I²C-Bus—Interface with HPC

National Semiconductor
Application Note 561
Hubert Utz
November 1989

INTRODUCTION

There are many applications in which microcontrollers are used as a central processor. These systems are designed with the following aspects:

— reduce and minimize system costs

— provide system flexibility

— simple connections to other peripheral devices (no high speed requirements)

A serial bus structure fulfills the above subjects.

The National Semiconductor microcontroller family provides the MICROWIRE/PLUS™ interface as a synchronous serial line to communicate with peripherals.

Another important serial bus is the I²C-Bus (Inter IC-Bus) which was developed by Valvo/Philips. It is mainly used in the customer area. This article describes a simple I²C-Bus interface with National's microcontroller family HPC 16xxx and two different software routines to work the interface:

a. Softwarepolling

b. Using the MICROWIRE™ shift register

THE I²C-Bus

The I²C-Bus is a bidirectional two line serial communication bus. The two wires, SDA (serial data) and SCL (serial clock) carry information between the different devices connected to the bus.

The devices can operate either as a receiver or a transmitter, depending on their functions.

The I²C-Bus also supports multimaster mode. Each device has its own 7-bit address.

This address consists commonly of a fixed hardwired part (4 Bits chip intern) and a variable address part (3 Pins of the device).

The I²C-Bus is based on the following definitions:

—TRANSMITTER:	the device which sends the data to the serial data line
—RECEIVER:	the device which receives the data from the serial data line
—MASTER:	the device which starts a transfer, supplies the clock signals and terminates a current transfer cycle
—SLAVE:	the device which is addressed by the master
—MULTIMASTER:	more than one device can get the master to control the serial data bus and the serial clock bus
—ARBITRATION:	if more than one device simultaneously tries to control the bus, a simple arbitration procedure takes place, so that only one device can get the master
—SYNCHRONIZATION:	procedure to synchronize the clock signals of two or more devices (slow slaves)

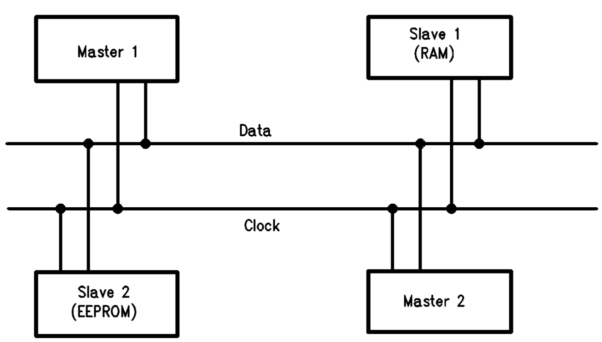
The maximum transmission rate is 100 kbit/s.

The maximum number of devices connected to the bus is limited by the maximum bus capacitance of 400 pF (typical device capacitance 10 pF).

Start-and Stop Conditions

The bus is not busy if both data- and clock lines remain HIGH because there are only two lines available, the start- and stop conditions have special timing definitions between these two lines:

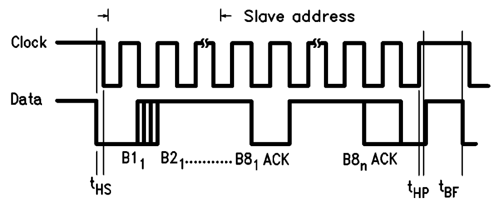
—start conditions:	HIGH-to-LOW transition of the data line, while the clock line is in a HIGH state.
—stop conditions:	LOW-to-HIGH transition of the data line, while the clock line is in a HIGH state.



TL/DD/10080–1

FIGURE 1. I²C-Bus Configurations

MICROWIRE™, MICROWIRE/PLUS™ and MOLE™ are trademarks of National Semiconductor Corporation.



TL/DD/10080–2

Startcondition

— Clock-, Dataline high (Bus free)

— change Dataline from high to low level

— after $t_{HS\ Min} = 4\ \mu s$ the master supplies the clock

Acknowledge

— transmitting Device releases the Dataline

— the receiving Device pulls the Dataline low during ACK-clock if there is no error

— if there is no ACK the master will generate a Stopcondition to abort the transfer

Stopcondition

— clockline goes high

— after $t_{HP\ Min} = 4.7\ \mu s$ datalines goes high

— the master remains the Data-, Clockline high

— next Startcondition after $t_{FB\ Min} = 4.7\ \mu s$ possible

FIGURE 2. I²C-Bus Timing

The master always generates the start and stop conditions. After the start condition the bus is in the busy state. The bus becomes free after the stop condition.

DATA BIT TRANSFER

After a start condition 'S' one databit is transferred during each clock pulse. The data must be stable during the HIGH-period of the clock. The data line can only change when the clock line is at a LOW level.

Normally each data transfer is done with 8 data bits and 1 acknowledge bit (byte format with acknowledge).

ACKNOWLEDGE

Each data transfer needs to be acknowledged. The master generates the acknowledge clock pulse. The transmitter releases the data line (SDA = HIGH) during the acknowledge clock pulse. If there was no error detected, the receiver will pull down the SDA-line during the HIGH period of the acknowledge clock pulse.

If a slave receiver is not able to acknowledge, the slave will keep the SDA line HIGH and the master can then generate a STOP condition to abort the transfer.

If a master receiver keeps the SDA line HIGH, during the acknowledge clock pulse the master signals the end of data transmission and the slave transmitter release the data line to allow the master to generate a STOP- condition.

ARBITRATION

Only in multi master systems.

If more than one device could be master and more than one wants to access the bus, an arbitration procedure takes place: if a master transmits a HIGH level and another master transmits a LOW level the master with the LOW level will get the bus and the other master will release the bus and the clockline immediately and switches to the slave receiver mode. This arbitration could carry on through many bits (address bits and data bits are used for arbitration).

FORMATS

There are three data transfer formats supported:

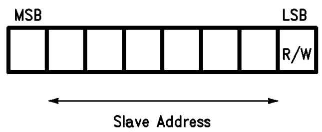
— master transmitter writes to slave receiver; no direction change

— master reads immediate after sending the address byte

— combined format with multiple read or write transfers (see . . .)

ADDRESSING

The 7-bit address of an I²C device and the direction of the following data is coded in the first byte after the start condition:

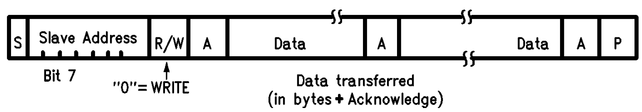


TL/DD/10080–3

A ''0'' on the least significant bit means that the master will write information to the selected Slave address device: a ''1'' means that the master will read data from the slave.

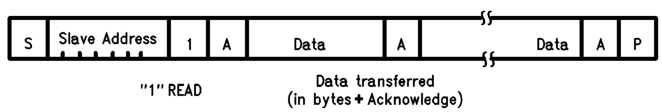
Some slave addresses are reserved for future use. These are all addresses with the bit combinations 1111XXX and 0000XXX. The address 00000000 is used for a general call address, for example to initialize all I²C devices (refer to I²C bus specification for detailed information).

—Master Transmits to Slave, No Direction Change



TL/DD/10080–4

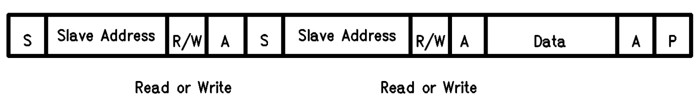
—Master Reads Slave Immediately after First Byte



TL/DD/10080–5

The master becomes a master receiver after first ACK

—Combined Formats



TL/DD/10080–6

nbytes Data + ACK nbytes Data + ACK

S = Startcondition A = Acknowledge P = Stopcondition

FIGURE 3. I²C-Bus Transfer Formats

TIMING

The master can generate a maximum clock frequency of 100 kHz. The minimum LOW period is defined with 4.17 μs, the minimum HIGH period width is 4 μs, the maximum rise time on SDA and SCL is 1 μs and the maximum fall time on SDA and SCL is 300 ns.

Figure 4 shows the detailed timing requirements.

Symbol	Parameter	Min	Max	Units
f_{SCL}	SCL Clock Frequency	0	100	kHz
t_{BUF}	Time the Bus Must Be Free before a New Transmission Can Start	4.7		μs
t_{HD}; STA	Hold Time Start Condition. After This Period the First Clock Pulse Is Generated	4.0		μs
t_{LOW}	The LOW Period of the Clock	4.7		μs
t_{SU}; STA	Setup Time for Start Condition (Only Relevant for a Repeated Start Condition)	4.7		μs
t_{HD}; DAT	Hold Time DATA for CBUS Compatible Masters (See Also NOTE, Section 8.1.3.) for I^2C Device	5 0*		μs μs
t_{SU}; DAT	Setup Time Data	250		ns
t_R	Rise Time of Both SDA and SCL Lines		1	μs
t_F	Fall Time of Both SDA and SCL Lines		300	ns
t_{SU};STO	Setup Time for Stop Condition	4.7		μs

All values referred to $V_{IH\ Min}$ = 3.0V and $V_{IL\ Min}$ = 1.5V levels at 5V supply voltage

*Note that a transmitter must internally provide at least a hold time to bridge the undefined region (max. 300 ns) of the falling edge of SCL.

FIGURE 4. I^2C-Bus Timing Requirements

—Two Wire Serial Bus with

*Data line
*Clock line

—Features:

*Multimaster Bus (Master/Slave)
*Busarbitration
*Transfer rate up to 100 kbits/s
*Bytetransfers
*Protocols with
—Start Condition
—Address
—Ackn.
—Data
—Ackn. (Each Byte)
—Stop Condition

*Read, Write, Multiple R/W

FIGURE 5. I²C-Bus Features

The I²C test hardware uses the following components:

2 x PC F 8570: 256 x 8-Bit RAM
RAM 1: Address: 1010000X
RAM 2: Address: 1010010X

2 x PC F 8582: 256 x 8 Bit EEPROM
EEPROM 1: Address: 1010001X
EEPROM 2: Address: 1010011X

2 x PC F 8574: Remote 8-Bit I/O Expander
I/O 1: Address: 0100000X Used as 8-Bit LED Output Port
I/O 2: Address: 0100001X Used as 8-Bit Input Port

1 x HCT04:	Inverter
1 x LS05:	Inverter, Open Collector
8 x LEDs:	Connected Via Pull Up Resistors to Output Pins of PCF 8574
2 x Rp:	Pull Up Resistors for Clock Line and Data Line
8 Switches:	Connected Via Pull Up Resistors to Input Pins of PCF 8574
1 x Pin Grid Socket:	Socket for MOLE™ Connection
1 x Power Connector:	5V Power Supply

Four I/O lines of the HPC are used to connect a HPC-MOLE or a HPC-Designer Kit to the I²C-board: SO, SI, P0, and SK. SO drives the data bus line; SDA and P0 drive the clock bus line SCL.

The data on the SDA line is monitored by the input SI and the I²C-bus clock is available at input SK.

SI, SO and SK are μWire interface lines.

P0 is used as a continuous timer output during the transfer. All rise and fall times meet the I²C-bus specification. The highest I²C-clock frequency you can get with a 17 MHz HPC oscillator/4 Waitstates is ca. 20 kHz.

I²C-Bus Software HPC

*Master only Mode

*Tested I²C-Clock Frequency 16 kHz–20 kHz

*3 Possible Formats Supported
—Read
—Write
—Multiple Read or Write

*Two Program Versions
—Programmed I/O
—Interrupt Driven I/O

*Demo Loop:
—Write Output
—RAM test
—Read Input
—Increment Output or
Set Output = Input

*IRQ Driven Program Uses μWire Shift register

*Message Field:

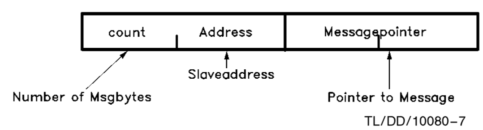
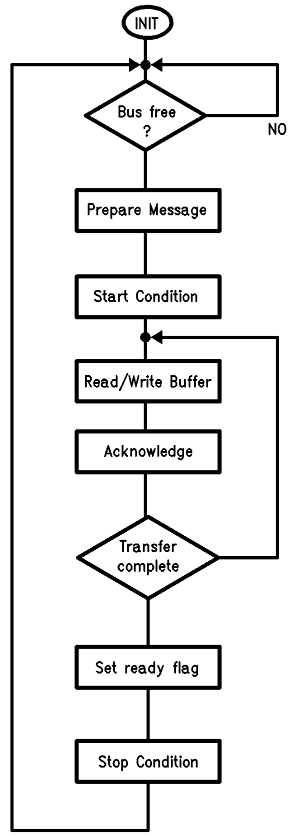


Figure 7. Software Features



TL/DD/10080–8

FIGURE 8. Programmed I/O Flowchart

```
        ;****************************************************
        ;*  INTER-IC-BUS (I2C-BUS) WITH HPC  : APPL.NOTE
        ;****************************************************

        ;* 12.10.87                                        HU
        ;* 20.10.87                                        HU

        .INCLD  HPC16083.MAP     ; MEMORY MAP FOR HPC16083

        ;DEFINITIONS
        CLK = 32                 ;CLOCK LOW/HIGH TIME = 33us

        .MACRO SAVE_AB

        PUSH    A                ;SAVE A-REG
        PUSH    B                ;SAVE B-REG

        .ENDM

        .MACRO SAVE_ABX

        SAVE_AB                  ;SAVE REGS A,B
        PUSH    X                ;SAVE X-REG

        .ENDM

        .MACRO RESTORE_AB

        POP     B                ;RESTORE B-REG
        POP     A                ;RESTORE A_REG

        .ENDM

        .MACRO RESTORE_ABX

        POP     X                ;RESTORE X-REG
        RESTORE_AB               ;RESTORE REGS B,A

        .ENDM

        .SECT   B1,BASE          ;DEFINE BASEPAGE SECTION

WBUF1:  .DSW    1                ;WORDBUFFER FOR I2C-TABLE
WBUF2:  .DSW    1                ;WORDBUFFER FOR I2C-TABLE
INDEX:  .DSW    1                ;POINTER TO THE NEXT TABLEENTRY
STATUS: .DSB    1                ;STATUSBYTE
DUMMY:  .DSB    1                ;DUMMY BYTE
WPORT:  .DSB    1                ;8-BIT VALUE WRITE TO PORT0
RPORT:  .DSB    1                ;8-BIT VALUE READ FROM PORT1
WRBUFF: .DSB    10               ;RAM WRITE BUFFER
RDBUFF: .DSB    10               ;RAM READ BUFFER

        .ENDSECT

        .SECT   I2C,ROM16
```

TL/DD/10080–9

```
        ;****************************************************
        ;* INIT : THIS SUBROUTINE INITIALIZES TIMER T4, TIMER T5
        ;*        AND THE uWIRE-INTERFACE TO OPERATE AS I2C-BUS
        ;*
        ;* INPUT : NONE
        ;* OUTPUT : NONE
        ;* USED REGS : A, B, X   ( ALL REGS ARE SAVED )
        ;****************************************************

INIT:   SAVE_ABX                ;SAVE REGS A,B,X
        LD      X,#PWMODE       ;ADDR. PWMODE-REG -> X
        LD      B,#PORTP        ;ADDR. PORTP-REG -> B
        LD      A,#0C           ;VALUE TO STOP TIMER
        ST      A,[X].B         ;STOP T4, NO IRQ, ACK TIP-FLAG
        NOP
        ST      A,[X].B         ;MAKE SHURE TIP-FLAGS ARE CLEARED
        LD      A,#01           ;DISABLE TOGGLE AND SET OUTPUT HIGH
        ST      A,[B].B         ;ON PINS PO AND P1
        SBIT    3,[B].B         ;TOGGLE ON AT P0
        LD      T4.W,#CLK-1     ;LOAD T4 (33us)
        LD      R4.W,#CLK-1     ;LOAD R4 (33us)
        SBIT    5,PORTB.B       ;DATALINE OUTPUT = HIGH
        SBIT    5,DIRB.B        ;B5 = OUTPUT
        RBIT    5,BFUN.B        ;NO ALTERNATE FUNCTION SELECTED
        RBIT    6,DIRB.B        ;B6 = INPUT
        RBIT    6,BFUN.B        ;
        RESTORE_ABX             ;RESTORE REGS X,B,A
        RET


RWI2C:  PUSH    K               ;SAVE REG K
        SAVE_ABX                ;SAVE-REGISTER
        LD      B,#PORTB        ;ADDRESS OF PORTB -> REG B
RWRST:  LD      STATUS.B,#0     ;RESET STATUSBYTE
        JSR     TSTBUS          ;BUS FREE ?
        IFC
        JP      RWERR           ;IF ERROR -> EXIT
        LD      A,[X+].W        ;GET 2 BYTES OF TABLE
        ST      A,WBUF1.W       ;SAVE TABLE CONTENTS
        IFBIT   0,A             ;TEST RECEIVE/TRANSMIT BIT
        JP      RWRECV          ;BIT = 1 -> RECEIVE
        RBIT    0,STATUS.B      ;STATUS = TRANSMIT
        JP      RW01            ;CONTINUE AT RW01
RWRECV: SBIT    0,STATUS.B      ;STATUS = RECEIVE
RW01:   SBIT    1,STATUS.B      ;STATUS = FIRST BYTE
        LD      A,[X+].W        ;GET NEXT 2 BYTES OF TABLE
        ST      A,WBUF2.W       ;SAVE TABLE CONTENTS
        LD      A,X
        ST      A,INDEX.W       ;SAVE INDEX
        LD      A,[X].W         ;GET NEXT WORD OF TABLE
        IFEQ    A,#0            ;ANY MORE TO TRANSFER
        JP      RW02            ;NO, EXIT
        SBIT    3,STATUS.B      ;STATUS = MULTISTART
RW02:   SBIT    7,STATUS.B      ;STATUS = BUSY
```

TL/DD/10080–10

```
        RC                          ;CLR CARRY = NO ERROR

        JSR     STRTCD              ;STARTCONDITION
        IFC
        JP      STPERR
        LD      X,WBUF2.W           ;X = BUFFERINDEX
        LD      A,WBUF1.W           ;A.B = ADDRESS
        JSR     TRANSF              ;TRANSMITT 1.BYTE = ADDRESS
        IFC
        JP      STPERR
        DECSZ   WBUF1+1.B           ;DECREMENT BYTECOUNT
        NOP                         ;DUMMY FOR DECSZ
        IFBIT   0,STATUS.B          ;STATUS = RECEIVE ?
        JP      RCVE                ;YES -> RCVE
TRMIT:  LD      A,[X+].B            ;GET NEXT BYTE
        DECSZ   WBUF1+1.B           ;DECREMENT BYTECOUNT
        JP      TRM1                ;BYTECOUNT <> 0
        SBIT    2,STATUS.B          ;FLAG LAST BYTE
TRM1:   JSR     TRANSF              ;SEND BYTE
        IFC                         ;TEST ERROR
        JP      STPERR              ;ERROR DETECTED
        IFBIT   2,STATUS.B          ;LAST BYTE ?
        JP      TRM2                ;YES
        JP      TRMIT               ;NO -> TRANSFER AGAIN
TRM2:   IFBIT   3,STATUS.B          ;MULTISTART ?
        JP      TRM3                ;YES
        JP      TRM6                ;NO -> STOPCONDITION
TRM3:   SBIT    5,[B].B             ;DATALINE = HIGH
TRM4:   IFBIT   6,[B].B             ;WAIT UNTIL CLOCKLINE = HIGH
        JP      TRM5                ;CLOCK = HIGH
        JP      TRM4                ;CLOCK = LOW
TRM5:   SBIT    2,PWMODE.B          ;STOP TIMER 4
        LD      T4.W,#2*CLK-1       ;SET START TIME
        LD      X,INDEX.W           ;GET NEXT TABLEENTRY
        JP      RWRST               ;PERFORM NEXT STARTCONDITION

TRM6:   JP      RWEND
RCVE:   DECSZ   WBUF1+1.B           ;DECREMENT BYTECOUNT
        JP      RCV1                ;BYTECOUNT <> 0
        SBIT    2,STATUS.B          ;FLAG LAST BYTE
RCV1:   JSR     RECEIV              ;GET 1 BYTE
        ST      A,[X].B             ;PUT BYTE TO BUFFER
        INC     X                   ;X += 1
        IFC                         ;ERROR ?
        JP      STPERR              ;YES -> STOPCONDITION
        IFBIT   2,STATUS.B          ;LAST BYTE FLAGGED ?
        JP      TRM2                ;YES, CHECK MULTISTART
        JP      RCVE                ;GET NEXT BYTE
RWEND:  RC                          ;NO ERROR
STPERR: JSR     STOPCD              ;STOPCONDITION

RWERR:  RESTORE_ABX                 ;RESTORE REGISTER
        POP     K                   ;RESTORE REG K
        RET
```

TL/DD/10080–11

```
STRTCD: IFBIT   5,PORTI.B       ;TEST DATALINE
        JP      STRT01          ;IF HIGH -> CONTINUE
STRTER: SC                      ;ELSE ERROR
        RET
STRT01: IFBIT   6,[B].B         ;TEST CLOCKLINE
        JP      STRT02          ;IF HIGH -> CONTINUE
        JP      STRTER          ;ELSE ERROR
STRT02: RBIT    5,[B].B         ;DATALINE = LOW
        AND     PWMODE.B,#0FB   ;START TIMER 4
STRT03: IFBIT   6,[B].B         ;WAIT UNTIL CLOCK = LOW
        JP      STRT03
        RC                      ;SIGNAL NO ERROR
        RET

TRANSF: IFBIT   7,A             ;TEST FOR THE NEXT DATA
        JP      TRNF1           ;PUT DATALINE HIGH
        RBIT    5,[B].B         ;PUT DATALINE LOW
        JP      TRNF2
TRNF1:  SBIT    5,[B].B         ;PUT DATALINE HIGH
TRNF2:  SWAP    A
        SWAP    A               ;EXCHANGE LOWER/HIGHER BYTE
        LD      K,#8            ;SET LOOP COUNT
        SHL     A               ;DUMMY SHIFT
        JP      TRF2            ;JUMP INTO THE LOOP
TRF1:   SHL     A               ;SHIFT MSB -> CARRY
        IFC
        SBIT    5,[B].B         ;DATALINE = HIGH
        IFNC
        RBIT    5,[B].B         ;DATALINE = LOW
TRF2:   IFBIT   6,[B].B         ;WAIT UNTIL CLOCK HIGH
        JP      TRF3            ;CLOCK = HIGH
        JP      TRF2            ;CLOCK = LOW
TRF3:   IFBIT   6,[B].B         ;WAIT UNTIL CLOCK = LOW
        JP      TRF3            ;CLOCK = HIGH
        DECSZ   K               ;DECREMENT LOOP COUNT
        JP      TRF1            ;NEXT BIT
        JSR     GETACK          ;LOOK FOR ACKNOWLEDGE
        RET

SETACK: RBIT    5,[B].B         ;DATALINE = LOW
        RC
        JP      ACK01
GETACK: SBIT    5,[B].B         ;DATALINE = HIGH
        RC
ACK01:  IFBIT   6,[B].B         ;WAIT UNTIL CLOCK = HIGH
        JP      ACK02           ;CLOCK = HIGH
        JP      ACK01           ;CLOCK = LOW , WAIT
ACK02:  IFBIT   5,PORTI.B       ;TEST DATALINE
        SC                      ;FLAG EROR IF HIGH
ACK03:  IFBIT   6,[B].B         ;WAIT UNTIL CLOCK = LOW
        JP      ACK03
        SBIT    5,[B].B         ;DATALINE = HIGH
        RET
```

TL/DD/10080–12

```
RECEIV: PUSH    K               ;SAVE REG K
        LD      K,#8            ;SET LOOP COUNT
REC1:   RC
REC2:   IFBIT   6,[B].B         ;WAIT UNTIL CLOCK HIGH
        JP      REC3            ;CLOCK = HIGH
        JP      REC2            ;CLOCK = LOW
REC3:   IFBIT   5,PORTI.B       ;TEST DATALINE
        SC                      ;IF HIGH SET CARRY
        RLC     A               ;ROTATE LEFT WITH CARRY
REC4:   IFBIT   6,[B].B         ;WAIT UNTIL CLOCK = LOW
        JP      REC4            ;CLOCK = HIGH
        DECSZ   K               ;DECREMENT LOOP COUNT
        JP      REC1            ;NEXT BIT
        IFBIT   2,STATUS.B      ;LAST BYTE FLAGGED ?
        JP      REC5            ;YES, NO ACKNOWLEDGE
        JSR     SETACK          ;SET ACKNOWLEDGE
        JP      REC6
REC5:   JSR     GETACK          ;LOOK FOR ACKNOWLEDGE
REC6:   POP     K               ;RESTORE REG K
        RET


STOPCD: RBIT    5,[B].B         ;DATALINE = LOW
STOP01: IFBIT   6,[B].B         ;WAIT UNTIL CLOCK = HIGH
        JP      STOP02          ;CLOCK = HIGH -> STOP02
        JP      STOP01          ;WAIT
STOP02: SBIT    2,PWMODE.B      ;STOP TIMER 4
        LD      T4.W,#CLK-1     ;INITIALIZE T4 TO STARTCONDITION
        RBIT    7,STATUS.B      ;STATUS = I2CBUS NOT BUSY
        SBIT    5,[B].B         ;PERFORM STOPCONDITION
        RET

TSTBUS: RC
        IFBIT   5,PORTI.B       ;TEST DATALINE
        JP      TST1
        SC
TST1:   IFBIT   6,[B].B         ;TEST CLOCKLINE
        JP      TST2
        SC
TST2:   RET

RESET:  LD      ENIR.B,#0       ;DISABLE ALL INTERRUPTS
        LD      PWMODE.W,#0CCCC ;STOP AND CLEAR ALL TIMERS
        LD      TMMODE.W,#0CCCC
START:  JSR     INIT
        LD      B,#WRBUFF
        LD      K,#WRBUFF+8     ;CLEAR 9 BYTES
START0: CLR     A
        XS      A,[B+].W
        JP      START0
        LD      RDBUFF.B,#0     ;SET READADDRESS TO 0

        LD      WPORT.B,#0FF    ;INITIALISE PORT1 AS INPUT
        LD      X,#INIPO1
        JSR     RWI2C
```

TL/DD/10080–13

```
        LD      WPORT.B,#0FF    ;PUT ALL LED'S OFF
START1: LD      X,#WRPO0
        JSR     RWI2C

        LD      X,#WRRAM0       ;WRITE TO RAM
        JSR     RWI2C           ;START TRANSMISSION
        JSR     WAIT

        LD      X,#RDRAM0       ;READ RAM0
        JSR     RWI2C
        JSR     WAIT

        ADD     WRBUFF.B,#8     ;WRITE/READ NEXT 8 BYTES RAM
        ADD     RDBUFF.B,#8
        IFEQ    WRBUFF.B,#0     ;IF WRAP
        DECSZ   WPORT.B         ;DECREMENT LED VALUE
        NOP                     ;ONLY DECREMENT

        LD      X,#RDPO1        ;READ INPUT
        JSR     RWI2C
        JSR     WAIT

        IFBIT   7,RPORT         ;IF BIT SET FREE-RUN-LED
        JP      START1
        LD      WPORT.B,RPORT.B ;ELSE COPY INPUT TO OUTPUT

        JP      START1

WAIT:   PUSH    X               ;SAVE X-REG
        LD      X,#010          ;INITIALIZE WAITLOOP
WAIT1:  DECSZ   X
        JP      WAIT1
        POP     X               ;RESTORE X-REG
        RET

INIPO1: .DW     0242,WPORT,0            ;INITIALIZE PORT1 AS INPUT
RDPO1:  .DW     0243,RPORT,0            ;READ 1 BYTE FROM PORT1
WRPO0:  .DW     0240,WPORT,0            ;WRITE 1 BYTE TO PORT0
WRRAM0: .DW     0AA0,WRBUFF,0           ;WRITE 8 BYTES TO RAM
RDRAM0: .DW     02A0,WRBUFF,0AA1,RDBUFF+1,0     ;READ 10 BYTES

        .END RESET
```

TL/DD/10080–14

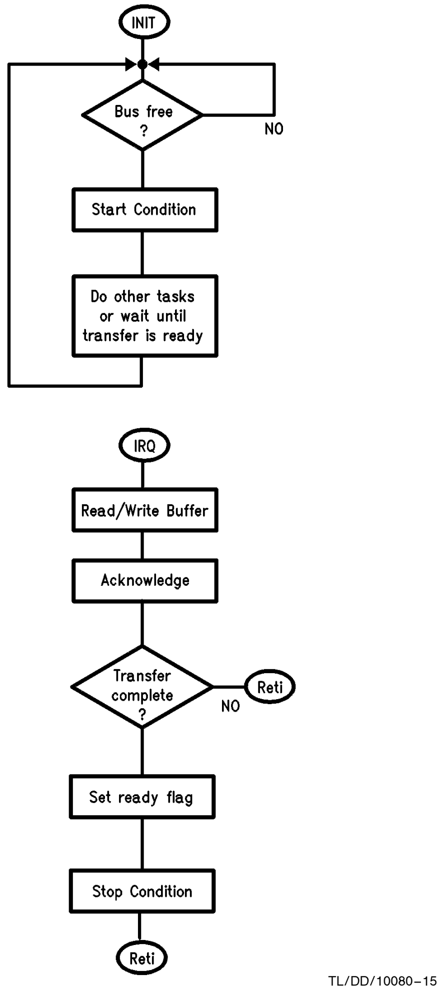


FIGURE 9. Interrupt Driver Flowchart

```
        ;****************************************************
        ;*  INTER-IC-BUS (I2C-BUS) WITH HPC  : APPL.NOTE
        ;*  using uWire interface with interrupt
        ;****************************************************

        ;* 12.10.87                                      HU
        ;* 20.10.87                                      HU
        ;* 04.11.87                                      HU
        ;* 08.02.88                                      HU

        .INCLD  HPC16083.MAP    ; MEMORY MAP FOR HPC16083

        ;DEFINITIONS
        CLK = 30                ;CLOCK LOW/HIGH TIME = 33us

        .MACRO SAVE_AB

        PUSH    A               ;SAVE A-REG
        PUSH    B               ;SAVE B-REG

        .ENDM

        .MACRO SAVE_ABX

        SAVE_AB                 ;SAVE REGS A,B
        PUSH    X               ;SAVE X-REG

        .ENDM

        .MACRO RESTORE_AB

        POP     B               ;RESTORE B-REG
        POP     A               ;RESTORE A_REG

        .ENDM

        .MACRO RESTORE_ABX

        POP     X               ;RESTORE X-REG
        RESTORE_AB              ;RESTORE REGS B,A

        .ENDM

        .SECT   B1,BASE         ;DEFINE BASEPAGE SECTION

WBUF1:  .DSW    1               ;WORDBUFFER FOR I2C-TABLE
WBUF2:  .DSW    1               ;WORDBUFFER FOR I2C-TABLE
INDEX:  .DSW    1               ;POINTER TO THE NEXT TABLEENTRY
STATUS: .DSB    1               ;STATUSBYTE
DUMMY:  .DSB    1               ;DUMMY BYTE
WPORT:  .DSB    1               ;8-BIT VALUE WRITE TO PORT0
RPORT:  .DSB    1               ;8-BIT VALUE READ FROM PORT1
WRBUFF: .DSB    10              ;RAM WRITE BUFFER
RDBUFF: .DSB    10              ;RAM READ BUFFER
```

TL/DD/10080–16

```
        .ENDSECT

        .SECT   I2C,ROM16

        ;*****************************************************
        ;* INIT : THIS SUBROUTINE INITIALIZES TIMER T4, TIMER T5
        ;*        AND THE uWIRE-INTERFACE TO OPERATE AS I2C-BUS
        ;*
        ;* INPUT : NONE
        ;* OUTPUT : NONE
        ;* USED REGS : A, B, X   ( ALL REGS ARE SAVED )
        ;*****************************************************

INIT:   SAVE_ABX                ;SAVE REGS A,B,X
        LD      X,#PWMODE       ;ADDR. PWMODE-REG -> X
        LD      B,#PORTP        ;ADDR. PORTP-REG -> B
        LD      A,#0CC          ;VALUE TO STOP TIMERS
        ST      A,[X].B         ;STOP T4, T5, NO IRQ, ACK TIP-FLAG
        NOP
        ST      A,[X].B         ;MAKE SHURE TIP-FLAGS ARE CLEARED
        LD      A,#011          ;DISABLE TOGGLE AND SET OUTPUT HIGH
        ST      A,[B].B         ;ON PINS PO AND P1
        SBIT    3,[B].B         ;TOGGLE ON AT P0
        SBIT    7,[B].B         ;TOGGLE ON AT P1
        LD      T4.W,#CLK-1     ;LOAD T4 (33us)
        LD      R4.W,#CLK-1     ;LOAD R4 (33us)
        LD      T5.W,#17*CLK-1  ;9-BIT SHIFT TIME (STARTCONDITION)
        LD      R5.W,#18*CLK-1  ;9-BIT SHIFT TIME (NORMAL MODE)
        SBIT    5,PORTB.B       ;DATALINE OUTPUT = HIGH
        SBIT    5,DIRB.B        ;B5 = OUTPUT
        RBIT    5,BFUN.B        ;NO ALTERNATE FUNCTION SELECTED
        RBIT    6,DIRB.B        ;B6 = INPUT
        SBIT    6,BFUN.B        ;SELECT SK-INPUT
        LD      A,DIVBY.B       ;SET UWIRE-DEVIDE
        AND     A,#0F0
        OR      A,#02           ;SET CLKI /16
        ST      A,DIVBY.B       ;STORE NEW VALUE
        SBIT    1,IRCD.B        ;ACTIVATE UWIRE
INIT1:  IFBIT   0,IRPD.B        ;TEST IF READY
        JP      INIT2           ;YES CONTINUE
        JP      INIT1           ;NO WAIT
INIT2:  RBIT    1,IRCD.B        ;SELECT SLAVE MODE
        RBIT    6,BFUN.B        ;
        SBIT    4,[X].B         ;ENABLE T5-IRQ
        OR      ENIR.B,#021     ;ENABLE GLOBAL TIMER IRQ
        RESTORE_ABX             ;RESTORE REGS X,B,A
        RET


RWI2C:  PUSH    A               ;SAVE A-REGISTER
        LD      STATUS.B,#0     ;RESET STATUSBYTE
        LD      A,[X+].W        ;GET 2 BYTES OF TABLE
        JSR     STRTCD          ;PERFORM STARTCONDITION
        IFC
```

TL/DD/10080–17

```
        JP      RWERR           ;IF ERROR -> EXIT
        SBIT    5,BFUN.B        ;ENABLE SO-OUTPUT
        ST      A,WBUF1.W       ;SAVE TABLE CONTENTS
        IFBIT   0,A             ;TEST RECEIVE/TRANSMIT BIT
        JP      RWRECV          ;BIT = 1 -> RECEIVE
        RBIT    0,STATUS.B      ;STATUS = TRANSMIT
        JP      RW01            ;CONTINUE AT RW01
RWRECV: SBIT    0,STATUS.B      ;STATUS = RECEIVE
RW01:   SBIT    1,STATUS.B      ;STATUS = FIRST BYTE
        DECSZ   WBUF1+1.B       ;DEC BYTECOUNT
        JP      RW02            ;MORE THAN 1 BYTE TO PROCESS
        SBIT    2,STATUS.B      ;STATUS = LAST BYTE
RW02:   LD      A,[X+].W        ;GET NEXT 2 BYTES OF TABLE
        ST      A,WBUF2.W       ;SAVE TABLE CONTENTS
        LD      A,X
        ST      A,INDEX.W       ;SAVE INDEX
        LD      A,[X].W         ;GET NEXT WORD OF TABLE
        IFEQ    A,#0            ;ANY MORE TO TRANSFER
        JP      RW03            ;NO, EXIT
        SBIT    3,STATUS.B      ;STATUS = MULTISTART
RW03:   SBIT    7,STATUS.B      ;STATUS = BUSY
        RC                      ;CLR CARRY = NO ERROR
RWERR:  POP     A               ;RESTORE A-REGISTER
        RET

STRTCD:
        IFBIT   5,PORTI.B       ;TEST DATALINE
        JP      STRT01          ;IF HIGH -> CONTINUE
STRTER: SC                      ;ELSE ERROR
        RET
STRT01: IFBIT   6,PORTB.B       ;TEST CLOCKLINE
        JP      STRT02          ;IF HIGH -> CONTINUE
        JP      STRTER          ;ELSE ERROR
STRT02: RBIT    5,PORTB.B       ;DATALINE = LOW
        AND     PWMODE.B,#0BB   ;START TIMER 4 AND 5
STRT03: IFBIT   6,PORTB.B       ;WAIT UNTIL CLOCK = LOW
        JP      STRT03
        ST      A,SIO.B         ;WRITE 1 BYTE TO SIO AND ENABLE SHIFT
        RC                      ;SIGNAL NO ERROR
        RET

TIMIRQ:
        IFBIT   5,PWMODE.B      ;TIMER 5 IRQ ?
        JP      T1IRQ           ;YES, CONTINUE
        JP      IRQRET          ;NO TIMER IRQ
T1IRQ:  SBIT    5,PORTB.B
        RBIT    5,BFUN.B
        SBIT    7,PWMODE.B      ;ACK IRQ
        SAVE_ABX                ;SAVE REGS A,B,X
        LD      B,#PORTB        ;PORTB-ADDR -> REG B
        LD      X,#STATUS       ;STATUS-ADDR -> REG-X
        PUSH    PSW.W           ;
        IFBIT   2,[X].B         ;LAST BYTE ?
        JP      LAST            ;YES -> JUMP
        IFBIT   1,[X].B         ;FIRST BYTE ?
```

TL/DD/10080–18

```
        JP      FIRST           ;YES -> JUMP
        IFBIT   0,[X].B         ;RECEIVEMODE ?
        JP      RECVE           ;YES -> JUMP
        JSR     GETACK          ;ACKNOWLEDGE
        IFC                     ;ERROR ?
        JP      STOPCD          ;STOP TRANSMISSION
TRMIT:  LD      A,[WBUF2].B     ;GET NEXT BYTE
        ST      A,SIO.B         ;ENABLE SHIFT
        JP      TINC            ;-> INCREMENT POINTERS
RECVE:  LD      A,SIO.B         ;GET DATA
        ST      A,[WBUF2].B     ;PUT INTO BUFFER
        JSR     SETACK          ;ACKNOWLEDGE
        IFC                     ;ERROR ?
        JP      STOPCD          ;STOP TRANSMISSION
        LD      SIO.B,#0FF      ;ENABLE SHIFT
TINC:   SBIT    5,BFUN.B        ;SELECT ALTERNATE MODE
TINC1:  INC     WBUF2.W         ;INC BUFFERPOINTER
TDEC:   DECSZ   WBUF1+1.B       ;DEC BYTECOUNT
        JP      T5IR02          ;MORE THAN 1 BYTE TO PROCESS
        SBIT    2,[X].B         ;STATUS = LAST BYTE
T5IR02: JP      IRQEND          ;EXIT AND WAIT FOR NEXT IRQ

LAST:   IFBIT   0,[X].B         ;RECEIVE ?
        JP      LASTRD          ;YES -> JUMP
LAST1:  JSR     GETACK          ;ACKNOWLEDGE
        IFBIT   3,[X].B         ;RESTART ?
        JP      RESTRT          ;YES -> JUMP
        JP      STOPCD          ;STOP TRANSMISSION

LASTRD: LD      A,SIO.B         ;GET LAST CHARACTER
        ST      A,[WBUF2].B     ;PUT INTO BUFFER
        JP      LAST1

FIRST:  LD      A,[WBUF2].B     ;GET NEXT BYTE
        JSR     GETACK          ;ACKNOWLEDGE
        IFC                     ;ERROR ?
        JP      STOPCD          ;STOP TRANSMISSION
        RBIT    1,[X].B         ;RESET FIRST BYTE FLAG
        IFBIT   0,[X].B         ;RECEIVE ?
        JP      IRRCV           ;YES -> JUMP
        ST      A,SIO.B         ;ENABLE SHIFT
        SBIT    5,BFUN.B        ;SELECT ALTERNATE FUNCTION
        JP      TINC1           ;-> INCREMENT POINTERS

IRRCV:  LD      SIO.B,#0FF      ;ACTIVATE SHIFT
        SBIT    5,BFUN.B        ;SELECT ALTERNATE FUNCTION
        JP      TDEC

IRQEND: POP     PSW.W
        RESTORE_ABX             ;RESTORE REGS X,B,A
IRQRET: RETI


RESTRT: LD      X,INDEX.W       ;GET NEXT POINTER TO ENTRYTABLE
        SBIT    5,[B].B         ;DATALINE = HIGH
```

TL/DD/10080–19

```
        RBIT    5,BFUN.B        ;DISABLE SO-OUTPUT
REST01: IFBIT   6,[B].B         ;WAIT UNTIL CLOCK = HIGH
        JP      REST02          ;CLOCK = HIGH -> REST02
        JP      REST01          ;WAIT
REST02: SBIT    2,PWMODE.B      ;STOP TIMER 4
        SBIT    6,PWMODE.B      ;STOP TIMER 5
        LD      T4.W,#2*CLK-1   ;LOAD TIMER 4
        LD      T5.W,#18*CLK-1  ;LOAD TIMER 5
        JSR     RWI2C           ;INITIALIZE READ/WRITE TO I2C-BUS
        JP      IRQEND

STOPCD: RBIT    5,[B].B         ;DATALINE = LOW
        RBIT    5,BFUN.B        ;DISABLE SO-OUTPUT
STOP01: IFBIT   6,[B].B         ;WAIT UNTIL CLOCK = HIGH
        JP      STOP02          ;CLOCK = HIGH -> STOP02
        JP      STOP01          ;WAIT
STOP02: SBIT    2,PWMODE.B      ;STOP TIMER 4
        SBIT    6,PWMODE.B      ;STOP TIMER 5
        LD      T4.W,#CLK-1     ;INITIALIZE T4 TO STARTCONDITION
        LD      T5.W,#17*CLK-1  ;INITIALIZE T5 TO STARTCONDITION
        RBIT    7,[X].B         ;STATUS = I2CBUS NOT BUSY
        SBIT    5,[B].B         ;PERFORM STOPCONDITION
        JP      IRQEND


SETACK: RBIT    5,[B].B         ;DATALINE = LOW
        RC
        JP      ACK01
GETACK: SBIT    5,[B].B         ;DATALINE = HIGH
        RC
ACK01:  IFBIT   6,[B].B         ;WAIT UNTIL CLOCK = HIGH
        JP      ACK02           ;CLOCK = HIGH
        JP      ACK01           ;CLOCK = LOW , WAIT
ACK02:  IFBIT   5,PORTI.B       ;TEST DATALINE
        SC                      ;FLAG EROR IF HIGH
ACK03:  IFBIT   6,[B].B         ;WAIT UNTIL CLOCK = LOW
        JP      ACK03
        SBIT    5,[B].B         ;DATALINE = HIGH
        RET

TSTBUS: RC
        IFBIT   5,PORTI.B       ;TEST DATALINE
        JP      TST1
        SC
TST1:   IFBIT   6,[B].B         ;TEST CLOCKLINE
        JP      TST2
        SC
TST2:   RET


RESET:  LD      ENIR.B,#0       ;DISABLE ALL INTERRUPTS
        LD      PWMODE.W,#0CCCC ;STOP AND CLEAR ALL TIMERS
        LD      TMMODE.W,#0CCCC
START:  JSR     INIT
        LD      B,#WRBUFF
```

TL/DD/10080–20

```
         LD       K,#WRBUFF+8      ;CLEAR 9 BYTES
START0:  CLR      A
         XS       A,[B+].W
         JP       START0
         LD       RDBUFF.B,#0      ;SET READADDRESS TO 0

         LD       WPORT.B,#0FF     ;INITIALISE PORT1 AS INPUT
         LD       X,#INIPO1
         JSR      RWI2C
         JSR      WAIT

         LD       WPORT.B,#0FF     ;PUT ALL LED'S OFF
START1:  LD       X,#WRPO0
         JSR      RWI2C
         JSR      WAIT

         LD       X,#WRRAM0        ;WRITE TO RAM
         JSR      RWI2C            ;START TRANSMISSION
         JSR      WAIT

         LD       X,#RDRAM0        ;READ RAM0
         JSR      RWI2C
         JSR      WAIT

         ADD      WRBUFF.B,#8      ;WRITE/READ NEXT 8 BYTES RAM
         ADD      RDBUFF.B,#8
         IFEQ     WRBUFF.B,#0      ;IF WRAP
         DECSZ    WPORT.B          ;DECREMENT LED VALUE
         NOP                       ;ONLY DECREMENT

         LD       X,#RDPO1         ;READ INPUT
         JSR      RWI2C
         JSR      WAIT

         IFBIT    7,RPORT          ;IF BIT SET FREE-RUN-LED
         JP       START1
         LD       WPORT.B,RPORT.B ;ELSE COPY INPUT TO OUTPUT

         JP       START1

WAIT:    PUSH     X                ;SAVE X-REG
         LD       X,#010           ;INITIALIZE WAITLOOP
         IFC
         INC      DUMMY.B
WAIT1:   IFBIT    7,STATUS.B       ;WAIT UNTIL READY
         JP       WAIT1
WAIT2:   DECSZ    X
         JP       WAIT2
         POP      X                ;RESTORE X-REG
         RET

INIPO1:  .DW      0242,WPORT,0              ;INITIALIZE PORT1 AS INPUT
RDPO1:   .DW      0243,RPORT,0              ;READ 1 BYTE FROM PORT1
WRPO0:   .DW      0240,WPORT,0              ;WRITE 1 BYTE TO PORT0
WRRAM0:  .DW      0AA0,WRBUFF,0             ;WRITE 8 BYTES TO RAM
```

TL/DD/10080–21

```
RDRAM0:  .DW      02A0,WRBUFF,0AA1,RDBUFF+1,0      ;READ 10 BYTES


         .IPT     5,TIMIRQ         ;SET TIMER IRQ ENTRY
         .END RESET
```

TL/DD/10080–22

```
;
;****** INCLUDE FILE HPC16083.MAP *******
;
;****** HPC-REGISTER DEFINITIONS ********

        PSW     = 0C0       ;PROCESSOR STATUS REGISTER
;       SP      = 0C4       ;STACK POINTER
;       PC      = 0C6       ;PROGRAM COUNTER
;       A       = 0C8       ;ACCUMULATOR
;       K       = 0CA       ;K REGISTER
;       B       = 0CC       ;B REGISTER
;       X       = 0CE       ;X REGISTER
        PORTA   = 0E0       ;PORTA DATA / OUTPUT BUFFER
        DIRA    = 0F0       ;PORTA DIRECTION / INPUT BUFFER
        PORTB   = 0E2       ;PORTB DATA REGISTER
        DIRB    = 0F2       ;PORTB DIRECTION REGISTER
        BFUN    = 0F4       ;PORTB ALTERNATE FUNCTION REG
        PORTI   = 0D8       ;PORTI DATA REGISTER
        PORTD   = 0104      ;PORTD DATA REGISTER
        PORTP   = 0152      ;PORTP REGISTER
        ENIR    = 0D0       ;INTERRUPT ENABLE REGISTER
        IRCD    = 0D4       ;INTERRUPT AND CAPTURE CONDITION REG
        IRPD    = 0D2       ;INTERRUPT PENDING REGISTER
        HLTEN   = 0DC       ;HALT ENABLE CONTROL CIRCUIT
        DIVBY   = 018E      ;DIVIDE BY REGISTER
        PWMODE  = 0150      ;PULSE WIDTH MODE REGISTER
        TMMODE  = 0190      ;TIMER MODE REGISTER
        I2CR    = 0184      ;I2 CAPTURE REGISTER / R1
        I3CR    = 0182      ;I3 CAPTURE REGISTER / T1
        I4CR    = 0180      ;I4 CAPTURE REGISTER
        EICR    = 015E      ;EI CAPTURE REGISTER
        EICON   = 015C      ;EI CONFIGURATION REGISTER
        T0CON   = 0192      ;T0 CAPTURE CONFIGURATION REG
        T2      = 0188      ;TIMER2
        R2      = 0186      ;TIMER2 MODULUS REGISTER
        T3      = 018C      ;TIMER3
        R3      = 018A      ;TIMER3 MODULUS REGISTER
        T4      = 0140      ;TIMER4
        R4      = 0142      ;TIMER4 MODULUS REGISTER
        T5      = 0144      ;TIMER5
        R5      = 0146      ;TIMER5 MODULUS REGISTER
        T6      = 0148      ;TIMER6
        R6      = 014A      ;TIMER6 MODULUS REGISTER
        T7      = 014C      ;TIMER7
        R7      = 014E      ;TIMER7 MODULUS REGISTER
        WD      = 0194      ;WATCHDOG REGISTER
        SIO     = 0D6       ;SERIAL INPUT OUTPUT SHIFT REG
        ENU     = 0120      ;UART CONTROL AND STATUS REGISTER
        ENUI    = 0122      ;UART INTERRUPT AND CLOCK SOURCE REG
        RBUF    = 0124      ;UART RECEIVE BUFFER
        TBUF    = 0126      ;UART TRANSMIT BUFFER
        ENUR    = 0128      ;UART RECEIVE CONTROL AND STATUS REG
        UPIC    = 0E6       ;UPI CONTROL REGISTER
```

TL/DD/10080–23

National Semiconductor Corporation
1111 West Bardin Road
Arlington, TX 76017
Tel: 1(800) 272-9959
Fax: 1(800) 737-7018

National Semiconductor Europe
Fax: (+49) 0-180-530 85 86
Email: cnjwge@tevm2.nsc.com
Deutsch Tel: (+49) 0-180-530 85 85
English Tel: (+49) 0-180-532 78 32
Français Tel: (+49) 0-180-532 93 58
Italiano Tel: (+49) 0-180-534 16 80

National Semiconductor Hong Kong Ltd.
13th Floor, Straight Block,
Ocean Centre, 5 Canton Rd.
Tsimshatsui, Kowloon
Hong Kong
Tel: (852) 2737-1600
Fax: (852) 2736-9960

National Semiconductor Japan Ltd.
Tel: 81-043-299-2309
Fax: 81-043-299-2408