Interfacing the LM12454/8 Data Acquisition System Chips to Microprocessors and Microcontrollers

National Semiconductor
Application Note 906
Farid Saleh
August 1993

TABLE OF CONTENTS

1.0 INTRODUCTION

The LM12454/8 family of data acquisition system (DAS) chips offers a fully differential self-calibrating 12-bit+ sign A/D converter with differential reference, 4 or 8 input analog multiplexer and extensive flexible and programmable logic. The logic embodies different units to perform specific tasks, for instance:

— An instruction RAM for stand-alone execution (after being programmed by the host) with programmable acquisition time, input selection, 8-bit or 12-bit conversion mode, etc.

— Limit registers for comparison of the inputs against high and low limits in "watchdog" mode.

— A 32-word FIFO register for storage of conversion results.

— Interrupt control logic with interrupt generation for 8 different conditions.

— A 16-bit timer register.

— Circuitry for synchronizing signal acquisition with external events.

— A parallel microprocessor interface with selectable 8-bit or 16-bit data access.

Because of its functionality and flexibility, working with the LM12454/8 family may appear to be an overwhelming task at first glance. However, this is not the case when the user gains a basic understanding of the device's functional units and the philosophy of its operation. This note shows how easy it is to use the LM12454/8 family and walks the user through the straightforward steps of the interfacing and programming of the device.

The LM12454/8 family has 6 members. The members and their differences are shown in Table I. For simplicity, the DAS abbreviation will be used throughout this Application Note as a generic name for any member of the family. Similarly, the drawings illustrate only the 8 input versions of the family. Note that this Application Note should be used in conjunction with the device data sheet and assumes the reader has some degree of familiarity with the device. However, a brief overview of the DAS and information related to the subjects being discussed are given here.

2.0 GENERAL OVERVIEW

2.1 The DAS Programming Model

Figure 1 illustrates the functional block diagram or user programming model of the DAS. (This diagram is not meant to reflect the actual implementation of the DAS internal building blocks.) The DAS model consists of the following blocks:

— A flexible analog multiplexer with differential output at the front end of the device.

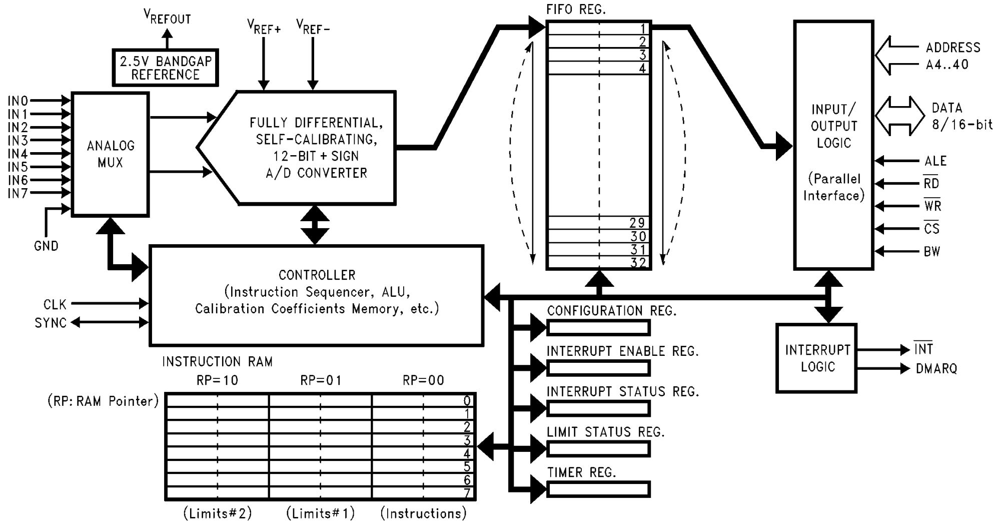
TABLE I: Members of the LM12454/8 Family

Device Number	Clock Frequency (Max, MHz)	Operating Supply Voltage (V)	Number of MUX Inputs	Internal Reference	Low Voltage Flag
LM12454	5	5.0 ±10%	4	Yes	Yes
LM12458	5	5.0 ±10%	8	Yes	Yes
LM12H454	8	5.0 ±10%	4	Yes	Yes
LM12H458	8	5.0 ±10%	8	Yes	Yes
LM12L454	6	3.3 ±10%	4	No	No
LM12L458	6	3.3 ±10%	8	No	No

HPC™ is a trademark of National Semiconductor Corporation.

Interfacing the LM12454/8 Data Acquisition System Chips to Microprocessors and Microcontrollers

AN-906



TL/H/11908–1

FIGURE 1. DAS Functional Block Diagram, Programming Model

— A fully-differential, self-calibrating 12-bit+ sign A/D converter.

— A 32-word FIFO register as the output data buffer.

— An instruction RAM that can be programmed to repeatedly perform a series of conversions and comparisons on the selected input channels.

— A series of registers for overall control and configuration of the DAS operation and indication of internal operational status.

— Interrupt generation logic to request service from the processor under specified conditions.

— Parallel interface logic for input/output operations between the DAS and the processor. All the registers shown in the diagram can be read and most of them can also be written to by the user through the input/output block.

— A controller unit that controls the interactions of the different blocks inside the DAS and performs the conversion, comparison and calibration sequences.

The DAS has 3 different modes of operation: 12-bit+ sign conversion, 8-bit+ sign conversion and 8-bit+ sign comparison, (also called "watchdog" mode). In the watchdog mode no conversion is performed, but the DAS samples an input and compares it with the values of the two limits stored in the Instruction RAM. If the input voltage is above or below the limits (as defined by the user) an interrupt can be generated to indicate a fault condition.

The INSTRUCTION RAM is divided into 8 separate words, each with 48 (3x16) bit length. Each word is separated into three 16-bit sections. Each word has a unique address and different sections of the instruction are selected by the 2-bit RAM pointer (RP) in the configuration register. As shown in *Figure 1*, the Instruction RAM sections are labeled Instructions, Limits #1 and Limits #2. The Instruction section holds operational information such as; the input channels to be selected, the mode of operation for each instruction, and how long the acquisition time should be. The other two sections are used in the watchdog mode and the user defined limits are stored in them. Each watchdog instruction has 2 limits associated with it (usually the low and high limits, but two low or two high limits may be programmed instead). The DAS can start executing from Instruction 0 and continue executing the next instructions up to any user specified instruction and, then "loops back" to Instruction 0. This means that not all 8 instructions need to be executed in the loop. The cycle may be repeatedly executed until stopped by the user. The user should access the Instruction RAM only when the instruction sequencer is stopped.

The FIFO Register is used to store the results of the conversion. This register is "read only" and all the locations are accessed through a single address. Each time a conversion is performed the result is stored in the FIFO and the FIFO's internal write pointer points to the next location. The pointer rolls back to location 1 after a write to location 32. The same flow occurs when reading from the FIFO. The internal FIFO writes and the external FIFO reads do not affect each other's pointer locations.

The CONFIGURATION Register is the main "control panel" of the DAS. Writing 1s and 0s to the different bits of the Configuration Register commands the DAS to perform different actions such as start or stop the sequencer, reset the pointers and flags, enter standby mode for low power consumption, calibrate offset and linearity, and select sections of the RAM.

The INTERRUPT ENABLE Register lets the user activate up to 8 sources for interrupt generation. It also holds two user programmable values. One is the number of conversions to be stored in the FIFO register before the generation of the data ready interrupt. The other value is the instruction number that generates an interrupt when the sequencer reaches that instruction.

The INTERRUPT STATUS and LIMIT STATUS Registers are "read only" registers. They are used as vectors to indicate which conditions have generated the interrupt and what limit boundaries have been passed. Note that the bits are set in the status registers upon occurrence of their corresponding interrupt conditions, regardless of whether the condition is enabled for external interrupt generation.

The TIMER Register can be programmed to insert a delay before execution of each instruction. A bit in the Instruction register enables or disables the insertion of the delay before the execution of an instruction.

Appendix A shows all the DAS accessible registers and a brief description of their bits assignments. These bit assignments are discussed in detail in the data sheet and are repeated here for reference. There are also empty register models available on the same pages that can be used as a programming tool. The designer can fill these register models with ‘‘1s’’ and ‘‘0s’’ during design based on system requirements. The user can also use these sheets for design documentation.

2.2 Programming Procedure

The DAS is designed for control by a processor. However, the functionality of the DAS off loads the processor to a great extent, resulting in reduction of the software overhead. At the start, the processor downloads a set of operational instructions to the DAS' RAM and registers and then gives a start command to the DAS. The DAS performs continuous conversions and/or comparisons as dictated by the instructions and loads the conversion results in the FIFO. From this point the processor has two basic options for interaction with the DAS. The DAS can generate an interrupt to the processor when the predetermined number of conversion results are stored in the FIFO or when any other interrupt conditions have occurred. The processor will then service the interrupt by reading the FIFO or taking corrective action, depending on the nature of the interrupt. Alternatively, rather than responding to an interrupt, the processor at any time can read the data or give a new command to the DAS.

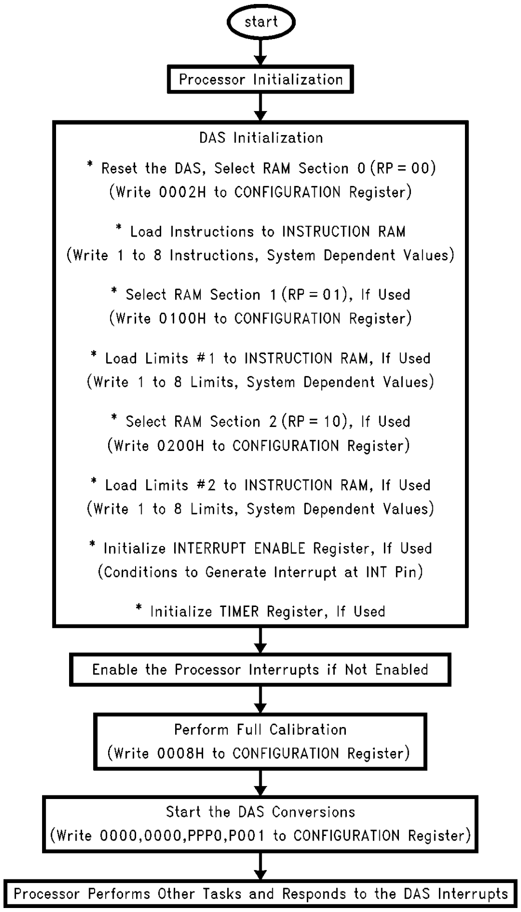
Defining a general programming procedure is not practical due to the extreme flexibility of the DAS and the variety of the applications. However, the following typical procedure demonstrates the basic concepts of the DAS start-up routine:

- — Reset the DAS by setting the RESET bit and select RAM section ‘‘00’’ through the Configuration register.
- — Load instructions to the Instruction RAM (1 to 8 instructions).
- — Select RAM section ‘‘01’’ (if used) through the Configuration register.
- — Load limits #1, 1 to 8 values (if used).
- — Select the RAM section ‘‘10’’ (if used) through the Configuration register.
- — Load limits #2, 1 to 8 values (if used).
- — Initialize the Interrupt Enable register, by selecting the conditions to generate an interrupt at the $\overline{\text{INT}}$ pin (if used).
- — Program the Timer register for required delay (if used).
- — Start the sequencer operation by setting the START bit in the Configuration register. Set the other bits in the Configuration register as required at the same time.

After the DAS starts operating, the processor may respond to interrupts from the DAS or it may interrogate the DAS at any time.

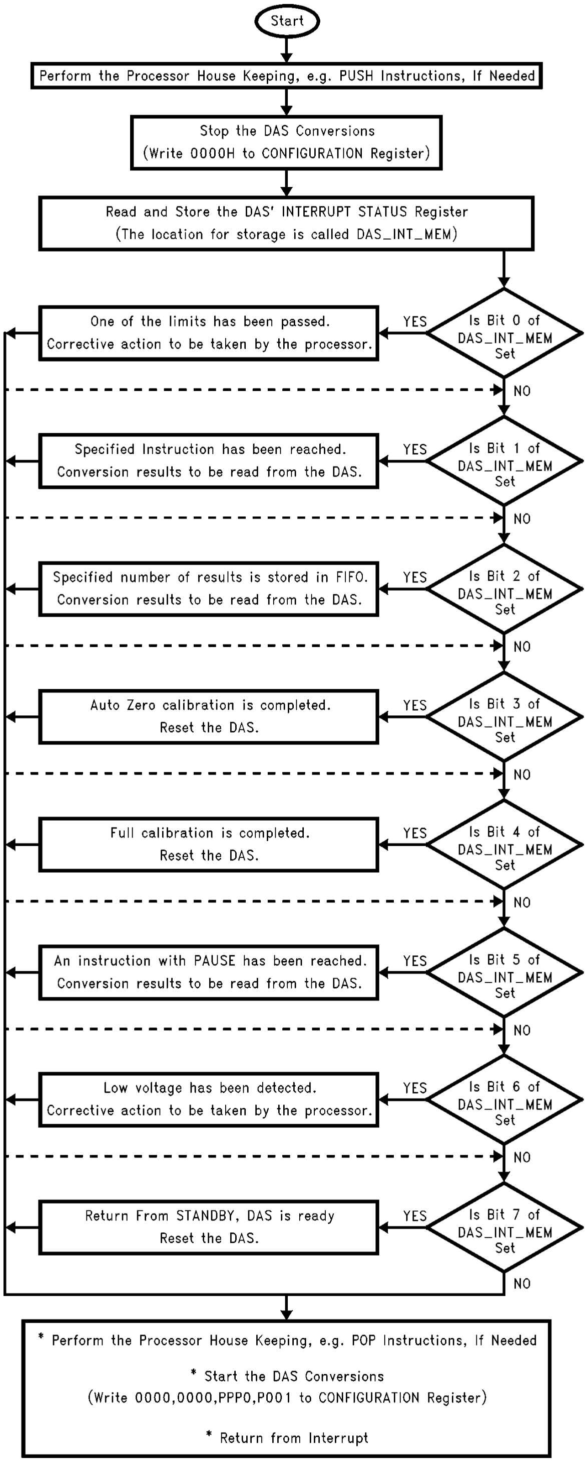
2.3 A Typical Program Flowchart and Alternative Approaches

A typical DAS program flowchart is shown in *Figure 2*. *Figure 2a* shows the initialization of the DAS and the start of the conversions. *Figure 2b* shows the general form of the DAS interrupt service routine. It is assumed that the DAS interacts with the processor through an interrupt line. This means the host processor generally is busy with other tasks and responds to the DAS through its interrupt service routine.


start
Processor Initialization
DAS Initialization
* Reset the DAS, Select RAM Section 0 (RP = 00)
(Write 0002H to CONFIGURATION Register)
* Load Instructions to INSTRUCTION RAM
(Write 1 to 8 Instructions, System Dependent Values)
* Select RAM Section 1 (RP = 01), If Used
(Write 0100H to CONFIGURATION Register)
* Load Limits #1 to INSTRUCTION RAM, If Used
(Write 1 to 8 Limits, System Dependent Values)
* Select RAM Section 2 (RP = 10), If Used
(Write 0200H to CONFIGURATION Register)
* Load Limits #2 to INSTRUCTION RAM, If Used
(Write 1 to 8 Limits, System Dependent Values)
* Initialize INTERRUPT ENABLE Register, If Used
(Conditions to Generate Interrupt at INT Pin)
* Initialize TIMER Register, If Used
Enable the Processor Interrupts if Not Enabled
Perform Full Calibration
(Write 0008H to CONFIGURATION Register)
Start the DAS Conversions
(Write 0000,0000,PPP0,P001 to CONFIGURATION Register)
Processor Performs Other Tasks and Responds to the DAS Interrupts

TL/H/11908–2

FIGURE 2a. A Typical Program Flowchart for the DAS Initialization and Start of Conversions


Start
Perform the Processor House Keeping, e.g. PUSH Instructions, If Needed
Stop the DAS Conversions
(Write 0000H to CONFIGURATION Register)
Read and Store the DAS' INTERRUPT STATUS Register
(The location for storage is called DAS_INT_MEM)
Is Bit 0 of DAS_INT_MEM Set
YES
One of the limits has been passed.
Corrective action to be taken by the processor.
NO
Is Bit 1 of DAS_INT_MEM Set
YES
Specified Instruction has been reached.
Conversion results to be read from the DAS.
NO
Is Bit 2 of DAS_INT_MEM Set
YES
Specified number of results is stored in FIFO.
Conversion results to be read from the DAS.
NO
Is Bit 3 of DAS_INT_MEM Set
YES
Auto Zero calibration is completed.
Reset the DAS.
NO
Is Bit 4 of DAS_INT_MEM Set
YES
Full calibration is completed.
Reset the DAS.
NO
Is Bit 5 of DAS_INT_MEM Set
YES
An instruction with PAUSE has been reached.
Conversion results to be read from the DAS.
NO
Is Bit 6 of DAS_INT_MEM Set
YES
Low voltage has been detected.
Corrective action to be taken by the processor.
NO
Is Bit 7 of DAS_INT_MEM Set
YES
Return From STANDBY, DAS is ready
Reset the DAS.
NO
* Perform the Processor House Keeping, e.g. POP Instructions, If Needed
* Start the DAS Conversions
(Write 0000,0000,PPP0,P001 to CONFIGURATION Register)
* Return from Interrupt

TL/H/11908–3

FIGURE 2b. A Typical Program Flowchart for the DAS Interrupt Service Routine

There is a processor initialization step at the start of the flowchart. It is included as a reminder that some specific processor initialization may be needed just for interaction with the DAS.

The DAS initialization steps are a series of write operations to the DAS registers. These steps are the same as mentioned in Section 2.2.

A full calibration cycle is usually performed after setting the DAS' registers. This is required for 12-bit accuracy. You may choose to perform one full calibration at power up, or periodic calibrations at specified time intervals, or condition-based calibrations, e.g., calibrations after a specified change in temperature. Calibration is done by writing the appropriate control code to the Configuration register. A full calibration cycle takes about 1 ms (989 µs) with a 5 MHz clock and about 0.6 ms (618 µs) with an 8 MHz clock. You can insert a delay after starting a calibration cycle, or can detect the end of calibration by an interrupt, or by reading the Interrupt Status register for the corresponding flag bit. In the flowchart, the end-of-calibration detection is handled in the interrupt service routine. The full calibration cycle affects some of the DAS' internal flags and pointers that will influence the execution of the first instruction after calibration. To avoid false instruction execution, the DAS should be reset after a calibration cycle. This is shown on the flowchart for the interrupt service routine.

After a calibration, the DAS is ready to start conversions. Conversion is initiated by writing to the configuration register and setting the START bit to "1". The data to be written to the configuration register is shown in binary format in the flowchart. The bits shown by "P" (program) are the control bits that determine different modes of operation during conversions. All the other bits should be programmed as shown. As mentioned before, the host processor can perform data manipulation and other control tasks after starting the DAS, and will respond to the DAS interrupts as required.

At the start of the interrupt service routine (*Figure 2b*), a zero is written to START bit in the Configuration register, to stop the conversion. Stopping the conversion is not necessarily needed unless it is required for accuracy or timing purposes. Generally the results of conversions will be noisier and less accurate if reads or writes to and from the DAS are performed while it is converting. However, the degree of this inaccuracy depends on many aspects of system design and is not easy to quantify. Power supply and ground routing, supply bypassing, speed of logic transitions on the bus, the logic family being used, and the loading (resistive and capacitive) on the data bus being driven by the DAS, all can affect conversion noise. Nevertheless, reading during conversions has been shown not to cause serious accuracy problems in most systems.

There are two timing issues regarding the reading during conversion.

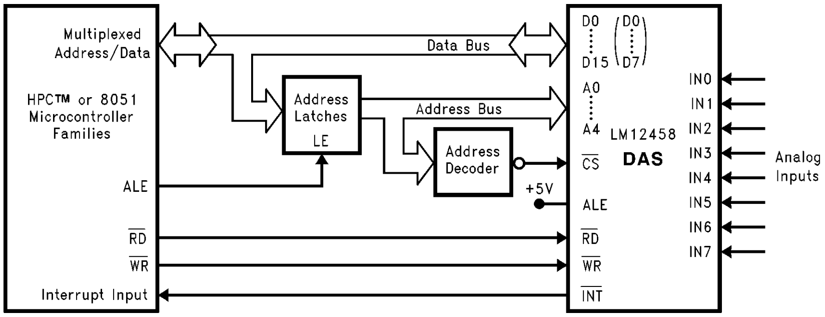
During any read or write from or to the DAS, the DAS internal clock will stop while the $\overline{CS}$ is low. This is done for synchronization between external and internal bus activities, thus preventing internal conflicts. Note that reads and writes are asynchronous to internal bus activities. A pause of internal clock cycles will increase the total acquisition plus conversion time for each instruction. The amount of this time increase is variable and is not easily predictable, because the processor and the DAS work asynchronously. As a result, the user should not perform reads during conversions if the fixed time intervals between the signal acquisitions are critical in the system performance.

The second timing issue depends on the speed of the conversions and the speed of the read cycles from the FIFO. The rule is to read the FIFO fast enough that old data will not be overwritten with new data during continuous conversions.

Returning to the flowchart, the main task of the interrupt service routine is to read the DAS' Interrupt Status register and test its bits for the source of the interrupt. The interrupt service routine shows all the interrupt bits in the DAS being tested. However, real systems often use only a few number of the interrupts, so the extra bit tests should be eliminated from the routine. Also, the sequence in which the bits are tested depends on the priority level of the interrupts in the system. The tasks to be performed for each interrupt are mainly system related and are not elaborated upon in the flowchart. If conversions are stopped at the start of the interrupt service, one possibility is to restart conversion before returning from the interrupt service routine. Otherwise conversions will be restarted again at some other point in the system routines.

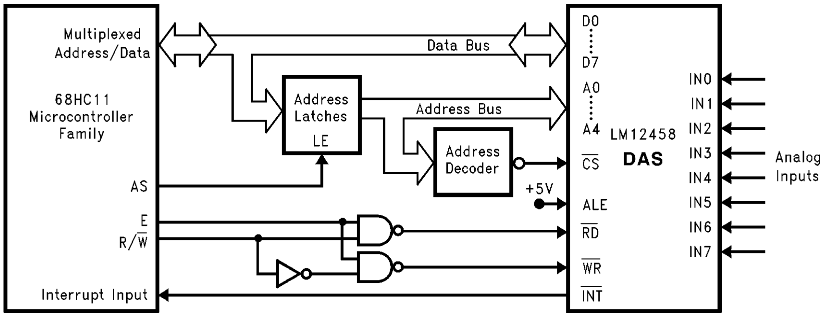
2.4 The DAS/Processor Interface

The interface between the processor and the DAS is similar to a memory or I/O interface. Some possible DAS/microcontroller interface schemes are shown in *Figures 3*, *4* and *5*.



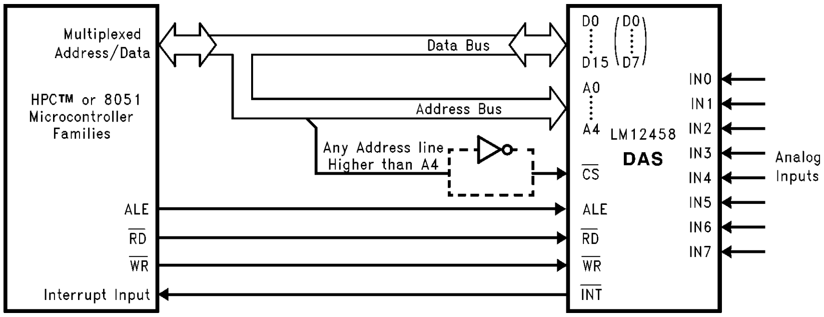
TL/H/11908–4

FIGURE 3. LM12458 to HPC or 8051 Microcontroller Interface



TL/H/11908–5

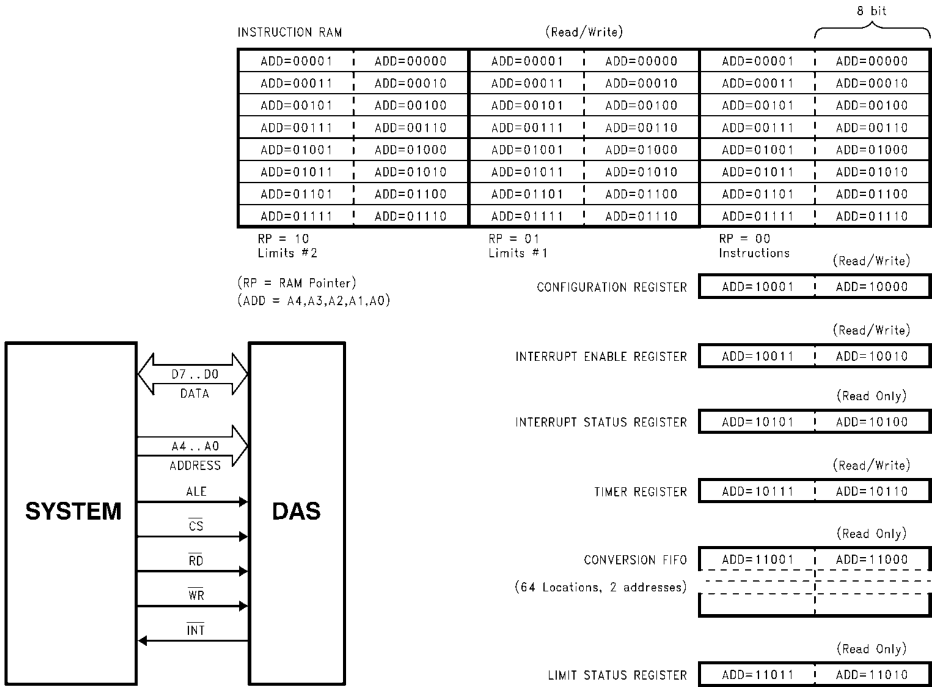
FIGURE 4. LM12458 to 68HCII Microcontroller Interface



TL/H/11908–6

FIGURE 5. LM12458 to HPC or 8051 Microcontroller Interface (Minimum System)

From the processor's point of view, the DAS is a group of I/O registers with specific addresses. *Figure 6* illustrates the DAS registers with their address assignments and the DAS interface buses and control signals. The DAS provides standard architecture for address, data and control buses for parallel interface to processors. The DAS can be interfaced to both multiplexed and non-multiplexed address/data bus architectures. An ALE input and internal latches allow the DAS to interface to a multiplexed address/data bus when external address latches are not required by the system. The DAS can be accessed in either 8-bit or 16-bit data width. BW (Bus Width) input pin selects the 8-bit or 16-bit access. In 8-bit access mode, each 16-bit I/O register is accessed in 2 cycles. Address line A0 selects the lower or upper portions of a 16-bit register. In 16-bit access mode, address line A0 is a ''don't care''. As shown in the *Figures 6a* and *6b*, the DAS appears to the processor as 14 separate 16-bit or 28 separate 8-bit I/O locations.

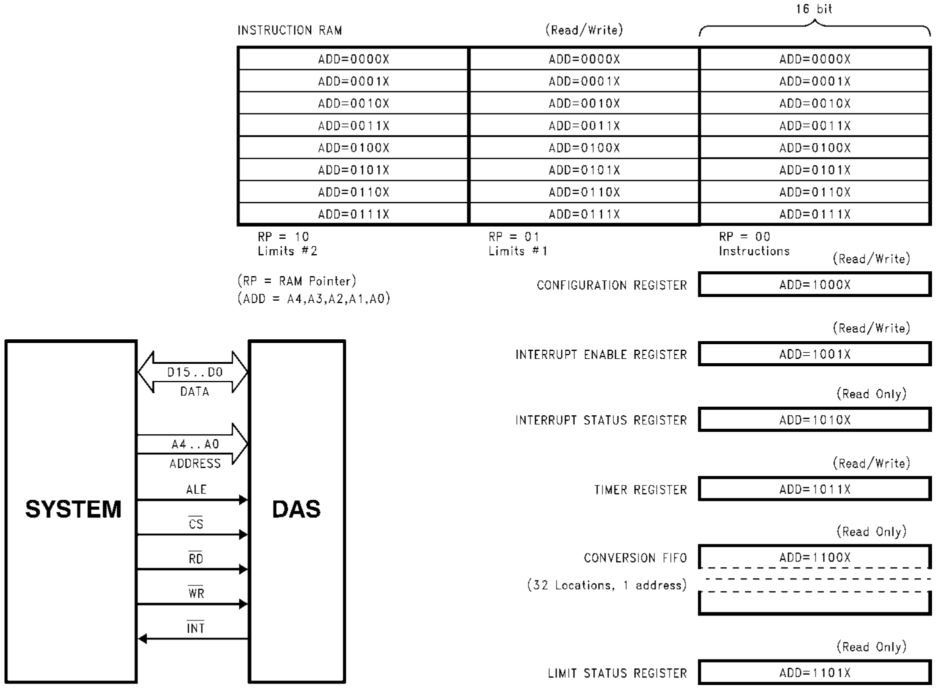


ADD=00001	ADD=00000	ADD=00001	ADD=00000	ADD=00001	ADD=00000
ADD=00011	ADD=00010	ADD=00011	ADD=00010	ADD=00011	ADD=00010
ADD=00101	ADD=00100	ADD=00101	ADD=00100	ADD=00101	ADD=00100
ADD=00111	ADD=00110	ADD=00111	ADD=00110	ADD=00111	ADD=00110
ADD=01001	ADD=01000	ADD=01001	ADD=01000	ADD=01001	ADD=01000
ADD=01011	ADD=01010	ADD=01011	ADD=01010	ADD=01011	ADD=01010
ADD=01101	ADD=01100	ADD=01101	ADD=01100	ADD=01101	ADD=01100
ADD=01111	ADD=01110	ADD=01111	ADD=01110	ADD=01111	ADD=01110

Register	Access		
CONFIGURATION REGISTER	(Read/Write)	ADD=10001	ADD=10000
INTERRUPT ENABLE REGISTER	(Read/Write)	ADD=10011	ADD=10010
INTERRUPT STATUS REGISTER	(Read Only)	ADD=10101	ADD=10100
TIMER REGISTER	(Read/Write)	ADD=10111	ADD=10110
CONVERSION FIFO (64 Locations, 2 addresses)	(Read Only)	ADD=11001	ADD=11000
LIMIT STATUS REGISTER	(Read Only)	ADD=11011	ADD=11010

TL/H/11908–7

FIGURE 6a. DAS Registers, Address Assignments, Interface Buses and Control Signals for 8-Bit Bus Width



ADD=0000X	ADD=0000X	ADD=0000X
ADD=0001X	ADD=0001X	ADD=0001X
ADD=0010X	ADD=0010X	ADD=0010X
ADD=0011X	ADD=0011X	ADD=0011X
ADD=0100X	ADD=0100X	ADD=0100X
ADD=0101X	ADD=0101X	ADD=0101X
ADD=0110X	ADD=0110X	ADD=0110X
ADD=0111X	ADD=0111X	ADD=0111X

Register	Access	
CONFIGURATION REGISTER	(Read/Write)	ADD=1000X
INTERRUPT ENABLE REGISTER	(Read/Write)	ADD=1001X
INTERRUPT STATUS REGISTER	(Read Only)	ADD=1010X
TIMER REGISTER	(Read/Write)	ADD=1011X
CONVERSION FIFO (32 Locations, 1 address)	(Read Only)	ADD=1100X
LIMIT STATUS REGISTER	(Read Only)	ADD=1101X

TL/H/11908–8

FIGURE 6b. DAS Registers, Address Assignments, Interface Buses and Control Signals for 16-Bit Bus Width

The interface should provide the address, data and control signals to the DAS with the following requirements:

— An address decoder is needed to generate a chip-select for the DAS within the required address range.

— The switching relationship between ALE, $\overline{CS}$, $\overline{RD}$, $\overline{WR}$, address bus and data bus should satisfy the DAS timing requirements. (Please refer to the data sheet for timing requirements.)

— When the DAS is working in an interrupt-driven I/O environment, a suitable service request link between the DAS and the system should be provided. This can be as simple as connecting the DAS' $\overline{INT}$ output to a processor's interrupt input or as sophisticated as using interrupt arbitration logic (interrupt controller) in systems that have many I/O devices.

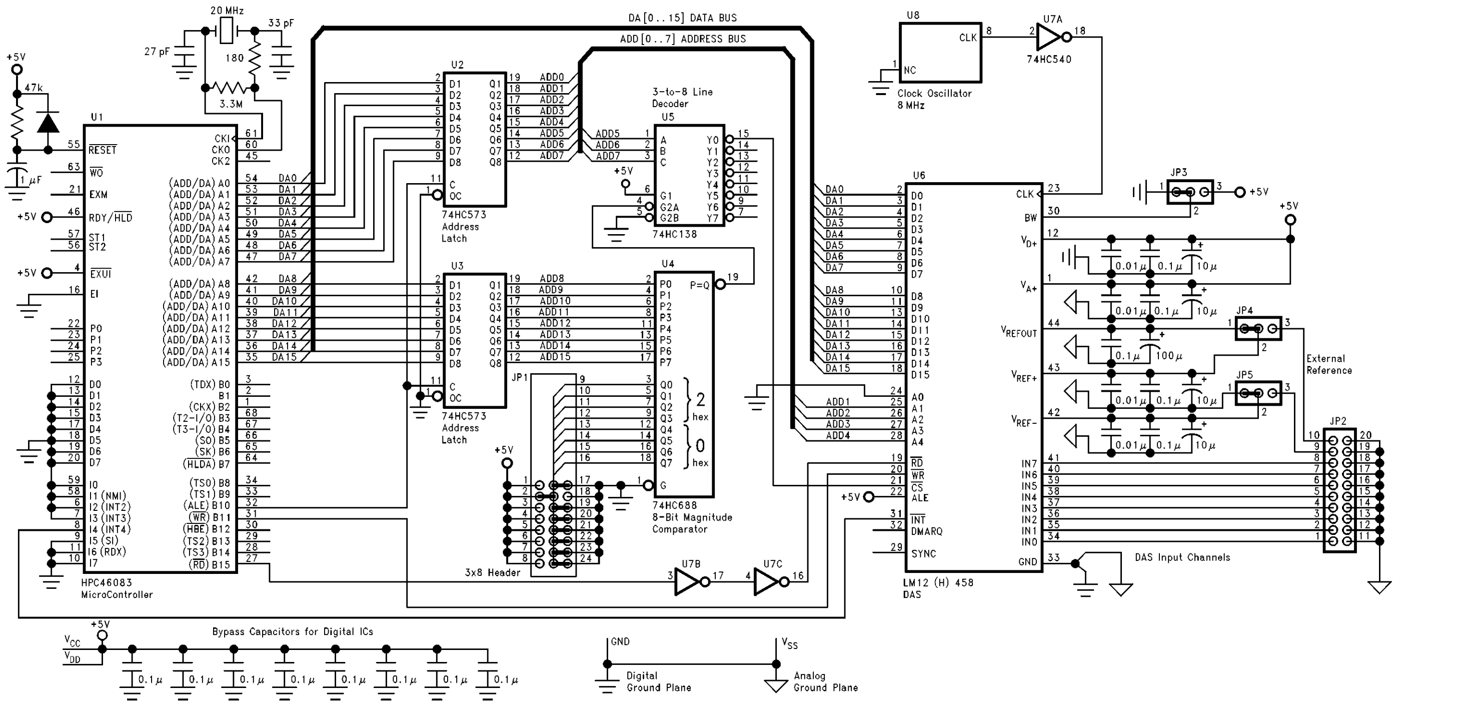
Figure 3 illustrates the generic interface for the National Semiconductor's HPC family of 16-bit microcontrollers and the 8051 family of 8-bit microcontrollers. *Figure 4* illustrates the interface for the 68HC11 family of microcontrollers or the processors with similar control bus architecture. The circuits in *Figures 3* and *4* are the maximum system schemes assuming the microcontroller is accessing other peripherals in addition to the DAS, therefore external address latches and an address decoder are required to select the DAS as well as the other peripherals. The size and complexity of the address decoder, however, depends on the system. In a minimum system scheme, the DAS can be interfaced to the microcontroller with minimal external logic for address latches and address decoder. This is shown in *Figure 5*. Note that the DAS' $\overline{CS}$ signal is also latched with the ALE inside the DAS, so a higher order address bit can be used to drive $\overline{CS}$ input. In this scheme a wide range of addresses (with many bits as ''don't cares'') are used to access the DAS. Care must be taken not to use this address range for any other memory or I/O locations. For example, lets assume bit A15 is used for $\overline{CS}$, and must be 1 (inverter in place) to select the DAS. As a result, all the 32k of the upper address range is used for the DAS. However, address bits A5 to A14 are ''don't cares'' and the DAS can be mapped anywhere within the upper 32k of the address range.

3.0 INTERFACING THE DAS TO HPC MICROCONTROLLERS

In this section we are going to develop a detailed interface circuit between the HPC46083 microcontroller and the DAS. The HPC46083 is a member of the HPC family of high performance 16-bit microcontrollers. The HPC family is available in a variety of versions suitable for specific applications. The reader is encouraged to refer to HPC family data sheets for complete information, available versions, and their specifications. The HPC46083, a 16-bit microcontroller with 16-bit multiplexed data and address lines, is one of the simplest members of the family. It is a complete microcontroller containing all the necessary system timing, internal logic, ROM, RAM, and I/O, and is optimized for implementing dedicated control functions in a variety of applications. Its architecture recognizes a single 64k byte of address space containing all the memory, registers and I/O addresses (memory-mapped I/O). The addressing space of the first 512 bytes (0000H to 01 FFH) contains 256 bytes of on-chip user RAM and internal registers. (The address values are given in hexadecimal format with suffix ''H'' as an indicator.) The last 8 kbytes of the address space (E000H to FFFFH) are on-chip ROM used mainly for program storage. In the following applications, the HPC46083 is setup for the expanded mode (as opposed to the single-chip mode) of operation that allows the external address range (0200H to DFFFH) to be accessed. The external data bus in the HPC family is configurable as 8-bit or 16-bit, allowing it to efficiently interface with a variety of peripheral devices.

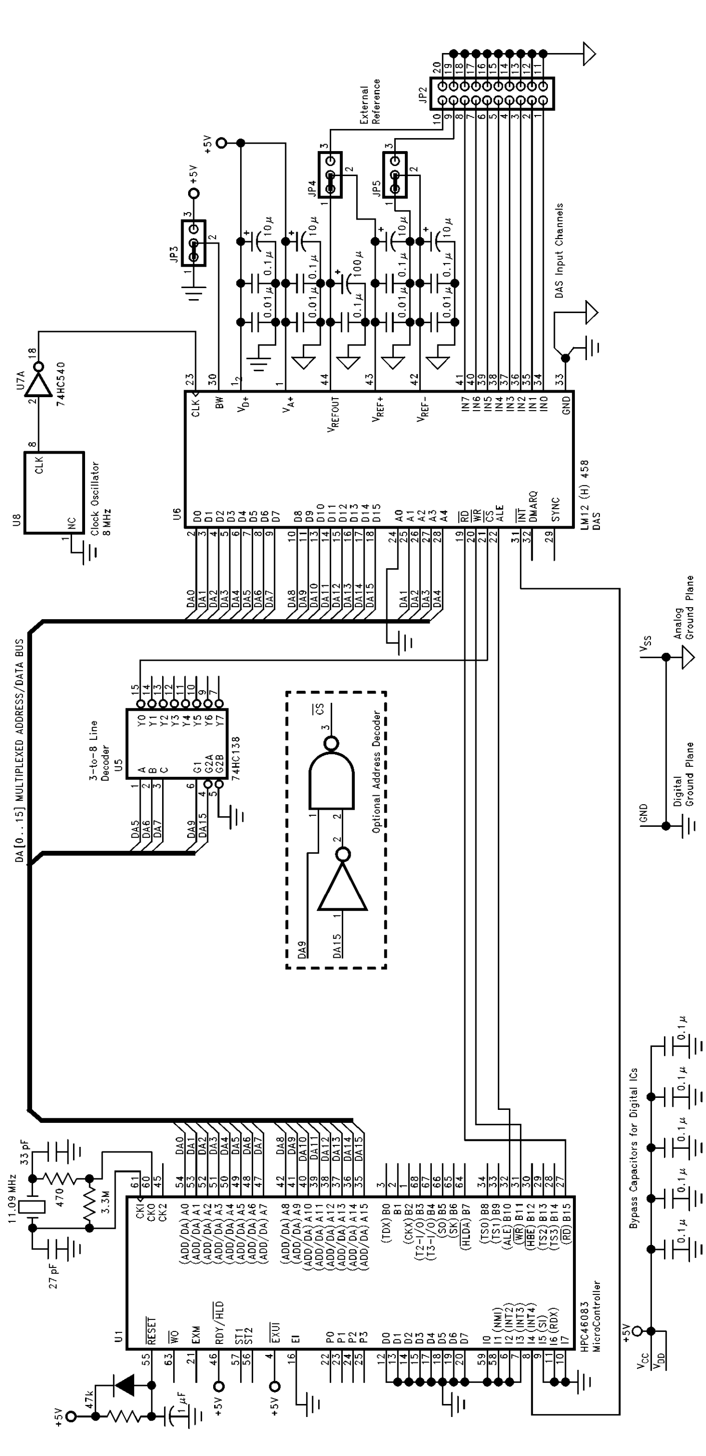
Interrupt handling is accomplished by the HPC46083's vector interrupt scheme. There are eight possible interrupt sources for the HPC46083. Four of these are maskable external interrupt inputs. These inputs can be programmed for different schemes, e.g., interrupt at low level, high level, rising edge or falling edge. One of these interrupts is used for interface with the DAS. The term ''HPC'' will be used throughout the remainder of this discussion to refer to the HPC46083.

Two different interface circuits are presented in *Figures 7* and *8*. The first circuit in *Figure 7* uses complete address decoding with the external address latches. This scheme assumes the HPC is accessing other devices using other address ranges. The circuit in *Figure 8* assumes the DAS is the only (or one of a few) peripherals interfaced to the HPC, so incomplete address decoding is used for minimum interface logic. Note that the address decoding schemes used in these circuits are only two of many different possibilities and are presented as generic forms of address decoding. These circuits are used as vehicles to illustrate the issues regarding the interface, and different schemes with other logic families or PAL devices can also be used for interface circuits.



TL/H/11908–9

FIGURE 7. The DAS/HPC Microcontroller Interface (Complete Address Decoding)



TL/H/11908–10

FIGURE 8. The DAS/HPC Microcontroller Interface (Minimal Address Decoding)

3.1 Complete Address Decoding

Figure 7 shows the circuit with complete address decoding to generate the DAS' $\overline{CS}$ signal. The DAS is accessed as memory mapped I/O at the start of the external address range (0200H to 021BH), and 16-bit data access is selected for the DAS. External address latches, U2 and U3, (74HC573) are used for the HPC's multiplexed 16-bit data/address lines. As a result, the ALE input of the DAS is tied high. An 8-bit magnitude comparator, U4, (74HC688) decodes the high order address byte [A15...A8] by comparing it with the logic input from the address range selector jumpers on header JP1. The jumper setting shown in *Figure 7* is 02H. The output of the magnitude comparator enables 3-to-8 line decoder chip, U5, (74HC138). The 3-to-8 line decoder inputs are address lines A5 to A7. The output Y0 of the 3-to-8 line decoder is activated for the 32 locations of address space from 0200H to 021FH. This output (Y0) is used for the DAS' $\overline{CS}$ input. Address lines A1 to A4 are directly connected to the DAS address inputs. The A0 input of the DAS is tied to ground since 16-bit data access is used and A0 is a ''don't care''. The DAS uses 28 bytes of address locations with addresses from 0200H to 021BH.

The DAS signal timing requires that its $\overline{CS}$ be active at least 20 ns before $\overline{RD}$ or $\overline{WR}$. This requirement cannot be met with the HPC running at 20 MHz clock frequency. In order to compensate for the propagation delays in the address latches and decoder, 2 inverting buffers (U7), are placed in the $\overline{RD}$ control line. The need for these inverters will be discussed further in the following Timing Analysis section. The $\overline{WR}$ line does not require this delay compensation.

The DAS' $\overline{INT}$ output drives the INT4 input of the HPC. This allows the DAS to request service when acquired data is ready or for any other condition for which processor attention is needed. The selection of INT4 is arbitrary, and any other external interrupt input could have been used. In a real system, selection of a processor interrupt input will be based on the number of interrupt driven I/O devices and the priority for each device.

The DAS' clock is driven with an 8 MHz crystal clock module. The output of the clock module is separately buffered for the DAS. This keeps the DAS' clock clean and minimizes interference that might be generated and induced by other devices using the same clock line.

3.2 Minimal Address Decoding

The circuit in *Figure 8* does not use the external address latches (U2, U3), the 8-bit magnitude comparator (U4), and the address setting jumpers (JP1). The 3-to-8 line decoder (U5) is still used and is enabled by the address/data lines DA9 and DA15. DA9 should be ''1'' to enable U5. This is selected to prevent address conflict between the DAS and the HPC internal RAM and registers, which use the address range 0000H to 01FFH with DA9 equal to ''0''. DA15 should be ''0'' to enable U5. This is selected to prevent conflict between the DAS and the HPC internal ROM, which uses the address range E000H to FFFFH with DA15 equal to ''1''. The Y0 output of U5 is still driving the DAS' $\overline{CS}$ input. The ALE output of the HPC directly drives the DAS' ALE input. The ALE latches the address and $\overline{CS}$ lines on the DAS' internal latches at the start of any data transfer cycle with the DAS.

The DAS can still be accessed with the same address range in the *Figure 7* circuit, which is 0200H to 021BH. However, many address bits are ''don't care'' in this case. The binary form of the DAS register addresses for the circuit in *Figure 8* is: 0XXX,XX1X,000P,PPP0. The P's indicate program bits, these will be programmed to select different registers. The X's are ''don't care'' bits. The rest of the bits should be programmed as shown.

The 3-to-8 line decoder (U5) outputs, Y1 to Y7, can still be used to access other peripherals, those peripherals should have internal latches for the address and chip-select as well. For example, up to eight DAS chips can be interfaced to an HPC using the circuit in *Figure 8*, for monitoring and data logging of 64 analog input channels. If the interface is for one DAS only, the DAS' $\overline{CS}$ input can be generated with minimum of 2 gates, as shown within the dashed-lines on *Figure 8*, replacing the U5.

The critical DAS timing requirement for the circuit in *Figure 8* is the address and chip-select setup time to ALE going low. The HPC running at 20 MHz clock frequency cannot satisfy this timing specification. As a result, the clock speed of the HPC is lowered to 11.09 MHz to meet the DAS requirement. This point is also discussed further in the Timing Analysis section.

3.3 Timing Analysis

The user should perform a timing analysis along with the interface hardware design to ensure proper transaction of information between the processor and the DAS. For example the buffers in the $\overline{RD}$ input of the DAS in *Figure 7* are necessary to ensure proper timing. Similarly, the clock frequency of the HPC in *Figure 8* was reduced to ensure proper timing. For every new circuit design the DAS timing specifications for read and write cycles should be compared with the HPC (or the processor being used) timing specifications. Any mismatch between the timing characteristics must be compensated by hardware design changes or software techniques.

3.3.1 Complete Address Decoding Circuit

Study of the switching characteristics of the DAS and the HPC (running at 20 MHz) shows that the read cycle timing is more stringent than that of the write cycle. The timing diagram (*Figure 9*) shows the timing relationship for the signals involved in a read cycle. The first three signals are generated by the HPC (ALE, ADD/DATA, $\overline{RD}$), and are shown with their minimum timing relationships. The three other signals are $\overline{CS}$ and $\overline{RD}$ received by the DAS, and DATA(DAS) which is sent to the HPC. The $\overline{CS}$' longest propagation delay through the address latch (U3), magnitude comparator (U4), and 3-to-8 line decoder (U5) starts from the moment an address becomes valid. This propagation delay is, typically, 54 ns up to worst case of 116 ns.

Note: Maximum propagation delays for 74HC series logic devices are for 4.5V supply, 50 pF load and −40°C to 85°C temperature range. Typical values are for the same conditions at 25°C.

This delay, referred to the falling edge of the $\overline{RD}$(HPC), is 16 ns to 78 ns. The DAS requires that its $\overline{RD}$ becomes active 20 ns after its $\overline{CS}$. This requirement compels insertion of delay on the HPC's $\overline{RD}$ line before it is received by the DAS. Referenced to the HPC's $\overline{RD}$ signal falling edge the DAS' should receive a $\overline{RD}$ signal that is delayed by 36 ns to 98 ns (16 + 20 ns to 78 + 20 ns). Inverting buffers U7B and U7C

(75HC540) provide $\overline{\text{RD}}$ signal delay between the HPC and the DAS. The two buffers' propagation delay specification is typically 24 ns and the maximum is 50 ns. This is less than the 36 ns to 98 ns requirement drawn from the analysis. However, practical measurements have shown that this delay is sufficient within a temperature range of 0°C to 50°C. This discrepancy results from the fact that the operating conditions on the specification sheets (loading capacitance, supply voltage) for the logic devices are more severe than the ones in the practical circuit. However, if the circuit is to perform reliably in worst case conditions, extra delay may be inserted in the $\overline{\text{RD}}$ line.

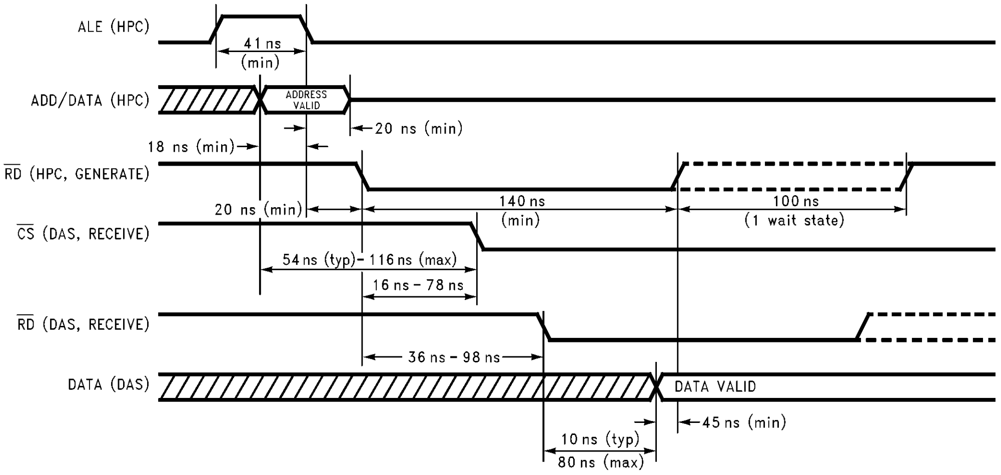
The second timing requirement is the data setup time referenced to the rising edge of the HPC's $\overline{\text{RD}}$ line. The DAS data outputs will be valid from a typical 10 ns to a maximum of 80 ns after the falling edge of its $\overline{\text{RD}}$ line. The HPC requires 45 ns of setup time and has a $\overline{\text{RD}}$ pulse width of 140 ns (1 wait state), resulting in 95 ns of total delay in the $\overline{\text{RD}}$ buffers and data latency of the DAS. Again, at the extreme limits of the operating conditions, the valid data might miss the 45 ns of required setup time, so the insertion of 1 extra wait state (100 ns) in the read cycle of the HPC is required. The design shown here is an example to demonstrate necessary design considerations and may not be the best possible solution for every application.

The circuit of *Figure 7* was implemented and tested using the "HPC Designer's Kit" development system. The development system performs the HPC real time emulation and all of the HPC's features are available for use in the application. The HPC Designer's Kit also closely resembles the real processor's switching characteristics.

Figure 10 shows scope photos of the $\overline{\text{CS}}$, $\overline{\text{RD}}$ and $\overline{\text{WR}}$ signals from the *Figure 7* circuit, using the HPC development system. *Figure 10a* shows a read and a write cycle when no inverter is added in the $\overline{\text{RD}}$ line. There is plenty of setup time for $\overline{\text{CS}}$ to $\overline{\text{WR}}$ but not for $\overline{\text{CS}}$ to $\overline{\text{RD}}$. *Figure 10b* shows a close look of the $\overline{\text{CS}}$ to $\overline{\text{RD}}$ setup time of *Figure 10a*. The setup time is 18 ns (at room temperature), very close to 20 ns, but not enough margin for circuit and temperature variations. *Figure 10c* shows a close look of the $\overline{\text{CS}}$ and $\overline{\text{RD}}$ signals after adding the inverters in the $\overline{\text{RD}}$ line. The setup time has increased to 30 ns with 10 ns of margin to cover for circuit variations and temperature changes.

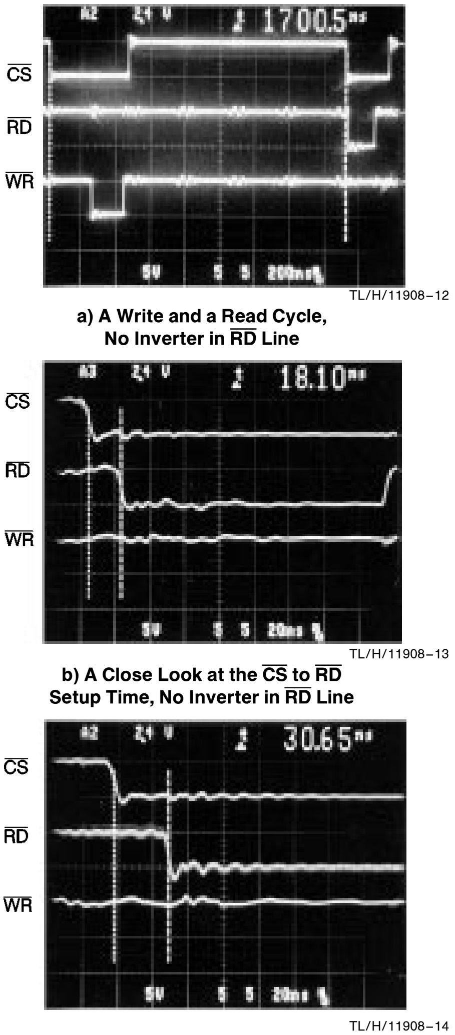
3.3.2 Minimal Address Decoding Circuit

Study of the switching characteristics for the circuit of *Figure 8* shows that the DAS address and $\overline{\text{CS}}$ setup times to ALE low are not satisfied when the HPC is running at 20 MHz. The HPC generates a valid address only 18 ns (min) before its ALE goes low at this speed (see *Figure 9*). The DAS needs 40 ns of setup time. The solution is reducing the HPC's clock frequency. The HPC running at 10 MHz will have a minimum setup time of 43 ns. There is some extra delay for the DAS' $\overline{\text{CS}}$ through U5 that must also be considered. This is the input to output propagation delay of U5, from the moment that address lines become valid to the point that the DAS' $\overline{\text{CS}}$ (Y0 output of U5) goes low.



TL/H/11908–11

FIGURE 9. DAS/HPC Interface Timing Diagram (Complete Address Decoding)



TL/H/11908–12

a) A Write and a Read Cycle, No Inverter in $\overline{\text{RD}}$ Line

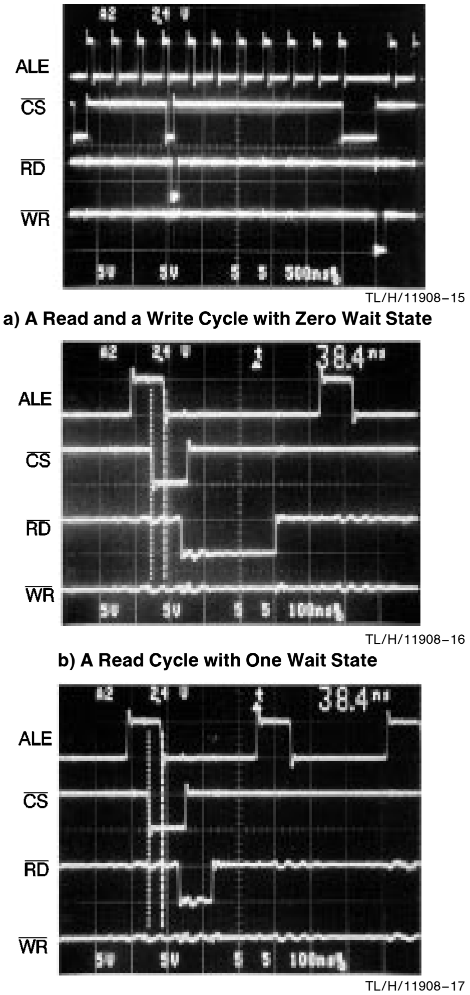
TL/H/11908–13

b) A Close Look at the $\overline{\text{CS}}$ to $\overline{\text{RD}}$ Setup Time, No Inverter in $\overline{\text{RD}}$ Line

TL/H/11908–14

c) A Close Look at the $\overline{\text{CS}}$ to $\overline{\text{RD}}$ Setup Time, 2 Inverters in $\overline{\text{RD}}$ Line

FIGURE 10. Scope Photos of $\overline{\text{CS}}$, $\overline{\text{RD}}$ and $\overline{\text{WR}}$ Signals at the DAS, *Figure 7* Circuit



TL/H/11908–15

a) A Read and a Write Cycle with Zero Wait State

TL/H/11908–16

b) A Read Cycle with One Wait State

TL/H/11908–17

c) A Read Cycle with Zero Wait State

FIGURE 11. Scope Photos of ALE, $\overline{\text{CS}}$, $\overline{\text{RD}}$ and $\overline{\text{WR}}$ Signals at the DAS, *Figure 8* Circuit

Practical measurements have shown reliable data transfer between the DAS and the HPC with the HPC running at 11.09 MHz clock at room temperature.

As a result of the HPC's lower clock frequency, the external data transfer can be performed with zero wait state, as opposed to the circuit of *Figure 7* where 1 wait state is essential. This speeds up the external read and write cycles and is especially useful when multiple successive reads are performed from the FIFO.

The circuit of *Figure 8* was also implemented and tested using the HPC development system. *Figure 11* shows scope photos of the ALE, $\overline{\text{CS}}$, $\overline{\text{RD}}$ and $\overline{\text{WR}}$ signals at the DAS inputs for the *Figure 8* circuit.*Figure 11a* shows a read and a write cycle with zero wait states. Notice that the read and write pulse rising edges occur after the $\overline{\text{CS}}$ signal. This does not matter since $\overline{\text{CS}}$ is internally latched. *Figures 11b* and *11c* give a more detailed view of the read cycles with 1 and 0 wait states, demonstrating about 180 ns shorter read cycle with no wait states. There is also about 38 ns of $\overline{\text{CS}}$ to

ALE low setup time. This is still about 2 ns less than the published DAS specifications. Although, practical tests resulted in reliable data transfer, to ensure dependable operation for the extremes of the circuit parameters and temperature variations, designers should use 10 MHz or less clock frequency for the HPC.

4.0 A SYSTEM EXAMPLE: A SEMICONDUCTOR FURNACE

In this application example the DAS measures the inputs from five sensors in a semiconductor furnace. We assume one of the circuits in *Figures 7* or *8* is used as the furnace data acquisition and control system. The system requirements will be defined and based on these requirements the DAS programming values for the DAS registers will be specified. A typical assembly routine for the HPC will also be presented for the DAS initialization and data capture.

Figure 12 shows a diagram of a typical measurement arrangement in a semiconductor furnace. A flow sensor measures the gas flow in the furnace chamber's duct. A pressure sensor measures the pressure in the chamber. Three temperature sensors measure the furnace temperature at the middle and each end of the furnace.

4.1 System Requirements and Assumptions

To control the operation of the furnace the following five measurements must be made:

— Absolute temperature at T1, with 12-bit resolution.

— Relative temperature, T1 to T2, with 12-bit + sign resolution.

— Relative temperature, Tl to T3, with 12-bit + sign resolution.

— Gas flow, F, through the chamber, with 8-bit resolution.

— Pressure, P, in the chamber, with 8-bit resolution.

There are three alarm conditions that are also being monitored:

— Gas flow, F, exceeds a maximum limit.

— Gas flow, F, drops below a minimum limit.

— Pressure, P, exceeds a maximum limit.

The following assumptions are also made for the system:

— All the signals from the sensors are conditioned (gain and offset adjusted) to provide voltage levels within 0V–2.5V for the DAS inputs.

— The output of all signal conditioning circuits are single ended with respect to analog ground.

— The signal at the output of the flow sensor signal conditioner has 600Ω source impedance.

— The DAS reference voltage is 2.5V, i.e., $V_{REF+} = 2.5V$ and $V_{REF-} = AGND$.

— The circuits in *Figure 7* or *8* (either one) is used for the furnace measurement and monitoring system. The following discussions and the program codes will be valid for both circuits.

— An approximate throughput rate of 50 Hz is desired for each set of measurement results (a set of results every 20 ms). However, due to the slow varying nature of the input signals, precisely controlled throughput rate is not essential for proper system performance.

4.2 DAS Setup and Register Programming

Based on the system requirements, we can proceed with the DAS setup and register settings.

The five sensor outputs are assigned to the first five DAS inputs:

— IN0: T1

— IN1: T2

— IN2: T3

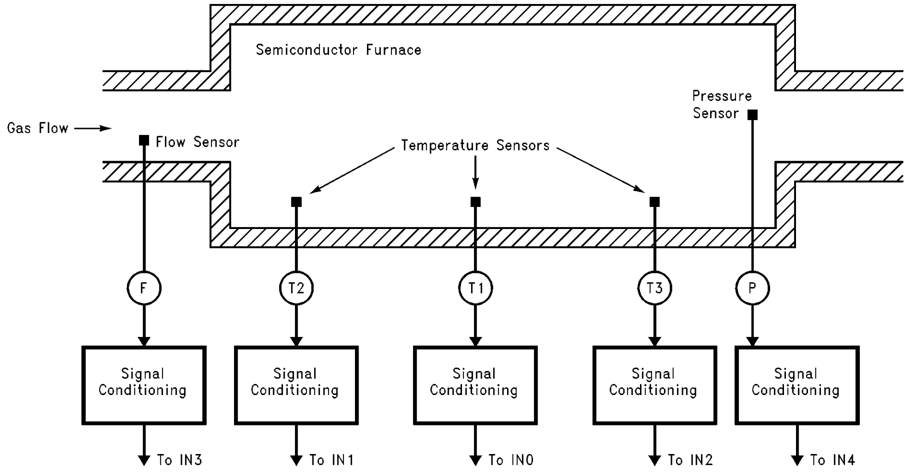
— IN3: F

— IN4: P

— IN5: Not used - Tied to GND

— IN6: Not used - Tied to GND

— IN7: Not used - Tied to GND



Single Ended 0V–2.5V Signals to the DAS Inputs

TL/H/11908–22

FIGURE 12. Diagram of a Typical Measurement Arrangement in a Semiconductor Furnace

Seven DAS instructions are needed for measurement and limit monitoring. Five perform the conversions and two perform the "watchdog" function for comparison of "F" and "P" against programmed limits. Note that the variable "F" needs only 1 instruction to monitor both high and low limits.

The following procedures are assumed for system operations:

— The seven instructions are executed in sequence from 0 to 6 with zero delay between them.

— After execution of instruction #6 the DAS loops back to instruction #0 and continues. Each loop is called an instruction loop.

— A delay is added, using the Timer register, before instruction #0 to provide 50 Hz throughput rate.

— Each instruction loop generates 5 conversion results. The FIFO is filled with 30 (6 sets of 5) results and is then read by the microcontroller. This is done by having an interrupt (from the DAS to the HPC) when a specified number of results are contained in the FIFO.

— Conversion will not be stopped during FIFO reads. Reads are performed during last comparison instruction (#6) and during the delay before instruction #0. The reads add extra delay after each six instruction loop, but the amount of the delay is negligible compared to the 20 ms loop duration. (See Section 2.3 for a discussion on reading during conversion and the interruption of the internal clock during reads and writes.)

The input from the flow sensor has a 600Ω source impedance, so it requires additional acquisition time. Referring to the equation in the DAS data sheet, for the 8-bit and "watchdog" mode, the acquisition time value (D) programmed in bits D12 through D15 of the Instruction register should be equal to 2 ($D = 0.36 \times Rs(k\Omega) \times f_{clk}(MHz) = 0.36 \times 0.6 \times 8 = 1.73$).

Now the contents of the DAS registers can be specified.

INSTRUCTION REGISTER:

— Sync and Pause bits are not used.

Instruction Register definition:

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
Acquisition Time				W-dog	8/12	Timer	Sync	V_{IN}^{-}			V_{IN}^{+}			Paus.	Loop

Instruction #0: Measuring T1, Single Ended, 12-Bit, Timer enabled

V_{IN}^{+} = IN0 (T1), V_{IN}^{-} = AGND

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

Instruction #1: Measuring T1−T2, Differential mode, 12-Bit + sign

V_{IN}^{+} = IN0 (T1), V_{IN}^{-} = IN1 (T2)

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

Instruction #2: Measuring T1−T3, Differential mode, 12-Bit + sign

V_{IN}^{+} = IN0 (T1), V_{IN}^{-} = IN2 (T3)

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Instruction #3: Measuring F, Single Ended, 8-Bit, D = 2

V_{IN}^{+} = IN3 (F), V_{IN}^{-} = AGND

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
0	0	1	0	0	1	0	0	0	0	0	0	1	1	0	0

Instruction #4: Watchdog mode, F, Single Ended, D = 2

V_{IN}^{+} = IN3 (F), V_{IN}^{-} = AGND

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0

Instruction #5: Measuring P, Single Ended, 8-Bit

V_{IN}^{+} = IN4 (P), V_{IN}^{-} = AGND

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0

Instruction #6: Watchdog mode, P, Single Ended, Loop bit enabled

V_{IN}^{+} = IN4 (P), V_{IN}^{-} = AGND

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1

Instruction #7: Not Used

Instructions #4 and #6 also have limit values. Instruction #4 has two limit values and instruction #6 has only one limit value. These values are referred to as F_MIN, F_MAX, P_MAX.

Instruction RAM, Limits definition:

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
Don't Care						>/<	Sign	Limit							

Instruction #4, Limit #1

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	1	0	F_MAX							

Instruction #4, Limit #2

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0	F_MIN							

Instruction #6, Limit #1

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	1	0	P_MAX							

Instruction #6, Limit #2, Limit value equal negative full-scale to prevent false interrupts.

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

INTERRUPT ENABLE REGISTER:

- INT0: Comparison Limit: Enable
- INT1: Instruction Number: Disable
- INT2: FIFO Full: Enable
- INT3: Auto Zero Complete: Disable
- INT4: Calibration Complete: Enable
- INT5: Pause: Disable
- INT6: Low Supply: Disable
- INT7: Standby Return: Disable
- Programmed instruction number: 0, Not used
- Programmed number of results in FIFO: 30 (11110 binary)

Interrupt Enable Register

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
Number of results in FIFO					Instruction Number			INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0
1	1	1	1	0	0	0	0	0	0	0	1	0	1	0	1

CONFIGURATION REGISTER:

— No auto zero or calibration before each conversion.

— Instruction number on bits D13 to D15 of the conversion results.

— Sync bit is not used and can be programmed as either input or output.

Configuration Register, Start Conversion Command

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
Don't Care				Diag.	Test	RAM Pointer		Sync I/O	A/Z Each	Chan Mask	Stand-by	Full Cal	Auto Zero	Reset	Start
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Configuration Register, Reset Command

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Configuration Register, Full Calibration Command

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

Configuration Register, RAM bank 1 Selection Command (Conversion is stopped)

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Configuration Register, stopping the Conversion Command

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TIMER REGISTER:

To calculate the timer preset value, we must first calculate the total instruction execution time. The following table shows the number of clock cycles for each instruction. Please see datasheet, Section 4.0 (Sequencer), for the discussion of the states and their duration.

Instruction #	State 0	State 1	State 7	State 6	State 4	State 5	Number of Clock Cycles
0	1	1	9			44	55
1	1	1	9			44	55
2	1	1	9			44	55
3	1	1	2			21	25
4	1	1	2	5	1	5	15
5	1	1	6			21	29
6	1	1	6	5	1	5	19
						Total:	253

There is a total of 253 clock cycles required for instruction execution. The Timer delay includes a fixed 2 clock cycles that must be added to 253, resulting in

$$253 + 2 = 255 \text{ clock cycles.}$$

The total time for 255 clock cycles would be:

$$255 \times 1/8 \text{ MHz} = 31.875 \ \mu\text{s.}$$

This time must be subtracted from 20 ms to get the Timer delay.

$$20 \text{ ms} - 31.875 \ \mu\text{s} = 19.968 \text{ ms.}$$

A Timer count is 32 clock cycles, with an 8 MHz clock (125 ns period), each Timer count is

$$32 \times 125 \text{ ns} = 4 \ \mu\text{s.}$$

The timer preset value is then

$$19.968 \text{ ms} \div 4 \ \mu\text{s} = 4992,$$

with a hex value of 1380H.

Once the required contents of the DAS registers have been determined, the next step is to program the processor to interact with the DAS.

4.3 Microcontroller Programming

Interaction between a microcontroller and the DAS is basically accomplished by read and write operations, the microcontroller's external data transfer instructions are used for communications with the DAS.

A write operation to the DAS needs 2 variables: The DAS register address and the data to be written. A read operation from the DAS needs only the register address. The DAS register address and data will be referred to as DAS__REG__ADD and DAS__DATA in the following examples.

Examples of assembly mnemonics for the DAS read and write operations are presented below for the HPC, 8051, and 68HC11 microcontroller families:

The HPC family can directly write 16-bit data to a memory-mapped I/O. Its main data transfer instruction is "LD" (load). Each read or write is only one instruction.

Write:

```
    LD    DAS_REG_ADD,#DAS_DATA
```

Read:

```
    LD    [destination],DAS_REG_ADD
```

The 8051 family accesses an external memory-mapped I/O indirectly through the DPTR (data pointer) register. The data should be preloaded to the accumulator for writes and the accumulator is the destination for reads. The transfer is 8-bit and two cycles are needed for 16 bits of data. The basic instruction are "MOV" (move), "MOVX" (move external) and "INC" (increment).

Write:

```
    MOV    DPTR,#DAS_REG_ADD
    MOV    A,DAS_DATA(low byte)
    MOVX   @DPTR,A
    INC    DPTR
    MOV    A,DAS_DATA(high byte)
    MOVX   @DPTR,A
```

Read:

```
      MOV DPTR,#DAS_REG_ADD
      MOVX A,@DPTR
      MOV [destination],A
      INC DPTR
      MOVX A,@DPTR
```

The above examples assume 16-bit addressing, however registers R0 or R1 of the 8051 can be used in place of DPTR for 8-bit addressing.

The 68HC11 family can access external memory mapped I/O directly. The data should be preloaded to the accumulator for writes and the accumulator is the destination for reads. Although, the 68HC11 is an 8-bit processor, it has 16-bit data transfer instructions that uses a double accumulator (A + B, called D) and performs 2 transfer cycles using a single instruction. The basic instructions are "LDD" (load double accumulator) and "STD" (store double accumulator).

Write:

```
    LDD    #DAS_DATA
    STD    DAS_REG_ADD
```

Read:

```
    LDD    DAS_REG_ADD
```

4.3.1 HPC Assembly Routines for the Semiconductor Furnace Example

The program listing in *Figure 13* is the HPC assembly routine for the semiconductor furnace application example. The program contains only the DAS initialization and the DAS interrupt service routine. A complete program for the application will have all the data manipulation and control functions, which are not discussed here.

The routines closely follow the procedure in the flowcharts of *Figure 2* and are extensively commented to be self-explanatory. The routines also separated according to the flowchart sections. The main difference between the routines and the flowchart is that the DAS is not stopped at the start of the interrupt service routine.

The interrupt service routine uses the IFBIT (if bit is true) instruction to test for the state of the interrupt status bits. This is very handy for control applications.

The READ__FIFO routine reads the FIFO contents and stores them to a specified block of memory starting at the location called DAS__RESULT. The size of the block and, consequently, the number of FIFO locations being read is programmable (30 locations on the listing). The routine is only 5 lines of assembly. It uses the multi-function "XS" (exchange and skip) instruction to perform the data transfer, address increment or decrement, and a compare for decision making. The first LD instruction in the READ__FIFO routine loads the HPC's B and K registers with the starting address and the ending address of the memory block. The second instruction reads a word (16-bit) from the FIFO to "A" (accumulator). The XS instruction stores A in the memory location pointed to by B, increments B by 2 (for 2 bytes) and then compares B with K to test for the end of the memory block. If the end of the block has not been reached, the program jumps back to load the next word from the FIFO, otherwise the program skips the "JP" (jump) instruction and returns from the service routine.

```
                         #NOHEADING
                         #PWIDTH= 112

                         ;****************************************************************************
                         ;*                                                                          *
                         ;* AN HPC ASSEMBLY ROUTINE FOR THE SEMICODUCTOR APPLICATION EXAMPLE FOR THE *
                         ;*   APPLICATION NOTE "INTERFACING THE LM12454/8 DATA ACQUISITION SYSTEM    *
                         ;*             CHIPS TO MICROPROCESSORS AND MICROCONTROLLERS"               *
                         ;*                                                                          *
                         ;* BY: FARID SALEH                                                          *
                         ;* DATE: 7/27/93                                                            *
                         ;****************************************************************************


                         ;****** HPC REGISTERS SYMBOLIC DEFINITIONS ******

00C0                             PSW     = 0C0H          ;PROCESSOR STATUS REGISTER
                         ;       SP      = 0C4H          ;STACK POINTER
                         ;       PC      = 0C6H          ;PROGRAM COUNTER
                         ;       A       = 0C8H          ;ACCUMULATOR
                         ;       K       = 0CAH          ;K REGISTER
                         ;       B       = 0CCH          ;B REGISTER
                         ;       X       = 0CEH          ;X REGISTER
00E2                             PORTB   = 0E2H          ;PORT B DATA REGISTER
00F2                             DIRB    = 0F2H          ;PORT B DIREGTION REGISTER
00F4                             BFUN    = 0F4H          ;PORT B ALTERNATE FUNCTION REGISTER
00D0                             ENIR    = 0D0H          ;INTERRUPT ENABLE REGISTER
00D4                             IRCD    = 0D4H          ;INTERRUPT / INPUT CAPTURE CONDITION REGISTER
00D2                             IRPD    = 0D2H          ;INTRUPT PENDING REGISTER

                         ;****** DAS REGISTERS / VARIABLES / CONSTANTS  SYMBOLIC DEFINITIONS ******

0200                             INSTR0  = 0200H         ;DAS INSTRUCTION REGISTER ADDRESSES
0202                             INSTR1  = 0202H         ;READ/WRITE REGISTERS
0204                             INSTR2  = 0204H         ; "
0206                             INSTR3  = 0206H         ; "
0208                             INSTR4  = 0208H         ; "
020A                             INSTR5  = 020AH         ; "
020C                             INSTR6  = 020CH         ; "
020E                             INSTR7  = 020EH         ; "

0210                             CONFIG  = 0210H         ;DAS CONFIGURATION REGISTER ADDRESS, R/W
0212                             INTEN   = 0212H         ;DAS INTERRUPT ENABLE REG. ADDRESS, R/W
0214                             INTSTAT = 0214H         ;DAS INTERRUPT STATUS REG. ADD. READ ONLY
0216                             TIMER   = 0216H         ;DAS TIMER REG. ADDRESS, R/W
0218                             FIFO    = 0218H         ;DAS FIFO ADDRESS, READ ONLY
021A                             LMTSTAT = 021AH         ;DAS LIMIT STATUS REG. ADD. READ ONLY

01C0                             DAS_RESULT = 01C0H      ;START ADDRESS OF TOP 64 LOCATION OF HPC
                                                         ;ON-CHIP RAM TO STORE CONVERSION RESULTS
00BC                             DAS_INT_MEM = 00BCH     ;HPC MEMORY LOCATION TO STORE INTERRUPT
                                                         ;STATUS REGISTER
00BE                             DAS_LIM_MEM = 00BEH     ;HPC MEMORY LOCATION TO STORE LIMIT STATUS
                                                         ;REGISTER
001E                             FIFO_CNT = 30           ;NUNBER OF RESULTS IN FIFO, A DECIMAL VALUE
1380                             TIMER_SET = 01380H      ;TIMER PRESET VALUE

00FF                             F_MAX   = 0FFH          ;HIGH LIMIT FOR GAS FLOW
0000                             F_MIN   = 000H          ;LOW LIMIT FOR GAS FLOW
00FF                             P_MAX   = 0FFH          ;HIGH LIMIT FOR PRESSURE

00BA                             FLAGS   = 00BAH         ;GENERAL SOFTWARE FLAGS BYTE
0000                             CAL_FLG = 0             ;BIT 0 OF FLAGS BYTE FOR CALLIBRATION

0000                     .SECT PDASTST,ROM16

```

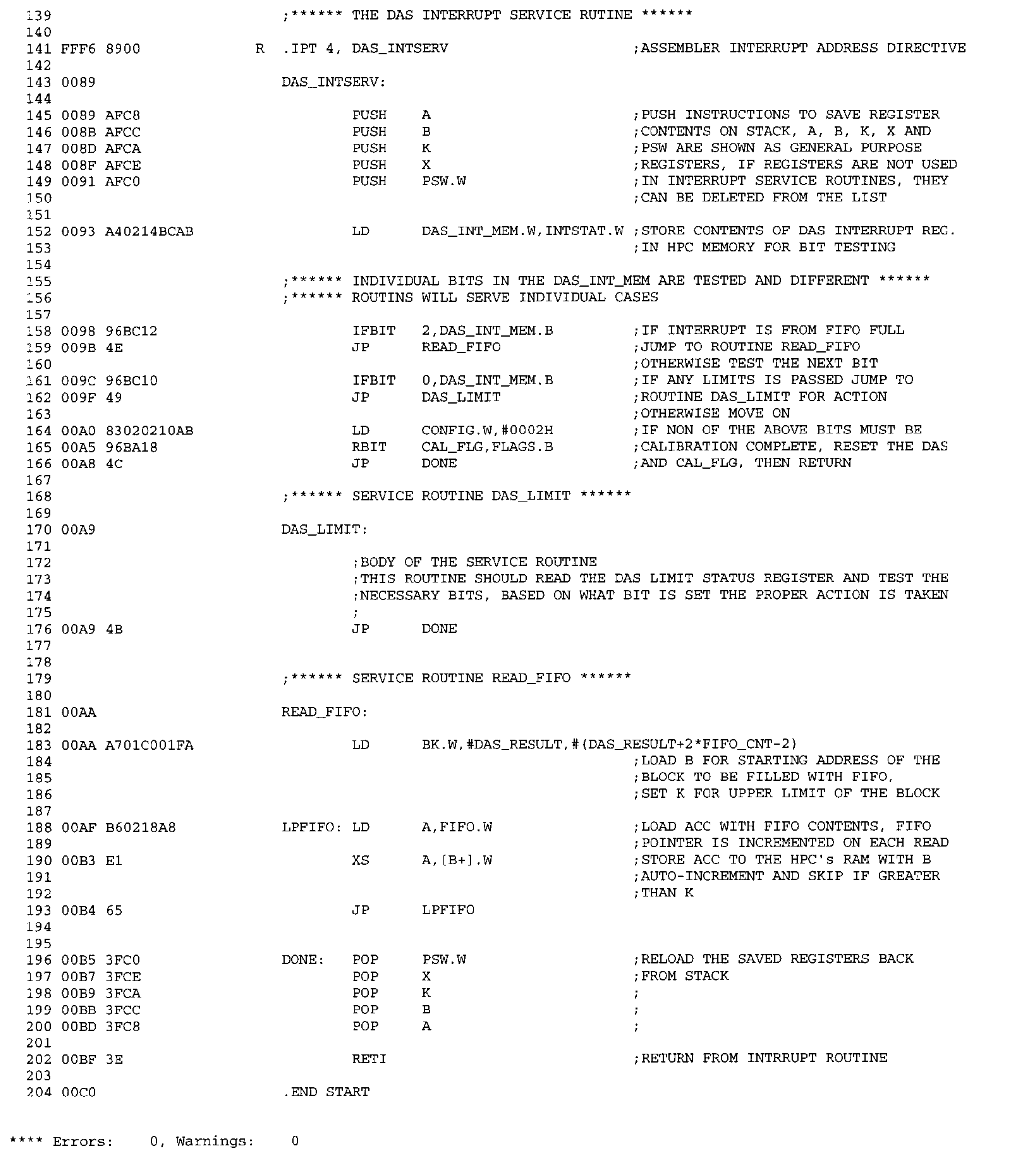
TL/H/11908–18

FIGURE 13. HPC Assembly Program Listing (Continued)

```
                        ;******  HPC INITIALIZATION ******

0000                    START:
0000 9718C0                     LD      PSW.B,#018H             ;PROCESSOR STATUS WORD,EXPANDED MODE
                                                                ;1 WAIT STATE, CAN BE ZERO WAIT STATE
                                                                ;FOR CIRCUIT IN FIGURE 8, #01CH
0003 9700D0                     LD      ENIR.B,#00H             ;DISABLE ALL INTERRUPTS
0006 9700D2                     LD      IRPD.B,#00H             ;CLEAR ANY INTERRUPT PENDING BIT
0009 9700D4                     LD      IRCD.B,#00H             ;SET I4 FOR HIGH TO LOW EDGE DETECT
000C B70000E2                   LD      PORTB.W,#00H            ;PORT B ALL ZERO
0010 B7FFFFF2                   LD      DIRB.W,#0FFFFH          ;PORT B ALL OUTPUTS, B10,11,12 AND 15
                                                                ;ARE PREDEFINED DUE TO EXPANDED MODE
                                                                ;NOTE: PORT A IS ALSO PREDEFINED
0014 B70000F4                   LD      BFUN.W,#00H             ;PORT B NO ALTERNATE FUNCTION

                        ;****** DAS REGISTERS INITIALIZATION ******

0018 83020210AB                 LD      CONFIG.W,#0002H         ;RESET, SELECT RAM SECTION 0, RP=00

001D 8702000200AB               LD      INSTR0.W,#0200H         ;INSTRUCTIONS INITIALIZATION, VALUES
0023 83200202AB                 LD      INSTR1.W,#0020H         ;ARE AS SPECIFIED ON THE SYSTEM
0028 83400204AB                 LD      INSTR2.W,#0040H         ;DESIGN
002D 87240C0206AB               LD      INSTR3.W,#0240CH        ; "
0033 87280C0208AB               LD      INSTR4.W,#0280CH        ; "
0039 870410020AAB               LD      INSTR5.W,#0410H         ; "
003F 870811020CAB               LD      INSTR6.W,#0811H         ; "

0045 8701000210AB               LD      CONFIG.W,#0100H         ;SELECT RAM SECTION 1, RP=01

004B 8702FF0208AB               LD      INSTR4.W,#(0200H+F_MAX) ;HIGH LIMIT FOR INSTRUCTION 4
0051 8702FF020CAB               LD      INSTR6.W,#(0200H+P_MAX) ;HIGH LIMIT FOR INSTRUCTION 6

0057 8702000210AB               LD      CONFIG.W,#0200H         ;SELECT RAM SECTION 2, RP=10

005D 83000208AB                 LD      INSTR4.W,#(0000H+F_MIN) ;LOW LIMIT FOR INSTRUCTION 4
0062 870100020CAB               LD      INSTR6.W,#0100H         ;LOW LIMIT FOR INSTRUCTION 6
                                                                ;NEGATIVE FULL-SCALE, NOT USED

0068 87F0150212AB               LD      INTEN.W,#((FIFO_CNT*2048)+015H)
                                                                ;SHIFT FIFO_CNT TO MSBs OF HIGH
                                                                ;BYTE, THEN ADD LOW BYTE (015H),
                                                                ;DAS INT # 0, 2 AND 4 ARE ENABLED
006E 8713800216AB               LD      TIMER.W,#TIMER_SET      ;TIMER INITIALIZED WITH PRESET VALUE

                        ;****** ENABLING HPC INTERRUPT #4 ******

0074 9711D0                     LD      ENIR.B,#011H            ;ENABLE HPC GLOBAL AND INTERRUPT #4

                        ;****** DAS FULL CALIBRATION ******

0077 96BA08                     SBIT    CAL_FLG,FLAGS.B         ;SET CALIBRATION FLAG FOR PROGRAM
                                                                ;CONTROL
007A 83080210AB                 LD      CONFIG.W,#0008H         ;DAS CALIBRATION IS STARTED

007F 96BA10             WAIT1:  IFBIT   CAL_FLG,FLAGS.B         ;CHECK FOR CAL_FLG, IF 1 WAIT,
                                                                ;IDEL LOOP UNTIL CALIBRATION IS DONE
0082 63                         JP      WAIT1                   ;AND INTRRUPT FROM DAS IS RECEIVED,
                                                                ;IN A COMPLETE PROGRAM, PROCESSOR
                                                                ;CAN DO OTHER TASKS

                        ;****** STARTING THE CONVERSIONS ******

0083 83010210AB                 LD      CONFIG.W,#0001H         ;START BIT = 1, DAS STARTS

0088 60                 WAIT2:  JP      WAIT2                   ;IDEL LOOP FOR HPC TO WAIT FOR DAS
                                                                ;INTERRUPT
                                                                ;THIS IS MAINLY A TEST STATEMENT HERE
                                                                ;AND IN A COMPLETE PROGRAM PROCESSOR
                                                                ;IS DOING OTHER TASKS


```

TL/H/11908–19

FIGURE 13. HPC Assembly Program Listing (Continued)



```
                         ;****** THE DAS INTERRUPT SERVICE RUTINE ******

FFF6 8900             R  .IPT 4, DAS_INTSERV                     ;ASSEMBLER INTERRUPT ADDRESS DIRECTIVE

0089                     DAS_INTSERV:

0089 AFC8                        PUSH    A                       ;PUSH INSTRUCTIONS TO SAVE REGISTER
008B AFCC                        PUSH    B                       ;CONTENTS ON STACK, A, B, K, X AND
008D AFCA                        PUSH    K                       ;PSW ARE SHOWN AS GENERAL PURPOSE
008F AFCE                        PUSH    X                       ;REGISTERS, IF REGISTERS ARE NOT USED
0091 AFC0                        PUSH    PSW.W                   ;IN INTERRUPT SERVICE ROUTINES, THEY
                                                                 ;CAN BE DELETED FROM THE LIST

0093 A40214BCAB                  LD      DAS_INT_MEM.W,INTSTAT.W ;STORE CONTENTS OF DAS INTERRUPT REG.
                                                                 ;IN HPC MEMORY FOR BIT TESTING

                         ;****** INDIVIDUAL BITS IN THE DAS_INT_MEM ARE TESTED AND DIFFERENT ******
                         ;****** ROUTINS WILL SERVE INDIVIDUAL CASES

0098 96BC12                      IFBIT   2,DAS_INT_MEM.B         ;IF INTERRUPT IS FROM FIFO FULL
009B 4E                          JP      READ_FIFO               ;JUMP TO ROUTINE READ_FIFO
                                                                 ;OTHERWISE TEST THE NEXT BIT
009C 96BC10                      IFBIT   0,DAS_INT_MEM.B         ;IF ANY LIMITS IS PASSED JUMP TO
009F 49                          JP      DAS_LIMIT               ;ROUTINE DAS_LIMIT FOR ACTION
                                                                 ;OTHERWISE MOVE ON
00A0 83020210AB                  LD      CONFIG.W,#0002H         ;IF NON OF THE ABOVE BITS MUST BE
00A5 96BA18                      RBIT    CAL_FLG,FLAGS.B         ;CALIBRATION COMPLETE, RESET THE DAS
00A8 4C                          JP      DONE                    ;AND CAL_FLG, THEN RETURN

                         ;****** SERVICE ROUTINE DAS_LIMIT ******

00A9                     DAS_LIMIT:

                                 ;BODY OF THE SERVICE ROUTINE
                                 ;THIS ROUTINE SHOULD READ THE DAS LIMIT STATUS REGISTER AND TEST THE
                                 ;NECESSARY BITS, BASED ON WHAT BIT IS SET THE PROPER ACTION IS TAKEN
                                 ;
00A9 4B                          JP      DONE


                         ;****** SERVICE ROUTINE READ_FIFO ******

00AA                     READ_FIFO:

00AA A701C001FA                  LD      BK.W,#DAS_RESULT,#(DAS_RESULT+2*FIFO_CNT-2)
                                                                 ;LOAD B FOR STARTING ADDRESS OF THE
                                                                 ;BLOCK TO BE FILLED WITH FIFO,
                                                                 ;SET K FOR UPPER LIMIT OF THE BLOCK

00AF B60218A8            LPFIFO: LD      A,FIFO.W                ;LOAD ACC WITH FIFO CONTENTS, FIFO
                                                                 ;POINTER IS INCREMENTED ON EACH READ
00B3 E1                          XS      A,[B+].W                ;STORE ACC TO THE HPC's RAM WITH B
                                                                 ;AUTO-INCREMENT AND SKIP IF GREATER
                                                                 ;THAN K
00B4 65                          JP      LPFIFO


00B5 3FC0                DONE:   POP     PSW.W                   ;RELOAD THE SAVED REGISTERS BACK
00B7 3FCE                        POP     X                       ;FROM STACK
00B9 3FCA                        POP     K                       ;
00BB 3FCC                        POP     B                       ;
00BD 3FC8                        POP     A                       ;

00BF 3E                          RETI                            ;RETURN FROM INTRRUPT ROUTINE

00C0                     .END START


**** Errors:    0, Warnings:    0
```

TL/H/11908–20

FIGURE 13. HPC Assembly Program Listing

APPENDIX A: Registers Bit Assignments and Programmer's Notes

CONFIGURATION REGISTER (Read/Write):

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
Don't Care				Diag.	Test	RAM Pointer		Sync I/O	A/Z Each	Chan Mask	Stand-by	Full Cal	Auto Zero	Reset	Start

D0: - Start: 0 = Stops the instruction execution. 1 = Starts the instruction execution

D1: - Reset: When set to 1, resets the Start bit, also resets all the bits is status registers and resets the instruction pointer to zero, will automatically reset itself to zero after 2 clock pulses

D2: - Auto-Zero: When set to 1 a long auto-zero calibration cycle is performed

D3: - Full Calibration: When set to 1 a full calibration cycle is performed

D4: - Standby: When set to 1 the chip goes to low-power standby mode, resetting the bit will return the chip to active mode after a power-up delay

D5: - Channel Mask: 0 = Bits 13 to 15 of the conversion result hold the instruction number to which the result belongs, 1 = Bits 13 to 15 of the result hold the extended sign bit

D6: - A/Z Each: When set to 1 a short auto-zero cycle if performed before each conversion

D7: - Sync I/O: 0 = Sync pin is input, 1 = Sync pin is output

D9–D8: - RAM Pointer: Selects the sections of the instruction RAM, 00 = Instructions, 01 = Limits #1, 10 = Limits #2

D10: - This bit is used for production testing, must be kept zero for normal operation

D11: - Diagnostic: When set to 1 perform diagnostic conversion along with a properly selected instruction

D15–D12: - Don't care

PROGRAMMER'S NOTES:

Configuration Registers: Address: Symbol:
Note:

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Hexadecimal value:

Configuration Register: Address: Symbol:
Note:

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Hexadecimal value:

Configuration Register: Address: Symbol:
Note:

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Hexadecimal value:

APPENDIX A: Registers Bit Assignments and Programmer's Notes

INSTRUCTION RAM (Read/Write): (Continued)

Instruction:

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
Acquisition Time				W-dog	8/12	Timer	Sync	$V_{IN}-$			$V_{IN}+$			Paus.	Loop

D0: - Loop: 0 = Go to next instruction, 1 = Loop back to instruction #0

D1: - Pause: 0 = No pause, 1 = Pause, don't do the instruction, Start bit in configuration register resets to 0 when a pause encountered, a 1 written to Start bit restarts the instruction execution

D4–D2: - $V_{IN}+$: Selects which input channel is connected to A/D's non-inverting input

D7–D5: - $V_{IN}-$: Selects which input channel is connected to A/D's inverting input

D8: - Sync: 0 = Normal operation, internal timing, 1 = S/H and conversion (comparison) timing, is controlled by SYNC input pin

D9: - Timer: 0 = No timer operation, 1 = Instruction execution halts until timer counts down to zero

D10: - 8/12: 0 = 12-bit + sign resolution, 1 = 8-bit + sign resolution

D11: - Watchdog: 0 = No watchdog comparision, 1 = Instruction performs watchdog comparisons

D15–D12: - Acquisition Time: Determines S/H acquisition time
For 12-bit + sign: (9 + 2D) clock cycles, For 8-bit + sign: (2+2D) clock cycles
D = Content of D15–D12, R_S = Input source resistance
For 12-bit + sign: D = 0.45 x R_S[kΩ] x f_{CLK}[MHz]
For 8-bit + sign: D = 0.36 x R_S[kΩ] x f_{CLK}[MHz]

PROGRAMMER'S NOTES:

Instruction # 0: Address: Symbol:
Note:

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Hexadecimal value:

Instruction # 1: Address: Symbol:
Note:

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Hexadecimal value:

Instruction # 2: Address: Symbol:
Note:

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Hexadecimal value:

Instruction #3: Address: Symbol:
Note:

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Hexadecimal value:

APPENDIX A: Registers Bit Assignments and Programmer's Notes

INSTRUCTION RAM (Read/Write): (Continued)

PROGRAMMER'S NOTES:

Instruction # 4: Address: Symbol:
Note:

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Hexadecimal value:

Instruction # 5: Address: Symbol:
Note:

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Hexadecimal value:

Instruction # 6: Address: Symbol:
Note:

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Hexadecimal value:

Instruction # 7: Address: Symbol:
Note:

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Hexadecimal value:

APPENDIX A: Registers Bit Assignments and Programmer's Notes

INSTRUCTION RAM (Read/Write): (Continued)

Limits # 1

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
Don't Care						>/<	Sign	Limit							

D7–D0: - Limit: 8-bit limit value

D8: - Sign: Sign bit for limit value, 0 = Positive, 1 = Negative

D9: - >/<: High or low limit determination, 0 = Inputs lower than limit generate interrupt, 1 = Inputs higher than limit generate interrupt

D15–D10 - Don't Care

PROGRAMMER'S NOTES:

Instruction # 0, Limit # 1: Address: Symbol:

Note:

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Hexadecimal value:

Instruction # 1, Limit # 1: Address: Symbol:

Note:

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Hexadecimal value:

Instruction # 2, Limit # 1: Address: Symbol:

Note:

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Hexadecimal value:

Instruction # 3, Limit # 1: Address: Symbol:

Note:

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Hexadecimal value:

Instruction # 4, Limit # 1: Address: Symbol:

Note:

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Hexadecimal value:

Instruction # 5, Limit # 1: Address: Symbol:

Note:

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Hexadecimal value:

Instruction # 6, Limit # 1: Address: Symbol:

Note:

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Hexadecimal value:

Instruction # 7, Limit # 1: Address: Symbol:

Note:

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Hexadecimal value:

APPENDIX A

INSTRUCTION RAM (Read/Write): (Continued)

Limits #2

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
Don't Care						>/<	Sign	Limit							

D7–D0: - Limit: 8-bit limit value

D8: - Sign: Sign bit for limit value, 0 = Positive, 1 = Negative

D9: - >/<: High or low limit determination, 0 = Inputs lower than limit generate interrupt, 1 = Inputs higher than limit generate interrupt

D15–D10 - Don't Care

PROGRAMMER'S NOTES:

Instruction # 0, Limit # 2: Address: Symbol:
Note:

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Hexadecimal value:

Instruction # 1, Limit # 2: Address: Symbol:
Note:

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Hexadecimal value:

Instruction # 2, Limit # 2: Address: Symbol:
Note:

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Hexadecimal value:

Instruction # 3, Limit # 2: Address: Symbol:
Note:

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Hexadecimal value:

Instruction # 4, Limit # 2: Address: Symbol:
Note:

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Hexadecimal value:

Instruction # 5, Limit # 2: Address: Symbol:
Note:

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Hexadecimal value:

Instruction # 6, Limit # 2: Address: Symbol:
Note:

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Hexadecimal value:

Instruction # 7, Limit # 2: Address: Symbol:
Note:

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Hexadecimal value:

APPENDIX A: Registers Bit Assignments and Programmer's Notes

INTERRUPT ENABLE REGISTER (Read/Write):

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
Number of results in FIFO to Generate Interrupt (INT2)					Instruction Number to Generate Interrupt (INT1)			INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0

Bits # 0 to 7 enable interrupt generaton for the following conditions when the bit is set to 1.

D0: - INT0: Generates interrupt when a limit is passed in watchdog mode

D1: - INT1: Generates interrupt when the programmed instruction (D10−D8) is reached for execution

D2: - INT2: Generates interrupt when number of conversion results in FIFO is equal to the programmed value (D15−D11)

D3: - INT3: Generates interrupt when an auto-zero cycle is completed

D4: - INT4: Generates interrupt when a full calibration cycle is completed

D5: - INT5: Generates interrupt when a pause condition is encountered

D6: - INT6: Generates interrupt when low power supply is detected

D7: - INT7: Generates interrupt when the chip is returned from standby and is ready

D10−D8: - Programmable instruction number to generate an interrupt when that instruction is reached foR execution

D15−D11: - Programmable number of conversion results in the FIFO to generate an interrupt

PROGRAMMER's NOTES:

Interrupt Enable Register: Address: Symbol:

Note:

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Hexadecimal value:

Interrupt Enable Register: Address: Symbol:

Note:

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Hexadecimal value:

TIMER REGISTER (Read/Write):

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
N = Timer Preset Value															

Timer delays the execution of an instruction if Timer bit is set in the instruction.
The time delay in number of clock cycles is:

Delay = 32 x N + 2 [Clock Cycles]

PROGRAMMER's NOTES:

Timer Register: Address: Symbol

Note:

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Hexadecimal value:

Timer Register: Address: Symbol:

Note:

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Hexadecimal value:

APPENDIX A: Registers Bit Assignments and Programmer's Notes

FIFO REGISTER (Read only):

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
Instruction Number or Extended Sign			Sign	Conversion Result											

D11–D0: - Conversion Result:
For 12-bit + sign: 12-bit result value
For 8-bit + sign: D11–D4 = result value, D3–D0 = 1110

D12: - Sign: Conversion result sign bit, 0 = Positive, 1 = Negative

D15–D13: - Instruction number associated with the conversion result or the extended sign bit for 2's complement arithmetic, selected by bit D5 (Chan Mask) of Configuration Register

PROGRAMMER'S NOTES:

FIFO Register: Address: Symbol:
Note:

INTERRUPT STATUS REGISTER (Read only):

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
Number of results in FIFO					Instruction Number Being executed			INST7	INST6	INST5	INST4	INST3	INST2	INST1	INST0

BITS # 0 to 7 are interrupt flags (vectors) that will be set to 1 when the following conditions occur. The bits set to 1 whether the interrupt is enabled or disabled in the Interrupt Enable register. The bits reset to 0 when the register is read, or by a device reset through Configuration register.

D0: - INST0: Is set to 1 when a limit is passed in watchdog mode

D1: - INST1: Is set to 1 when the programmed instruction (D10–D8) is reached for execution

D2: - INST2: Is set to 1 when number of conversion results in FIFO is equal to the programmed value (D15–D11)

D3: - INST3: Is set to 1 when an auto-zero cycle is completed

D4: - INST4: Is set to 1 when a full calibration cycle is completed

D5: - INST5: Is set to 1 when a pause condition is encountered

D6: - INST6: Is set to 1 when low power supply is detected

D7: - INST7: Is set to 1 when the chip is returned from standby and is ready

D10–D8: - Holds the instruction number being executed or will be executed during a Pause or Timer delay

D15–D11: - Holds the present number of conversion results in the FIFO while the device is running

PROGRAMMER'S NOTES:

Interrupt Status Register: Address: Symbol:
Note:

Interfacing the LM12454/8 Data Acquisition System Chips to Microprocessors and Microcontrollers

AN-906

APPENDIX A: Registers Bit Assignments and Programmer's Notes

LIMIT STATUS REGISTER (Read only):

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
Limits #2: Status								Limits #1: Status							

The bits in this register are limit flags (vectors) that will be set to 1 when a limit is passed. The bits are associated to individual instruction limits as indicated below.

D0: - Limit # 1 of Instruction # 0 is passed
D1: - Limit # 1 of Instruction # 1 is passed
D2: - Limit # 1 of Instruction # 2 is passed
D3: - Limit # 1 of Instruction # 3 is passed
D4: - Limit # 1 of Instruction # 4 is passed
D5: - Limit # 1 of Instruction # 5 is passed
D6: - Limit # 1 of Instruction # 6 is passed
D7: - Limit # 1 of Instruction # 7 is passed
D8: - Limit # 2 of Instruction # 0 is passed
D9: - Limit # 2 of Instruction # 1 is passed
D10: - Limit # 2 of Instruction # 2 is passed
D11: - Limit # 2 of Instruction # 3 is passed
D12: - Limit # 2 of Instruction # 4 is passed
D13: - Limit # 2 of Instruction # 5 is passed
D14: - Limit # 2 of Instruction # 6 is passed
D15: - Limit # 2 of Instruction # 7 is passed

PROGRAMMER'S NOTES:

Limit Status Register: Address Symbol:
Note:

LIFE SUPPORT POLICY

NATIONAL'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT OF NATIONAL SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

National Semiconductor Corporation
2900 Semiconductor Drive
P.O. Box 58090
Santa Clara, CA 95052-8090
Tel: 1(800) 272-9959
TWX: (910) 339-9240

National Semiconductor GmbH
Livry-Gargan-Str. 10
D-82256 Fürstenfeldbruck
Germany
Tel: (81-41) 35-0
Telex: 527649
Fax: (81-41) 35-1

National Semiconductor Japan Ltd.
Sumitomo Chemical Engineering Center
Bldg. 7F
1-7-1, Nakase, Mihama-Ku
Chiba-City,
Ciba Prefecture 261
Tel: (043) 299-2300
Fax: (043) 299-2500

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 Semiconductores Do Brazil Ltda.
Rue Deputado Lacorda Franco
120-3A
Sao Paulo-SP
Brazil 05418-000
Tel: (55-11) 212-5066
Telex: 391-1131931 NSBR BR
Fax: (55-11) 212-1181

National Semiconductor (Australia) Pty, Ltd.
Building 16
Business Park Drive
Monash Business Park
Nottinghill, Melbourne
Victoria 3168 Australia
Tel: (3) 558-9999
Fax: (3) 558-9998

National does not assume any responsibility for use of any circuitry described, no circuit patent licenses are implied and National reserves the right at any time without notice to change said circuitry and specifications.