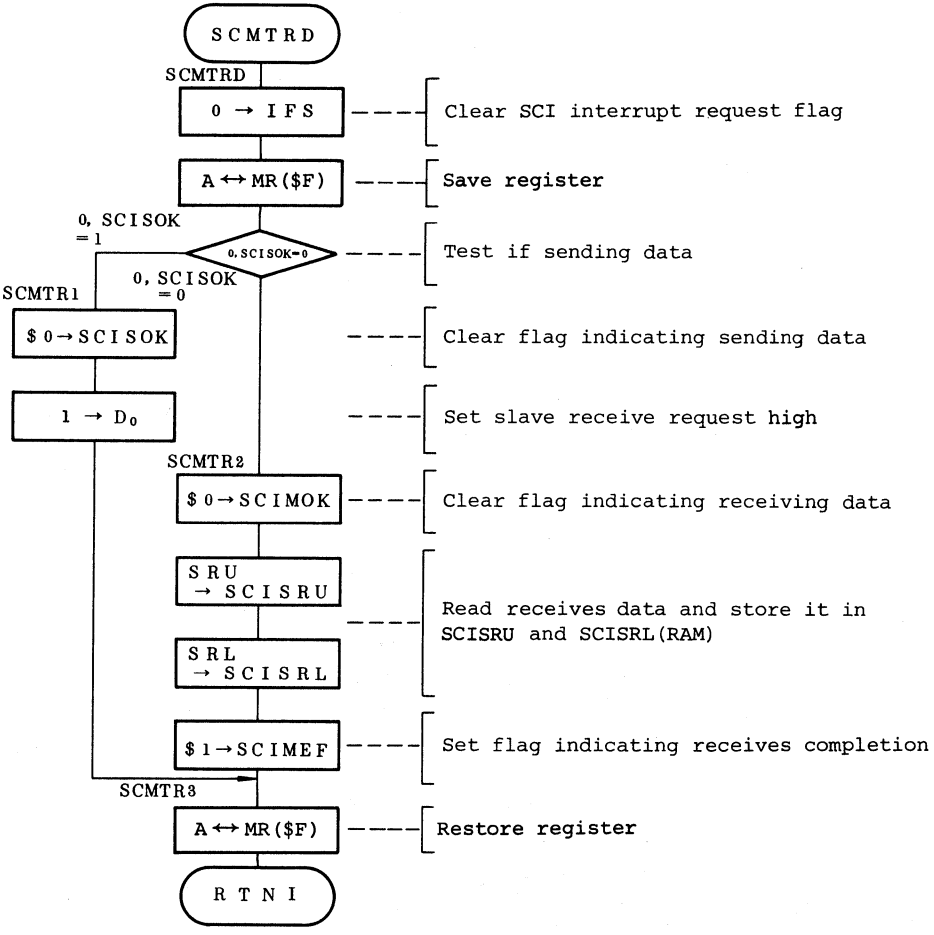
(1) Checks slave receives request signal to test if data is received after outputting the signal.

(2) Tests if master receives request signal has been output from slave system.

(3) If output, sets SCI interrupt request flag, and stores serial data in return argument after checking that serial data is received.

Program Module Name: SCI MASTER RECEIVE	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: SCMTRD

Flowchart:



11.4 SUBROUTINES

This application example calls no subroutines.

11.5 PROGRAM LISTING

```
ST-NO   OBJECT    ADRS   SOURCE STATEMENTS

00001   37E       0000           LLEN     132
00002                            TITLE    CLOCKED SYNCHRONOUS SCI (INTERNAL CLOCK)
00003                  *
00004                  ****   RAM ALLOCATION   ********************************
00005                  *
00006                  SCISOK   EQU      $030          FLAG FOR SENDING DATA
00007                  SCIMOK   EQU      $031          FLAG FOR RECEIVING DATA
00008                  SCISRU   EQU      $032          UPPER RECEIVER DATA
00009                  SCISRL   EQU      $033          LOWER RECEIVED DATA
00010                  SCIMTD   EQU      $034          SENDING DATA FLAG
00011                  SCIMEF   EQU      $035          RECEIVING DATA COMPLETE FLAG
00012                  *
00013                  ****   SYMBOL DEFINITIONS   ***************************
00014                  *
00015                  PMR      EQU      $004          PORT MODE REG.
00016                  SMR      EQU      $005          SERIAL MODE REG.
00017                  SRL      EQU      $006          LOWER SERIAL DATA REG.
00018                  SRU      EQU      $007          UPPER SERIAL DATA REG.
00019                  IMS      EQU      $1,$003       INTERRUPT MASK OF SERIAL
00020                  IFS      EQU      $0,$003       INTERRUPT REQUEST FLAG OF SERIAL
00021                  IE       EQU      $0,$000       INTERRUPT ENABLE FLAG
00022                  ************************************************************
00023                  *                                                          *
00024                  *              VECTOR ADDRESSES                            *
00025                  *                                                          *
00026                  ************************************************************
00027                  *
00028                           ORG      $0000
00029                  *
00030   150 010 0000           JMPL     SCIMMN        RESET
00031   150 010 0002           JMPL     SCIMMN        INTO
00032   150 010 0004           JMPL     SCIMMN        INT1
00033   150 010 0006           JMPL     SCIMMN        TIMER-A
00034   150 010 0008           JMPL     SCIMMN        TIMER-B
00035   000     000A           NOP
00036   000     000B           NOP
00037   150 068 000C           JMPL     SCMTRD        SERIAL
00038                  ************************************************************
00039                  *                                                          *
00040                  *              MAIN PROGURAM : SCIMMN                      *
00041                  *                                                          *
00042                  ************************************************************
00043                  *
00044                           ORG      $0010
00045                  *
00046   0F0     0010   SCIMMN   LWI      $0            INITIALIZE W REGISTER
00047   1A0 030 0011           LMID     $0,SCISOK     INITIALIZE RAM
00048   1A0 031 0013           LMID     $0,SCIMOK
00049   1A0 032 0015           LMID     $0,SCISRU
00050   1A0 033 0017           LMID     $0,SCISRL
00051   1A0 034 0019           LMID     $0,SCIMTD
00052   1A0 035 001B           LMID     $0,SCIMEF
00053   2E0     001D           SEDD     $0            INITIALIZE DO PORT
00054   2E1     001E           SEDD     $1
00055   185 003 001F           SEMD     IMS           INITIALIZE IMS
00056   1A3 004 0021           LMID     $3,PMR        INITIALIZE PMR
00057   1AD 005 0023           LMID     $D,SMR        SELECT PRESCALER AS 1/2.SYSTEM CLOCK 1/4
00058   188 003 0025           REMD     IFS           CLEAR SERIAL INTERRUPT REQUEST FLAG
00059   189 003 0027           REMD     IMS           CLEAR SERIAL INTERRUPT MASK
00060   184 000 0029           SEMD     IE            ENABLE INTERRUPT
00061   206     002B           LBI      $6            STORE OUTPUT DATA
00062   211     002C           LYI      $1
00063   1A1 034 002D           LMID     $1,SCIMTD     SET SENDING DATA FLAG
00064   160 03E 002F   SCIMN1   CALL     SCIMTR        OUTPUT SCI DATA
00065   18C 035 0031           TMD      $0,SCIMEF     TEST RECEIVING COMPLETE FLAG
00066   335     0033           BR       SCIMN2
00067   32F     0034           BR       SCIMN1
00068   190 007 0035   SCIMN2   LAMD     SRU           STORE RECEIVE DATA IN RAM
00069   194 032 0037           LMAD     SCISRU
00070   190 006 0039           LAMD     SRL
00071   194 033 003B           LMAD     SCISRL
00072   33D     003D   PEND     BR       PEND          END OF PROGRAM
00073                  ************************************************************
00074                  *                                                          *
```

```
00075                    *          NAME : SCIMTR (SCI MASTER TRANSFER)          *
00076                    *                                                       *
00077                    *********************************************************
00078                    *                                                       *
00079                    *        ENTRY : B REGISTER,Y REGISTER (TRANSFER DATA) *
00080                    *      RETURNS : NOTHING                                *
00081                    *                                                       *
00082                    *********************************************************
00083   18C 030   003E   SCIMTR   TMD      $0,SCISOK     TEST IF SENDING DATA
00084   35E       0040            BR       SCIM7
00085   18C 031   0041            TMD      $0,SCIMOK     TEST IF RECEIVING DATA
00086   35E       0043            BR       SCIM7
00087   2A1       0044            TDD      $1            TEST MASTER RECEIVING REQUEST
00088   347       0045            BR       SCIM1
00089   360       0046            BR       SCIM9
00090   18C 034   0047   SCIM1    TMD      $0,SCIMTD     TEST IF SENDING DATA
00091   34B       0049            BR       SCIM2
00092   35E       004A            BR       SCIM7
00093   260       004B   SCIM2    REDD     $0            CLEAR SLAVE RECEIVE REQUEST
00094   230       004C            LAI      $0            EXECUTE IOMS SOFTWARE TIMER
00095   284       004D   SCIM3    AI       $4
00096   350       004E            BR       SCIM4
00097   34D       004F            BR       SCIM3
00098   2A1       0050   SCIM4    TDD      $1            TEST MASTER RECEIVE REQUEST
00099   353       0051            BR       SCIM5
00100   35F       0052            BR       SCIM8
00101   1A1 030   0053   SCIM5    LMID     $1,SCISOK     SET SENDING DATA FLAG
00102   1A0 034   0055            LMID     $0,SCIMTD     CLEAR RECEIVING DATA FLAG
00103   048       0057            LAB                    STORE SENDING DATA IN RAM
00104  .194 007   0058            LMAD     SRU
00105   0AF       005A            LAY
00106   194 006   005B            LMAD     SRL
00107   148       005D   SCIM6    STS                    SCI START
00108   010       005E   SCIM7    RTN
00109   2E0       005F   SCIM8    SEDD     $0            SET SLAVE RECEIVE REQUEST
00110   230       0060   SCIM9    LAI      $0            EXECUTE IOMS SOFTWARE TIMER
00111   284       0061   SCIM10   AI       $4
00112   364       0062            BR       SCIM11
00113   361       0063            BR       SCIM10
00114   1A1 031   0064   SCIM11   LMID     $1,SCIMOK     SET RECEIVING DATA FLAG
00115   150 05D   0066            JMPL     SCIM6
00116                    *********************************************************
00117                    *                                                       *
00118                    *          NAME : SCMTRD (SCI MASTER RECEIVE)           *
00119                    *                                                       *
00120                    *********************************************************
00121                    *                                                       *
00122                    *         ENTRY : NOTHING                               *
00123                    *       RETURNS : SCISRU (UPPER RECEIVED DATA)          *
00124                    *                 SCISRL (LOWER RECEIVED DATA)          *
00125                    *                 SCIMEF (SCIMEF=0:TRUE,SCIMEF=1:FALSE) *
00126                    *                                                       *
00127                    *********************************************************
00128   188 003   0068   SCMTRD   REMD     IFS            CLEAR SERIAL INTRRUPT REQUEST FLAG
00129   2FF       006A            XMRA     $F             SAVE A
00130   18C 030   006B            TMD      $0,SCISOK      TEST IF SENDING DATA
00131   36F       006D            BR       SCMTR1
00132   374       006E            BR       SCMTR2
00133   1A0 030   006F   SCMTR1   LMID     $0,SCISOK      CLEAR SENDING DATA FLAG
00134   2E0       0071            SEDD     $0             SET SLAVE RECEIVING REQUEST FLAG
00135   150 080   0072            JMPL     SCMTR3
00136   1A0 031   0074   SCMTR2   LMID     $0,SCIMOK      RECEIVING DATA FLAG
00137   190 007   0076            LAMD     SRU            STORE RECEIVE DATA IN RAM
00138   194 032   0078            LMAD     SCISRU
00139   190 006   007A            LAMD     SRL
00140   194 033   007C            LMAD     SCISRL
00141   1A1 035   007E            LMID     $1,SCIMEF      SET RECEIVING COMPLETE FLAG
00142   2FF       0080   SCMTR3   XMRA     $F             RESTORE A
00143   011       0081            RTNI
00144                    *
00145                            END
```

SECTION 12. LIQUID CRYSTAL DRIVER (HD61100A) CONTROL

12.1 HARDWARE DESCRIPTION

12.1.1 Function

Controls the HD61100A liquid crystal driver and displays "9876543210" on an LCD display.

12.1.2 Microcomputer Operation

The HMCS404C sends display data and control signals from to the HD61100A to display graphics on the LCD. Signals M and CL1 of the HD61100A and signal COMMON of the liquid crystal are controlled through port R7. In addition, the HD61100A control signals (signals CL_2, DL) are controlled using the clock synchronous SCI (serial communication interface) of port R7 to enable sending of display data to the HD61100A.

12.1.3 Peripheral Devices

HD61100A LCD Driver: Performs static drive on an 8-segment × 10-digit LCD.

12.1.4 Circuit Diagram

LCD driver (HD61100A) control circuit is shown in Fig. 12.1.

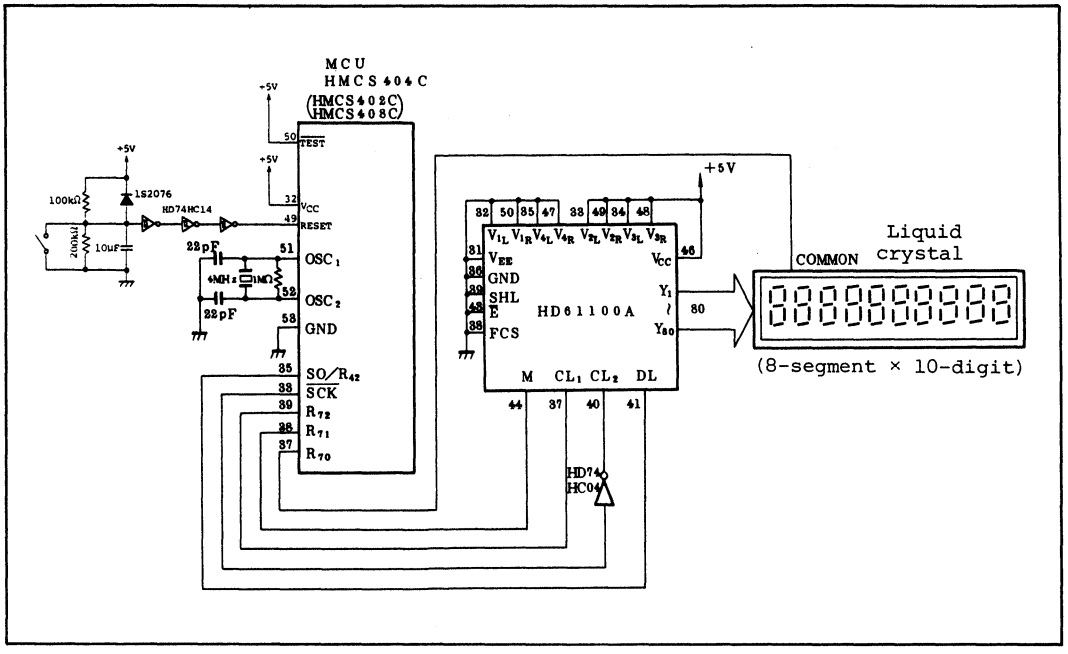


Fig. 12.1. LCD Driver (HD61100A) Control Circuit

12.1.5 Pin Functions

Pin functions at the interface between the HMCS404C and the HD61100A are shown in Table 12.1.

Table 12.1. Pin Functions

Pin Name (HMCS404C)	Input/ Output	Active Level (High or Low)	Function	Pin Name (HD61100A LCD)
R_{71}	Output	——	Outputs alternate signal for LCD driving output	M
R_{72}	Output	——	Resets counter, outputs synchronous signal of latch clock for display data	CL_1
$\overline{SCK}$	Output	——	Outputs shift clock for display data	CL_2
SO	Output	——	Inputs display data	DL

12.1.6 Hardware Operation

Timing chart of the HMCS404C, LCD, and the HD61100A is shown in Fig. 12.2.

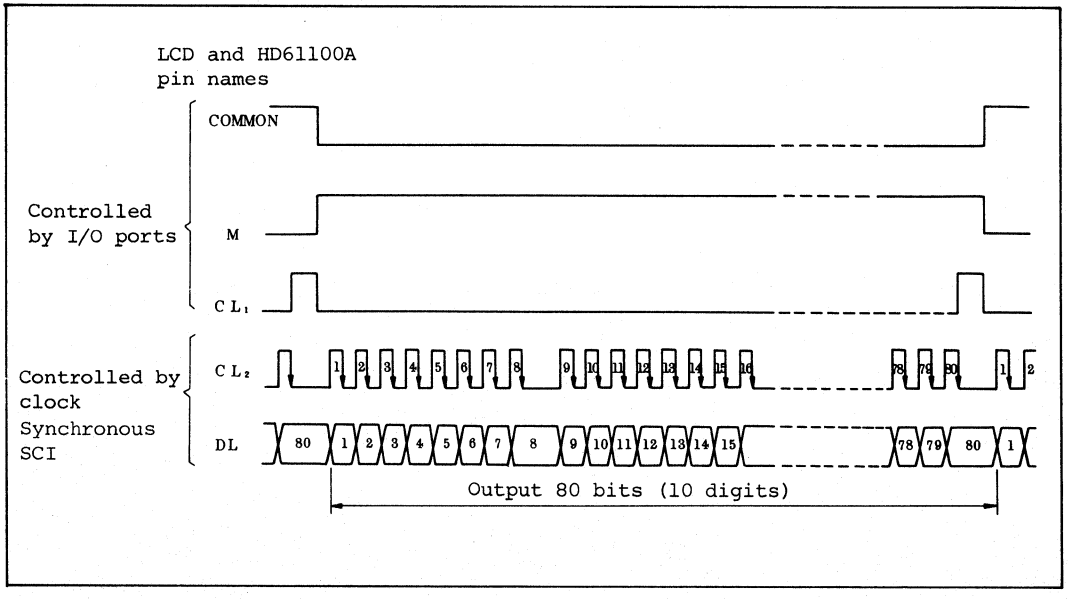


Fig. 12.2. Timing Chart of HMCS404C, LCD, and HD61100A

12.2 SOFTWARE DESCRIPTION

12.2.1 Program Module Configuration

The program module configuration for character display on LCD is shown in Fig. 12.3.

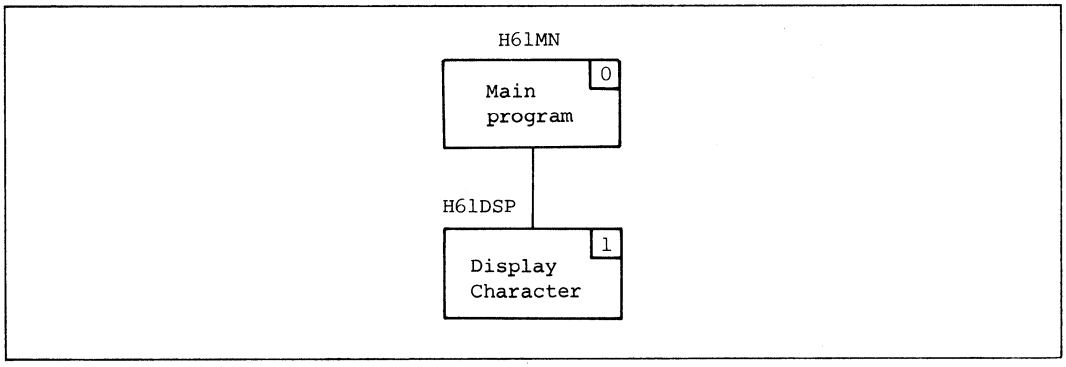


Fig. 12.3. Program Module Configuration

12.2.2 Program Module Functions

Program module functions are summarized in Table 12.2.

Table 12.2. Program Module Functions

No.	Program Module Name	Label	Function
0	**Main Program**	H61MN	Performs static drive on an 8-segment × 10-digit LCD. Initializes control registers and data addresses used for the interface between the HMCS404C and the HD61100A
1	**Display Character**	H61DSP	Performs static drive of LCD using the HD61100A and displays numerals

12.2.3 Program Module Process Flow (Main Program)

The flowchart in Fig. 12.4. shows the procedure for displaying numerals on or LCD as performed by the program module in Fig. 12.3.

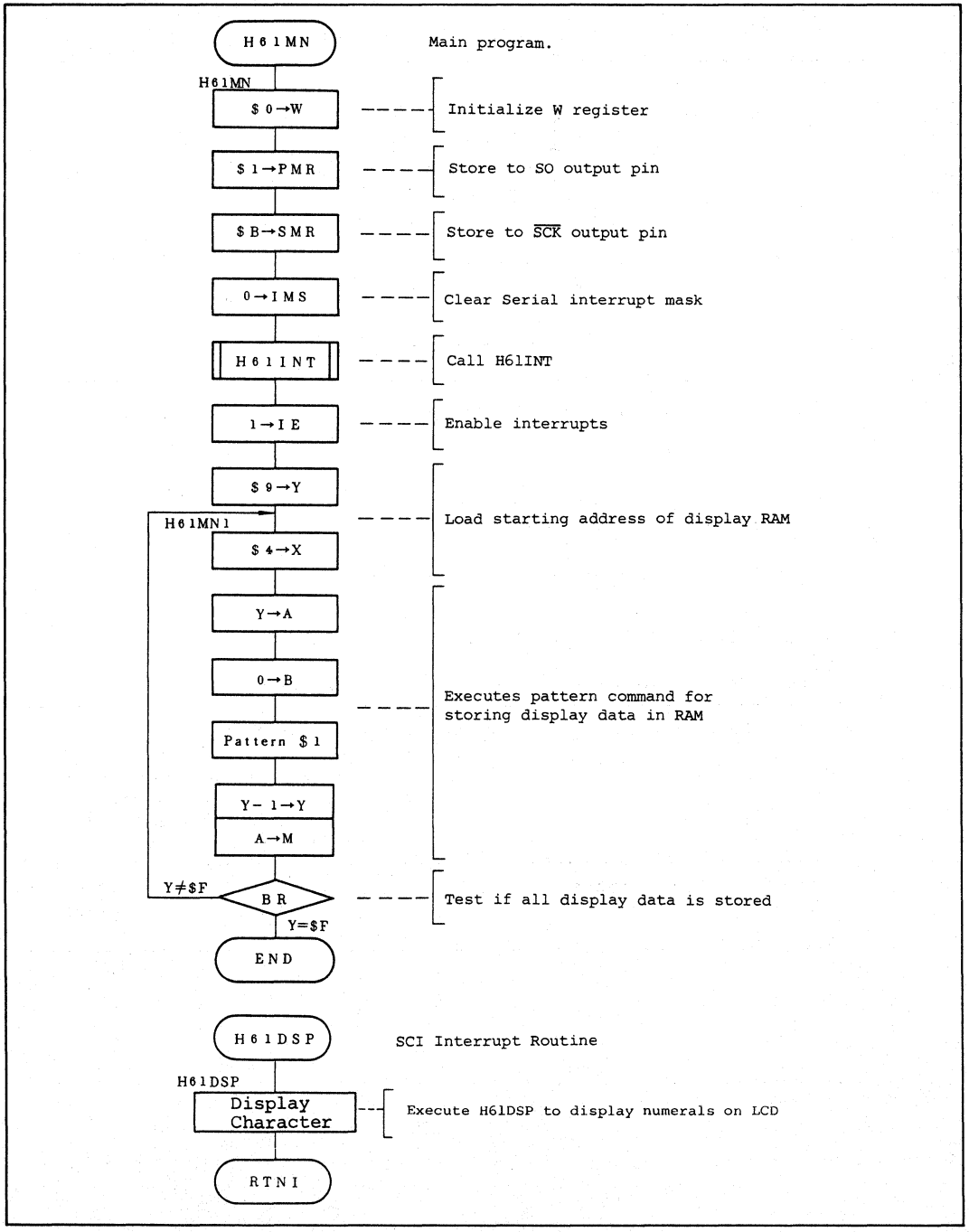


Fig. 12.4. Program Module Flowchart

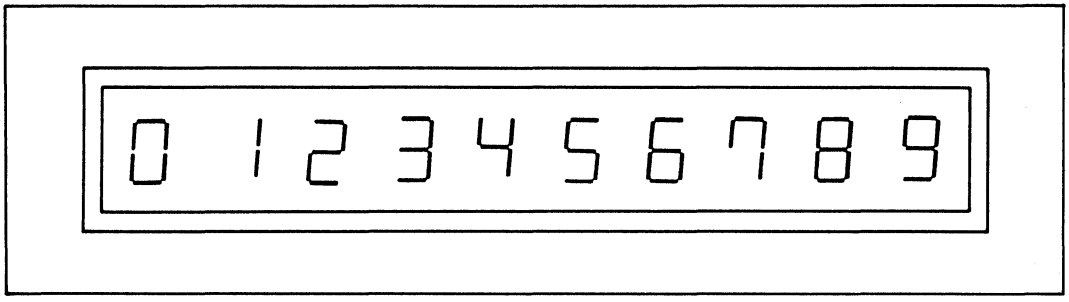


Fig. 12.5. Example of H61MN Execution

Program Module Name:	MCU:	Label:
Display Character	HMCS402C/ HMCS404C/HMCS408C	H61DSP

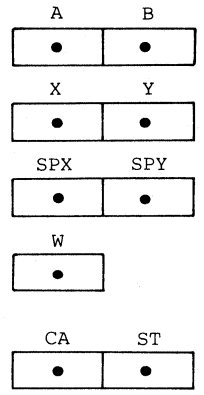
Function:

Sends display data to the HD61100A and displays characters on the LCD.

Arguments:

Contents		1 digit = 4 bits Storage Location	No. of Digits
Entry	Dispaly data	DDATA (RAM)	10
Returns	——	——	——

Changes in CPU Registers and Flags:



● : Not Affected
× : Undefined
↕ : Result

Specifications:

1 word = 10 bits

ROM (Words): 57
RAM (Digits): 2
Stack (Digits): 0
No. of cycles: 435
Reentrant: No
Relocatable: No
Interrupt OK?: No

Description:

1. Function Details

(1) Argument details

DDATA(RAM): Holds 10 digits of display data.

(2) Example of H61DSP execution is shown in Fig. 12.6. If entry argument is as shown in part ① of Fig. 12.6. characters are displayed as shown in part ② of Fig. 12.6.

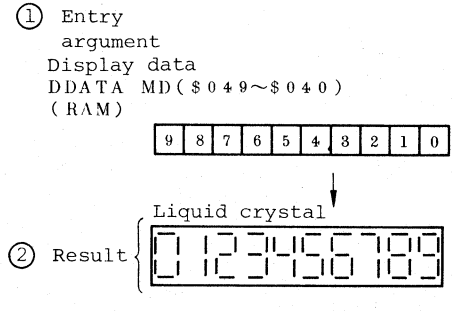


Fig. 12.6. Example of H61DSP Execution

Specifications Notes:

Program Module Name: Display Character	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: H61DSP

Description:

(3) H61DSP calls neither the program modules nor subroutines.

2. User Notes

The following procedure must be performed before H61DSP execution.

(1) Initializes clock synchronous SCI to send display data.

(2) Sets bit IE to enable SCI interrupts.

(3) Clears IMS.

(4) Executes STS command to generate SCI interrupts.

3. RAM Allocation

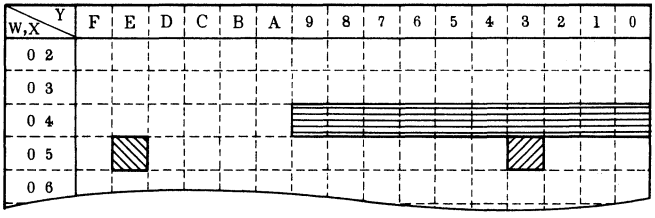


Fig. 12.7. RAM Allocation

Label	RAM	Description
DDATA	MD($049) – MD($040)	Stores display data
CNTR	b3 b0 MD($05E)	Used as Y register pointer to display data and as a counter indicating number of interrupts
MFLG	b3 b0 MD(0,$053)	Used as a test flag indicating whether M signal will be high or low Flag Function is shown in Table 12.3.

Program Module Name:	MCU:	Label:
Display Character	HMCS402C/ HMCS404C/HMCS408C	H61DSP

Description:

Table 12.3. Flag Functions

Label	Bit/Label Bit 0	Function
M F L G	0	Indicates M signal is low
	1	Indicates M signal is high

4. Sample Application

```
                ⋮
        LMID    $1, PMR
        LMID    $B, SMR  ..... Initialize SCI
        REMD    IMS
        LMID    $9, CNTR
        LAI     $5
        LRA     $7       ..... Control M signal, CLI signal, common
        LAI     $2               signal
        LRA     $7
        SEMD    MFLG
        LAMD    $049
        LBI     $0
        P       $2       ..... Store segment data in SRL, SRV
        LMAD    SRL
        LAB
        LMAD    SRU
        STS              ..... Start SCI
        SEMD    IE       ..... Enable interrupts
   Store display data
   in display RAM
                ⋮
        ORG     $200
        DC      $177, $141, $1B8, $1E8, $1C5,  ..... Segment data
                $1E6, $1F6, $148, $1F7, $1E7
```

Program Module Name: Display Character	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: H61DSP

Description:

5. Basic Operation

(1) 10 digits of display data are sent to the HD61100A to display numerals on an 8 segments × 10 digits LCD. Shift clock and data signal are controlled by the block synchronous SCI of the HMCS404C.

(2) Display data is stored in display of RAM before execution. Each SCI interrupt executes display of 1 byte of data.

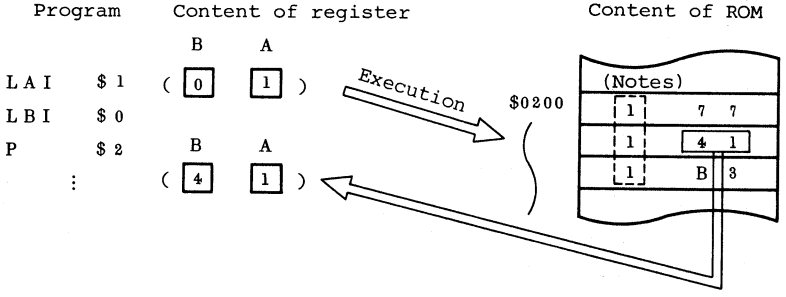
(3) Pointer to display RAM and counter for number of interrupts are decremented every interrupt. CNTR(RAM) is reinitialized each time 10 interrupts are executed.

(4) The first enabling interrupts are performed by the main program. From then on, after execute SCI command SCI interrupts are generated automatically each time segment data are outputted.

(5) Indicates MFLG is status of M signal of HD61100A.

(6) Data stored at the address indicated by Accumulator and B register is transferred to Accumulator and B register, using the table look-up function of the pattern generation instruction (P).

(7) Lower 8 bits of word stores segment data.

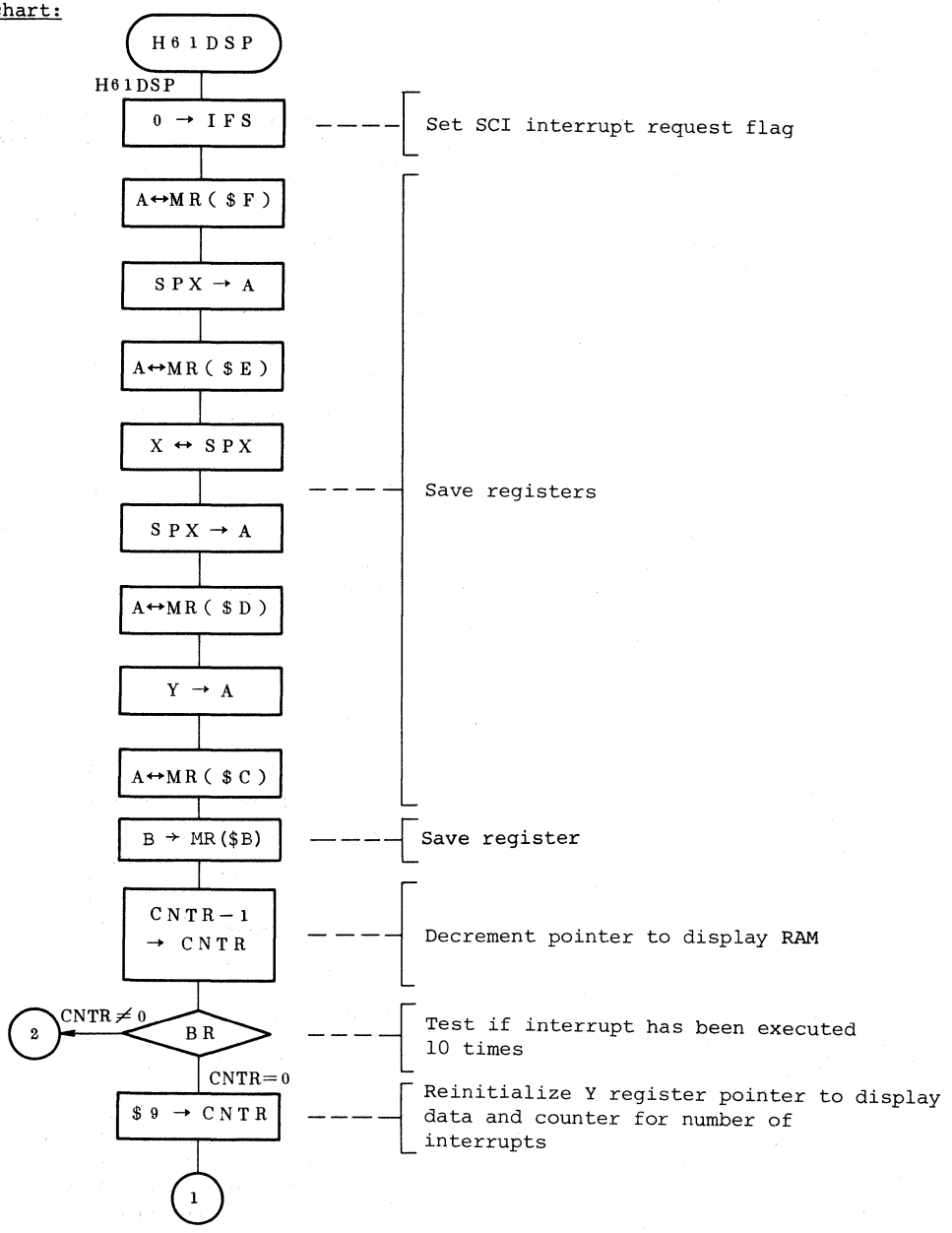


After executing a P instruction in the above program sequence, $41 is contained in B register and Accumulator.
($41 is stored in lower 8 bits of word located at $0201).

(Note) If dotted area (bit 8) is $1 as shown above, ROM data is transferred to Accumulator and B register after executing the P instruction.

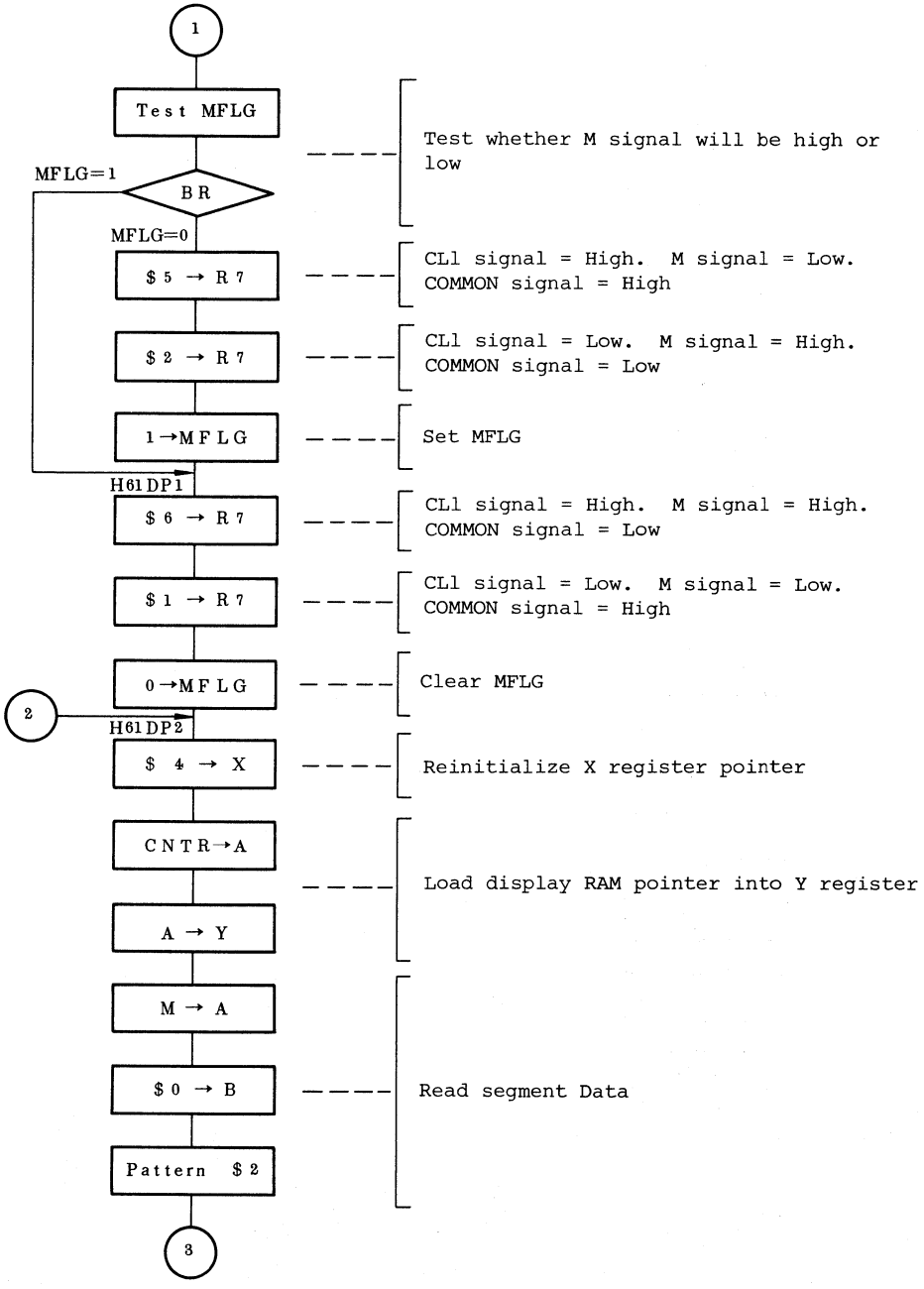
Program Module Name: Display Character	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: H61DSP

Flowchart:



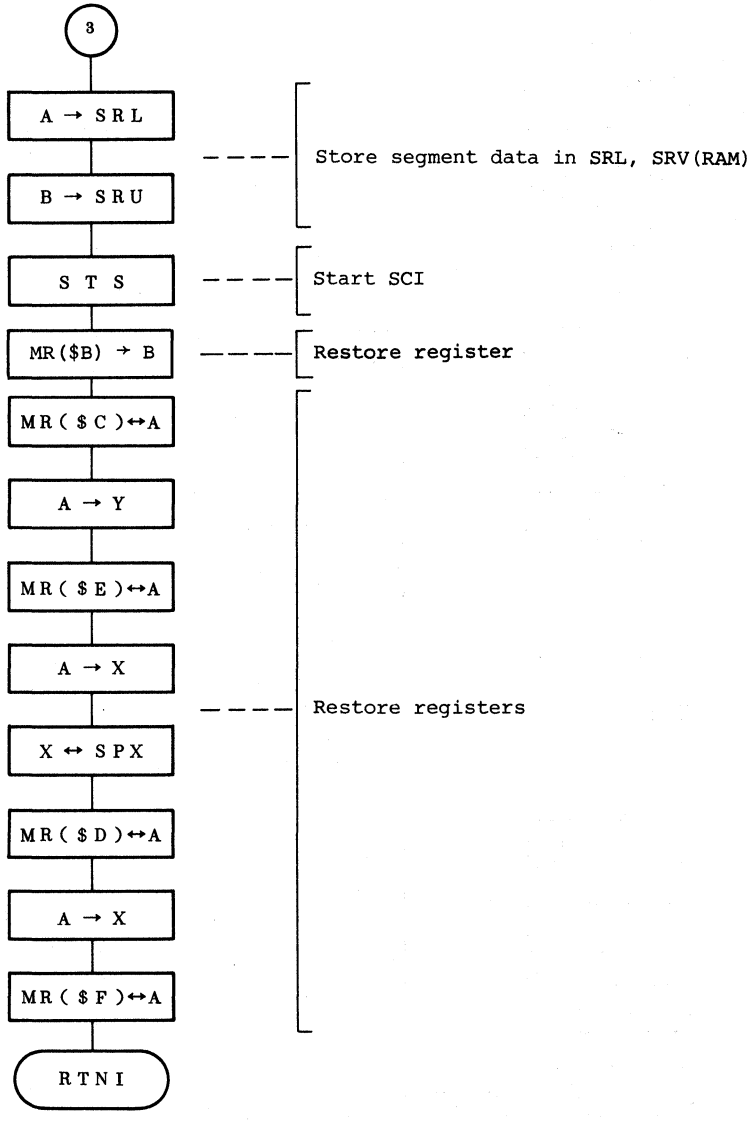
Program Module Name: Display Character	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: H61DSP

Flowchart:



Program Module Name: Display Character	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: H61DSP

Flowchart:



12.4 SUBROUTINE DESCRIPTION

Subroutine Name: Initialize	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: H61INT

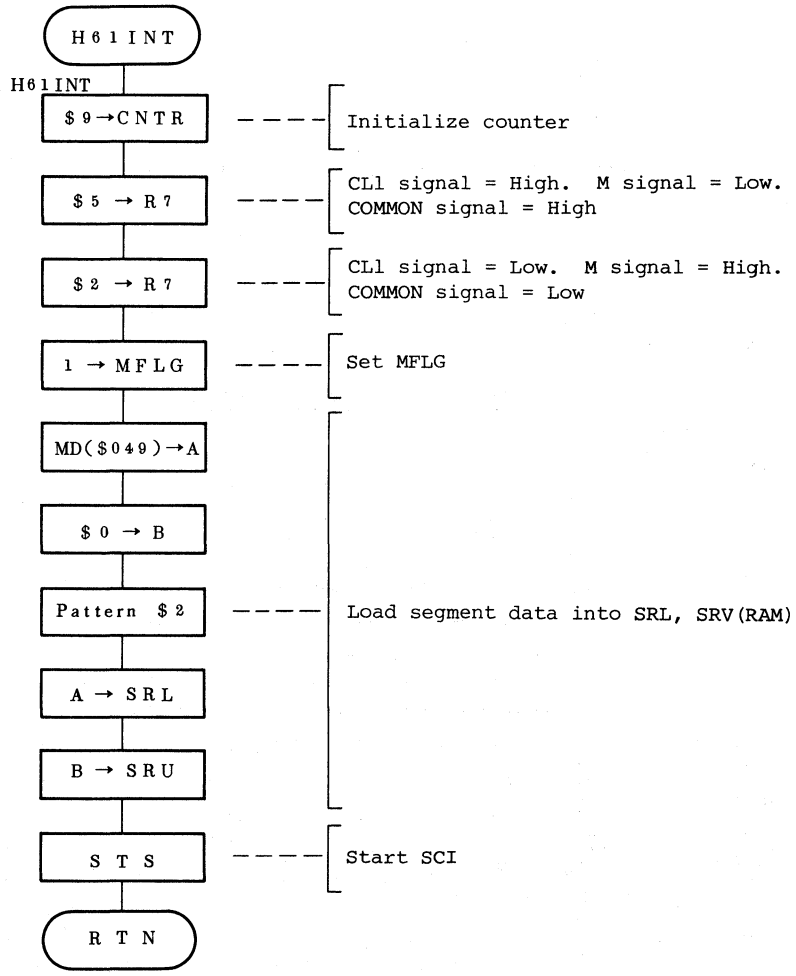
Function: Stores display data

Basic Operation:

(1) Initializes counter, CL1, M, COMMON signal, MFLG and others.

(2) Executes only once is this routine.

Program Module Using This Subroutine: ——



```
ST-NO   OBJECT    ADRS   SOURCE STATEMENTS

00001   011       0000           LLEN     132
00002                            TITLE    LIQUID CRYSTAL DRIVER (HD61100) CONTROL
00003                  *
00004                  ****   RAM ALLOCATION   **********************************
00005                  *
00006                  CNTR     EQU      $05E          COUNTER FOR DISPLAY
00007                  MFLG     EQU      0.$053        M SIGNAL JUDGMENT FLAG
00008                  *
00009                  ****   SYMBOL DEFINITIONS   ****************************
00010                  *
00011                  PMR      EQU      $004          PORT MODE REGISTER
00012                  SMR      EQU      $005          SERIAL MODE REGISTER
00013                  SRL      EQU      $006          LOWER SERIAL DATA REGISTER
00014                  SRU      EQU      $007          UPPER SERIAL DATA REGISTER
00015                  IMS      EQU      1.$003        IM OF SERIAL
00016                  IFS      EQU      0.$003        IF OF SERIAL
00017                  IE       EQU      0.$000        ENABLE INTERRUPT
00018                  ************************************************************
00019                  *                                                          *
00020                  *               VECTOR ADDRESSES                           *
00021                  *                                                          *
00022                  ************************************************************
00023                  *
00024                           ORG      $0000
00025                  *
00026   150 010   0000         JMPL     H61MN         RESET
00027   150 010   0002         JMPL     H61MN         INT0
00028   150 010   0004         JMPL     H61MN         INT1
00029   150 010   0006         JMPL     H61MN         TIMER-A
00030   150 010   0008         JMPL     H61MN         TIMER-B
00031   000       000A         NOP
00032   000       000B         NOP
00033   150 023   000C         JMPL     H61DSP        SERIAL
00034                  ************************************************************
00035                  *                                                          *
00036                  *               MAIN PROGRAM : H61MN                       *
00037                  *                                                          *
00038                  ************************************************************
00039                  *
00040                           ORG      $0010
00041                  *
00042   0F0       0010 H61MN    LWI      $0            INITIALIZE W REGISTER
00043   1A1 004   0011         LMID     1.PMR         SELECT SD
00044   1AB 005   0013         LMID     $B.SMR        SELECT SCK
00045   189 003   0015         REMD     IMS           CLEAR SERIAL INTERRUPT MASK
00046   160 05F   0017         CALL     H61INT
00047   184 000   0019         SEMD     IE            ENABLE INTERRUPTS
00048   219       001B         LYI      $9            STORE DESTINATION
00049   224       001C H61MN1  LXI      $4
00050   0AF       001D         LAY
00051   200       001E         LBI      $0
00052   1B1       001F         P        $1            STORE DISPLAY DATA IN RAM
00053   0D0       0020         LMADY                  TEST IF ALL DISPLAY DATA IS STORED
00054   31C       0021         BR       H61MN1
00055   322       0022 PEND    BR       PEND          END OF PROGRAM
00056                  ************************************************************
00057                  *                                                          *
00058                  *         NAME : H61DSP (DISPLAY CHARACTER)                *
00059                  *                                                          *
00060                  ************************************************************
00061                  *                                                          *
00062                  *        ENTRY : DDATA (DISPLAY DATA)                      *
00063                  *      RETURNS : NOTHING                                   *
00064                  *                                                          *
00065                  ************************************************************
00066   188 003   0023 H61DSP  REMD     IFS           SET SCI INTERRUPT REQUEST FLAG
00067   2FF       0025         XMRA     $F            SAVE REGISTERS
00068   068       0026         LASPX
00069   2FE       0027         XMRA     $E
00070   001       0028         XSPX
00071   068       0029         LASPX
00072   2FD       002A         XMRA     $D
00073   0AF       002B         LAY
00074   2FC       002C         XMRA     $C
```

```
00075   048        002D            LAB
00076   2FB        002E            XMRA    $B
00077   190 05E    002F            LAMD    CNTR            DECREMENT DISPLAY RAM POINTER
00078   28F        0031            AI      $F
00079   194 05E    0032            LMAD    CNTR            TEST IF 10 TIMERS SCI INTERRUPT ARE COMPLETED
00080   347        0034            BR      H61DP2
00081   239        0035            LAI     $9              REINITIALIZE POINTER
00082   194 05E    0036            LMAD    CNTR
00083   18C 053    0038            TMD     MFLG            TEST MSIGNAL IS HIGH OR LOW
00084   341        003A            BR      H61DP1
00085   235        003B            LAI     $5
00086   2D7        003C            LRA     $7              CL1=1,M=0,COMMON=1
00087   232        003D            LAI     $2              CL1=0,M=1,COMMON=0
00088   2D7        003E            LRA     $7
00089   184 053    003F            SEMD    MFLG            SET MFLAG
00090   236        0041  H61DP1    LAI     $6              CL2=1,M=1,COMMON=0
00091   2D7        0042            LRA     $7              CL2=0,M=0,COMMON=1
00092   231        0043            LAI     $1
00093   2D7        0044            LRA     $7
00094   188 053    0045            REMD    MFLG            CLEAR MFLAG
00095   224        0047  H61DP2    LXI     $4              REINITIALIZE X REG POINTER
00096   190 05E    0048            LAMD    CNTR            LOAD DISPLAY RAM POINTER INTO Y REG
00097   0D8        004A            LYA
00098   090        004B            LAM                     READ SEGMENT DATA
00099   200        004C            LBI     $0
00100   1B2        004D            P       $2
00101   194 006    004E            LMAD    SRL             STORE SEGMENT DATA IN RAM
00102   048        0050            LAB
00103   194 007    0051            LMAD    SRU
00104   148        0053            STS                     SCI START
00105   2FB        0054            XMRA    $B              RESTORE REGISTERS
00106   0C8        0055            LBA
00107   2FC        0056            XMRA    $C
00108   0D8        0057            LYA
00109   2FE        0058            XMRA    $E
00110   0E8        0059            LXA
00111   001        005A            XSPX
00112   2FD        005B            XMRA    $D
00113   0E8        005C            LXA
00114   2FF        005D            XMRA    $F
00115   011        005E            RTNI
00116                      ********************************************************
00117                      *                                                      *
00118                      *      NAME : H61INT (START SCI)                       *
00119                      *                                                      *
00120                      ********************************************************
00121   1A9 05E    005F  H61INT    LMID    $9,CNTR         INITIALIZE COUNTER
00122   235        0061            LAI     $5              CL1=1,M=0,COMMON=1
00123   2D7        0062            LRA     $7
00124   232        0063            LAI     $2              CL1=0,M=1,COMMON=0
00125   2D7        0064            LRA     $7
00126   184 053    0065            SEMD    MFLG            SET MFLAG
00127   190 049    0067            LAMD    $049            LOAD SEGMENT DATA
00128   200        0069            LBI     $0
00129   1B2        006A            P       $2
00130   194 006    006B            LMAD    SRL             STORE SEGMENT DATA IN RAM
00131   048        006D            LAB
00132   194 007    006E            LMAD    SRU
00133   148        0070            STS                     SCI START
00134   010        0071            RTN
00135                      ********************************************************
00136                      *                                                      *
00137                      *                DATA TABLE                            *
00138                      *                                                      *
00139                      ********************************************************
00140                      *
00141                              ORG     $100
00142                      *                       RAM DATA
00143   100        0100            DC      $100
00144   101        0101            DC      $101
00145   102        0102            DC      $102
00146   103        0103            DC      $103
00147   104        0104            DC      $104
00148   105        0105            DC      $105
00149   106        0106            DC      $106
00150   107        0107            DC      $107
00151   108        0108            DC      $108
00152   109        0109            DC      $109
00153                      *
00154                              ORG     $200
00155                      *                       SEGMENT DATA
00156   177        0200            DC      $177            0
00157   141        0201            DC      $141            1
00158   1B3        0202            DC      $1B3            2
00159   1E3        0203            DC      $1E3            3
00160   1C5        0204            DC      $1C5            4
00161   1E6        0205            DC      $1E6            5
```

```
00162   1F6     0206            DC      $1F6        6
00163   143     0207            DC      $143        7
00164   1F7     0208            DC      $1F7        8
00165   1E7     0209            DC      $1E7        9
00166                   *
00167                           END
```

SECTION 13. HD61830 (LM200) GRAPHIC MODE

13.1 HARDWARE DESCRIPTION

13.1.1 Function

Initializes graphic mode and displays dot graphics on the LM200 liquid crystal module.

13.1.2 Microcomputer Operation

The HMCS404C transfers display data to the dot matrix liquid crystal graphic display controller LSI HD61830 (LCTC) from port R onto the LCTC data bus (DB_0 - DB_7), and transmits control signals E, R/W, and RS through port D. Port D and port R are controlled by software.

13.1.3 Peripheral Devices

HD61830 LCTC: Receives control signals and display data from the HMCS404C and in turn controls the HM6116 Display RAM and LM200.

LM200 Liquid Crystal Module: Receives graphic display data and control signals from the HD61830 LCTC. A resolution of 64 × 240 pixels is provided in LM200 graphic mode. In this application, the graphic figure shown in Fig. 13.5 is displayed.

13.1.4 Circuit Diagram

LCTC control circuit is shown in Fig. 13.1.

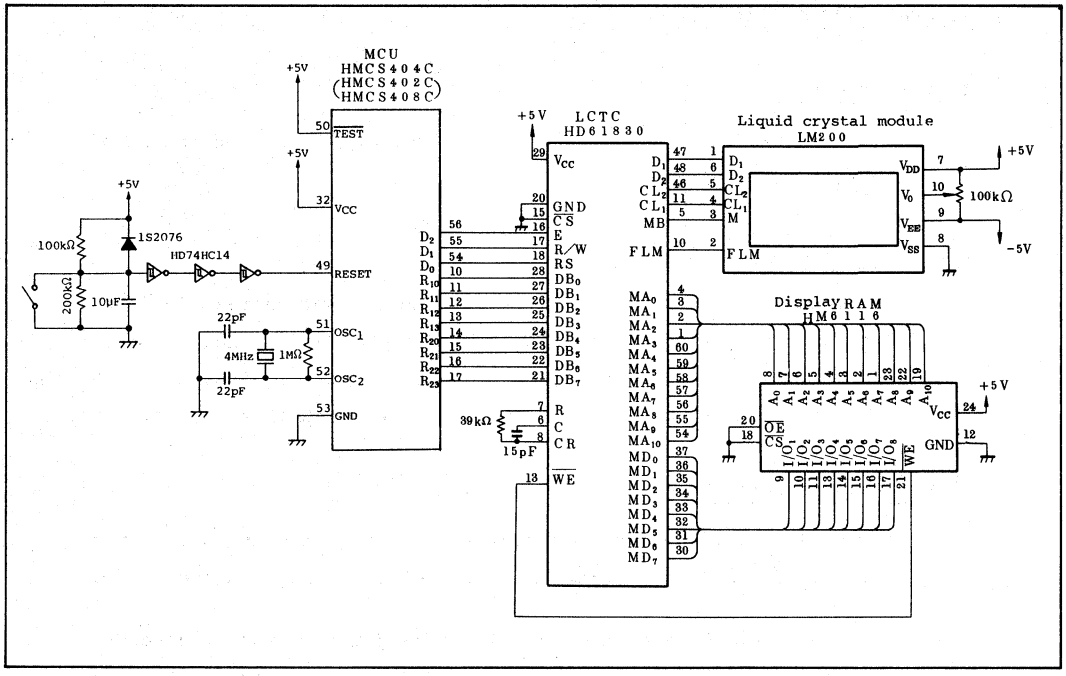


Fig. 13.1. LCTC Control Circuit

13.1.5 Pin Functions

Pin functions at the interface between the HMCS404C and LCTC are shown in Table 13.1.

Table 13.1. Pin Functions

Pin Name (HMCS404C)	Input/Output	Active Level (High or Low)	Function	Pin Name (LCTC)
D_2	Output	High	Enables signal	E
D_1	Output	High	Reads data	R/W
		Low	Writes data	
D_0	Output	High	Selects instruction register	RS
		Low	Selects data register	
R_{10}	Input/Output	-	Data lines	DB_0
R_{11}	Input/Output	-		DB_1
R_{12}	Input/Output	-		DB_2
R_{13}	Input/Output	-		DB_3
R_{20}	Input/Output	-		DB_4
R_{21}	Input/Output	-		DB_5
R_{22}	Input/Output	-		DB_6
R_{23}	Input/Output	-		DB_7

13.1.6 Hardware Operation

The timing chart for interfacing between the HMCS404C and each signal is shown in Fig. 13.2. ① and ② in Fig. 13.2. show timing for read and write.

① Data from LCTC can be read during ① period.

② Data can be written to LCTC at the falling edge of signal E.

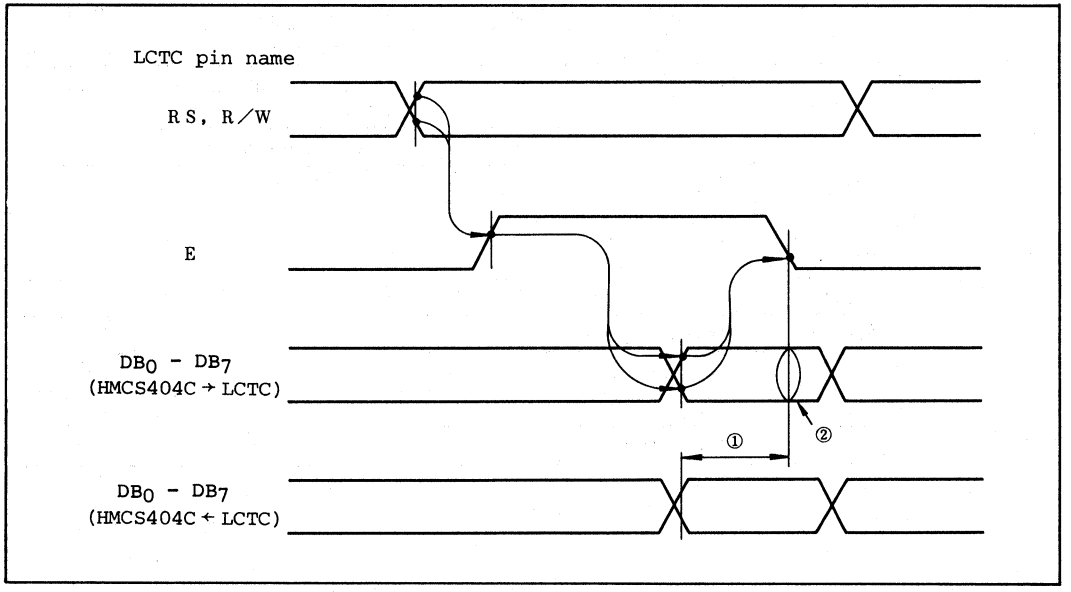


Fig. 13.2. HMCS404C LCTC Interface

13.2 SOFTWARE DESCRIPTION

13.2.1 Program Module Configuration

The program module configuration for graphic display on the liquid crystal module is shown in Fig. 13.3.

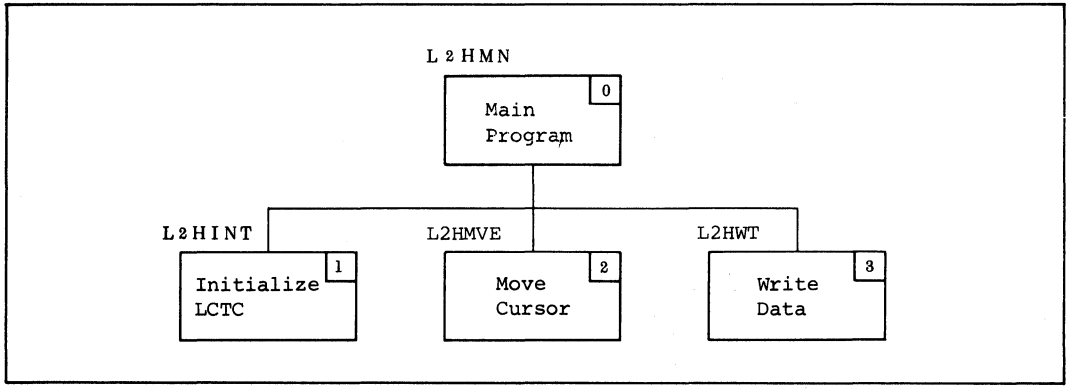


Fig. 13.3. Program Module Configuration

13.2.2 Program Module Functions

Program module functions are summerized in Table 13.2.

Table 13.2. Program Module Functions

No.	Program Module Name	Label	Function
0	Main Program	L2HMN	Demonstrates graphic display on LM200
1	Initialize LCTC	L2HINT	Initializes LCTC for graphic mode
2	Move Cursor	L2HMVE	Initializes LCTC cursor address
3	Write Data	L2HWT	Writes instructions and data to the LCTC

13.2.3 Program Module Process Flow (Main Program)

The following flowchart (Fig. 13.4) demonstrates the process for displaying graphics on the LM200 liquid crystal display, using the modules described above. Fig. 13.5 shows this applications display.

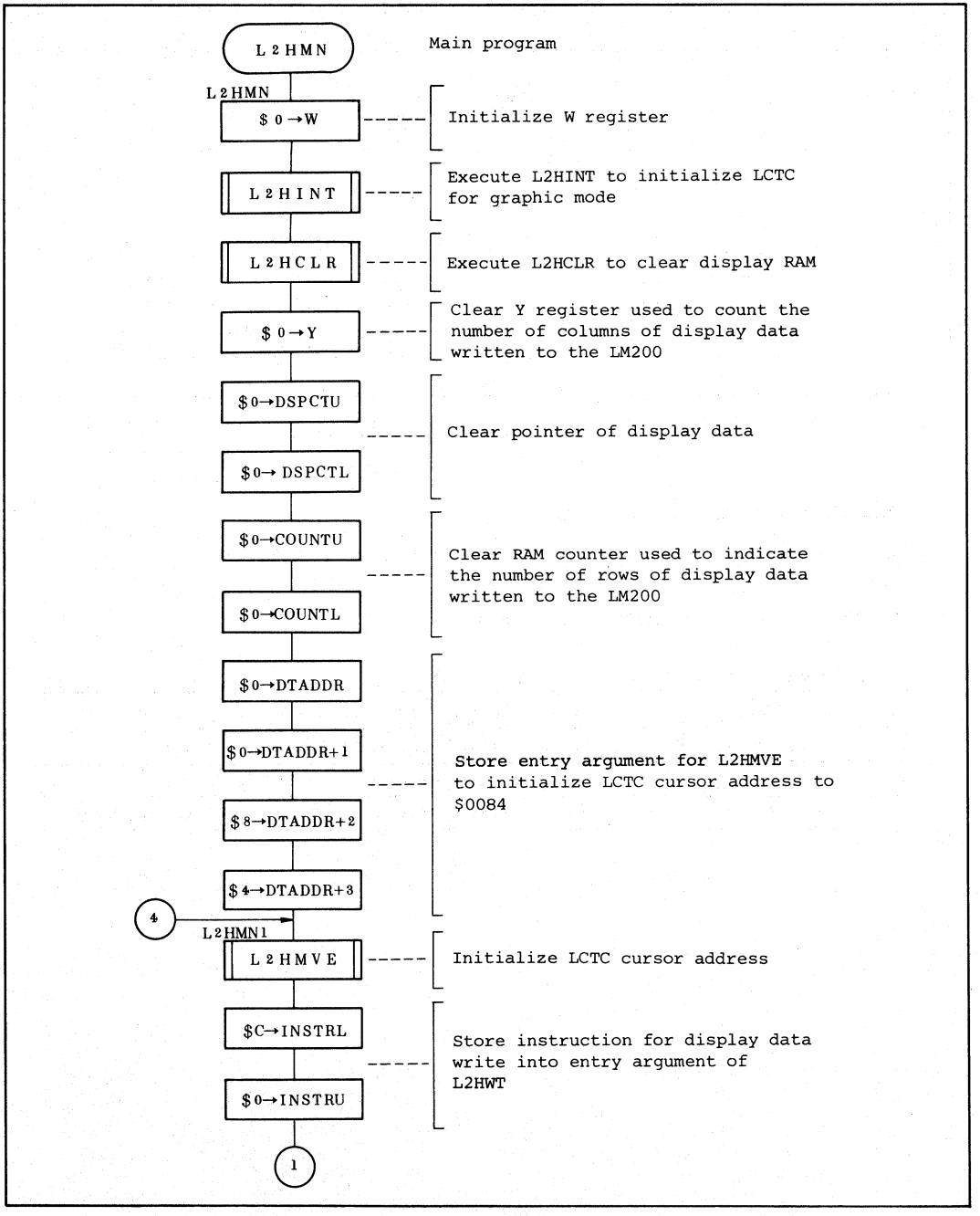


Fig. 13.4. Program Module Sample Application

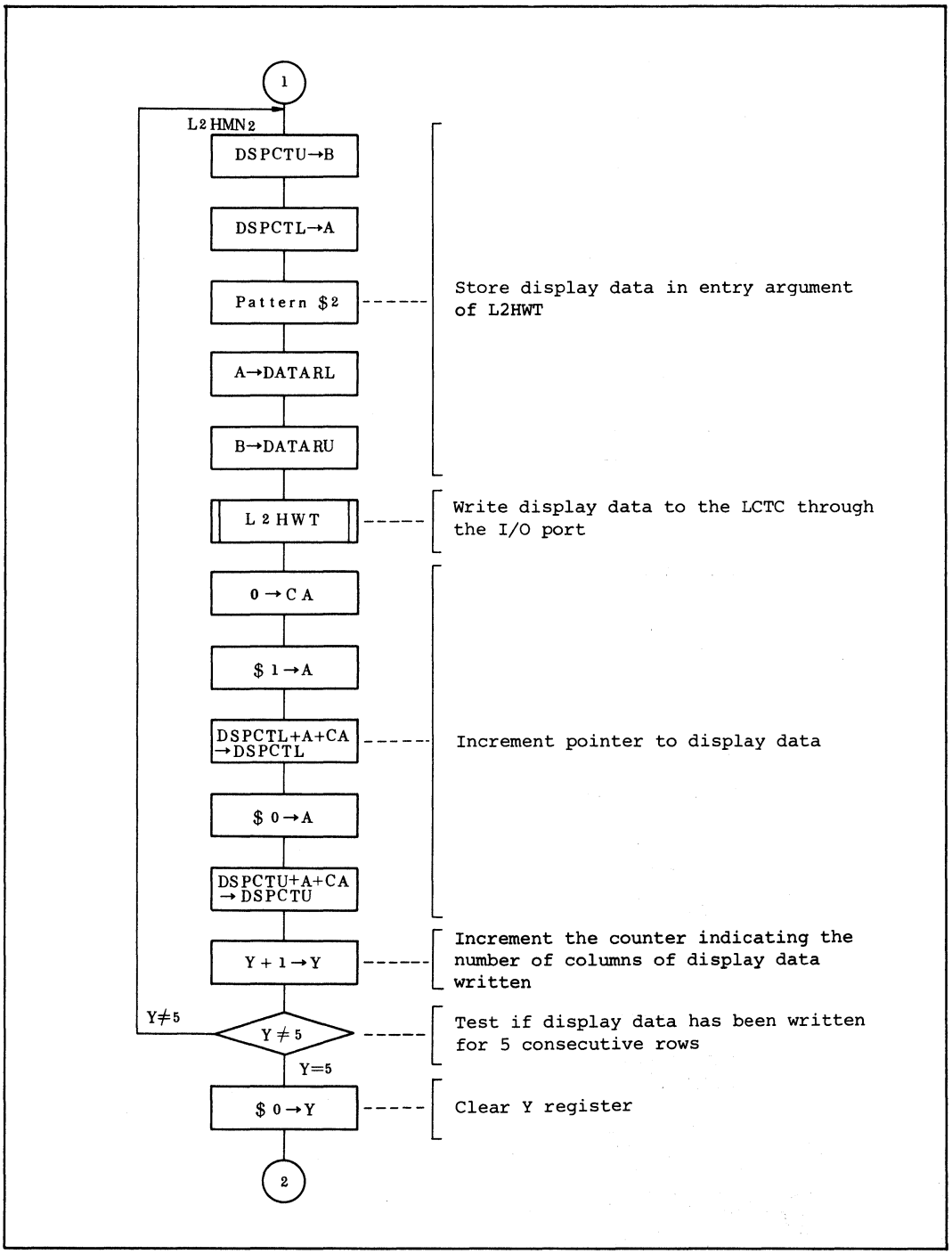


Fig. 13.4. Program Module Sample Application (Cont.)

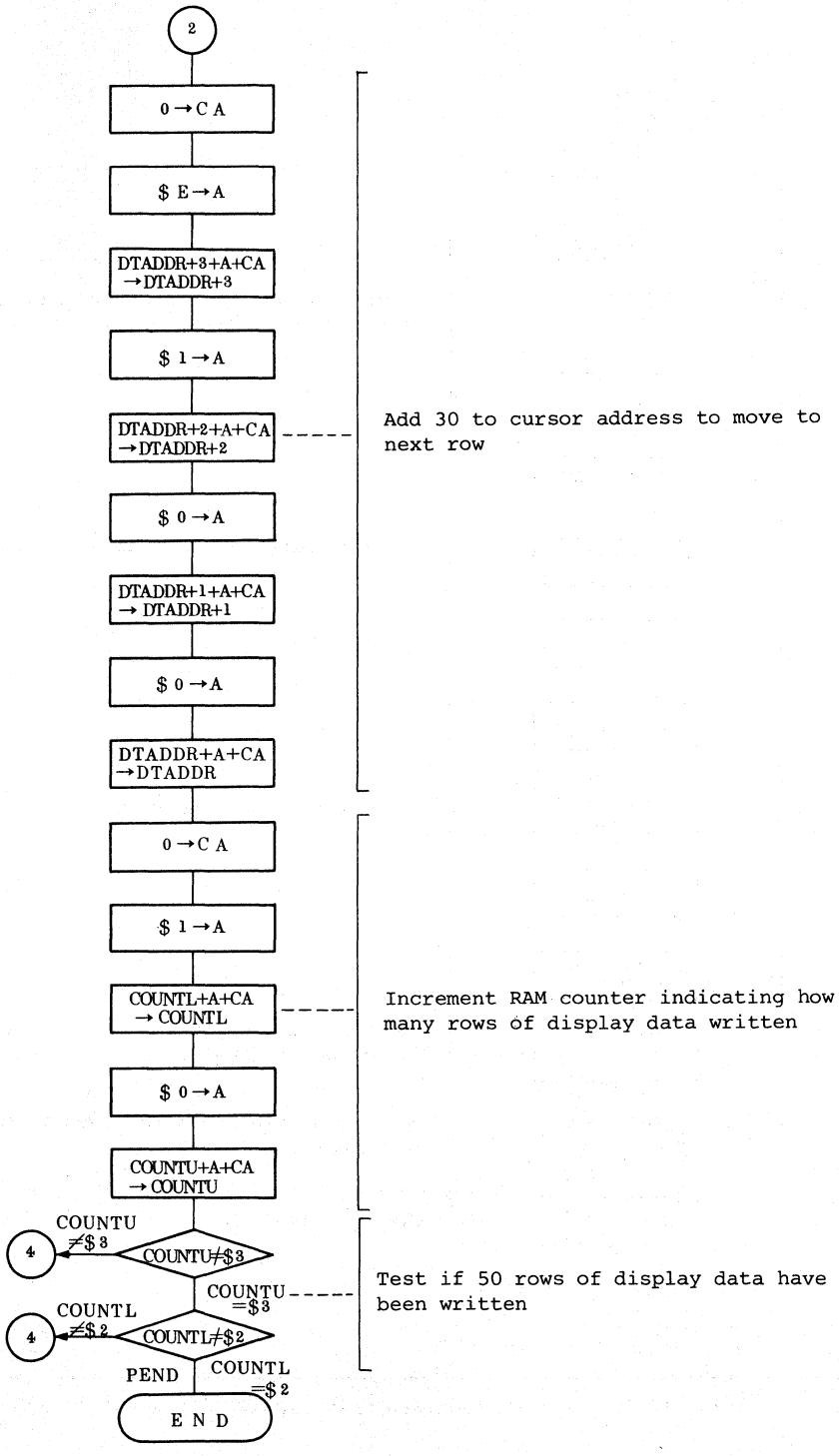


Fig. 13.4. Program Module Sample Application (Cont.)

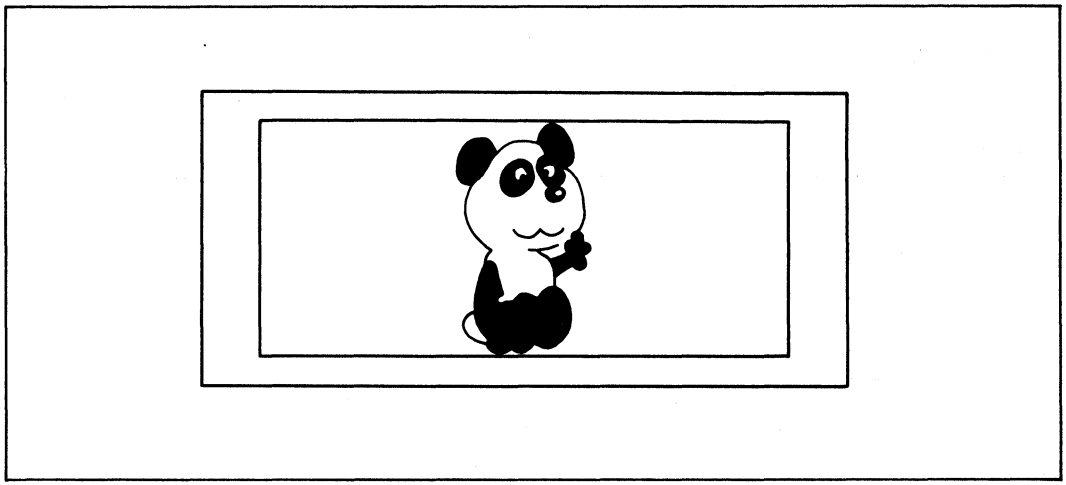


Fig. 13.5. Result of Program Module Execution

Program Module Name: Initialize LCTC

MCU: HMCS402C/ HMCS404C/HMCS408C

Label: L2HINT

Function: Initializes LCTC for graphic mode.

Arguments: None

Changes in CPU Registers and Flags:

A	B
×	×

X	Y
●	●

SPX	SPY
●	●

W
●

CA	ST
×	×

● : Not Affected
× : Undefined
↕ : Result

Specifications:

1 word = 10 bits

ROM (Words):	31
RAM (Digits):	2
Stack (Digits):	0
No. of cycles:	1538
Reentrant:	No
Relocatable:	No
Interrupt OK?:	Yes

Description:

1. Function Details

(1) Program module L2HINT has no arguments.

(2) After execution of L2HINT, the LCTC enters graphic mode and the LM200 display screen is cleared.

(3) L2HINT uses the subroutines shown in Table 13.3.

Table 13.3. Subroutines Used by L2HINT

Subroutine Name	Label Name	Function
Check Busy Flag	L2HBSY	Checks LCTC busy flag

Specifications Notes:

The number of cycles indicated is the minimum number of cycles required by subroutine L2HBSY.

Program Module Name: Initialize LCTC	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: L2HINT

Description:

2. User Notes

As this routine uses the P instruction (Pattern Generation Instruction), pay close attention to the data reference addresses of the data table.

3. RAM Allocation

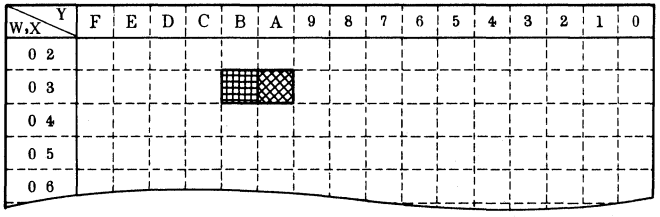


Fig. 13.6. RAM Allocation

Label	RAM	Description
INTA	b3 b0 MD($03A)	Holds lower digit of source address
INTB	b3 b0 MD($03B)	Holds upper digit of source address

4. Sample Application

```
        :
    CALL    L2HINT      ..... Call L2HINT
        :
```

Program Module Name: Initialize LCTC	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: L2HINT

Description:

5. Basic Operation

(1) The instruction and data elements shown in Table 13.4 must be loaded into the instruction and data registers in corresponding pains to initialize LCTC. Program module L2HINT loads data shown in Table 13.4.

(2) RS signal is used to switch between the two registers;
High: Instruction register, Low: Data register

Table 13.4. Data to be stored in LCTC

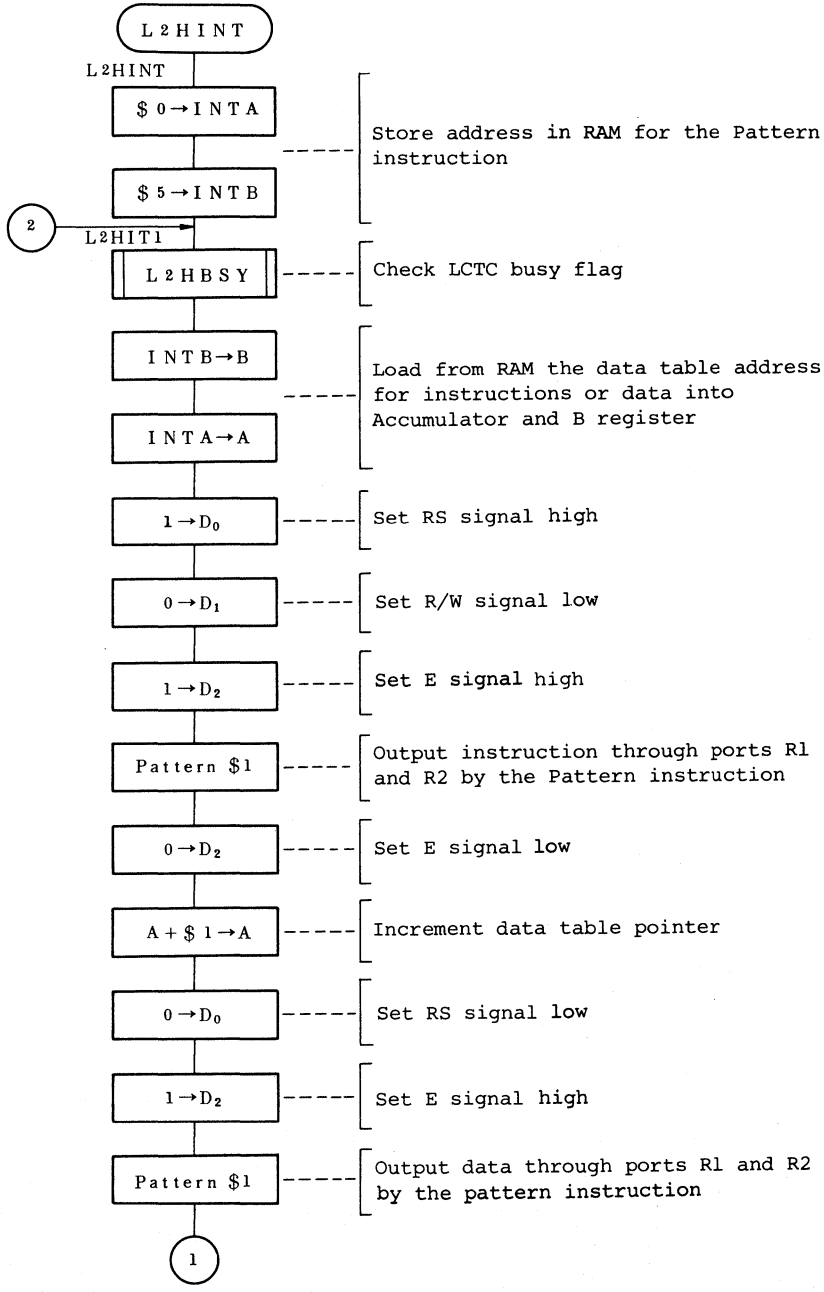
Instruction	Data	Function
$00	$32	Selects Display ON, Master mode and Graphic mode
$01	$07	Selects 8 bits of horizontal dots per character in display
$02	$1D	Selects 30 bytes of horizontal bytes in the graphic mode
$03	$1F	Selects 1/32 duty in multiplex display
$08	$00	Selects display starting address to $0000
$09	$00	
$0A	$00	Selects cursor address to $0000
$0B	$00	

(3) RS, R/W, and E signals are controlled by port D_0, port D_1 and port D_2.

(4) Initialization data shown in Table 13.4 is previously stored in the data table in ROM.

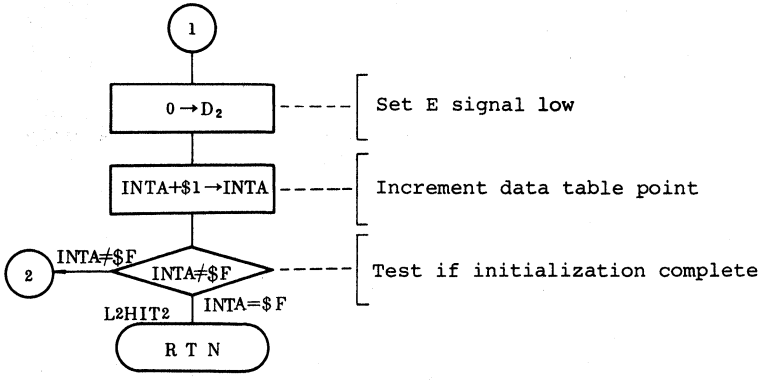
Program Module Name: Initialize LCTC	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: L2HINT

Flowchart:



Program Module Name: Initialize LCTC	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: L2HINT

Flowchart:



Program Module Name: Move Cursor	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: L2HMVE

Function: Loads cursor address value stored in DTADDR(RAM) into LCTC cursor address counter.

Arguments: 1 digit = 4 bits

	Contents	Storage Location	No. of Digits
Entry	Cursor address	DTADDR (RAM)	4
Returns	———	———	———

Changes in CPU Registers and Flags:

A	B
×	●

X	Y
●	●

SPX	SPY
●	●

W
●

CA	ST
×	×

● : Not Affected
× : Undefined
↕ : Result

Specifications:

1 word = 10 bits

ROM (Words):	29
RAM (Digits):	8
Stack (Digits):	4
No. of cycles:	254
Reentrant:	No
Relocatable:	No
Interrupt OK?:	Yes

Description:

1. Function Details

(1) Argument details
DTADDR(RAM): Holds cursor address value to be loaded into cursor address counter as 4-digits hexadecimal number.

(2) Program module L2HMVE loads cursor address value into cursor address counter to change cursor address on display.

(3) Program module L2HMVE calls other program modules and subroutines shown in Table 13.5.

Specifications Notes:

"No. of cycles" in "Specifications" represents the number of cycles needed when subroutine L2HBSY is executed by the minimum cycles.

Program Module Name: Move Cursor	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: L2HMVE

Description:

Table 13.5. Program Modules and Subroutines Called in L2HMVE

Program Module/ Subroutine Name	Label	Function
Write Data to LCTC	L2HWT	Writes data to LCTC through I/O port of the HMCS404C
Check Busy Flag	L2HBSY	Checks LCTC busy flag

2. User Notes

The RAM area for storing the 4-digits hexadecimal of the cursor address must be allocated.

3. RAM Allocation

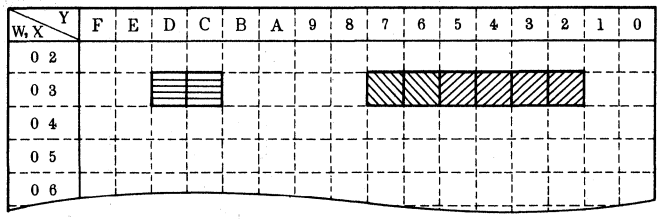


Fig. 13.7. RAM Allocation

Label	RAM	Description
DTADDR:DTADDR+1: DTADDR+2:DTADDR+3	b15 b0 MD($035,$034,$033,$032)	Holds cursor address to be stored in the cursor address counter; consists of 4 hexadecimal digits
INSTRL:INSTRU	b7 b0 MD($037,$036)	Holds value to initialize the LCTC instruction register
DATARU:DATARL	b7 b0 MD($03D,$03C)	Holds value to initialize the LCTC data register

Program Module Name: Move Cursor	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: L2HMVE

Description:

4. Sample Application

```
DTADDR    EQU     $032          ... Allocate RAM area for storing
                                      the 4-digit cursor address in
                                      the user program
            :
          LMID    $0, DTADDR    }
          LMID    $0, DTADDR+1  }  ... Load the cursor address stored
          LMID    $8, DTADDR+2  }      in the user program into entry
          LMID    $4, DTADDR+3  }      arguments

          CALL    L2HMVE        ... Call L2HMVE
            :
```

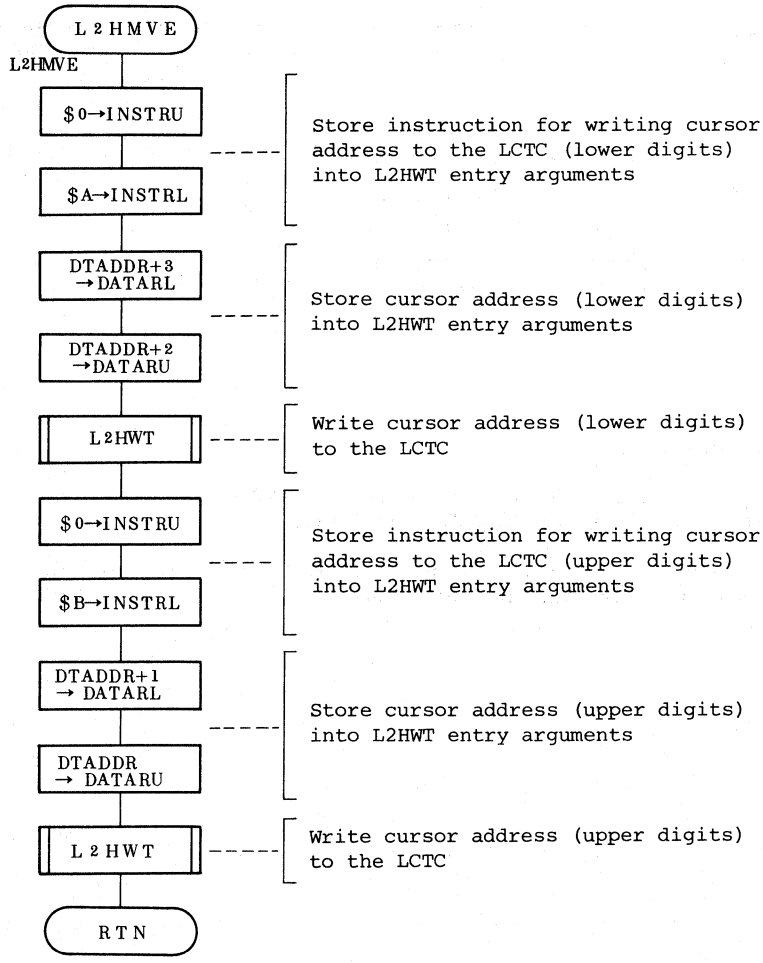
5. Basic Operation

(1) To effect display at any location on LM200, the cursor address must be written before writing display data.

(2) The cursor address consists of four digits. Program module L2HWT is used to first set the lower digit and then the upper digit.

Program Module Name: Move Cursor	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: L2HMVE

Flowchart:



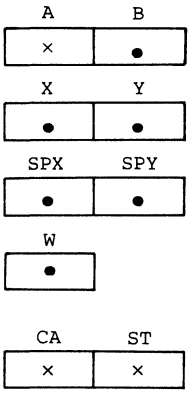
Program Module Name: Write Data	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: L2HWT

Function: Writes data or instructions to the LCTC through an HMCS404C I/O port under the control of signals RS, R/W and E.

Arguments:

Contents		Storage Location	1 digit = 4 bits No. of Digits
Entry	LCTC instruction	INSTRU (RAM)	1
		INSTRL (RAM)	1
	LCTC data	DATARU (RAM)	1
		DATARL (RAM)	1
Returns	—	—	—

Changes in CPU Registers and Flags:



● : Not Affected
× : Undefined
↕ : Result

Specifications:

1 word = 10 bits

ROM (Words):	22
RAM (Digits):	4
Stack (Digits):	4
No. of cycles:	96
Reentrant:	No
Relocatable:	No
Interrupt OK?:	Yes

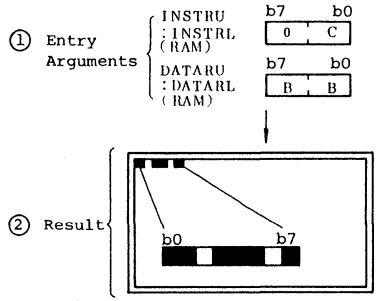
Description:

1. Function Details

(1) Argument details

INSTRU,INSTRL(RAM): Holds value to initialize the LCTC instruction register

DATARU,DATARL(RAM): Holds value to initialize the LCTC data register



Note: Display position is different from the cursor address

Fig. 13.8. Example of L2HWT Execution

Specifications Notes: The number of cycles indicated is the minimum number of cycles required by subroutine L2HBSY.

Program Module Name: Write Data	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: L2HWT

Description:

(2) Fig. 13.8 shows an example of program module L2HWT execution, in which display data $BB is written to the LCTC.

(3) L2HWT uses the subroutines shown in Table 13.6.

Table 13.6. Subroutines Used by Module L2HWT

Subroutine Name	Label Name	Function
Check Busy Flag	L2HBSY	Checks LCTC busy flag

2. RAM Allocation

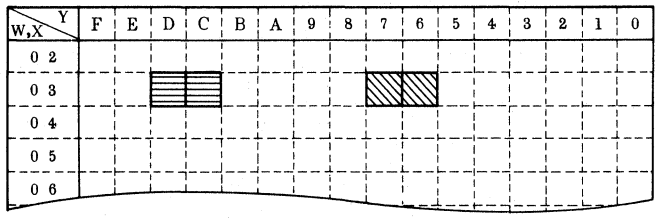
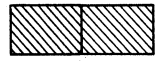
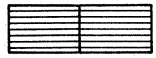


Fig. 13.9. RAM Allocation

Label	R A M	Description
INSTRU: INSTRL(RAM)	b7 b0  MD($037, $036)	Holds value to initialize the LCTC instruction register
DATARU: DATARL(RAM)	b7 b0  MD($03D, $03C)	Holds value to initialize the LCTC data register

Program Module Name: Write Data	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: L2HWT

Description:

3. Sample Application

```
WORK1U    EQU    $037      ┐
WORK1L    EQU    $036      ┘ ..... Allocates RAM area in the user program to store value to initialize the instruction register
WORK2U    EQU    $03D      ┐
WORK2L    EQU    $03C      ┘ ..... Allocates RAM area in the user program to store value to initialize the data register
           :
          LAMD    WORK1L    ┐
          LMAD    INSTRL    │
          LAMD    WORK1U    ├ ..... Store value to initialize the instruction register into entry arguments
          LMAD    INSTRU    ┘
          LAMD    WORK2L    ┐
          LMAD    DATARL    │
          LAMD    WORK2U    ├ ..... Store value to initialize the data register into entry arguments
          LMAD    DATARU    ┘
          CALL    L2HWT     ..... Calls L2HWT
           :
```

Program Module Name: Write Data	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: L2HWT

Description:

4. Basic Operation

(1) When writing data to the LCTC, the microcomputer must store data in the LCTC instruction and data registers in corresponding pairs. To switch between instruction and data registers, the RS signal is used (high for instruction and low for data).

Fig. 13.10 shows the flowchart for writing data to the LCTC.

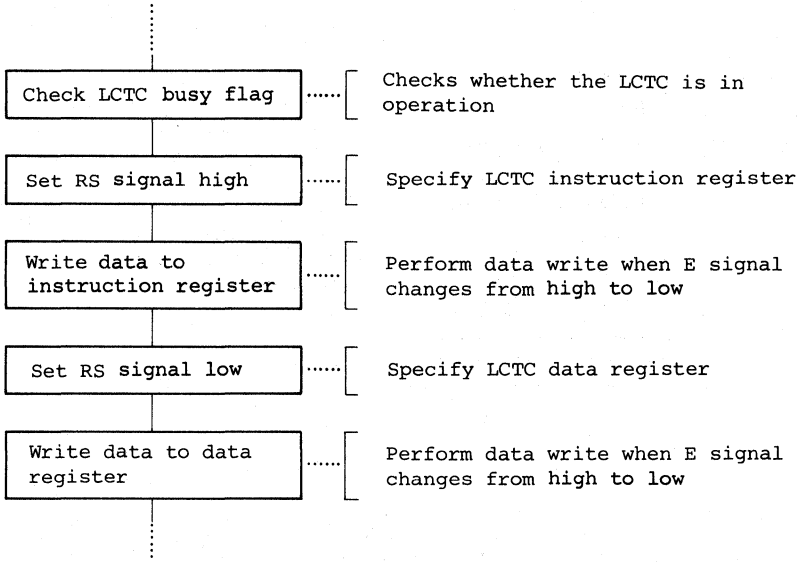


Fig. 13.10. Procedure for Writing Data to LCTC

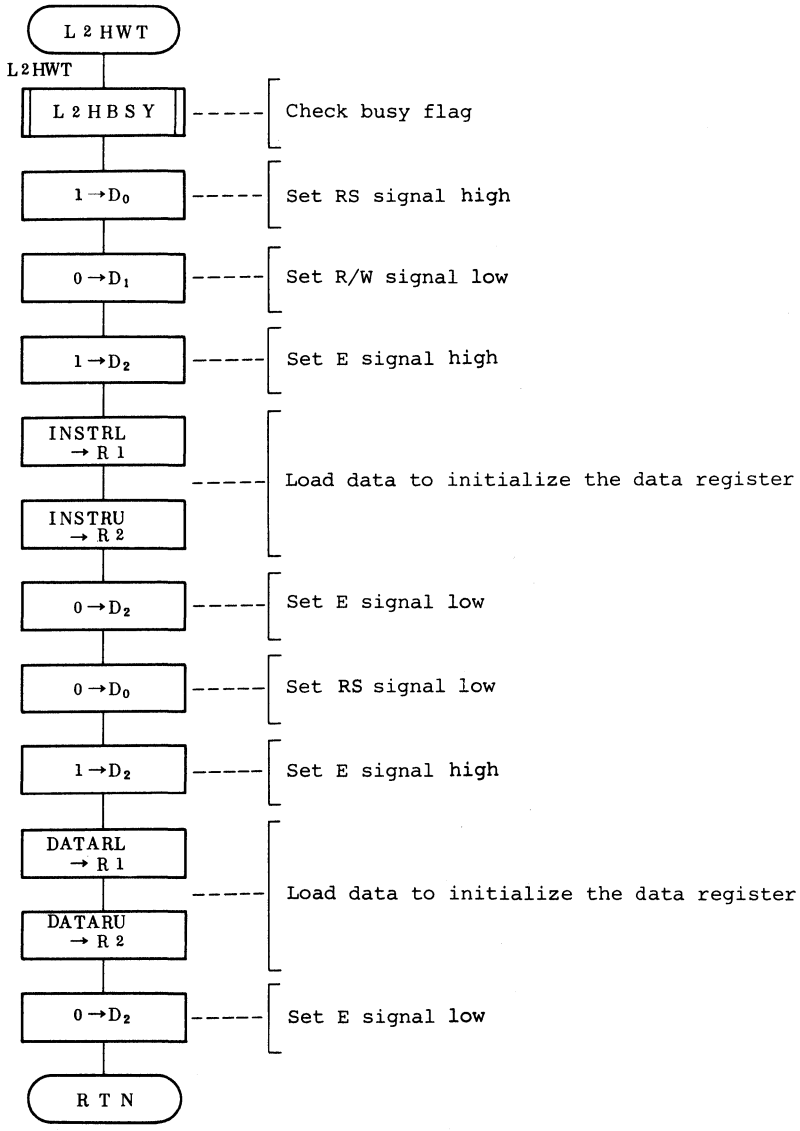
(2) When the LCTC is in operation, the microcomputer cannot write data to it. However, LCTC operation can be distinguished by the LCTC busy flag. Thus in this case, program module L2HWT calls subroutine L2HBSY to check the busy flag and determine whether the LCTC is in operation.

Busy flag = 1: LCTC is in operation, data cannot be written

Busy flag = 0: Data can be written to the LCTC

Program Module Name: Write Data	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: L2HWT

Flowchart:



13.4 SUBROUTINE DESCRIPTION

Subroutine Name:	MCU:	Label:
Check Busy Flag	HMCS402C/ HMCS404C/HMCS408C	L2HBSY

Function:

Checks whether LCTC is in operation and waits for ready state.

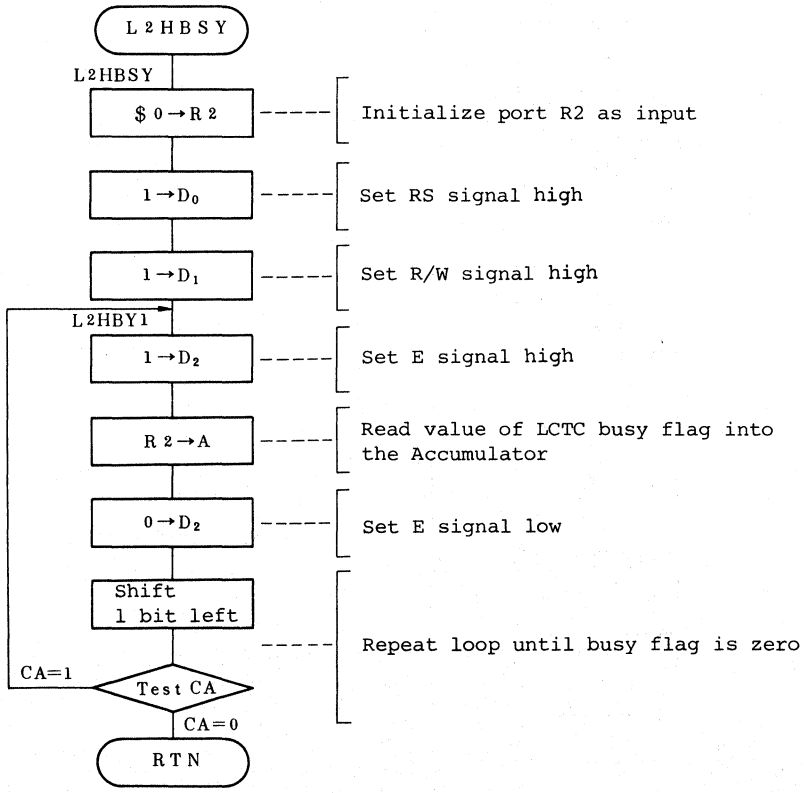
Basic Operation:

(1) Since the LCTC cannot be accessed by the microcomputer while it is in operation, the LCTC busy flag must be checked.

(2) The RS, R/W and E signals are controlled by port R2 to read the busy flag.

Program Module Using This Subroutine: L2HWT, L2HINT, L2HMVE

Flowchart:



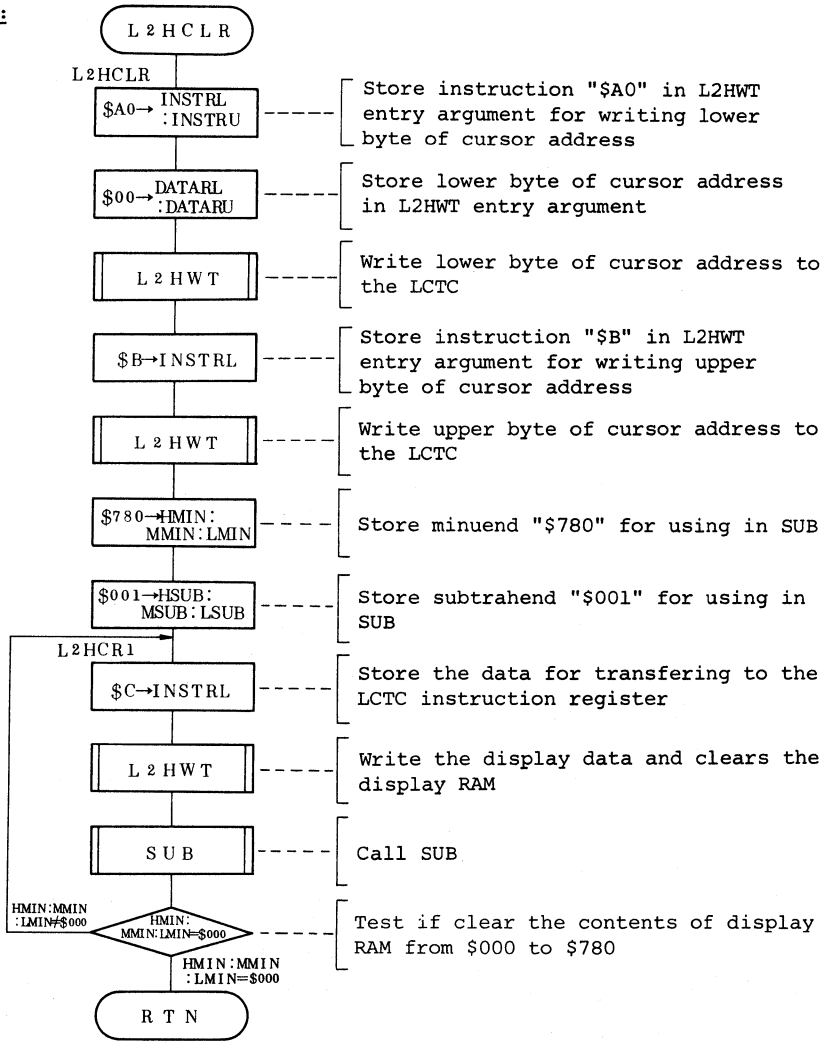
Subroutine Name: Clear Display	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: L2HCLR

Function: Clears the contents of display RAM in the LM200 and clears the liquid crystal display.

Basic Operation: Initializes cursor address to "$0000", calls L2HWT and stores "$00" consecutively up to "$0780" in RAM.

Program Module Using This Subroutine: L2HINT

Flowchart:



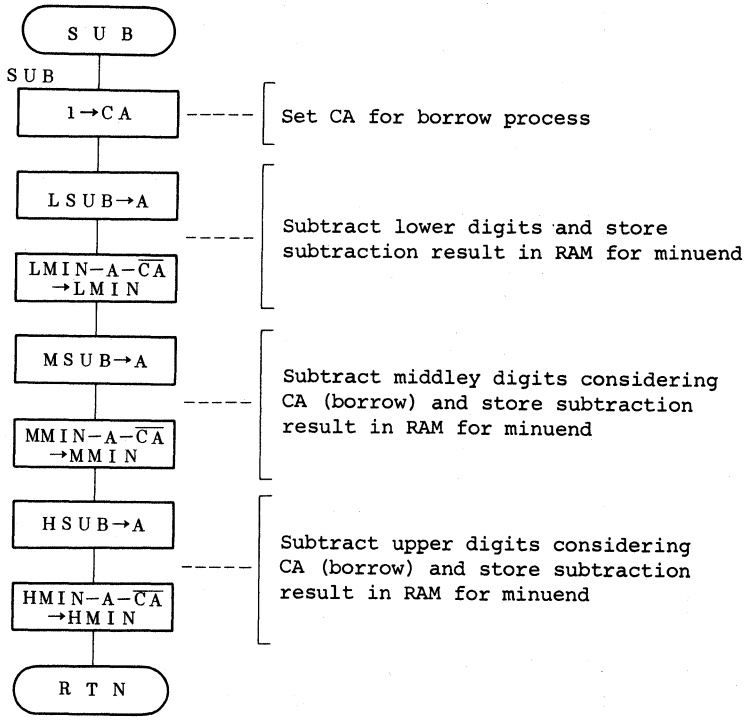
Subroutine Name:	MCU:	Label:
Subtract 12 Bits Binary Data	HMCS402C/ HMCS404C/HMCS408C	SUB

Function: Performs subtraction of 12 bits binary data in RAM, and stores result in RAM.

Basic Operation: When substraction is performed with 2 or more digits, the same operation sequence is repreated for each digit. Subtraction result is stored in RAM for minuend.

Program Module Using This Subroutine: None

Flowchart:



13.5 PROGRAM LISTING

```
ST-NO   OBJECT   ADRS   SOURCE STATEMENTS

00001   100      0000           LLEN     132
00002                           TITLE    HD61830 (LM200) GRAPHIC MODE
00003                  *
00004                  ****   RAM ALLOCATION   **********************************
00005                  *
00006                  COUNTU   EQU      $030          UPPER DIGIT OF ROW COUNTER
00007                  COUNTL   EQU      $031          LOWER DIGIT OF ROW COUNTER
00008                  DTADDR   EQU      $032          CURSOR ADDRESS
00009                  INSTRU   EQU      $036          UPPER LCTC INSTRUCTION REGISTER DATA
00010                  INSTRL   EQU      $037          LOWER LCTC INSTRUCTION REGISTER DATA
00011                  DSPCTU   EQU      $038          UPPER DIGIT OF DISPLAY DATA POINTER
00012                  DSPCTL   EQU      $039          LOWER DIGIT OF DISPLAY DATA POINTER
00013                  INTA     EQU      $03A          LOWER DIGIT OF SOURCE STARTING ADDR
00014                  INTB     EQU      $03B          UPPER DIGIT OF SOURCE STARTING ADDR
00015                  DATARU   EQU      $03C          UPPER LCTC DATA REGISTER DATA
00016                  DATARL   EQU      $03D          LOWER LCTC DATA REGISTER DATA
00017                  HMIN     EQU      $03E          UPPER DIGIT OF MINUEND
00018                  MMIN     EQU      $03F          MIDDLE MINUEND
00019                  LMIN     EQU      $040          LOWER DIGIT OF MINUEND
00020                  HSUB     EQU      $041          UPPER DIGIT OF SUBTRAHEND
00021                  MSUB     EQU      $042          MIDDLE DIGIT OF SUBTRAHEND
00022                  LSUB     EQU      $043          LOWER DIGIT OF SUBTRAHEND
00023                  ************************************************************
00024                  *                                                          *
00025                  *               VECTOR ADDRESSES                           *
00026                  *                                                          *
00027                  ************************************************************
00028                  *
00029                           ORG      $0000
00030                  *
00031   150 010  0000           JMPL     L2HMN         RESET
00032   150 010  0002           JMPL     L2HMN         INT0
00033   150 010  0004           JMPL     L2HMN         INT1
00034   150 010  0006           JMPL     L2HMN         TIMER-A
00035   150 010  0008           JMPL     L2HMN         TIMER-B
00036   000      000A           NOP
00037   000      000B           NOP
00038   150 010  000C           JMPL     L2HMN         SERIAL
00039                  ************************************************************
00040                  *                                                          *
00041                  *               MAIN PROGRAM : L2HMN                       *
00042                  *                                                          *
00043                  ************************************************************
00044                  *
00045                           ORG      $0010
00046                  *
00047   0F0      0010  L2HMN    LWI      $0            INITIALIZE W REGISTER
00048   160 06F  0011           CALL     L2HINT        INITIALIZE LCTC FOR GRAPHIC MODE
00049   160 0C1  0013           CALL     L2HCLR        CLEAR DISPLAY
00050   210      0015           LYI      $0            CLEAR COLUMN COUNTER
00051   1A0 038  0016           LMID     $0,DSPCTU     CLEAR DISPLAY DATA POINTER
00052   1A0 039  0018           LMID     $0,DSPCTL
00053   1A0 030  001A           LMID     $0,COUNTU     CLEAR ROW COUNTER
00054   1A0 031  001C           LMID     $0,COUNTL
00055   1A0 032  001E           LMID     $0,DTADDR     STORE CURSOR ADDRESS(UPPER)
00056   1A0 033  0020           LMID     $0,DTADDR+1
00057   1A8 034  0022           LMID     $8,DTADDR+2   STORE CURSOR ADDRESS(LOWER)
```

```
00058  1A4 035  0024             LMID     $4,DTADDR+3
00059  160 0A4  0026  L2HMN1     CALL     L2HMVE          WRITE CURSOR ADDRESS TO LCTC
00060  1AC 037  0028             LMID     $C,INSTRL       LOAD INSTRUCTION
00061  1A0 036  002A             LMID     $0,INSTRU
00062  190 038  002C  L2HMN2     LAMD     DSPCTU          LOAD DISPLAY DATA
00063  0C8      002E             LBA
00064  190 039  002F             LAMD     DSPCTL
00065  1B2      0031             P        $2              PATTERN
00066  194 03D  0032             LMAD     DATARL
00067  048      0034             LAB
00068  194 03C  0035             LMAD     DATARU
00069  160 08E  0037             CALL     L2HWT           WRITE DISPLAY DATA TO LCTC
00070  0EC      0039             REC                      INCREMENT POINTER
00071  231      003A             LAI      $1
00072  118 039  003B             AMCD     DSPCTL
00073  194 039  003D             LMAD     DSPCTL
00074  230      003F             LAI      $0
00075  118 038  0040             AMCD     DSPCTU
00076  194 038  0042             LMAD     DSPCTU
00077  05C      0044             IY                       INCREMENT COLUMN COUNTER
00078  075      0045             YNEI     $5              TEST IF COLUMN COUNTER =5
00079  32C      0046             BR       L2HMN2
00080  210      0047             LYI      $0              CLEAR COLUMN COUNTER
00081  0EC      0048             REC                      ADD CURSOR ADDRESS TO 30
00082  23E      0049             LAI      $E
00083  118 035  004A             AMCD     DTADDR+3
00084  194 035  004C             LMAD     DTADDR+3
00085  231      004E             LAI      $1
00086  118 034  004F             AMCD     DTADDR+2
00087  194 034  0051             LMAD     DTADDR+2
00088  230      0053             LAI      $0
00089  118 033  0054             AMCD     DTADDR+1
00090  194 033  0056             LMAD     DTADDR+1
00091  230      0058             LAI      $0
00092  118 032  0059             AMCD     DTADDR
00093  194 032  005B             LMAD     DTADDR
00094  0EC      005D             REC                      INCREMENT ROW COUNTER
00095  231      005E             LAI      $1
00096  118 031  005F             AMCD     COUNTL
00097  194 031  0061             LMAD     COUNTL
00098  230      0063             LAI      $0
00099  118 030  0064             AMCD     COUNTU
00100  194 030  0066             LMAD     COUNTU
00101  123 030  0068             INEMD    $3,COUNTU       TEST IF ROW COUNTER =50
00102  326      006A             BR       L2HMN1
00103  122 031  006B             INEMD    $2,COUNTL
00104  326      006D             BR       L2HMN1          LOOP UNTIL DISPLAY END
00105  36E      006E  PEND       BR       PEND            END OF PROGRAM
00106                             ********************************************************
00107                             *                                                      *
00108                             *         NAME : L2HINT (INITIALIZE LCTC)              *
00109                             *                                                      *
00110                             ********************************************************
00111                             *                                                      *
00112                             *        ENTRY : NOTHING                               *
00113                             *      RETURNS : NOTHING                               *
00114                             *                                                      *
```

```
00115                   ********************************************************
00116   1A0 03A  006F   L2HINT   LMID     $0,INTA        STORE LOWER DIGIT OF SOURCE ADDRESS
00117   1A5 03B  0071            LMID     $5,INTB        STORE UPPER DIGIT OF SOURCE ADDRESS
00118   160 0EB  0073   L2HIT1   CALL     L2HBSY         CHECK LCTC BUSY FLAG
00119   190 03B  0075            LAMD     INTB
00120   0C8      0077            LBA
00121   190 03A  0078            LAMD     INTA
00122   2E0      007A            SEDD     0              RS=1
00123   261      007B            REDD     1              R/W=0
00124   2E2      007C            SEDD     2              E=1
00125   000      007D            NOP
00126   1B1      007E            P        1              OUTPUT INSTRUCTION BY PATTERN INSTR
00127   000      007F            NOP
00128   262      0080            REDD     2              E=0
00129   281      0081            AI       $1             INCREMENT ADDRESS
00130   260      0082            REDD     0              RS=0
00131   2E2      0083            SEDD     2              E=1
00132   000      0084            NOP
00133   1B1      0085            P        1              OUTPUT DATA BY PATTERN INSTRUCTION
00134   000      0086            NOP
00135   262      0087            REDD     2              E=0
00136   281      0088            AI       $1             INCREMENT ADDRESS
00137   194 03A  0089            LMAD     INTA
00138   38D      008B            BRS      L2HIT2         BRANCH IF INTA=F
00139   373      008C            BRS      L2HIT1         BRANCH IF INTA/=F
00140   010      008D   L2HIT2   RTN
00141                   ********************************************************
00142                   *                                                      *
00143                   *          NAME : L2HWT (WRITE DATA TO LCTC)           *
00144                   *                                                      *
00145                   ********************************************************
00146                   *                                                      *
00147                   *         ENTRY : INSTRU (UPPER HALF OF INSTR TO LCTC) *
00148                   *                 INSTRL (LOWER HALF OF INSTR TO LCTC) *
00149                   *                 DATARU (UPPER DATA TO LCTC)          *
00150                   *                 DATARL (LOWER DATA TO LCTC)          *
00151                   *       RETURNS :  NOTHING                             *
00152                   *                                                      *
00153                   ********************************************************
00154   160 0EB  008E   L2HWT    CALL     L2HBSY         CHECK LCTC BUSY FLAG
00155   2E0      0090            SEDD     0              RS=1
00156   261      0091            REDD     1              R/W=0
00157   2E2      0092            SEDD     2              E=1
00158   190 037  0093            LAMD     INSTRL
00159   2D1      0095            LRA      1              STORE DATA WRITTEN TO INSTRUCTION REG
00160   190 036  0096            LAMD     INSTRU
00161   2D2      0098            LRA      2
00162   262      0099            REDD     2              E=0
00163   260      009A            REDD     0              RS=0
00164   2E2      009B            SEDD     2              E=1
00165   190 03D  009C            LAMD     DATARL
00166   2D1      009E            LRA      1              STORE DATA WRITTEN TO DATA REG
00167   190 03C  009F            LAMD     DATARU
00168   2D2      00A1            LRA      2
00169   262      00A2            REDD     2              E=0
00170   010      00A3            RTN
00171                   ********************************************************
```

```
00172                    *                                                     *
00173                    *          NAME : L2HMVE (SET CURSOR ADDRESS)         *
00174                    *                                                     *
00175                    *******************************************************
00176                    *                                                     *
00177                    *        ENTRY : DTADDR (CURSOR ADDRESS VALUE)        *
00178                    *      RETURNS : NOTHING                              *
00179                    *                                                     *
00180                    *******************************************************
00181   1A0 036  00A4    L2HMVE   LMID     $0,INSTRU      STORE INSTRUCTION
00182   1AA 037  00A6             LMID     $A,INSTRL
00183   190 035  00A8             LAMD     DTADDR+3       STORE DATA
00184   194 03D  00AA             LMAD     DATARL
00185   190 034  00AC             LAMD     DTADDR+2
00186   194 03C  00AE             LMAD     DATARU
00187   160 08E  00B0             CALL     L2HWT          WRITE LOWER CURSOR ADDR TO LCTC
00188   1A0 036  00B2             LMID     $0,INSTRU      STORE INSTRUCTION
00189   1AB 037  00B4             LMID     $B,INSTRL
00190   190 033  00B6             LAMD     DTADDR+1       STORE DATA
00191   194 03D  00B8             LMAD     DATARL
00192   190 032  00BA             LAMD     DTADDR
00193   194 03C  00BC             LMAD     DATARU
00194   160 08E  00BE             CALL     L2HWT          WRITE UPPER CURSOR ADDR TO LCTC
00195   010      00C0             RTN
00196                    *******************************************************
00197                    *                                                     *
00198                    *          NAME : L2HCLR (CLEAR DISPLAY)              *
00199                    *                                                     *
00200                    *******************************************************
00201   1AA 037  00C1    L2HCLR   LMID     $A,INSTRL      STORE INSTRUCTION
00202   1A0 036  00C3             LMID     $0,INSTRU
00203   1A0 03D  00C5             LMID     $0,DATARL      STORE LOWER CURSOR ADDRESS
00204   1A0 03C  00C7             LMID     $0,DATARU
00205   160 08E  00C9             CALL     L2HWT          WRITE LOWER CURSOR ADDR TO LCTC
00206   1AB 037  00CB             LMID     $B,INSTRL      STORE INSTRUCTION
00207   160 08E  00CD             CALL     L2HWT          WRITE UPPER CURSOR ADDR TO LCTC
00208   1A7 03E  00CF             LMID     $7,HMIN        STORE MINUEND
00209   1A8 03F  00D1             LMID     $8,MMIN
00210   1A0 040  00D3             LMID     $0,LMIN
00211   1A0 041  00D5             LMID     $0,HSUB        STORE SUBTRAHEND
00212   1A0 042  00D7             LMID     $0,MSUB
00213   1A1 043  00D9             LMID     $1,LSUB
00214   1AC 037  00DB    L2HCR1   LMID     $C,INSTRL      STORE DATA
00215   160 08E  00DD             CALL     L2HWT          WRITE DISPLAY DATA($00) TO RAM
00216   160 0F6  00DF             CALL     SUB
00217   120 03E  00E1             INEMD    $0,HMIN        TEST IF DISPLAY RAM IS CLEARED
00218   3DB      00E3             BRS      L2HCR1
00219   120 03F  00E4             INEMD    $0,MMIN
00220   3DB      00E6             BRS      L2HCR1
00221   120 040  00E7             INEMD    $0,LMIN
00222   3DB      00E9             BRS      L2HCR1
00223   010      00EA             RTN
00224                    *******************************************************
00225                    *                                                     *
00226                    *          NAME : L2HBSY (CHECK BUSY FLAG)            *
00227                    *                                                     *
00228                    *******************************************************
```

```
00229   230         00EB   L2HBSY   LAI       $0
00230   2D2         00EC             LRA       2              SELECT R2 AS INPUT
00231   2E0         00ED             SEDD      0              RS=1
00232   2E1         00EE             SEDD      1              R/W=1
00233   2E2         00EF   L2HBY1    SEDD      2              E=1
00234   252         00F0             LAR       2              READ LCTC BUSY FLAG
00235   262         00F1             REDD      2              E=0
00236   0A1         00F2             RDTL
00237   06F         00F3             TC
00238   3EF         00F4             BRS       L2HBY1         LOOP UNTIL BUSY FLAG=1
00239   010         00F5             RTN
00240                      ************************************************************
00241                      *                                                          *
00242                      *          NAME : SUB (SUBTRACT 12-BIT BINARY DATA)        *
00243                      *                                                          *
00244                      ************************************************************
00245   0EF         00F6   SUB       SEC                      SET CARRY
00246   190 043     00F7             LAMD      LSUB
00247   198 040     00F9             SMCD      LMIN           SUBTRACT LOWER DIGITS
00248   194 040     00FB             LMAD      LMIN           LMIN-LSUB->LMIN
00249   190 042     00FD             LAMD      MSUB
00250   198 03F     00FF             SMCD      MMIN           SUBTRACT MIDDLE DIGITS
00251   194 03F     0101             LMAD      MMIN           MMIN-MSUB->MMIN
00252   190 041     0103             LAMD      HSUB
00253   198 03E     0105             SMCD      HMIN           SUBTRACT UPPER DIGITS
00254   194 03E     0107             LMAD      HMIN           HMIN-HSUB->HMIN
00255   010         0109             RTN
00256                      ************************************************************
00257                      *                                                          *
00258                      *       DATA TABLE (DATA TO INITIALIZE LCTC)               *
00259                      *                                                          *
00260                      ************************************************************
00261                      *
00262                               ORG       $0150
00263                      *
00264   200         0150             DC        $200
00265   232         0151             DC        $232
00266   201         0152             DC        $201
00267   207         0153             DC        $207
00268   202         0154             DC        $202
00269   21D         0155             DC        $21D
00270   203         0156             DC        $203
00271   21F         0157             DC        $21F
00272   208         0158             DC        $208
00273   200         0159             DC        $200
00274   209         015A             DC        $209
00275   200         015B             DC        $200
00276   20A         015C             DC        $20A
00277   200         015D             DC        $200
00278   20B         015E             DC        $20B
00279   200         015F             DC        $200
00280                      ************************************************************
00281                      *                                                          *
00282                      *       DATA TABLE (DISPLAY DATA)                          *
00283                      *                                                          *
00284                      ************************************************************
00285                      *
```

```
00286                                   ORG     $0200
00287                           *
00288   100     0200            DC      $100
00289   100     0201            DC      $100
00290   180     0202            DC      $180
00291   10F     0203            DC      $10F
00292   100     0204            DC      $100
00293   1C0     0205            DC      $1C0
00294   100     0206            DC      $100
00295   1C0     0207            DC      $1C0
00296   11F     0208            DC      $11F
00297   100     0209            DC      $100
00298   1E0     020A            DC      $1E0
00299   101     020B            DC      $101
00300   1C0     020C            DC      $1C0
00301   11F     020D            DC      $11F
00302   100     020E            DC      $100
00303   1F8     020F            DC      $1F8
00304   103     0210            DC      $103
00305   1E7     0211            DC      $1E7
00306   13F     0212            DC      $13F
00307   100     0213            DC      $100
00308   1F8     0214            DC      $1F8
00309   1C7     0215            DC      $1C7
00310   1F8     0216            DC      $1F8
00311   13F     0217            DC      $13F
00312   100     0218            DC      $100
00313   1FC     0219            DC      $1FC
00314   137     021A            DC      $137
00315   1C0     021B            DC      $1C0
00316   17F     021C            DC      $17F
00317   100     021D            DC      $100
00318   1FE     021E            DC      $1FE
00319   10F     021F            DC      $10F
00320   180     0220            DC      $180
00321   17F     0221            DC      $17F
00322   100     0222            DC      $100
00323   1FE     0223            DC      $1FE
00324   107     0224            DC      $107
00325   180     0225            DC      $180
00326   13F     0226            DC      $13F
00327   100     0227            DC      $100
00328   1FE     0228            DC      $1FE
00329   103     0229            DC      $103
00330   1C3     022A            DC      $1C3
00331   13F     022B            DC      $13F
00332   100     022C            DC      $100
00333   1FF     022D            DC      $1FF
00334   183     022E            DC      $183
00335   1E7     022F            DC      $1E7
00336   13F     0230            DC      $13F
00337   100     0231            DC      $100
00338   1FF     0232            DC      $1FF
00339   1C1     0233            DC      $1C1
00340   12C     0234            DC      $12C
00341   11C     0235            DC      $11C
00342   100     0236            DC      $100
```

```
00343  1FE  0237  DC  $1FE
00344  1E0  0238  DC  $1E0
00345  168  0239  DC  $168
00346  12D  023A  DC  $12D
00347  100  023B  DC  $100
00348  17E  023C  DC  $17E
00349  170  023D  DC  $170
00350  1CE  023E  DC  $1CE
00351  14C  023F  DC  $14C
00352  100  0240  DC  $100
00353  13E  0241  DC  $13E
00354  1F0  0242  DC  $1F0
00355  1CC  0243  DC  $1CC
00356  18E  0244  DC  $18E
00357  100  0245  DC  $100
00358  110  0246  DC  $110
00359  1F0  0247  DC  $1F0
00360  10F  0248  DC  $10F
00361  10F  0249  DC  $10F
00362  101  024A  DC  $101
00363  110  024B  DC  $110
00364  1E0  024C  DC  $1E0
00365  187  024D  DC  $187
00366  113  024E  DC  $113
00367  101  024F  DC  $101
00368  110  0250  DC  $110
00369  1E0  0251  DC  $1E0
00370  183  0252  DC  $183
00371  11B  0253  DC  $11B
00372  102  0254  DC  $102
00373  108  0255  DC  $108
00374  180  0256  DC  $180
00375  101  0257  DC  $101
00376  10F  0258  DC  $10F
00377  102  0259  DC  $102
00378  108  025A  DC  $108
00379  100  025B  DC  $100
00380  100  025C  DC  $100
00381  106  025D  DC  $106
00382  102  025E  DC  $102
00383  108  025F  DC  $108
00384  100  0260  DC  $100
00385  100  0261  DC  $100
00386  100  0262  DC  $100
00387  102  0263  DC  $102
00388  108  0264  DC  $108
00389  100  0265  DC  $100
00390  100  0266  DC  $100
00391  100  0267  DC  $100
00392  101  0268  DC  $101
00393  110  0269  DC  $110
00394  100  026A  DC  $100
00395  100  026B  DC  $100
00396  100  026C  DC  $100
00397  101  026D  DC  $101
00398  110  026E  DC  $110
00399  100  026F  DC  $100
```

```
00400  140  0270  DC  $140
00401  108  0271  DC  $108
00402  101  0272  DC  $101
00403  120  0273  DC  $120
00404  180  0274  DC  $180
00405  1A0  0275  DC  $1A0
00406  184  0276  DC  $184
00407  100  0277  DC  $100
00408  120  0278  DC  $120
00409  100  0279  DC  $100
00410  113  027A  DC  $113
00411  1C3  027B  DC  $1C3
00412  101  027C  DC  $101
00413  140  027D  DC  $140
00414  100  027E  DC  $100
00415  10C  027F  DC  $10C
00416  1C0  0280  DC  $1C0
00417  101  0281  DC  $101
00418  180  0282  DC  $180
00419  101  0283  DC  $101
00420  100  0284  DC  $100
00421  1F8  0285  DC  $1F8
00422  103  0286  DC  $103
00423  100  0287  DC  $100
00424  106  0288  DC  $106
00425  1C0  0289  DC  $1C0
00426  1F7  028A  DC  $1F7
00427  107  028B  DC  $107
00428  100  028C  DC  $100
00429  102  028D  DC  $102
00430  13C  028E  DC  $13C
00431  1E0  028F  DC  $1E0
00432  107  0290  DC  $107
00433  100  0291  DC  $100
00434  101  0292  DC  $101
00435  140  0293  DC  $140
00436  1F8  0294  DC  $1F8
00437  101  0295  DC  $101
00438  100  0296  DC  $100
00439  103  0297  DC  $103
00440  180  0298  DC  $180
00441  1FF  0299  DC  $1FF
00442  103  029A  DC  $103
00443  180  029B  DC  $180
00444  107  029C  DC  $107
00445  100  029D  DC  $100
00446  1FF  029E  DC  $1FF
00447  103  029F  DC  $103
00448  180  02A0  DC  $180
00449  10F  02A1  DC  $10F
00450  100  02A2  DC  $100
00451  19F  02A3  DC  $19F
00452  103  02A4  DC  $103
00453  1C0  02A5  DC  $1C0
00454  10F  02A6  DC  $10F
00455  100  02A7  DC  $100
00456  10E  02A8  DC  $10E
```

```
00457  100  02A9  DC  $100
00458  1C0  02AA  DC  $1C0
00459  10F  02AB  DC  $10F
00460  100  02AC  DC  $100
00461  102  02AD  DC  $102
00462  100  02AE  DC  $100
00463  1C0  02AF  DC  $1C0
00464  11F  02B0  DC  $11F
00465  100  02B1  DC  $100
00466  102  02B2  DC  $102
00467  100  02B3  DC  $100
00468  1E0  02B4  DC  $1E0
00469  11F  02B5  DC  $11F
00470  100  02B6  DC  $100
00471  103  02B7  DC  $103
00472  100  02B8  DC  $100
00473  1E0  02B9  DC  $1E0
00474  10F  02BA  DC  $10F
00475  1C6  02BB  DC  $1C6
00476  107  02BC  DC  $107
00477  100  02BD  DC  $100
00478  1E0  02BE  DC  $1E0
00479  10F  02BF  DC  $10F
00480  1DF  02C0  DC  $1DF
00481  11E  02C1  DC  $11E
00482  100  02C2  DC  $100
00483  1E0  02C3  DC  $1E0
00484  1AF  02C4  DC  $1AF
00485  1FA  02C5  DC  $1FA
00486  11A  02C6  DC  $11A
00487  100  02C7  DC  $100
00488  1E0  02C8  DC  $1E0
00489  1EF  02C9  DC  $1EF
00490  1FA  02CA  DC  $1FA
00491  11A  02CB  DC  $11A
00492  100  02CC  DC  $100
00493  1D0  02CD  DC  $1D0
00494  177  02CE  DC  $177
00495  1FB  02CF  DC  $1FB
00496  11A  02D0  DC  $11A
00497  100  02D1  DC  $100
00498  1A8  02D2  DC  $1A8
00499  17B  02D3  DC  $17B
00500  1FB  02D4  DC  $1FB
00501  13F  02D5  DC  $13F
00502  100  02D6  DC  $100
00503  164  02D7  DC  $164
00504  1FD  02D8  DC  $1FD
00505  1FF  02D9  DC  $1FF
00506  13F  02DA  DC  $13F
00507  100  02DB  DC  $100
00508  1C2  02DC  DC  $1C2
00509  1FE  02DD  DC  $1FE
00510  1FF  02DE  DC  $1FF
00511  13F  02DF  DC  $13F
00512  100  02E0  DC  $100
00513  1C2  02E1  DC  $1C2
```

```
00514   1FF     02E2            DC      $1FF
00515   1FF     02E3            DC      $1FF
00516   11F     02E4            DC      $11F
00517   100     02E5            DC      $100
00518   184     02E6            DC      $184
00519   1FF     02E7            DC      $1FF
00520   1FF     02E8            DC      $1FF
00521   11F     02E9            DC      $11F
00522   100     02EA            DC      $100
00523   198     02EB            DC      $198
00524   1FF     02EC            DC      $1FF
00525   1FF     02ED            DC      $1FF
00526   10F     02EE            DC      $10F
00527   100     02EF            DC      $100
00528   160     02F0            DC      $160
00529   1FF     02F1            DC      $1FF
00530   1FF     02F2            DC      $1FF
00531   10F     02F3            DC      $10F
00532   100     02F4            DC      $100
00533   100     02F5            DC      $100
00534   1FE     02F6            DC      $1FE
00535   1DF     02F7            DC      $1DF
00536   107     02F8            DC      $107
00537   100     02F9            DC      $100
00538                   *
00539                           END
```

SECTION 14. LIQUID CRYSTAL MODULE (CYCLE MEASUREMENT)

14.1 HARDWARE DESCRIPTION

14.1.1 Function

Controls LCD module H2570 and displays "CMOS MCU HMCS400" on the liquid crystal display.

14.1.2 Microcomputer Operation

(1) Controls HD44780 (hereafter abbreviated LCD-II) data bus through ports R4 and R5.

(2) Controls LCD-II control signals (Signals RS, R/W and E) through port D.

(3) To control LCD-II data bus and control signals by HMCS404C software, there are no restrictions in terms of timing.

(4) From HMCS404C display data is transmitted to LCD-II in the form of ASCII code. Liquid crystal driver HD44100 and the liquid crystal display are automatically controlled by LCD-II which is in turn controlled by the HMCS404C.

14.1.3 Peripheral Devices

(1) LCD controller driver HD44780 (LCD-II): Controls dot matrix LCD of LCD module H2570.

(2) LCD driver HD44100: Drives LCD of LCD module H2570.

(3) LCD module H2570: Provides a display of 16 characters × 1 row.

14.1.4 Circuit Diagram

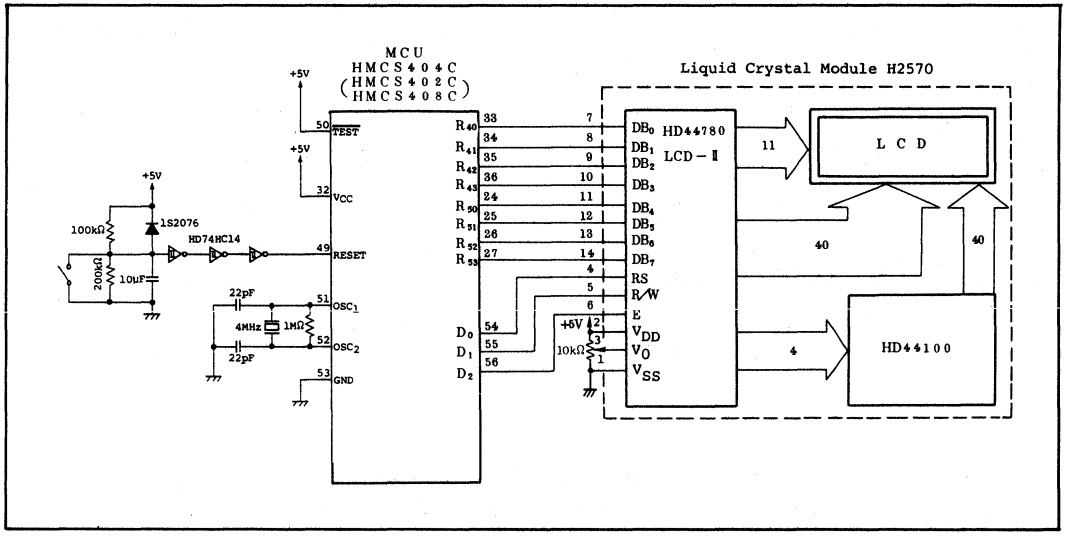


Fig. 14.1. H2570 Control Circuit

14.1.5 Pin Functions

Pin functions at the connecting interface of HMCS404C and LCD-II are shown in Table 14.1.

Table 14.1. Pin Functions

Pin Name (HMCS404C)	Input/ Output	Active level	Function	Pin Name (LCD-II)
D_0	Input/ Output	Low	Selects instruction register	RS
		High	Selects data register	
D_1		Low	Data writing (Microcomputer → LCD-II)	R/W
		High	Data reading (Microcomputer ← LCD-II)	
D_2		High	Enable signal	E
R_{40}		-	Data line	DB_0
R_{41}		-		DB_1
R_{42}		-		DB_2
R_{43}		-		DB_3
R_{50}		-		DB_4
R_{51}		-		DB_5
R_{52}		-		DB_6
R_{53}				DB_7

14.1.6 Hardware Operation

Control signals from both the HMCS404C and LCD-II is performed with the timing shown in Fig. 14.2.

① Data is written to LCTC at the falling edge of E.

② Data from LCD-II can be read during period T_1.

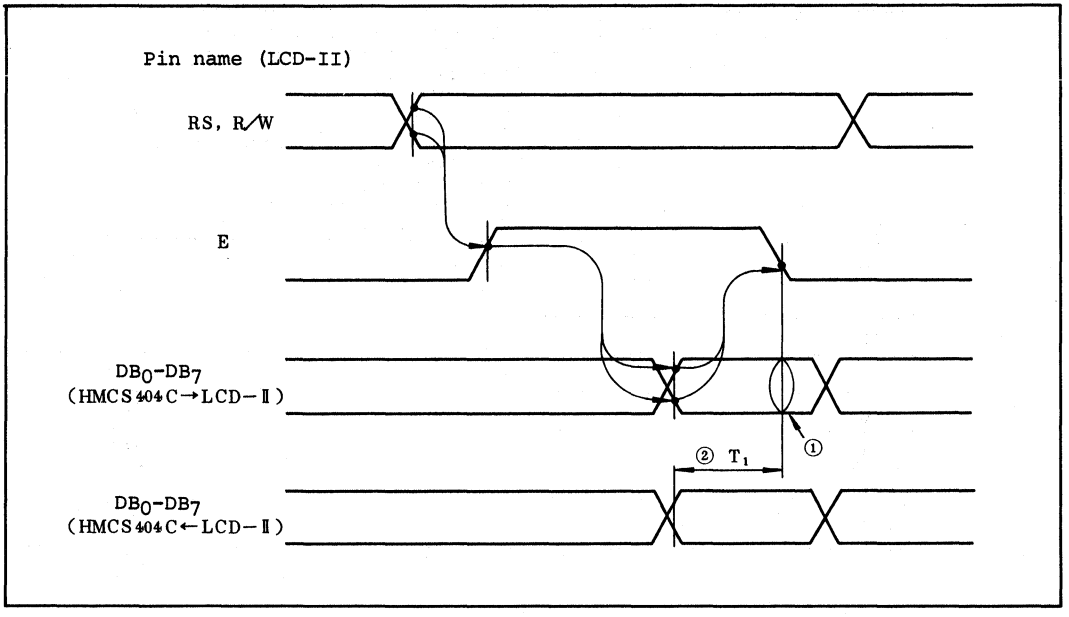


Fig. 14.2. HMCS404C ↔ LCD-II Interface Timing Chart

14.2 SOFTWARE DESCRIPTION

14.2.1 Program Module Configuration

Fig. 14.3 shows the program module configuration for performing display on the liquid crystal.

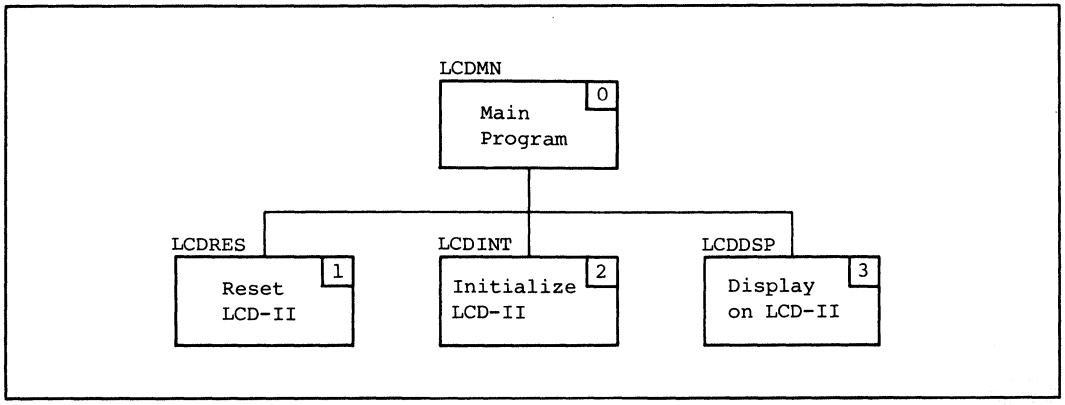


Fig. 14.3. Program Module Configuration

14.2.2 Program Module Functions

Outline of program module functions is shown in Table 14.2.

Table 14.2. Program Module Functions

No.	Program Module Name	Label	Function
0	Main Program	LCDMN	Main program which conducts character display on H2570
1	Reset LCD-II	LCDRES	Resets LCD-II according to instructions
2	Initialize LCD-II	LCDINT	Initializes LCD-II display mode
3	Display on LCD	LCDDSP	Sends ASCII code to LCD-II and displays on H2570

14.2.3 Program Module Process Flow (Main Program)

An example of conducting character display on H2570 using the module shown in Fig. 14.3 is shown in Fig. 14.4. Execution of the main program of Fig. 14.4 results in liquid crystal demonstration display as shown in Fig. 14.5.

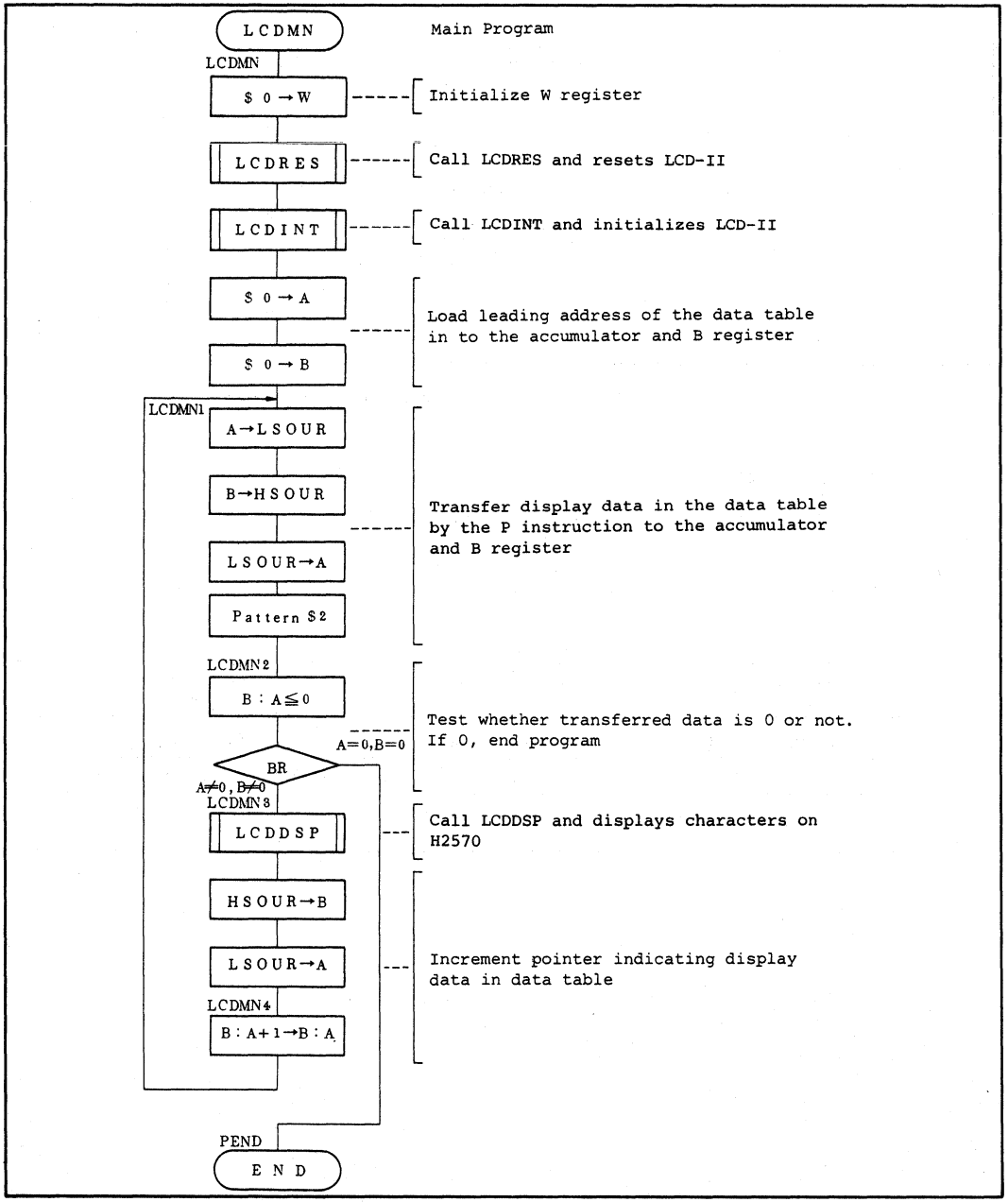


Fig. 14.4. Example of Program Module Sample Application

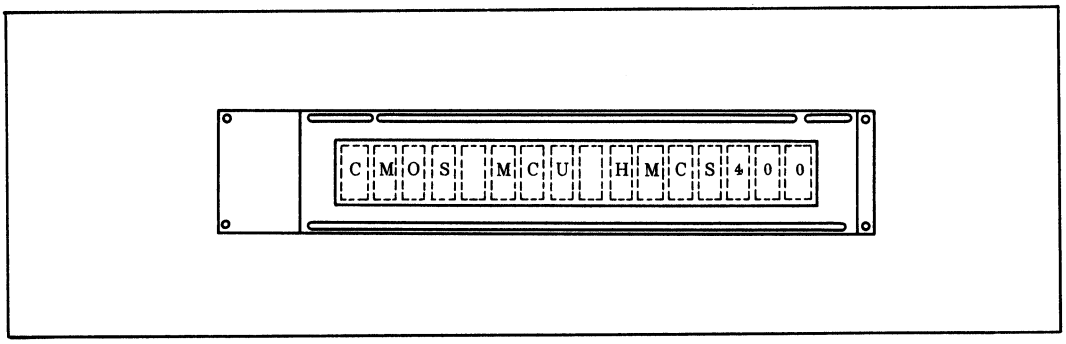


Fig. 14.5. Example of Liquid Crystal Display

14.3 PROGRAM MODULE DESCRIPTION

Program Module Name: Reset LCD-II	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: LCDRES

Function: Resets LCD-II by instruction.

Arguments: None

Changes in CPU Registers and Flags:

A	B
×	×

X	Y
●	×

SPX	SPY
●	×

W
●

CA	ST
×	×

● : Not Affected
× : Undefined
↕ : Result

Specifications:

1 word = 10 bits

ROM (Words):	26
RAM (Digits):	0
Stack (Digits):	0
No. of cycles:	33249
Reentrant:	No
Relocatable:	No
Interrupt OK?:	No

Description:

1. Function Details

   (1) There are no arguments in LCDRES.

   (2) LCD-II is to be reset without fail, even if power source requirements for normal operation of the built-in reset circuit are not being met.

   (3) Program module and subroutines are not being used by program module LCDRES.

2. User Notes: LCDRES is to be executed immediately following turning on of LCD-II power source.

Specifications Notes:

Program Module Name: Reset LCD-II	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: LCDRES

Description:

3. RAM Allocation

RAM is not used by program module LCDRES.

4. Sample Application

```
LCD-II power source ON
        :
    CALL    LCDRES  ·····  Call LCDRES
        :
```

5. Basic Operation

(1) LCD-II may, by the process shown in Fig. 14.6, be reset by instruction.

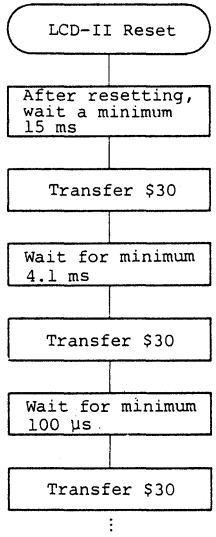


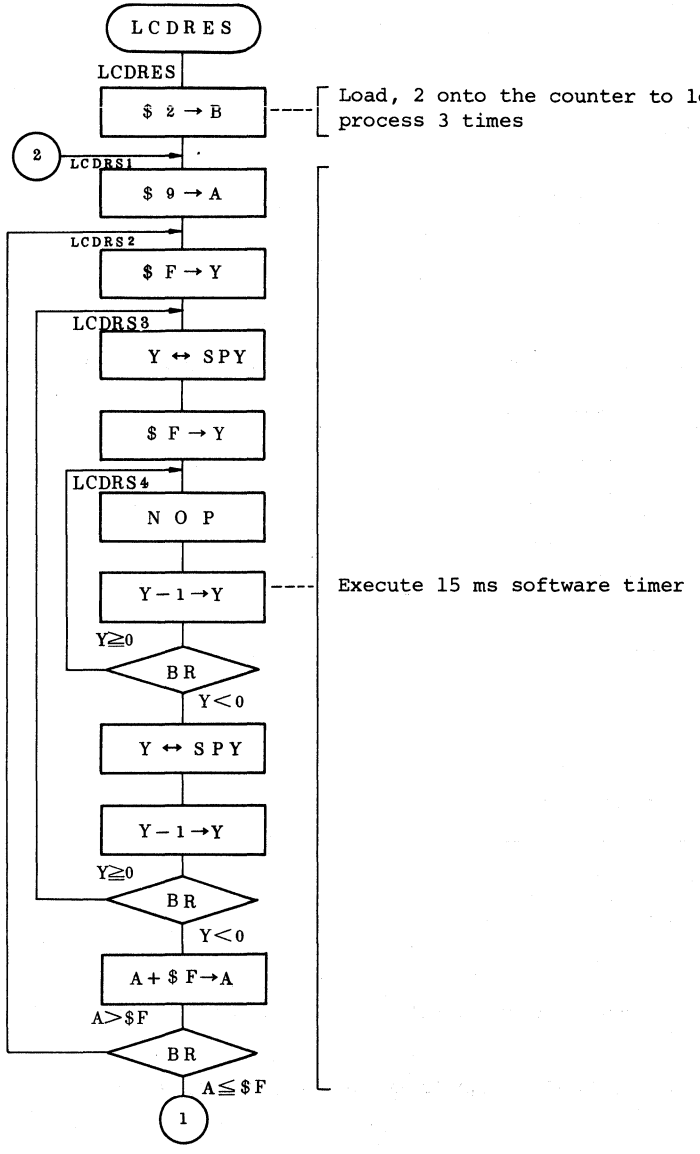
Fig. 14.6. Resetting of LCD-II by Software

(2) Controls RS, R/W and E signals at the I/O port by software and output $30.

(3) By loop processing of software timer which generates a delay time (15 ms), and by program which transfers $30, resetting of LCD-II is achieved as shown in Fig. 14.6.

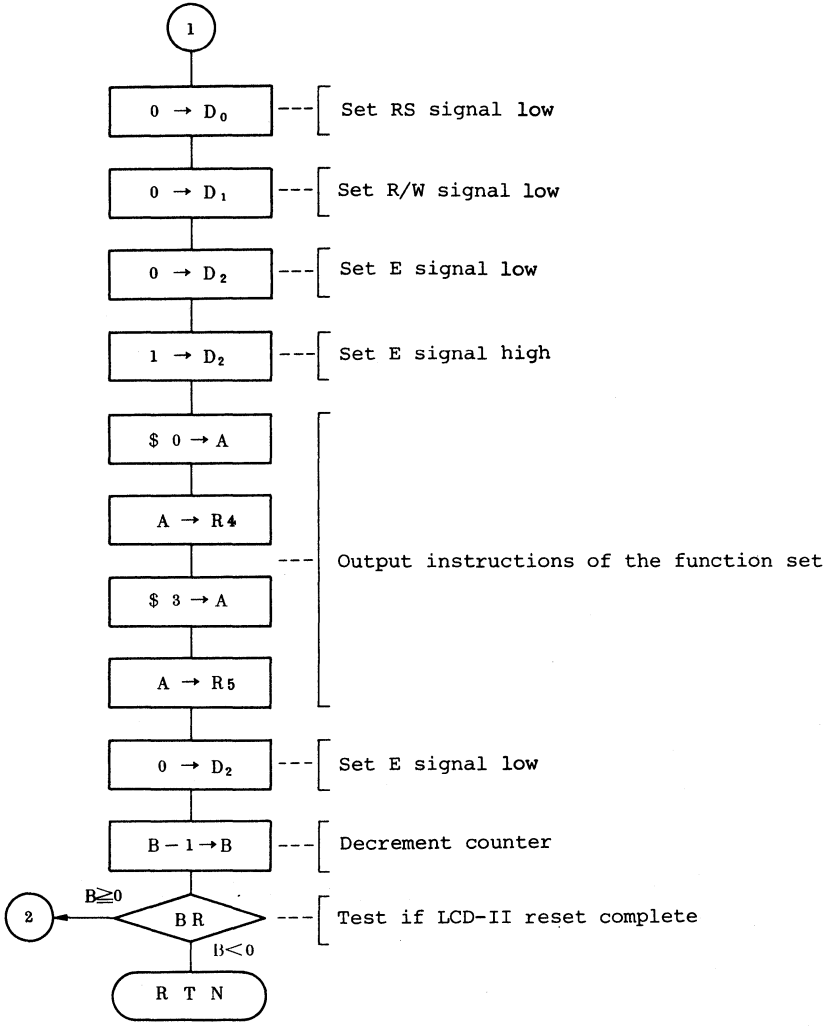
Program Module Name: Reset LCD-II	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: LCDRES

Flowchart:



Program Module Name: Reset LCD-II	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: LCDRES

Flowchart:



Program Module Name:	MCU:	Label:
Initialize LCD-II	HMCS402C/ HMCS404C/HMCS408C	LCDINT

Function: Initializes LCD-II display mode.

Arguments: None

Changes in CPU Registers and Flags:

A	B
×	×

X	Y
●	×

SPX	SPY
●	●

W
●

CA	ST
×	×

● : Not Affected
× : Undefined
↕ : Result

Specifications:

1 word = 10 bits

ROM (Words):	18
RAM (Digits):	0
Stack (Digits):	0
No. of cycles:	180
Reentrant:	No
Relocatable:	No
Interrupt OK?:	No

Description:

1. Function Details

   (1) There are no arguments in LCDINT.

   (2) Data in Table 14.3 is to be transferred to LCD-II and initialized in display mode.

Specifications Notes:

Program Module Name: Initialize LCD-II	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: LCDINT

Description:

Table 14.3. LCD-II Display Mode Data

Data	Function
$08	Display OFF
$01	All display cleared, DDRAM address set to $00
$07	Cursor movement direction to the right and display shifted
$90	DDRAM address set to $10
$18	Select display shift direction to the left
$0C	Display ON

(3) In program module LCDINT, the subroutine in Table 14.4 is used.

Table 14.4. Subroutine Used in Program Module LCDINT

Subroutine Name	Label	Functions
Busy check	LCDBSY	Check the LCD-II busy flag

2. User Notes

Resets LCD-II prior to execution of LCDINT.

3. RAM Allocation

RAM is not used program module LCDINT.

4. Sample Applications

```
        ⋮
        CALL    LCDRES  ..... Execute LCDRES and reset
                                LCD-II
        CALL    LCDINT  ..... Call LCDINT
        ⋮
        ORG     $100    )
        DC      $10C    )..... Secure data table
        DC      $118    )
        ⋮
```

Program Module Name:	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: LCDINT
Initialize LCD-II		

Description:

5. Basic Operation

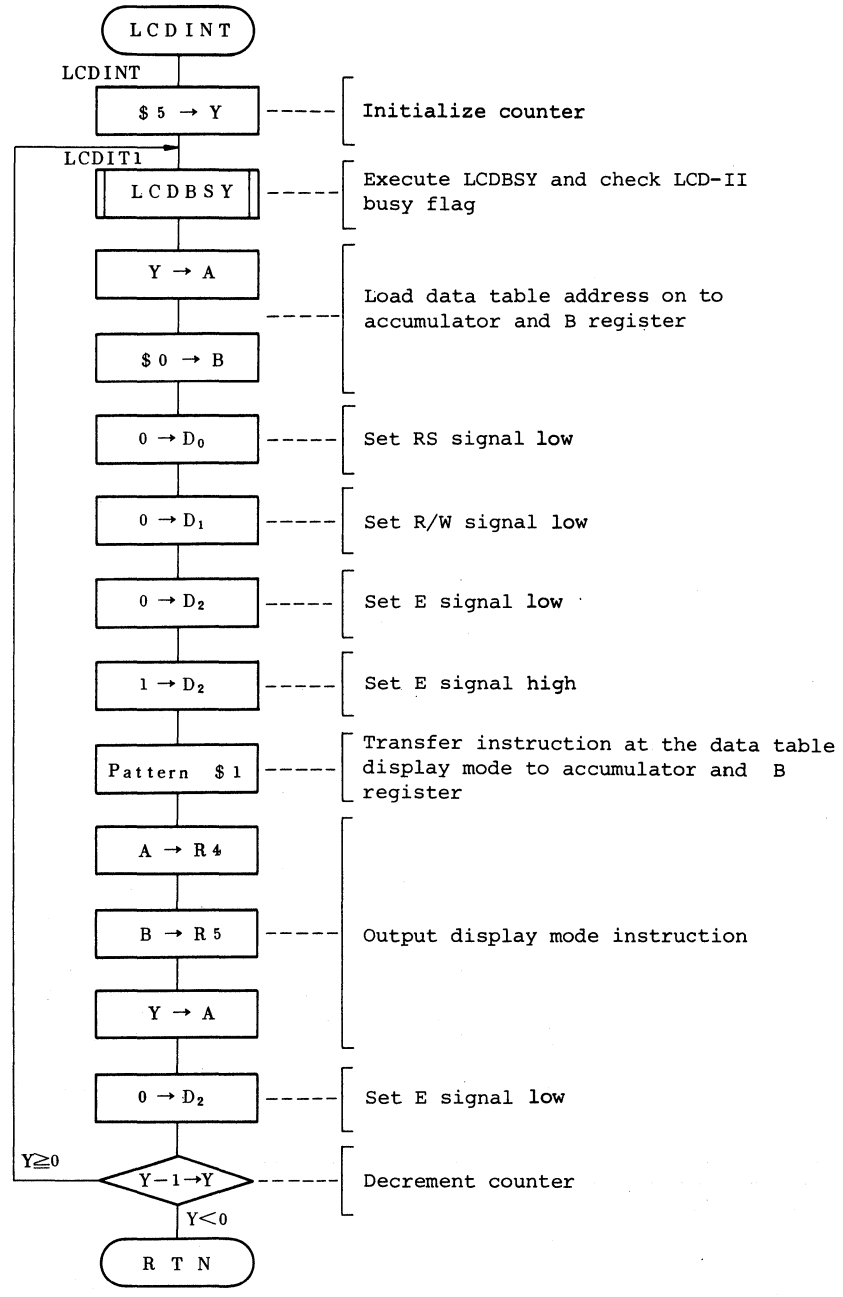
(1) Performs busy flag check every time before outputting data.

(2) Set data shown in Table 14.3 into port R4 and R5, control LCD-II signals E, R/W and RS through ports D0, D1 and D2, and initialize LCD-II with this data.

(3) Repeat (1) to (2) 6 consecutive times and initialize.

Program Module Name: Initialize LCD-II	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: LCDINT

Flowchart:



Program Module Name: Display on LCD	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: LCDDSP

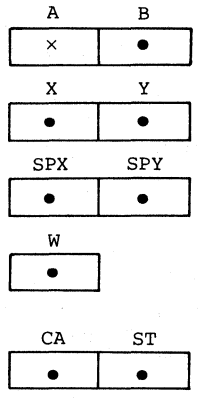
Function: Writes ASCII code in DDRAM of LCD-II and displays graphically on liquid crystal.

Arguments:

1 digit = 4 bits

Contents		Storage Location	No. of Digits
Entry	Display data (ASCII code) Upper	B	1
	Lower	A	1
Returns	—	—	—

Changes in CPU Registers and Flags:



● : Not Affected
× : Undefined
↕ : Result

Specifications:

1 word = 10 bits

ROM (Words):	13
RAM (Digits):	0
Stack (Digits):	0
No. of cycles:	29
Reentrant:	No
Relocatable:	No
Interrupt OK?:	No

Description:

1. Function Details

(1) Argument details

A: Stores the lower 4 bits of the ASCII code of the graphic display.

B: Stores the upper 4 bits of the ASCII code of the graphic display.

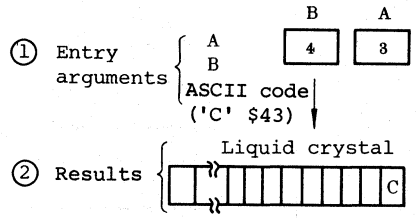


Fig. 14.7. Example of LCDDSP Execution

Specifications Notes:

Program Module Name: Display on LCD	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: LCDDSP

Description:

(2) Fig. 14.7 shows an example of program module LCDDSP. If the entry arguments are set in accumulator and B register as in Fig. 14.7-①, ASCII code is written to the current DDRAM address and graphically displayed on the liquid crystal.

(3) In program module LCDDSP, the subroutine shown in Table 14.5 is used.

Table 14.5. Subroutine Used in Program Module LCDDSP

Subroutine Name	Label	Function
Busy check	LCDBSY	Checks busy flag of LCD-II

2. User Notes

(1) Resets LCD-II.

(2) Initializes LCD-II display mode.

(3) Stores entry arguments.

3. RAM Allocation

RAM is not used by program module LCDDSP.

4. Sample Application

```
        :
CALL    LCDRES      ····· Execute LCDRES and reset LCD-II
CALL    LCDINT      ····· Execute LCDINT and initialize LCD-II
                          display mode
LAI     3           }
LBI     4           } ····· Load entry arguments
CALL    LCDDSP      ····· Call LCDDSP
        :
```

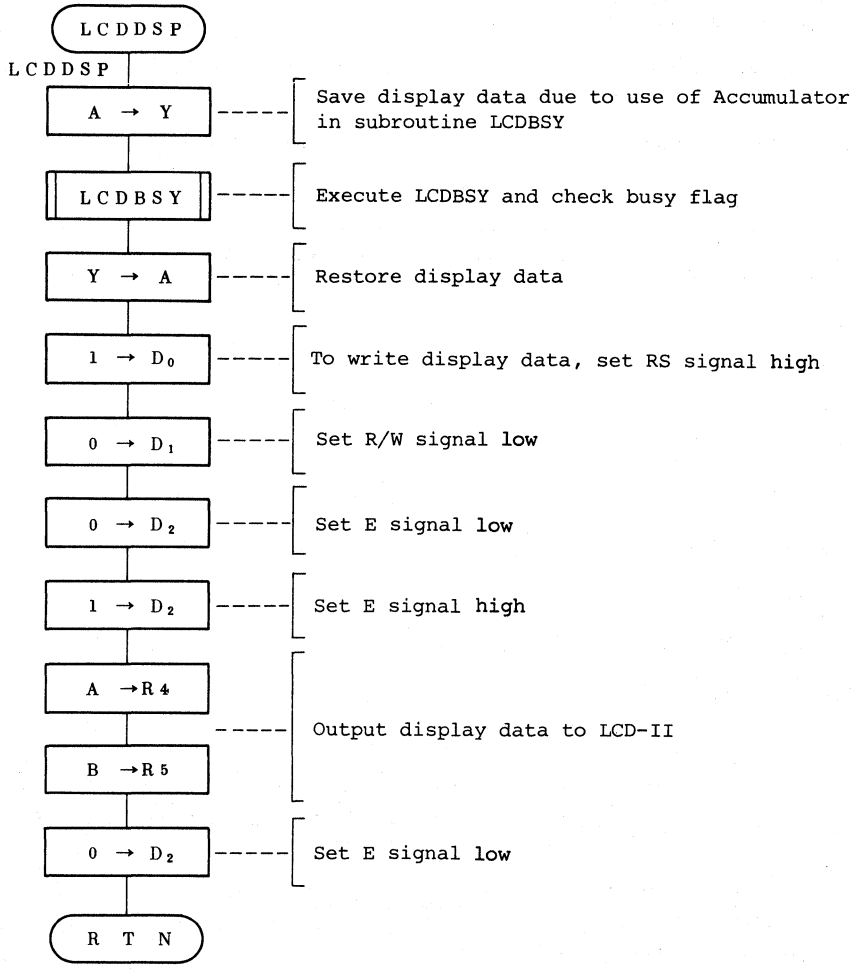
5. Basic Operation

(1) Checks LCD-II busy flag.

(2) Controls signals RS, R/W and E by software through port D and outputs display data.

Program Module Name: Display on LCD	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: LCDDSP

Flowchart:



14.4 SUBROUTINE DESCRIPTION

Subroutine Name: Busy Check	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: LCDBSY

Function: Checks if LCD-II is busy and waits for ready state.

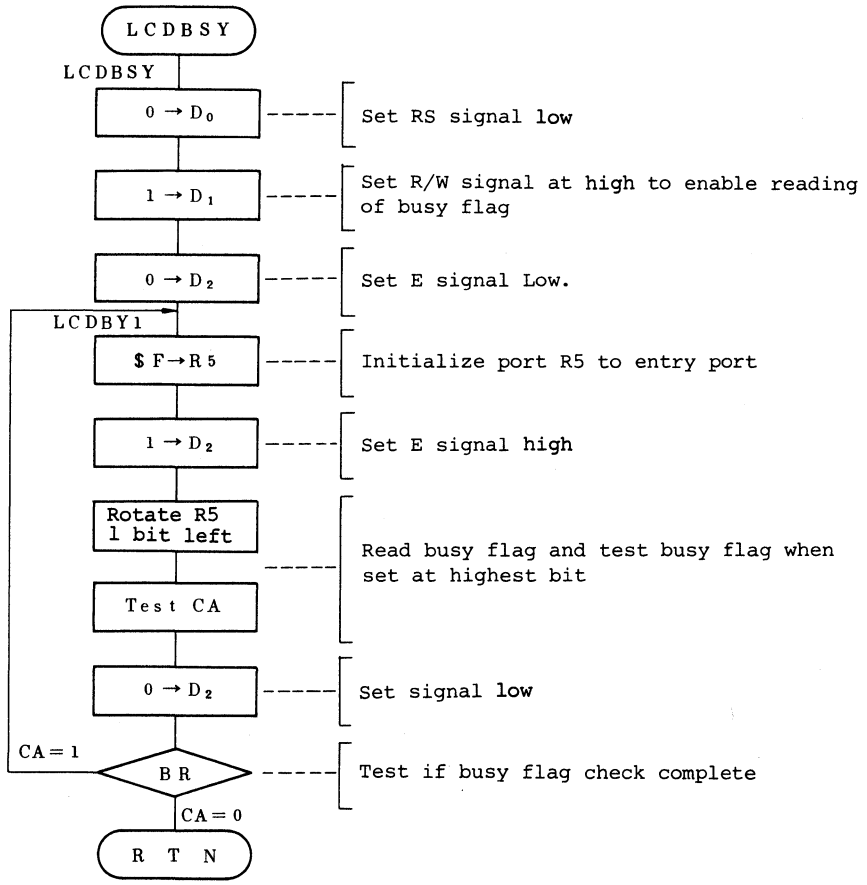
Basic Operation:

(1) Since, while LCD-II is in operation, it will not allow access from microcomputers, checks must be made on the LCD-II busy flag.

(2) Control of signals RS, R/W and E through port D is performed by software together with busy flag check.

Program Module Using This Subroutine: LCDDSP, LCDINT, LCDRES

Flowchart:



14.5 PROGRAM LISTING

```
ST-NO   OBJECT    ADRS   SOURCE STATEMENTS

00001   100       0000           LLEN     132
00002                            TITLE    LIQUID CRYSTAL MODULE (H2570) CONTROL
00003                  *
00004                  ****   RAM ALLOCATION   ****************************************
00005                  *
00006                  LSOUR    EQU      $020          WORK AREA FOR ACCA
00007                  HSOUR    EQU      $021          WORK AREA FOR B REGISTER
00008                  ************************************************************************
00009                  *                                                                      *
00010                  *               VECTOR ADDRESSES                                       *
00011                  *                                                                      *
00012                  ************************************************************************
00013                  *
00014                           ORG      $0000
00015                  *
00016   150 040   0000          JMPL     LCDMN         RESET
00017   150 040   0002          JMPL     LCDMN         INTO
00018   150 040   0004          JMPL     LCDMN         INT1
00019   150 040   0006          JMPL     LCDMN         TIMER A
00020   150 040   0008          JMPL     LCDMN         TIMER B
00021                  *
00022                           ORG      $000C
00023                  *
00024   150 040   000C          JMPL     LCDMN         SERIAL
00025                  ************************************************************************
00026                  *                                                                      *
00027                  *               MAIN PROGRAM : LCDMN                                   *
00028                  *                                                                      *
00029                  ************************************************************************
00030                  *
00031                           ORG      $0040
00032                  *
00033   0F0       0040  LCDMN    LWI      $0            INITIALIZE W REGISTER
00034   160 064   0041          CALL     LCDRES        RESET LCD-II
00035   160 07E   0043          CALL     LCDINT        INITIALIZE LCD-II
00036   230       0045          LAI      $0            LOAD DATA TABLE STARTING ADDR INTO ACCA
00037   200       0046          LBI      $0            LOAD DATA TABLE STARTING ADDR INTO B REG
00038   194 020   0047  LCDMN1   LMAD     LSOUR         ACCA ---> LSOUR
00039   048       0049          LAB
00040   194 021   004A          LMAD     HSOUR         B REGISTER ---> HSOUR
00041   190 020   004C          LAMD     LSOUR         LSOUR ---> ACCA
00042   1B2       004E          P        $2            MOVE DISPLAY TO ACCA & B REGISTER
00043   0D8       004F          LYA
00044   2B0       0050          ALEI     $0            TEST IF ACCA=0
00045   048       0051          LAB
00046   354       0052          BR       LCDMN2        IF 0, BRANCH TO LCDMN2
00047   356       0053          BR       LCDMN3        IF NOT 0, BRANCH TO LCDMN3
00048   2B0       0054  LCDMN2   ALEI     $0            TEST IF B REGISTER=0
00049   363       0055          BR       PEND          IF 0, BRANCH TO PEND
00050   0AF       0056  LCDMN3   LAY
00051   160 090   0057          CALL     LCDDSP        DISPLAY FIGURE ON LCD-II
00052   190 021   0059          LAMD     HSOUR         HSOUR ---> B REGISTER
00053   0C8       005B          LBA
00054   190 020   005C          LAMD     LSOUR         LSOUR ---> ACCA
00055   281       005E          AI       $1            ACCA + $1 ---> ACCA
00056   361       005F          BR       LCDMN4        BRANCH IF ACCA =$F
00057   347       0060          BR       LCDMN1
```

```
00058  04C       0061  LCDMN4   IB                      INCREMENT B REGISTER
00059  347       0062           BR        LCDMN1       LOOP UNTIL B REGISTER = $F
00060  363       0063  PEND     BR        PEND
00061                  ************************************************************
00062                  *                                                          *
00063                  *       NAME : LCDRES (RESET LCD-II)                       *
00064                  *                                                          *
00065                  ************************************************************
00066                  *                                                          *
00067                  *      ENTRY : NOTHING                                     *
00068                  *    RETURNS : NOTHING                                     *
00069                  *                                                          *
00070                  ************************************************************
00071  202       0064  LCDRES   LBI       $2           INITIALIZE LOOP COUNTER
00072  239       0065  LCDRS1   LAI       $9           EXECUTE 15MS SOFTWARE TIMER
00073  21F       0066  LCDRS2   LYI       $F
00074  002       0067  LCDRS3   XSPY
00075  21F       0068           LYI       $F
00076  000       0069  LCDRS4   NOP
00077  000       006A           NOP
00078  0DF       006B           DY
00079  369       006C           BR        LCDRS4
00080  002       006D           XSPY
00081  0DF       006E           DY
00082  367       006F           BR        LCDRS3
00083  28F       0070           AI        $F
00084  366       0071           BR        LCDRS2
00085  260       0072           REDD      $0           RS=0
00086  261       0073           REDD      $1           R/W=0
00087  262       0074           REDD      $2           E=0
00088  2E2       0075           SEDD      $2           E=1
00089  230       0076           LAI       $0
00090  2D4       0077           LRA       $4           WRITE INSTRUCTION DATA
00091  233       0078           LAI       $3
00092  2D5       0079           LRA       $5           WRITE INSTRUCTION DATA
00093  262       007A           REDD      $2           SET E=0
00094  0CF       007B           DB                     DECREMENT LOOP COUNTER
00095  365       007C           BR        LCDRS1       LOOP UNTIL LOOP COUNTER=0
00096  010       007D           RTN
00097                  ************************************************************
00098                  *                                                          *
00099                  *       NAME : LCDINT (INITIALIZE LCD-II)                  *
00100                  *                                                          *
00101                  ************************************************************
00102                  *                                                          *
00103                  *      ENTRY : NOTHING                                     *
00104                  *    RETURNS : NOTHING                                     *
00105                  *                                                          *
00106                  ************************************************************
00107  215       007E  LCDINT   LYI       $5           INITIALIZE LOOP COUNTER
00108  160 09D   007F  LCDIT1   CALL      LCDBSY       CHECK LCD-II BUSY FLAG
00109  0AF       0081           LAY                    LOAD DATA TABLE ADDRESS INTO ACCA
00110  200       0082           LBI       $0           LOAD DATA TABLE ADDRESS INTO B REG
00111  260       0083           REDD      $0           RS=0
00112  261       0084           REDD      $1           R/W=0
00113  262       0085           REDD      $2           E=0
00114  2E2       0086           SEDD      $2           E=1
```

```
00115  1B1       0087            P         $1              MOVE INSTRUCTION TO ACCA & B REG
00116  2D4       0088            LRA       $4              OUTPUT INSTRUCTION
00117  048       0089            LAB
00118  2D5       008A            LRA       $5
00119  0AF       008B            LAY
00120  262       008C            REDD      $2              E=0
00121  0DF       008D            DY                        DECREMENT LOOP COUNTER
00122  37F       008E            BR        LCDIT1          LOOP UNTIL LOOP COUNTER=0
00123  010       008F            RTN
00124                    ********************************************************************
00125                    *                                                                  *
00126                    *       NAME : LCDDSP (DISPLAY ON LCD-II)                          *
00127                    *                                                                  *
00128                    ********************************************************************
00129                    *                                                                  *
00130                    *      ENTRY : B REGISTER,ACCUMULATOR (DISPLAY DATA)               *
00131                    *    RETURNS : NOTHING                                             *
00132                    *                                                                  *
00133                    ********************************************************************
00134  0D8       0090  LCDDSP  LYA                       SAVE ACCA
00135  160 09D   0091          CALL      LCDBSY          CHECK BUSY FLAG
00136  0AF       0093          LAY                       RESTORE ACCA
00137  2E0       0094          SEDD      $0              RS=1
00138  261       0095          REDD      $1              R/W=0
00139  262       0096          REDD      $2              E=0
00140  2E2       0097          SEDD      $2              E=1
00141  2D4       0098          LRA       $4              OUTPUT DISPLAY DATA
00142  048       0099          LAB
00143  2D5       009A          LRA       $5
00144  262       009B          REDD      $2              E=0
00145  010       009C          RTN
00146                    ********************************************************************
00147                    *                                                                  *
00148                    *         NAME : LCDBSY (CHECK BUSY FLAG)                          *
00149                    *                                                                  *
00150                    ********************************************************************
00151  260       009D  LCDBSY  REDD      $0              RS=0
00152  2E1       009E          SEDD      $1              R/W=1
00153  262       009F          REDD      $2              E=0
00154  23F       00A0  LCDBY1  LAI       $F              SELECT R PORT AS INPUT
00155  2D5       00A1          LRA       $5
00156  2E2       00A2          SEDD      $2              E=1
00157  255       00A3          LAR       $5              READ BUSY FLAG
00158  0A1       00A4          ROTL
00159  06F       00A5          TC                        TEST CARRY
00160  262       00A6          REDD      $2              E=0
00161  3A0       00A7          BR        LCDBY1          LOOP UNTIL BUSY FLAG=0
00162  010       00A8          RTN
00163                    ********************************************************************
00164                    *                                                                  *
00165                    *          DATA TABLE                                              *
00166                    *                                                                  *
00167                    ********************************************************************
00168                    *
00169                            ORG       $0100
00170                    *
00171  10C       0100  $00     DC        $10C
```

```
00172   118     0101    $01     DC      $118
00173   191     0102    $02     DC      $191
00174   107     0103    $03     DC      $107
00175   101     0104    $04     DC      $101
00176   108     0105    $05     DC      $108
00177                   *
00178                           ORG     $0200
00179                   *
00180   143     0200    $00     DC      $143
00181   14D     0201    $01     DC      $14D
00182   14F     0202    $02     DC      $14F
00183   153     0203    $03     DC      $153
00184   120     0204    $04     DC      $120
00185   14D     0205    $05     DC      $14D
00186   143     0206    $06     DC      $143
00187   155     0207    $07     DC      $155
00188   120     0208    $08     DC      $120
00189   148     0209    $09     DC      $148
00190   14D     020A    $0A     DC      $14D
00191   143     020B    $0B     DC      $143
00192   153     020C    $0C     DC      $153
00193   134     020D    $0D     DC      $134
00194   130     020E    $0E     DC      $130
00195   130     020F    $0F     DC      $130
00196   100     0210    $10     DC      $100
00197                   *
00198                           END
```

SECTION 15. LOW POWER DISSIPATION MODE AND WATCHING TIMER EXECUTION USING THE HA1835A

15.1 HARDWARE DESCRIPTION

15.1.1 Function

Enters low power dissipation mode of the HMCS404C MCU and executes fail safe mode with the HA1835P watchdog timer.

15.1.2 Microcomputer Operation

Enters stop mode and standby mode and turns on LED indicating current mode, and counts down binary counter every second to display mode elapsed time on a set of LEDs; stop mode is released by reset input and standby mode by timer interrupt. Controls and outputs pulses to the HA1835P through port D_8.

15.1.3 Peripheral Devices

SW3: Sets stop mode in low power dissipation mode.

SW2: Sets standby mode in low power dissipation mode.

SW4: Stops output of pulses to the HA1835P in fail safe mode and executes system run away.

15.1.4 Circuit Diagram

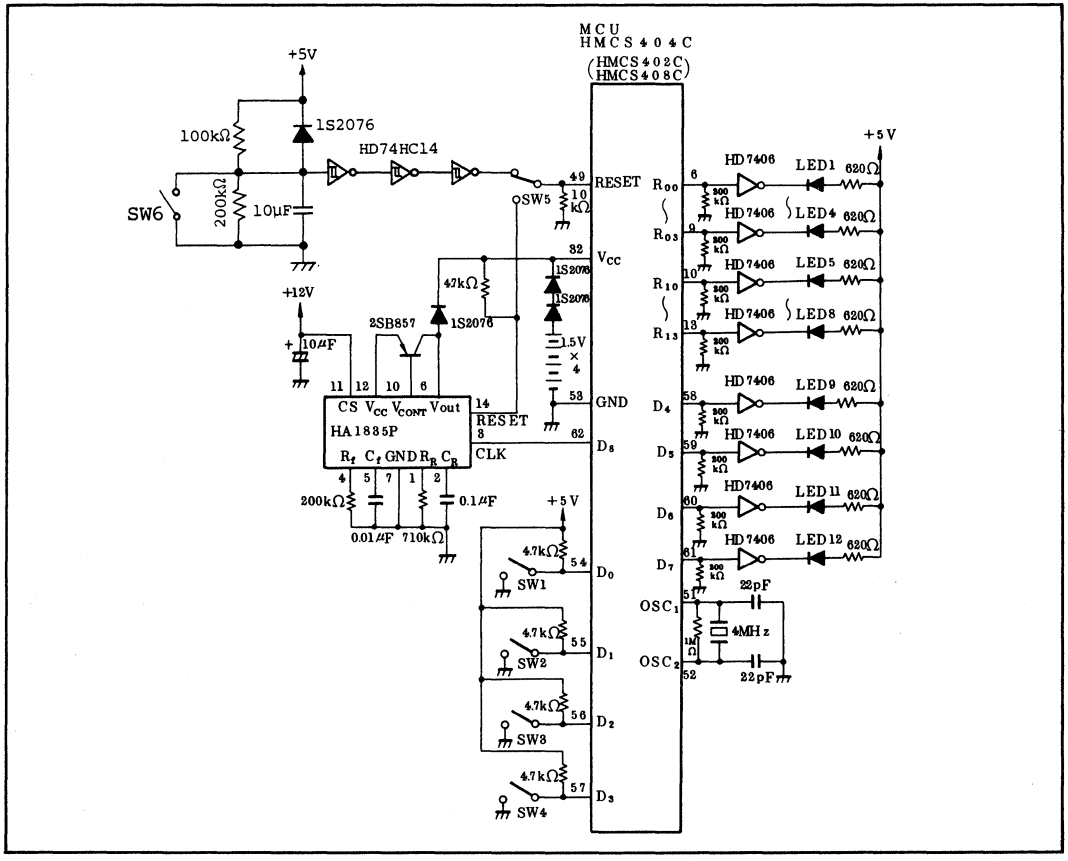


Fig. 15.1. Low Power Dissipation Mode and HA1835P Control

15.1.5 Pin Functions

HMCS404C pin functions, switches, LED names and HA1835P pin functions are shown in Table 15.1.

Table 15.1. Pin Functions

Pin Name HMCS404C	Input/ Output	Active Level (High or Low)	Function	Pin Name SW. LED HA1835P
D_0	Input	High	Controls HA1835P	SW1
		Specifies Low	Low power dissipation mode	
D_1	Input	High	Standby mode	SW2
D_2	Input	High	Stop mode	SW3
D_3	Input	High	Controls pulses for HA1835P	SW4
D_4	Output	High	Drives LED indicating watchdog timer mode control	LED9
D_5	Output	High	Drives LED indicating reset of watchdog timer mode	LED10
D_6	Output	Low	Drives LED during standby mode execution	LED11
D_7	Output	Low	Drives LED during stop mode execution	LED12
D_8	Output	-	Controls duty pulse for HA1835P CLK pin	CLK
R_{00}	Output	High	Indicates standby mode execution time.	LED1
R_{01}	Output	High	Drive LEDs indicating 8 bits binary counter	LED2
R_{02}	Output	High		LED3
R_{03}	Output	High		LED4
R_{10}	Output	High		LED5
R_{11}	Output	High		LED6
R_{12}	Output	High		LED7
R_{13}	Output	High		LED8
RESET	Input	Low	Controls HA1835P	SW5
		High	Low power dissipation mode	
RESET	Input	High	Resets from stop mode. Resets from HA1835P control	SW6

15.1.6 Hardware Operation

1. Low power dissipation

(a) Standby mode

Timing chart in standby mode is shown in Fig. 15.2.

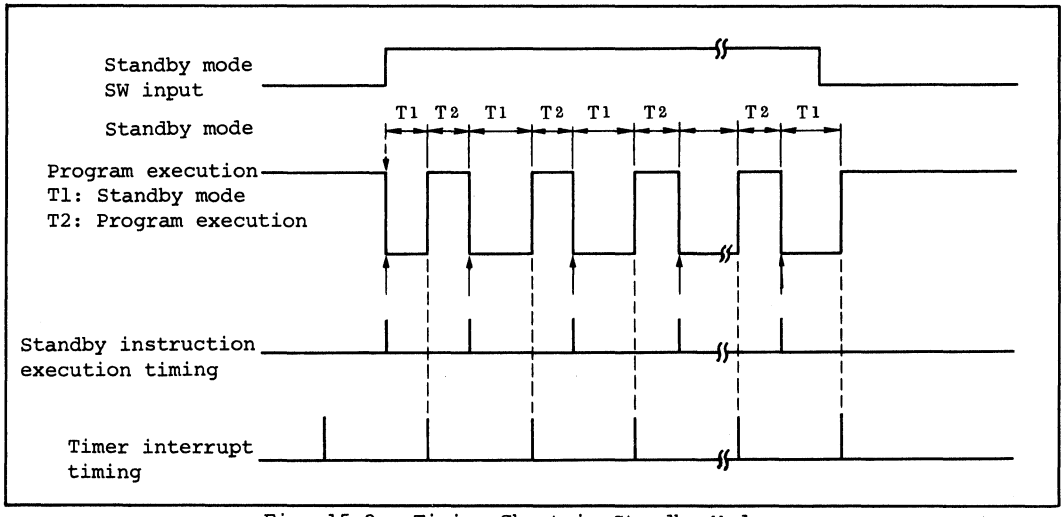


Fig. 15.2. Timing Chart in Standby Mode

(b) Stop mode

Timing chart in stop mode is shown in Fig. 15.3.

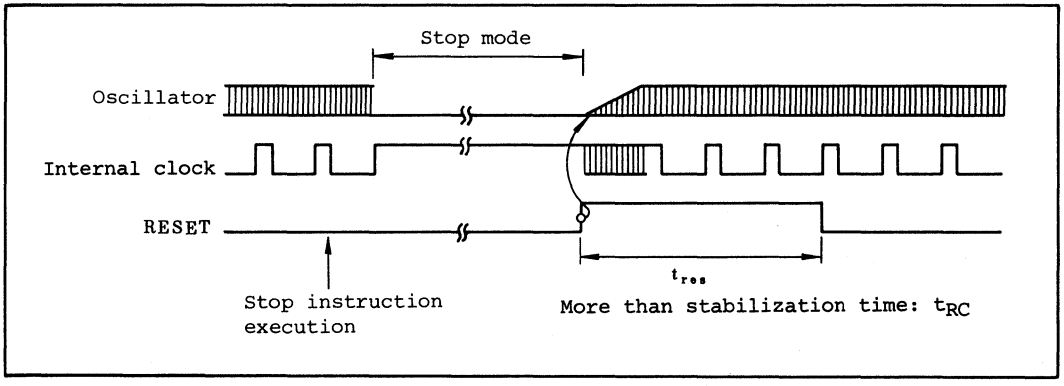


Fig. 15.3. Timing Chart in Stop Mode

2. HA1835P control

Timing chart of pulse output and reset pin for watchdog timer is shown in Fig. 15.4.

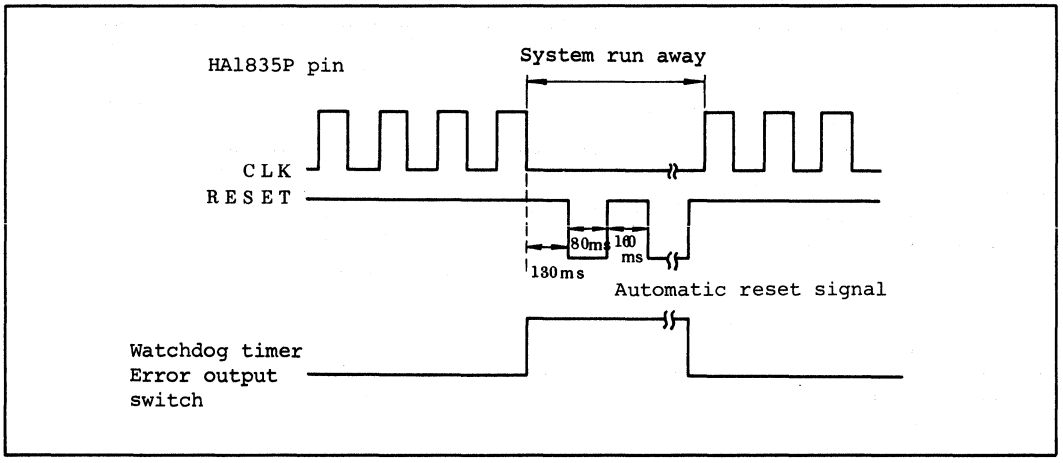


Fig. 15.4. Timing Chart of Pulse Output

15.2.1 Program Module Configuration

Program module configuration for switch board controlling of the HA1835P and low power dissipation mode is shown in Fig. 15.5.

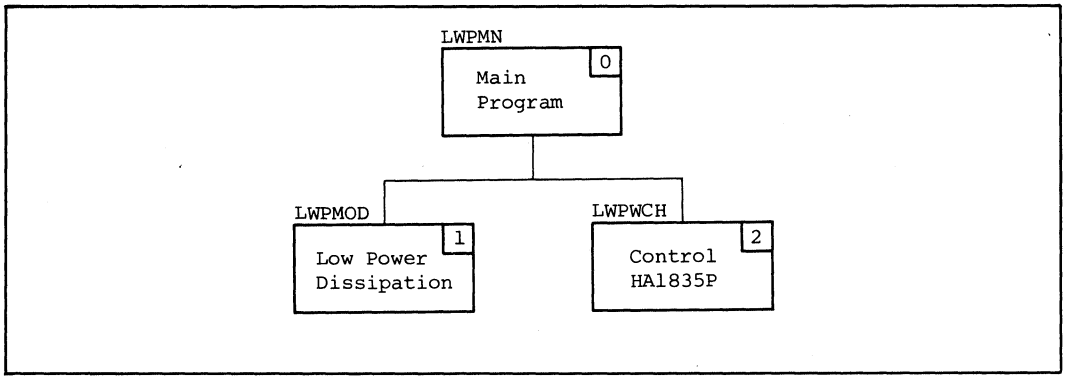


Fig. 15.5. Program Module Configuration

15.2.2 Program Module Functions

Program module functions are summarized in Table 15.2.

Table 15.2. Program Module Functions

No.	Program Module Name	Label	Function
0	Main Program	LWPMN	Demonstrates low power dissipation and HA1835P control
1	Low Power Dissipation	LWPMOD	Tests operation of low power dissipation mode
2	Control HA1835P	LWPWCH	Tests operation of watchdog timer using HA1835P

15.2.3 Program Module Process Flow (Main Program)

The flowchart in Fig. 15.6. is an example of low power dissipation and HA1835P control performed by the program module in Fig. 15.6.

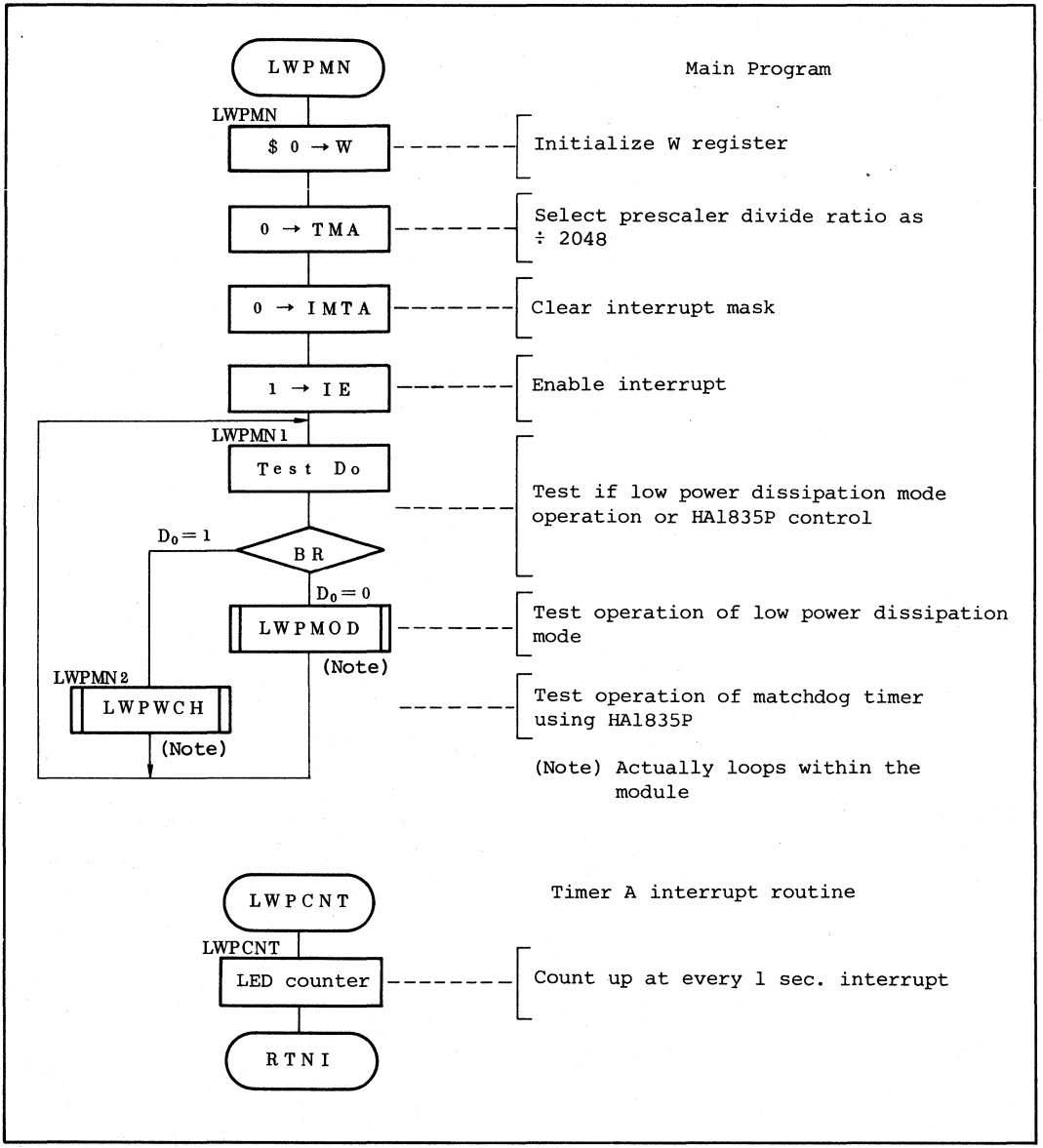


Fig. 15.6. Program Module Sample Application

15.3 PROGRAM MODULE DESCRIPTION

Program Module Name:	MCU:	Label:
Low Power Dissipation	HMCS402C/ HMCS404C/HMCS408C	LWPMOD

Function: Tests operation of low power dissipation mode.

Arguments: NONE

Changes in CPU Registers and Flags:

A	B
×	●

X	Y
●	●

SPX	SPY
●	●

W
●

CA	ST
×	×

● : Not Affected
× : Undefined
↕ : Result

Specifications:

1 word = 10 bits

ROM (Words):	34
RAM (Digits):	7
Stack (Digits):	4
No. of cycles:	23
Reentrant:	No
Relocatable:	No
Interrupt OK?:	Yes

Description:

1. Function Details

   (1) LWPMCD has no arguments.

   (2) Executes standby mode, stop mode by switch control. Turns on LED indicating execution of standby mode.

2. User Notes

   (1) Executes only one mode at a time.

   (2) Selects only one mode switch.

Specifications Notes: "No. of cycles" in "Specification" represents the number of cycles required when data is not stored in SBRAM(RAM) and each mode is not executed.

Program Module Name: Low Power Dissipation	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: LWPMOD

Description:

(3) Initializes timer.

3. RAM Allocation

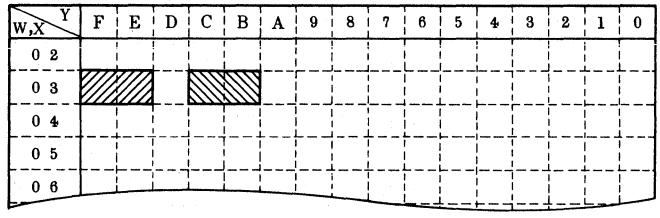


Fig. 15.7. RAM Allocation

Label	RAM	Description
ERRAM1	b3 b0 MD($03F)	Stores data indicating whether operation starts after activating power on or returns from stop mode
ERRAM2	b3 b0 MD($03E)	
CNTRL	b3 b0 MD($03C)	Uses upper and lower counter for counting up every 1 sec.
CNTRH	b3 b0 MD($03B)	

Program Module Name: Low Power Dissipation	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: LWPMOD

Description:

4. Sample Application

```
LAI     $0          }
LMAD    $008        }  ..... Store prescaler divide ratio
                        ..... as ÷ 2048
REMD    3, $001     ..... Clear interrupt mask flag
SEMD    0, $000     ..... Enable interrupt

CALL    LWPMOD      ..... Call LWPMOD
```

5. Basic Operation

(1) Enters stop mode by setting signal stop mode switch to high, returns with stop mode switch set to low and with RESET switch to on. Software checks whether operation starts after activating power on or returns from standby mode.

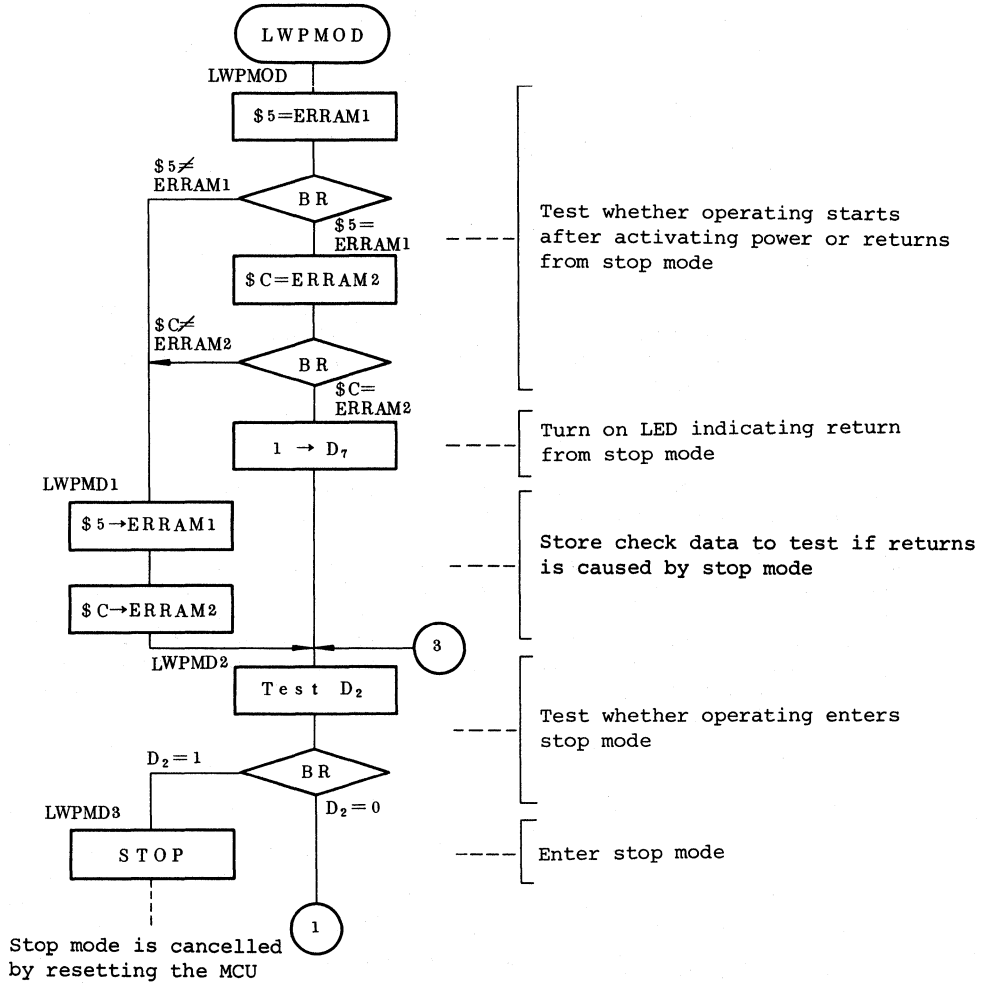
When operation returns from stop mode, turns on LED; if not, stores comparison data in ERRAM(RAM).

(2) Enters standby mode by setting signal standby mode switch to high, returns with standby mode switch set to low.

Returns at every timer interrupt and executes interrupt routine. When standby mode switch is low, enters standby mode after interrupt. Whether or not operation is returned at every timer interruption can be confirmed by LED counter incremented after return. When setting standby mode switch to high, turns on LED indicating standby mode and executes standby instruction. When low, returns from standby mode and turns off LED indicating standby mode.

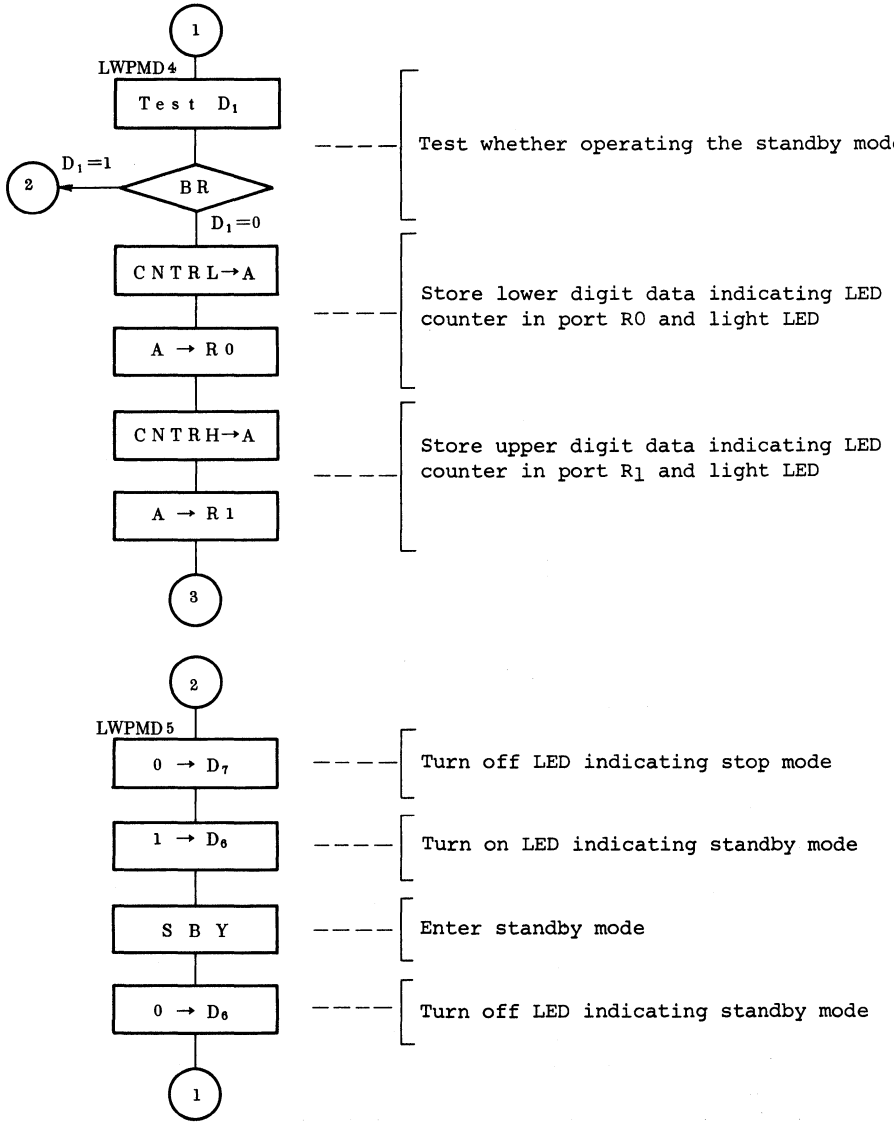
Program Module Name: Low Power Dissipation	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: LWPMOD

Flowchart:



Program Module Name: Low Power Dissipation	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: LWPMOD

Flowchart:



Program Module Name:	MCU:	Label:
Control HA1835P	HMCS402C/ HMCS404C/HMCS408C	LWPWCH

Function: Tests watchdog timer operation using HA1835P.

Arguments: NONE

Changes in CPU Registers and Flags:

A	B
●	×

X	Y
●	×

SPX	SPY
●	●

W
●

CA	ST
×	×

● : Not Affected
× : Undefined
↕ : Result

Specifications:

1 word = 10 bits

ROM (Words):	35
RAM (Digits):	3
Stack (Digits):	4
No. of cycles:	27
Reentrant:	No
Relocatable:	No
Interrupt OK?:	Yes

Description:

1. Function Details

(1) LWPWCH has no arguments.

(2) Controls HA1835P under switch control.

(a) When switch 4 is OFF, outputs pulses from port D_8.

(b) When switch 4 is ON, stops outputting pulses causing system MCU run away.

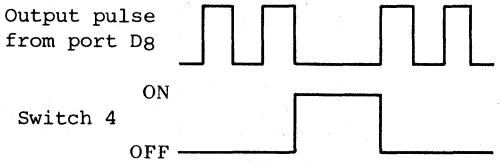


Fig. 15.8. Example of LWPWCH execution

Specifications Notes: "No. of cycles" in "Specifications" represents the number of cycles required when data is stored in ERRAM(RAM) and an pulse generation is selected by switch.

Program Module Name: Control HA1835P	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: LWPWCH

Description:

2. User Notes

Call LWPWCH after clearing RAM.

3. RAM Allocation

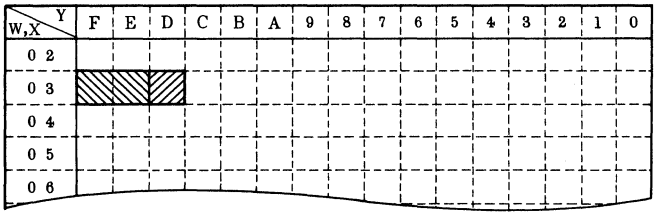


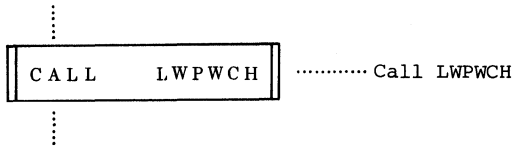
Fig. 15.9. RAM Allocation

Label	RAM	Description
ERRAM1	b3 b0 MD($03F)	Store check data indicating whether operation starts after activating power on or returns from watchdog timer error
ERRAM2	b3 b0 MD($03E)	
D8FLAG	b3 b0 MD($0,$03D)	Set status of port D8

Program Module Name: Control HA1835P	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: LWPWCH

Description:

4. Sample Application



5. Basic Operation

(1) Turns on LED indicating watchdog timer mode, after main switch is activated.

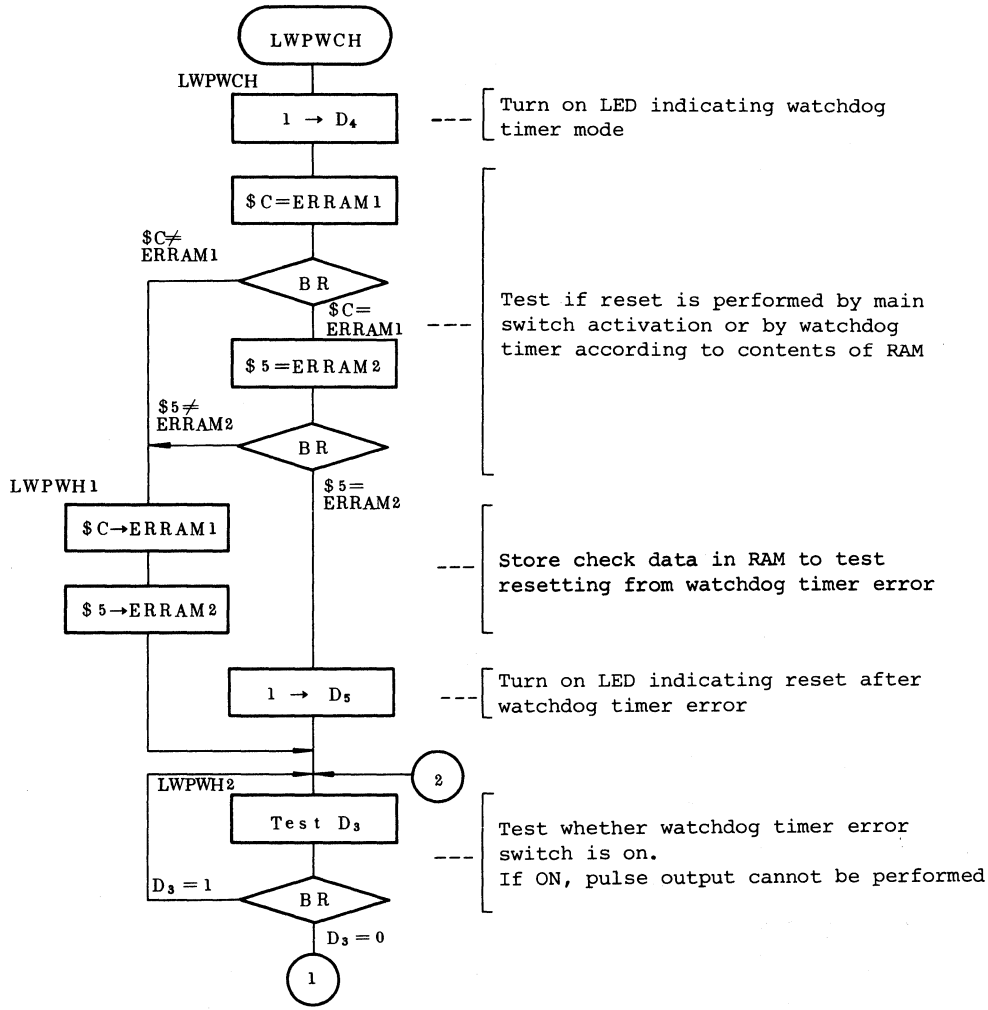
(2) When watchdog error switch is turned off, outputs pulse by 1 ms software timer. When turned on, stops pulse output and enter eternity loop.

(3) When turning off watchdog timer error switch, re-calls program after reset. If data in ERRAM(RAM) and decision data are the same, turns on LED indicating reset by watchdog timer error.

(4) Stores check data if it is not stored in ERRAM(RAM).

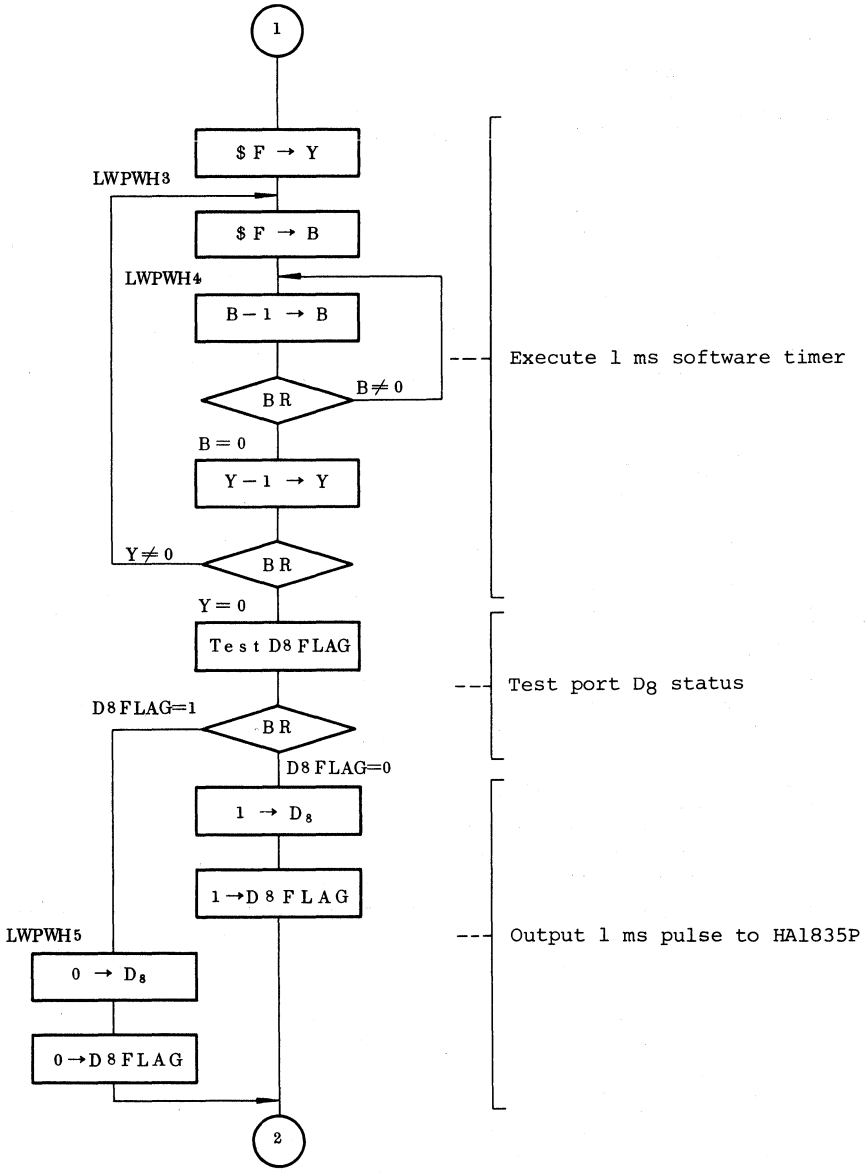
Program Module Name: Control HA1835P	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: LWPWCH

Flowchart:



Program Module Name: Control HA1835P	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: LWPWCH

Flowchart:



15.4 SUBROUTINE DESCRIPTION

Subroutine Name: LED Counter	MCU: HMCS402C/ HMCS404C/HMCS408C	Label: LWPCNT

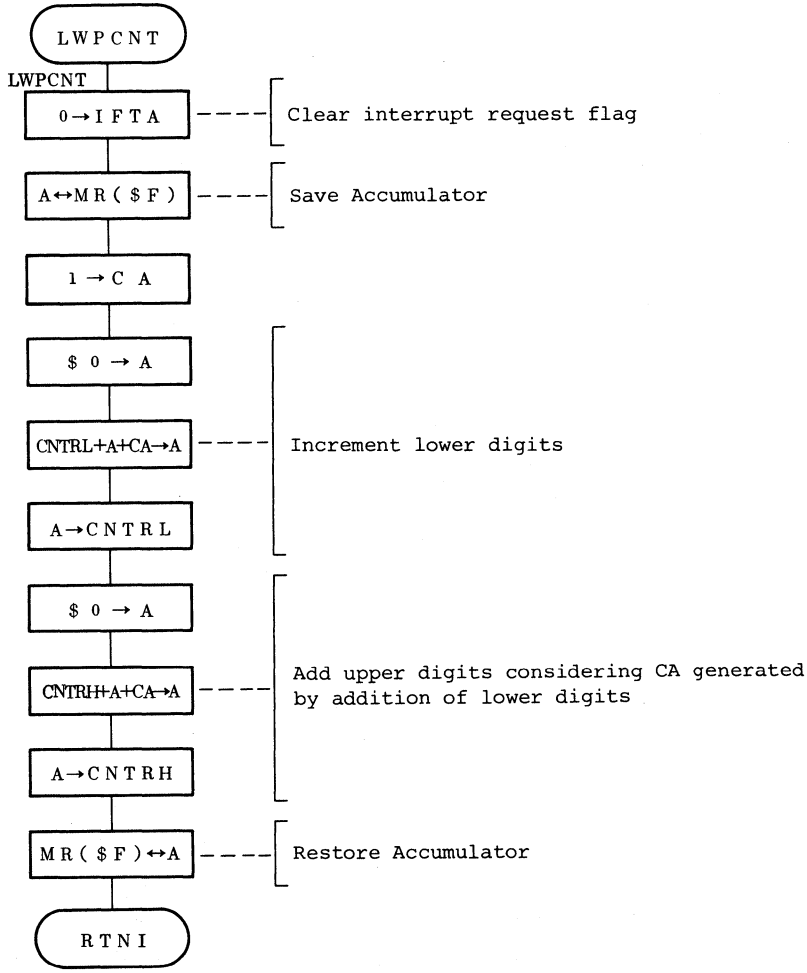
Function: Increments LED counter.

Basic Operation: (1) This subroutine is called for each 1 sec. timer interrupt.

(2) Increments 1 byte counter each second.

Program Module Using This Subroutine: None

Flowchart:



15.5 PROGRAM LISTING

```
ST-NO   OBJECT    ADRS   SOURCE STATEMENTS

00001   100       0000           LLEN     132
00002                            TITLE    LOW POWER DISSIPATION MODE AND HA1835P CONTROL
00003                  *
00004                  ****   RAM ALLOCATION   ****************************************
00005                  *
00006                  ERRAM1   EQU      $3F           WATCHDOG DATA RAM AREA
00007                  ERRAM2   EQU      $3E
00008                  D8FLAG   EQU      $0.$03D       D8 PORT FLAG
00009                  CNTRL    EQU      $3C           LOWER 1S COUNTER
00010                  CNTRH    EQU      $3B           UPPER 1S COUNTER
00011                  *
00012                  ****   SYMBOL DEFINITIONS   ************************************
00013                  *
00014                  TMA      EQU      $008          TIMER MODE REGISTER A
00015                  IFTA     EQU      2.$001        IF OF TIMER A
00016                  IMTA     EQU      3.$001        IM OF TIMER A
00017                  IE       EQU      0.$000        ENABLE INTERRUPT
00018                  ********************************************************************
00019                  *                                                                  *
00020                  *               VECTOR ADDRESSES                                   *
00021                  *                                                                  *
00022                  ********************************************************************
00023                  *
00024                           ORG      $0000
00025                  *
00026   150 100   0000         JMPL     LWPMN         RESET
00027   150 100   0002         JMPL     LWPMN         INTO
00028   150 100   0004         JMPL     LWPMN         INT1
00029   150 158   0006         JMPL     LWPCNT        TIMER-A
00030   150 100   0008         JMPL     LWPMN         TIMER-B
00031   000       000A         NOP
00032   000       000B         NOP
00033   150 100   000C         JMPL     LWPMN         SERIAL
00034                  ********************************************************************
00035                  *                                                                  *
00036                  *               MAIN PROGRAM : LWPMN                               *
00037                  *                                                                  *
00038                  ********************************************************************
00039                  *
00040                           ORG      $0100
00041                  *
00042   0F0       0100 LWPMN    LWI      $0            INITIALIZE W REGISTER
00043   1A0 008   0101         LMID     0.TMA         SELECT PRESCALER AS 1/2048
00044   18A 001   0103         REMD     IFTA          INITIALIZE IFTA
00045   18B 001   0105         REMD     IMTA          INITIALIZE IMTA
00046   184 000   0107         SEMD     IE            ENABLE INTERRUPTS
00047   2A0       0109 LWPMN1   TDD      $0            TEST IF LOW POWER DISIIPATION MODE OR HA1835P MODE
00048   30F       010A         BR       LWPMN2
00049   160 137   010B         CALL     LWPMOD        EXECUTE LOW POWER MODE
00050   150 109   010D         JMPL     LWPMN1
00051   160 113   010F LWPMN2   CALL     LWPWCH        EXECUTE HA1835P MODE
00052   150 109   0111         JMPL     LWPMN1
00053                  ********************************************************************
00054                  *                                                                  *
00055                  *       NAME : LWPWCH (CONTROL HA1835P)                            *
00056                  *                                                                  *
00057                  ********************************************************************
```

```
00058                    *                                                              *
00059                    *       ENTRY : NOTHING                                        *
00060                    *     RETURNS : NOTHING                                        *
00061                    *                                                              *
00062                    ****************************************************************
00063   2E4      0113    LWPWCH   SEDD     $4            TURN ON LED FOR WATCH DOG
00064   12C 03F  0114             INEMD    $C.ERRAM1     POWER ON RESET OR WATCH DOG RESET?
00065   31C      0116             BR       LWPWH1
00066   125 03E  0117             INEMD    $5.ERRAM2
00067   31C      0119             BR       LWPWH1
00068   2E5      011A             SEDD     $5            TURN ON LED FOR WATCH DOG TIMER RESET
00069   322      011B             BR       LWPWH2
00070   1AC 03F  011C    LWPWH1   LMID     $C.ERRAM1     LOAD DATA FOR POWER ON RESET
00071   1A5 03E  011E             LMID     $5.ERRAM2
00072   2E5      0120             SEDD     $5
00073   264      0121             REDD     $4
00074   2A3      0122    LWPWH2   TDD      $3
00075   322      0123             BR       LWPWH2        TEST IF WATCH DOG TIMER ERR OR SWITCH IS ON
00076   265      0124             REDD     $5
00077   21F      0125             LYI      $F            EXECUTE 1MS SOFTWARE TIMER
00078   20F      0126    LWPWH3   LBI      $F
00079   0CF      0127    LWPWH4   DB
00080   327      0128             BR       LWPWH4
00081   0DF      0129             DY
00082   326      012A             BR       LWPWH3
00083   18C 03D  012B             TMD      D8FLAG        READ D8 PORT CONDITION
00084   332      012D             BR       LWPWH5
00085   2E8      012E             SEDD     $8            OUTPUT PULSE TO HA1835P EVERY 1MS
00086   184 03D  012F             SEMD     D8FLAG
00087   335      0131             BR       LWPWH6
00088   268      0132    LWPWH5   REDD     $8
00089   188 03D  0133             REMD     D8FLAG
00090   150 122  0135    LWPWH6   JMPL     LWPWH2
00091                    ****************************************************************
00092                    *                                                              *
00093                    *       NAME : LWPMOD (LOW POWER DISSIPATION)                  *
00094                    *                                                              *
00095                    ****************************************************************
00096                    *                                                              *
00097                    *      ENTRY : NOTHING                                         *
00098                    *    RETURNS : NOTHING                                         *
00099                    *                                                              *
00100                    ****************************************************************
00101   264      0137    LWPMOD   REDD     $4
00102   125 03F  0138             INEMD    $5.ERRAM1     POWER ON RESET OR STOP MODE RESET?
00103   340      013A             BR       LWPMD1
00104   12C 03E  013B             INEMD    $C.ERRAM2
00105   340      013D             BR       LWPMD1
00106   2E7      013E             SEDD     $7            TURN ON LED  FOR STOP MODE RESET
00107   344      013F             BR       LWPMD2
00108   1A5 03F  0140    LWPMD1   LMID     $5.ERRAM1     LOAD DATA FOR STOP MODE RESET
00109   1AC 03E  0142             LMID     $C.ERRAM2
00110   2A2      0144    LWPMD2   TDD      $2            TEST IF STOP MODE EXECUTION
00111   347      0145             BR       LWPMD3
00112   348      0146             BR       LWPMD4
00113   14D      0147    LWPMD3   STOP
00114   2A1      0148    LWPMD4   TDD      $1            TEST IF STANDBY MODE EXECUTION
```

```
00115   352       0149            BR      LWPMD5
00116   190 03C   014A            LAMD    CNTRL           TURN ON LED
00117   2D0       014C            LRA     $0
00118   190 03B   014D            LAMD    CNTRH
00119   2D1       014F            LRA     $1
00120   150 144   0150            JMPL    LWPMD2
00121   267       0152  LWPMD5    REDD    $7              TURN OFF LED FOR STOP MODE
00122   2E6       0153            SEDD    $6              TURN OFF LED FOR STAND BY MODE
00123   14C       0154            SBY                     EXECUTE STANDBY MODE
00124   266       0155            REDD    $6              TURN OFF LED FOR STAND BY MODE
00125   150 148   0156            JMPL    LWPMD4
00126                   ********************************************************************
00127                   *                                                                  *
00128                   *       NAME : LWPCNT (INCREMENT COUNTER)                          *
00129                   *                                                                  *
00130                   ********************************************************************
00131   18A 001   0158  LWPCNT    REMD    IFTA            CLEAR INTERRUPT REQUEST FLAG
00132   2FF       015A            XMRA    $F              SAVE ACCA
00133   0EF       015B            SEC                     INCREMENT LSD OF COUNTER
00134   230       015C            LAI     $0
00135   118 03C   015D            AMCD    CNTRL
00136   194 03C   015F            LMAD    CNTRL
00137   230       0161            LAI     $0              INCREMENT MSD OF COUNTER
00138   118 03B   0162            AMCD    CNTRH
00139   194 03B   0164            LMAD    CNTRH
00140   2FF       0166            XMRA    $F              RESTORE ACCA
00141   011       0167            RTNI
00142                   *
00143                           END
```

INSTRUCTION SET

$a \rightarrow b$	Transfer from "a" to "b"
$a \leftrightarrow b$	Exchange between "a" and "b"
$\overline{X}$	Logical negation (NOT)
"1"	"High" level
"0"	"Low" level
LSB	Least Significant Bit
MSB	Most Significant Bit
NZ	None Zero
NB	No Borrow
OVF	Overflow
$\cap$	AND
$\cup$	OR
$\oplus$	Exclusive OR
$\neq$	Not Equal
$\leqq$	Less than or Equal
DIRECT	Addressing by the instruction operand in the ROM code
REGISTER	Addressing by the contents of Address Register
I/E	Interrupt enable flag
PC	Program counter
SP	Stack pointer
ST	Status flag
CA	Carry flag
A	Accumulator
B	B register
W	W register
X	X register
SPX	SPX register
Y	Y register
SPY	SPY register
M	Memory (RAM)
MR	Memory register
R	Data I/O pin or data register
D	Discrete I/O pin or discrete latch

NZ, NB, OVF: Status is set when NZ, NB, or OVF occurs

Symbolic Operands Used with Instruction Set Mnemonics

Symbol	Contents	Label	D	HD	B
n	RAM digit selection by 2 bits	○	○	○	○
m	RAM (M_R) digit and port selection by 4 bits	○	○	○	○
p	Replacement of contents of PC by 4 bits	○	○	○	○
a	Replacement of contents of PC by 6 bits	○	×	○	○
b	Replacement of contents of PC by 8 bits	○	×	○	○
d	Replacement of PC or RAM direct address by 10 bits	○	×	○	○
i	Immediate data by 4 bits (Note)	○	○	○	○
u	u = p + d (14 bits)	○	×	○	○

D: Decimal HD: Hexadecimal B: Binary

○ .... Can be used × .... Cannot be used

(Note) In case of LWI instruction, 2 bits.

Immediate Instruction

OPERATION	MNEMONIC	OPERATION CODE	FUNCTION	STATUS	WORD / CYCLE
Load A from Immediate	LAI i	1 0 0 0 1 1 i_3 i_2 i_1 i_0	i⟶A		1/1
Load B from Immediate	LBI i	1 0 0 0 0 0 i_3 i_2 i_1 i_0	i⟶B		1/1
Load Memory from Immediate	LMID i,d	0 1 1 0 1 0 i_3 i_2 i_1 i_0 d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	i⟶M		2/2
Load Memory from Immediate, Increment Y	LMIIY i	1 0 1 0 0 1 i_3 i_2 i_1 i_0	i→M,Y+1→Y	NZ	1/1

Register-to-Register Instruction

OPERATION	MNEMONIC	OPERATION CODE	FUNCTION	STATUS	WORD / CYCLE
Load A from B	LAB	0 0 0 1 0 0 1 0 0 0	B→A		1/1
Load B from A	LBA	0 0 1 1 0 0 1 0 0 0	A→B		1/1
Load A from Y	LAY	0 0 1 0 1 0 1 1 1 1	Y→A		1/1
Load A from SPX	LASPX	0 0 0 1 1 0 1 0 0 0	SPX→A		1/1
Load A from SPY	LASPY	0 0 0 1 0 1 1 0 0 0	SPY→A		1/1
Load A from MR	LAMR m	1 0 0 1 1 1 m_3 m_2 m_1 m_0	MR(m)→A		1/1
Exchange MR and A	XMRA m	1 0 1 1 1 1 m_3 m_2 m_1 m_0	MR(m)↔A		1/1

RAM Address Instruction

OPERATION	MNEMONIC	OPERATION CODE	FUNCTION	STATUS	WORD / CYCLE
Load W from Immediate	LWI i	0 0 1 1 1 1 0 0 i_1 i_0	i⟶W		1/1
Load X from Immediate	LXI i	1 0 0 0 1 0 i_3 i_2 i_1 i_0	i⟶X		1/1
Load Y from Immediate	LYI i	1 0 0 0 0 1 i_3 i_2 i_1 i_0	i⟶Y		1/1
Load X from A	LXA	0 0 1 1 1 0 1 0 0 0	A⟶X		1/1
Load Y from A	LYA	0 0 1 1 0 1 1 0 0 0	A⟶Y		1/1
Increment Y	IY	0 0 0 1 0 1 1 1 0 0	Y+1→Y	NZ	1/1
Decrement Y	DY	0 0 1 1 0 1 1 1 1 1	Y−1→Y	NB	1/1
Add A to Y	AYY	0 0 0 1 0 1 0 1 0 0	Y+A→Y	OVF	1/1
Subtract A from Y	SYY	0 0 1 1 0 1 0 1 0 0	Y−A→Y	NB	1/1
Exchange X and SPX	XSPX	0 0 0 0 0 0 0 0 0 1	X↔SPX		1/1
Exchange Y and SPY	XSPY	0 0 0 0 0 0 0 0 1 0	Y↔SPY		1/1
Exchange X and SPX,Y and SPY	XSPXY	0 0 0 0 0 0 0 0 1 1	X↔SPX,Y↔SPY		1/1

RAM Register Instruction

OPERATION	MNEMONIC	OPERATION CODE	FUNCTION	STATUS	WORD / CYCLE
Load A from Memory	LAM(XY)	0 0 1 0 0 1 0 0 y x	M→A, $\binom{X\leftrightarrow SPX}{Y\leftrightarrow SPY}$		1/1
Load A from Memory	LAMD d	0 1 1 0 0 1 0 0 0 0 $d_9\ d_8\ d_7\ d_6\ d_5\ d_4\ d_3\ d_2\ d_1\ d_0$	M→A		2/2
Load B from Memory	LBM(XY)	0 0 0 1 0 0 0 0 y x	M→B, $\binom{X\leftrightarrow SPX}{Y\leftrightarrow SPY}$		1/1
Load Memory from A	LMA(XY)	0 0 1 0 0 1 0 1 y x	A→M, $\binom{X\leftrightarrow SPX}{Y\leftrightarrow SPY}$		1/1
Load Memory from A	LMAD d	0 1 1 0 0 1 0 1 0 0 $d_9\ d_8\ d_7\ d_6\ d_5\ d_4\ d_3\ d_2\ d_1\ d_0$	A→M		2/2
Load Memory from A, Increment Y	LMAIY(X)	0 0 0 1 0 1 0 0 0 x	A→M, Y+1→Y(X↔SPX)	NZ	1/1
Load Memory from A, Decrement Y	LMADY(X)	0 0 1 1 0 1 0 0 0 x	A→M, Y−1→Y(X↔SPX)	NB	1/1
Exchange Memory and A	XMA(XY)	0 0 1 0 0 0 0 0 y x	M↔A, $\binom{X\leftrightarrow SPX}{Y\leftrightarrow SPY}$		1/1
Exchange Memory and A	XMAD d	0 1 1 0 0 0 0 0 0 0 $d_9\ d_8\ d_7\ d_6\ d_5\ d_4\ d_3\ d_2\ d_1\ d_0$	M↔A		2/2
Exchange Memory and B	XMB(XY)	0 0 1 1 0 0 0 0 y x	M↔B, $\binom{X\leftrightarrow SPX}{Y\leftrightarrow SPY}$		1/1

Note) (XY) and (x) have the meaning as follows:

(1) The instructions with (XY) have 4 mnemonics and 4 object codes for each. (example of LAM (XY) is given below.)

MNEMONIC	y	x	FUNCTION
LAM	0	0	
LAMX	0	1	X↔SPX
LAMY	1	0	Y↔SPY
LAMXY	1	1	X↔SPX, Y↔SPY

(2) The instructions with (x) have 2 mnemonics and 2 object codes for each. (example of LMAIY(X) is given below.)

MNEMONIC	x	FUNCTION
LMAIY	0	
LMAIYX	1	X↔SPX

Arithmetic Instruction

OPERATION	MNEMONIC	OPERATION CODE	FUNCTION	STATUS	WORD/CYCLE
Add Immediate to A	AI i	1 0 1 0 0 0 i_3 i_2 i_1 i_0	A+i→A	OVF	1/1
Increment B	IB	0 0 0 1 0 0 1 1 0 0	B+1→B	NZ	1/1
Decrement B	DB	0 0 1 1 0 0 1 1 1 1	B−1→B	NB	1/1
Decimal Adjust for Addition	DAA	0 0 1 0 1 0 0 1 1 0			1/1
Decimal Adjust for Subtraction	DAS	0 0 1 0 1 0 1 0 1 0			1/1
Negate A	NEGA	0 0 0 1 1 0 0 0 0 0	$\bar{A}$+1→A		1/1
Complement B	COMB	0 1 0 1 0 0 0 0 0 0	$\bar{B}$→B		1/1
Rotate Right A with Carry	ROTR	0 0 1 0 1 0 0 0 0 0			1/1
Rotate Left A with Carry	ROTL	0 0 1 0 1 0 0 0 0 1			1/1
Set Carry	SEC	0 0 1 1 1 0 1 1 1 1	1→CA		1/1
Reset Carry	REC	0 0 1 1 1 0 1 1 0 0	0→CA		1/1
Test Carry	TC	0 0 0 1 1 0 1 1 1 1		CA	1/1
Add A to Memory	AM	0 0 0 0 0 0 1 0 0 0	M+A→A	OVF	1/1
Add A to Memory	AMD d	0 1 0 0 0 0 1 0 0 0 d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	M+A→A	OVF	2/2
Add A to Memory with Carry	AMC	0 0 0 0 0 1 1 0 0 0	M+A+CA→A OVF→CA	OVF	1/1
Add A to Memory with Carry	AMCD d	0 1 0 0 0 1 1 0 0 0 d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	M+A+CA→A OVF→CA	OVF	2/2
Subtract A from Memory with Carry	SMC	0 0 1 0 0 1 1 0 0 0	M−A−$\overline{CA}$→A NB→CA	NB	1/1
Subtract A from Memory with Carry	SMCD d	0 1 1 0 0 1 1 0 0 0 d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	M−A−$\overline{CA}$→A NB→CA	NB	2/2
OR A and B	OR	0 1 0 1 0 0 0 1 0 0	A∪B →A		1/1
AND Memory with A	ANM	0 0 1 0 0 1 1 1 0 0	A∩M→A	NZ	1/1
AND Memory with A	ANMD d	0 1 1 0 0 1 1 1 0 0 d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	A∩M→A	NZ	2/2
OR Memory with A	ORM	0 0 0 0 0 0 1 1 0 0	A∪M→A	NZ	1/1
OR Memory with A	ORMD d	0 1 0 0 0 0 1 1 0 0 d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	A∪M→A	NZ	2/2
EOR Memory with A	EORM	0 0 0 0 0 1 1 1 0 0	A⊕M→A	NZ	1/1
EOR Memory with A	EORMD d	0 1 0 0 0 1 1 1 0 0 d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	A⊕M→A	NZ	2/2

Compare Instruction

OPERATION	MNEMONIC	OPERATION CODE	FUNCTION	STATUS	WORD/CYCLE
Immediate Not Equal to Memory	INEM i	0 0 0 0 1 0 i_3 i_2 i_1 i_0	i ≠ M	NZ	1/1
Immediate Not Equal to Memory	INEMD i,d	0 1 0 0 1 0 i_3 i_2 i_1 i_0 / d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	i ≠ M	NZ	2/2
A Not Equal to Memory	ANEM	0 0 0 0 0 0 0 0 1 0 0	A ≠ M	NZ	1/1
A Not Equal to Memory	ANEMD d	0 1 0 0 0 0 0 1 0 0 / d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	A ≠ M	NZ	2/2
B Not Equal to Memory	BNEM	0 0 0 1 0 0 0 1 0 0	B ≠ M	NZ	1/1
Y Not Equal to Immediate	YNEI i	0 0 0 1 1 1 i_3 i_2 i_1 i_0	Y ≠ i	NZ	1/1
Immediate Less or Equal to Memory	ILEM i	0 0 0 0 1 1 i_3 i_2 i_1 i_0	i ≦ M	NB	1/1
Immediate Less or Equal to Memory	ILEMD i,d	0 1 0 0 1 1 i_3 i_2 i_1 i_0 / d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	i ≦ M	NB	2/2
A Less or Equal to Memory	ALEM	0 0 0 0 0 1 0 1 0 0	A ≦ M	NB	1/1
A Less or Equal to Memory	ALEMD d	0 1 0 0 0 1 0 1 0 0 / d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	A ≦ M	NB	2/2
B Less or Equal to Memory	BLEM	0 0 1 1 0 0 0 1 0 0	B ≦ M	NB	1/1
A Less or Equal to Immediate	ALEI i	1 0 1 0 1 1 i_3 i_2 i_1 i_0	A ≦ i	NB	1/1

RAM Bit Manipulation Instruction

OPERATION	MNEMONIC	OPERATION CODE	FUNCTION	STATUS	WORD/CYCLE
Set Memory Bit	SEM n	0 0 1 0 0 0 0 1 n_1 n_0	1→M(n)		1/1
Set Memory Bit	SEMD n,d	0 1 1 0 0 0 0 1 n_1 n_0 / d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	1→M(n)		2/2
Reset Memory Bit	REM n	0 0 1 0 0 0 1 0 n_1 n_0	0→M(n)		1/1
Reset Memory Bit	REMD n,d	0 1 1 0 0 0 1 0 n_1 n_0 / d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	0→M(n)		2/2
Test Memory Bit	TM n	0 0 1 0 0 0 1 1 n_1 n_0		M(n)	1/1
Test Memory Bit	TMD n,d	0 1 1 0 0 0 1 1 n_1 n_0 / d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0		M(n)	2/2

ROM Address Instruction

OPERATION	MNEMONIC	OPERATION CODE	FUNCTION	STATUS	WORD/CYCLE
Branch on Status 1	BR b	1 1 b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0		1	1/1
Long Branch on Status 1	BRL u	0 1 0 1 1 1 p_3 p_2 p_1 p_0 / d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0		1	2/2
Long Jump Unconditionally	JMPL u	0 1 0 1 0 1 p_3 p_2 p_1 p_0 / d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0			2/2
Subroutine Jump on Status 1	CAL a	0 1 1 1 a_5 a_4 a_3 a_2 a_1 a_0		1	1/2
Long Subroutine Jump on Status 1	CALL u	0 1 0 1 1 0 p_3 p_2 p_1 p_0 / d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0		1	2/2
Table Branch	TBR p	0 0 1 0 1 1 p_3 p_2 p_1 p_0			1/1
Return from Subroutine	RTN	0 0 0 0 0 1 0 0 0 0			1/3
Return from Interrupt	RTNI	0 0 0 0 0 1 0 0 0 1	1→I/E CA RESTORE	ST	1/3

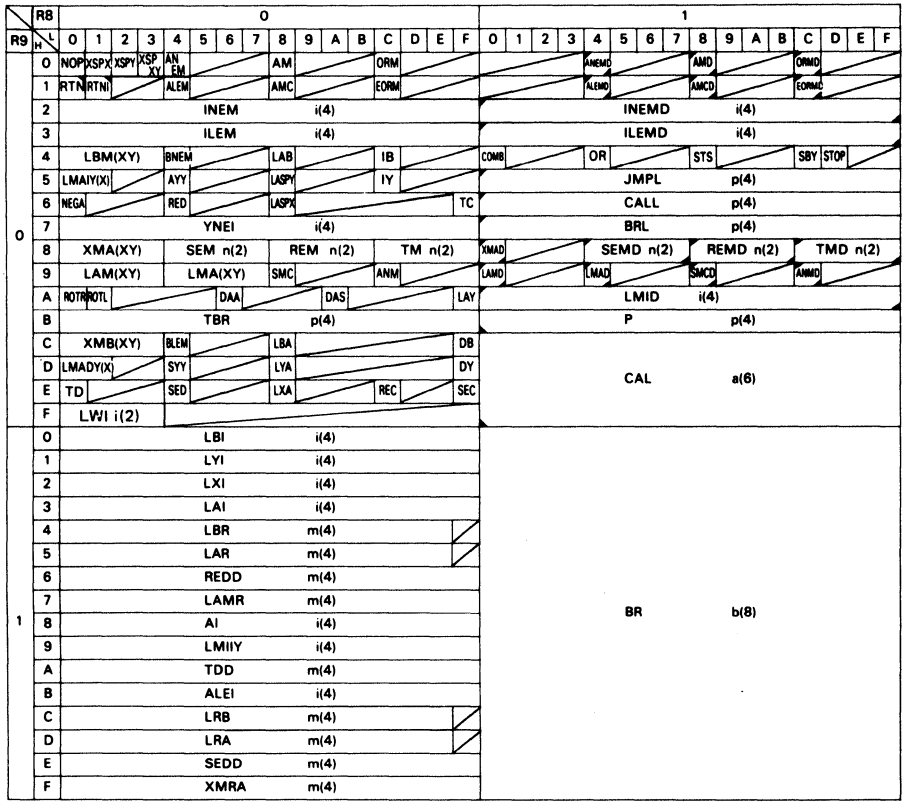
Input/Output Instruction

OPERATION	MNEMONIC	OPERATION CODE	FUNCTION	STATUS	WORD/CYCLE
Set Discrete I/O Latch	SED	0 0 1 1 1 0 0 1 0 0	1→D(Y)		1/1
Set Discrete I/O Latch Direct	SEDD m	1 0 1 1 1 0 m_3 m_2 m_1 m_0	1→D(m)		1/1
Reset Discrete I/O Latch	RED	0 0 0 1 1 0 0 1 0 0	0→D(Y)		1/1
Reset Discrete I/O Latch Direct	REDD m	1 0 0 1 1 0 m_3 m_2 m_1 m_0	0→D(m)		1/1
Test Discrete I/O Latch	TD	0 0 1 1 1 0 0 0 0 0		D(Y)	1/1
Test Discrete I/O Latch Direct	TDD m	1 0 1 0 1 0 m_3 m_2 m_1 m_0		D(m)	1/1
Load A from R-Port Register	LAR m	1 0 0 1 0 1 m_3 m_2 m_1 m_0	R(m)→A		1/1
Load B from R-Port Register	LBR m	1 0 0 1 0 0 m_3 m_2 m_1 m_0	R(m)→B		1/1
Load R-Port Register from A	LRA m	1 0 1 1 0 1 m_3 m_2 m_1 m_0	A→R(m)		1/1
Load R-Port Register from B	LRB m	1 0 1 1 0 0 m_3 m_2 m_1 m_0	B→R(m)		1/1
Pattern Generation	P p	0 1 1 0 1 1 p_3 p_2 p_1 p_0			1/2

Control Instruction

OPERATION	MNEMONIC	OPERATION CODE	FUNCTION	STATUS	WORD / CYCLE
No Operation	NOP	0000000000			1/1
Start Serial	STS	0101001000			1/1
Stand-by Mode	SBY	0101001100			1/1
Stop Mode	STOP	0101001101			1/1

Op-Code Map



... 1-word/2-cycle Instruction

... 1-word/3-cycle Instruction

... RAM Direct Address Instruction (2-word/2-cycle)

... 2-word/2-cycle Instruction

NOTES

NOTES

NOTES

HITACHI AMERICA, LTD.
SEMICONDUCTOR AND IC DIVISION

HEADQUARTERS

Hitachi, Ltd.
New Marunouchi Bldg., 5-1,
Marunouchi 1-chome
Chiyoda-ku, Tokyo 100, Japan
Tel: Tokyo (03) 212-1111
Telex: J22395, J22432, J24491,
J26375 (HITACHY)
Cable: HITACHY TOKYO

U.S. SALES OFFICE

Hitachi America, Ltd.
Semiconductor and IC Division
2210 O'Toole Avenue
San Jose, CA 95131
Tel: 408-435-8300
Telex: 17-1581
Twx: 910-338-2103
Fax: 408-435-2748
Fax: 408-435-2749
Fax: 408-435-2782

REGIONAL OFFICES

MID-ATLANTIC REGION

Hitachi America, Ltd.
1700 Galloping Hill Rd.
Kenilworth, NJ 07033
201/245-6400

NORTHEAST REGION

Hitachi America, Ltd.
5 Burlington Woods Drive
Burlington, MA 01803
617/229-2150

SOUTH CENTRAL REGION

Hitachi America, Ltd.
Two Lincoln Centre, Suite 865
5420 LBJ Freeway
Dallas, TX 75240
214/991-4510

NORTH CENTRAL REGION

Hitachi America, Ltd.
500 Park Blvd., Suite 415
Itasca, IL 60143
312/773-4864

NORTHWEST REGION

Hitachi America, Ltd.
2210 O'Toole Avenue
San Jose, CA 95131
408/435-2200

SOUTHWEST REGION

Hitachi America, Ltd.
18300 Von Karman Avenue, Suite 730
Irvine, CA 92715
714/553-8500

SOUTHEAST REGION

Hitachi America, Ltd.
4901 N.W. 17th Way, Suite 302
Fort Lauderdale, FL 33309
305/491-6154

DISTRICT OFFICES

- Hitachi America, Ltd.
  3800 W. 80th Street, Suite 1050
  Bloomington, MN 55431
  612/896-3444

- Hitachi America, Ltd.
  80 Washington St., Suite 302
  Poughkeepsie, NY 12601
  914/485-3400

- Hitachi America, Ltd.
  6 Parklane Blvd., #558
  Dearborn, MI 48126
  313/271-4410

- Hitachi America, Ltd.
  6161 Savoy Dr., Suite 850
  Houston, TX 77036
  713/974-0534

- Hitachi (Canadian) Ltd.
  2625 Queensview Dr.
  Ottawa, Ontario, Canada K2A 3Y4
  613/596-2777

- Hitachi America, Ltd.
  401 Harrison Oaks Blvd., Suite #317
  Cary, NC 27513
  919/481-3908

HITACHI®

We make things possible

Hitachi America, Ltd.
Semiconductor and IC Division
2210 O'Toole Avenue, San Jose, CA 95131
1-408-435-8300

10M

Printed in U.S.A.