Interfacing the LM12454/8 Data Acquisition System Chips to the 80C51 Family of Microcontrollers

National Semiconductor
Application Note 949
Nicholas Gray
May 1994

Table of Contents

1.0 INTRODUCTION

The LM12458 family of data acquisition system (DAS) components offers a self-calibrating, 12-bit + sign A/D converter with choice of single ended, fully differential, or mixed inputs, with on-chip differential reference, 4 or 8-input analog multiplexer, sample-and-hold, an impressive, flexible programmable logic system and a choice of speed/power combinations. The programmable logic has the circuitry to perform a number of tasks on its own, freeing the host processor for other tasks. This logic includes:

- An instruction RAM that allows the DAS to function on its own (after being programmed by the host processor) 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 the ''watchdog'' mode.
- A 32-word FIFO register to store conversion results until read by the host.
- Interrupt control logic with interrupt generation for 8 different conditions.
- A 16-bit timer register.
- Circuitry to synchronize signal acquisition with external events.
- A parallel microprocessor/microcontroller interface with selectable 8-bit or 16-bit data access.

Because the members of the LM12458 family are so versatile, working with them may appear, at first, to be an overwhelming task. However, gaining a basic understanding of the device will prove to be fairly easy and using it to be as easy as programming the host microprocessor or microcontroller.

The LM12458 family has 6 members, as shown in Table I. This Application Note will simply refer to the DAS or to the LM12458 as generic names for any member of the family. The drawings illustrate only the 8-input versions of the family, although references are meant to include the 4-input members as well. A brief overview of the DAS and information related to the subjects of discussion are given here, but this Application Note should be used in conjunction with the device data sheet and assumes that the reader has some familiarity with the device. Similarly, the 80C51 family of microcontrollers is discussed briefly here, but the assumption is that the reader is familiar with that device.

MCS™ is a trademark of Intel Corporation

 TL/H/12068 RRD-B30M75/Printed in U. S. A.

TABLE I. Members of the LM12454/8 Family

Device Number	Clock Frequency (Max, MHz)	Operating Supply Voltage	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

2.0 GENERAL OVERVIEW

The DAS has 3 different modes of operation: 12-bit + sign conversion, 8-bit + sign conversion, and 8-bit + sign comparison. The latter is referred to as the ''watchdog'' mode.

No conversion is performed in the watchdog mode, but the DAS samples the selected input(s) and compares it/them with values of the low and high limits stored in the instruction RAM. This comparison is done with a voltage comparator with one comparator input being the selected multiplexer input (pair) and the other input being the appropriate tap on the internal capacitive ladder of the converter. This tap is selected by a programmed value in the instruction register. If the input voltage is outside of the user defined and programmed minimum/maximum limits, an interrupt can be generated to indicate a fault condition, and the host processor could then service that interrupt, taking the appropriate action.

The 8 possible interrupts can be individually masked or enabled as desired by the user.

The DAS is designed to be controlled by a processor, but the DAS' functionality off loads most of the data acquisition burden from the processor, resulting in a great reduction of software and processor overhead. The processor downloads a set of operational instructions to the DAS' RAM and registers, then issues a start command to the DAS, which performs conversions and/or comparisons as indicated by the instructions, loading conversion results into the FIFO, while the processor is free to do other chores, or can be idled, if not needed.

There are two basic options at this point. The DAS can generate an interrupt to the processor when a predetermined number of conversion results are stored in the FIFO, or when any other interrupt conditions have occurred. The processor can then service the interrupt by reading the FIFO or taking whatever corrective action is appropriate. Alternatively, the processor can read the data or give a new command to the DAS without waiting for an interrupt from the DAS.

Internal operation ceases when the DAS is accessed by pulllng the $\overline{CS}$ pin low.

2.1 The DAS Programming Model

Figure 1 illustrates the functional block diagram or user programming model of the DAS, and is not intended to reflect the actual implementation of the internal building blocks. The Model contains the following blocks:

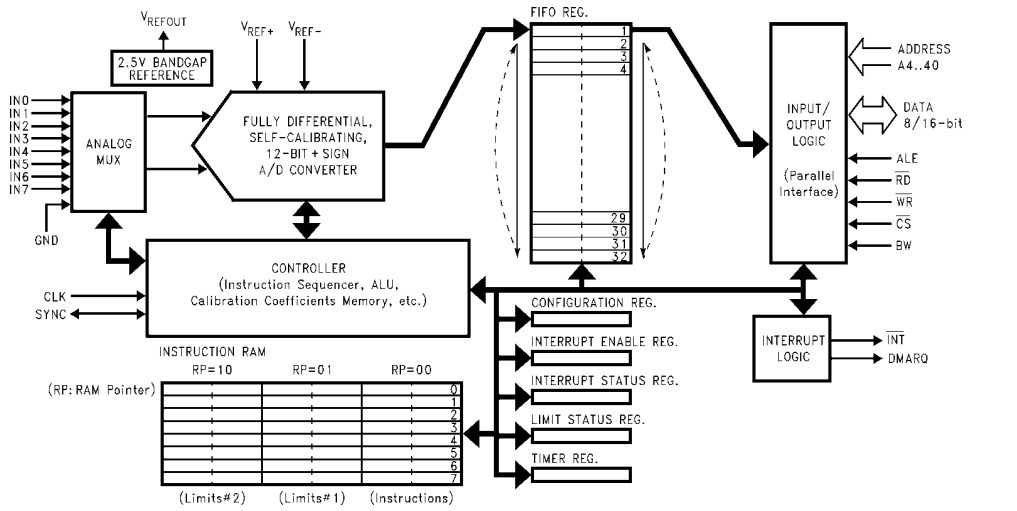
- A flexible analog multiplexer with differential output. The inputs can be programmed with any input referenced to any other input or to ground.
- A differential, self-calibrating 12-bit + sign A/D converter.
- A 32-word by 16-bit FIFO as the output data register.
- An instruction RAM that can be programmed for a single execution or to repeatedly perform a series of conversions and/or comparisons on the selected input channels.
- A series of registers for overall control and configuration of the DAS operation and indication of internal status.
- Interrupt generation logic to request service from the processor under specific conditions.
- Parallel interface logic for input/output operations between the DAS and the processor. All 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 various internal blocks, including performance of the conversion, comparison, and calibration sequences.

The INSTRUCTION RAM is divided into 8 separate words of 48 (3x16) bits. Each word consists of three 16-bit sections, each with a unique address. The three different sections of each word are selected by a 2-bit RAM Pointer (RP) for read/write operations. The RP is part of the Configuration Register.

Figure 1 shows the Instruction RAM sections labeled as **Instructions, Limits # 1,** and **Limits # 2**. The instruction section holds operational information such as the input channels to be selected, the mode of operation of that instruction, and how long the acquisition time should be. The Limits sections are used in the watchdog mode and hold user-defined limits. The watchdog limits are usually one high and one low limit, but it is possible to program two low or two high limits.

The DAS begins executing from Instruction 0 and continues executing subsequent instructions up to any user specified instruction, where it ''loops back'' to Instruction 0 or pauses, depending upon user programming. Not all 8 instructions need to be executed. The cycle may be repeatedly executed until stopped by the user, until the FIFO is full, or until the FIFO holds a user programmed number of conversion results.

The user should access the Instruction RAM only when the sequencer is stopped.



TL/H/12068–1

FIGURE 1. DAS Functional Block Diagram, Programming Model

The FIFO Register is used to store the results of the conversions. This register is ''read only'' to the user 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 0 after a write to location 31. A similar flow occurs during FIFO reads. The internal FIFO writing and the external FIFO reading do not affect each other's pointers.

The CONFIGURATION Register is the main ''control panel'' of the DAS. Writing data to the Configuration Register tells the DAS to perform operations 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 RAM sections.

The INTERRUPT ENABLE Register allows the user to activate any or all of the 8 interrupt sources. It also holds a user programmable value to indicate the number of conversions to be stored in the FIFO before a data ready interrupt is generated, and a user programmable value to indicate which instruction will generate an interrupt just before that instruction is to be executed.

The INTERRUPT STATUS and LIMIT STATUS Registers are ''read only'' and are used to indicate which conditions have generated the interrupt and what limits have been exceeded. The appropriate bits are set upon occurrence of their corresponding interrupt conditions whether or not that interrupt is enabled for external interrupt generation. All interrupt bits in the Interrupt Status Register are cleared whenever this register is read or a device reset is issued. The Limit Status Register is likewise cleared whenever it (Limit Status Register) is read or a device reset is issued.

The TIMER Register is used to insert a delay before execution of any selected instruction(s). This can be useful for reducing the generation of redundant data when converting slowly changing signals.

Appendix A shows the DAS accessible registers and a brief description of their bit assignments. Appendix A also has empty register models that can be used as a programming tool and for design documentation.

2.2 Programming Procedure

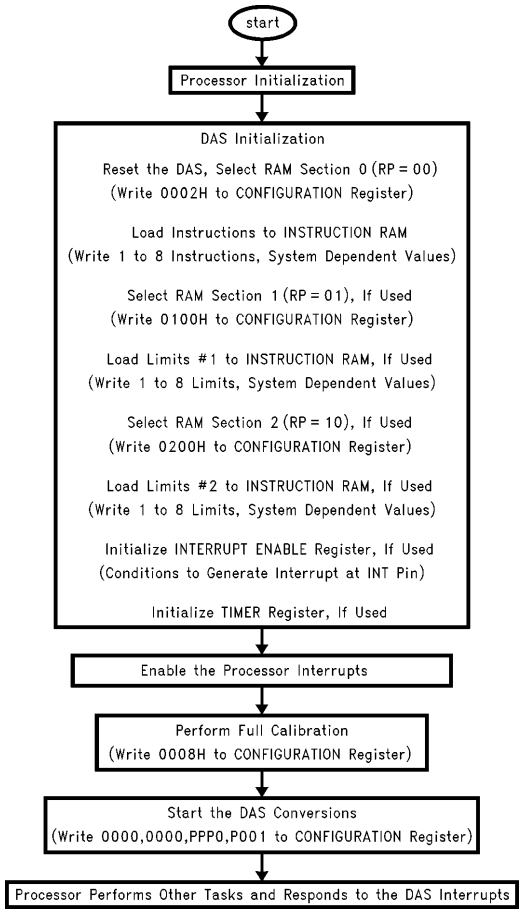
Defining a general programming procedure is not practical due to the high flexibility of the DAS and the broad variety of possible applications. However, the following typical procedure demonstrates the basic concepts of DAS programming:

- 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 to program the first set of watchdog limits.
- Load limits #1, 1 to 8 values (if used).
- Select RAM section ''10'' (if used) through the Configuration Register to program the second set of watchdog limits.
- Load limits #2, 1 to 8 values (if used).
- Initialize the Interrupt Enable Register by selecting the conditions to generate an interrupt at the $\overline{INT}$ pin (if interrupt 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 at the same time, as required.

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

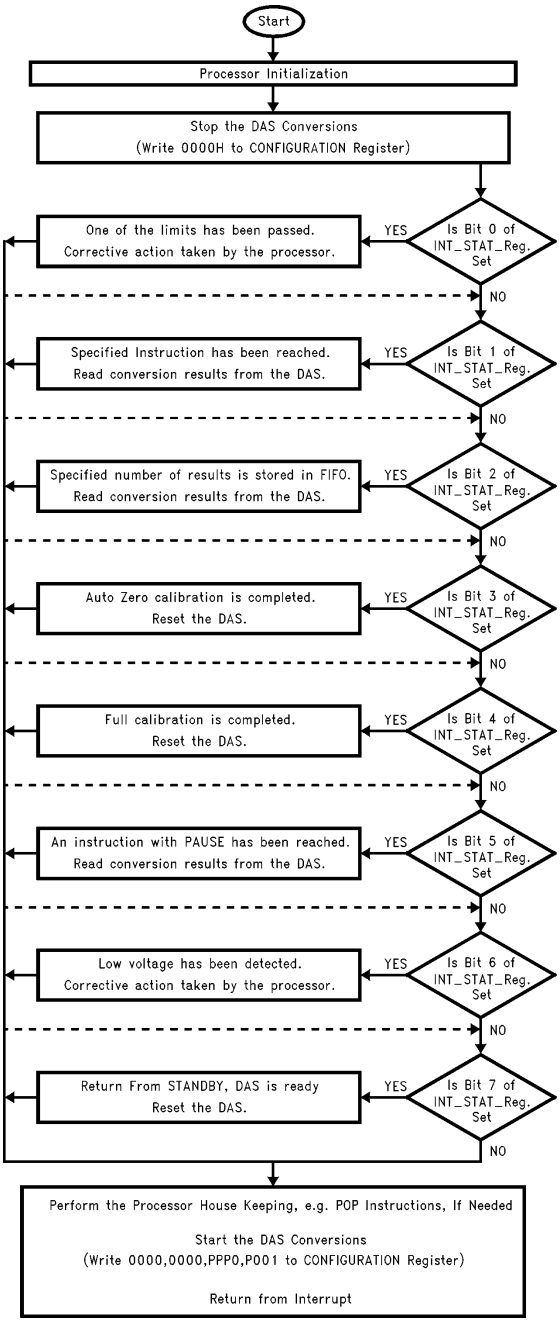
2.3 A Typical Program Flowchart

Figure 2 shows a typical DAS program flowchart as applied to the DAS. *Figure 2a* shows initialization of the DAS and the start of conversions, while *Figure 2b* illustrates one general form of the DAS interrupt service routine. It is assumed here that the processor reads the DAS registers and takes appropriate action only upon receiving an interrupt from the DAS, freeing the host processor for other tasks until an interrupt is generated.



TL/H/12068–2

FIGURE 2a. Typical Program Flow Chart for DAS Initialization and Sequencer Start



TL/H/12068–3

FIGURE 2b. Typical Program Flow Chart for DAS Interrupt Service Routine

The Processor Initialization step is used to define addresses, memory space and values, as well as any other system initialization needed for the application.

After the DAS is initialized by writing the appropriate information to it, the processor interrupts that will be used are enabled and a full calibration cycle is ordered for the DAS, which is required for 13-bit accuracy. One full calibration may be done at power up with or without calibrations at specified time intervals or upon occurrence of given conditions. A full calibration cycle takes about 1 ms with a 5 MHz clock, or about 0.6 ms with an 8 MHz clock. A delay can be inserted into the processor code after initiating a calibration cycle to allow for the calibration to be completed before starting the DAS sequencer, the processor can wait for a ''calibration complete'' interrupt from the DAS as is done with the example given in this Application Note, or the DAS' Interrupt Status Register can be read for the corresponding flag bit.

The full calibration cycle affects some of the DAS' internal flags and pointers, which will influence the execution of the first instruction after completion of calibration, so the DAS should be reset after a calibration cycle to avoid false instruction execution.

The sequencer is started by writing a ''1'' to the START bit in the Configuration Register. The bits shown as ''P'' (Program) in the flow chart are user defined and determine the different modes of operation during conversions. These will be discussed later. All other bits should be set as shown. Note that words of 16 bits are shown in the flow chart, and that the 80C51 family of microcontrollers will transfer only 8 bits of data at a time.

At the start of the interrupt service routine *(Figure 2b)*, a zero is written to the START bit in the Configuration register to stop conversion. This is not necessarily needed unless required for accuracy or timing purposes. Results of conversions can be noisier and less accurate if reads and/or writes of the DAS are performed while it is converting. The degree of any resulting inaccuracy depends upon many aspects of system design and is not easily quantified. However, 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 that need to be addressed.

Whenever the $\overline{CS}$ line of the DAS is low, as when reading or writing to the DAS, *the internal clock is gated off* to stop internal bus activities. This is done to prevent internal conflicts. External reads and writes are asynchronous to internal bus activities. This pause of the internal clock will increase the total acquisition plus conversion time for the interrupted instruction. The amount of this time increase is variable and is not easily predicted because the processor and DAS are not synchronized with each other.

Sometimes it is critical to maintain equal time intervals between successive readings of the same input, as when performing an FFT upon the data. There are two ways to insure that the time intervals between conversions remain constant. The first is to avoid reading the DAS while conversions are being performed. This could be difficult if a lot of data is being gathered as the FIFO has a 32 word limit.

The other way to insure equal time intervals between successive readings of the same input is to use the SYNC input of the DAS. To properly use this input, the I/O bit of the Configuration Register (bit 7) must be set, as must be the SYNC bit (bit 8) of the Instruction RAM. When this is done, the input is acquired at the rising edge of the SYNC input, and conversion begins at the rising edge of the next clock (CLK) input.

There are a couple of concerns when doing this. The first is that the SYNC signal period is long enough to allow execution of all programmed instructions. Second, there must be enough time after completion of one sequencer cycle for the processor to access the DAS and read the registers before another rising edge of the SYNC input. This second requirement, of course, is the most stringent and incorporates the first. Appendix B gives some insight into using the SYNC input.

The main task of the interrupt service routine is to read the DAS' Interrupt Status Register and test its interrupt bits for the source of the interrupt, followed by the appropriate application dependent action. The sequence of bit tests in a given system would depend upon the priority level of the interrupts in that system. Also, actual systems may not use all of the interrupts, so the extra bit tests may be eliminated from the routine. The actual tasks to be performed for each interrupt are system related, so cannot be specified here.

2.4 The DAS/Processor Interface

The 80C51 family of 8-bit microcontrollers is a popular one with a still growing product line. Reference to the 80C51 in this application note is intended to also include all derivatives, as well as the NMOS versions.

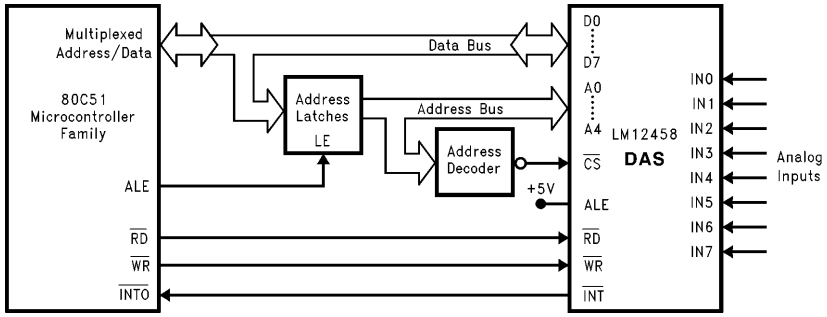
The interface between the processor and the DAS is similar to a memory or I/O interface; the 80C51 sees the DAS as a group of I/O registers with specific addresses. Some of the possible DAS/80C51 interface schemes are shown in *Figures 3*, *4* and *5*.

Figure 3 shows a generic interface scheme for the 80C51. This is a maximum system scheme, assuming that the microcontroller is accessing other peripherals or memory in addition to the DAS. In this case external address latches and an address decoder are required to select the DAS, other peripherals, or memory. The exact configuration of the address decoder, of course, would depend upon individual system requirements.

The DAS can interface with both multiplexed and non-multiplexed address/data bus architectures. The DAS' 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 rest of the system. *Figure 4* indicates how this might be implemented.

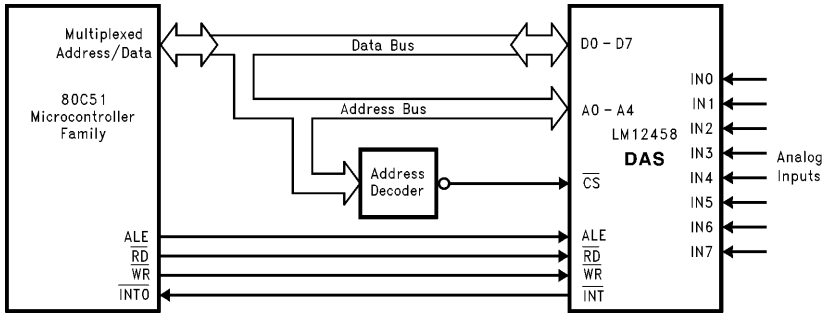
In smaller systems it is possible to use a simpler address decoding scheme, such as shown in the example of *Figure 4*, where all signals but a Chip Select come directly from the 80C51. The DAS latches the $\overline{CS}$ input signal with the ALE, so a higher order address bit may be used to drive the $\overline{CS}$, reducing the circuit to the ultimate in simplicity, as indicated in *Figure 5*. This does, however, limit the total available external address space.

The DAS can be accessed in either 8-bit or 16-bit data width by using the BW (Bus Width) pin of the DAS to select 8-bit or 16-bit access mode. Since the 80C51 is limited to 8-bit data transfers, the BW pin is tied high for 8-bit access when used with this family, and address line A0 selects the lower or upper byte of a 16-bit register. As shown in *Figure 6*, the DAS appears to the processor as a group of 28 separate 8-bit I/O locations. In 16-bit systems, the BW pin would be tied low, and address line A0 would be a ''don't care''. Note that each set of eight 16-bit Instruction RAM words has the



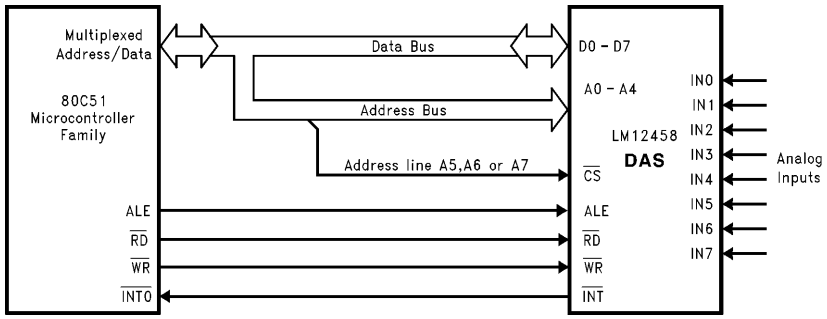
TL/H/12068–4

FIGURE 3. LM12458 to 80C51 Microcontroller Interface (Maximum System)



TL/H/12068–5

FIGURE 4. LM12458 to 80C51 Microcontroller Interface (Simplified System)



TL/H/12068–6

FIGURE 5. LM12458 to 80C51 Microcontroller Interface (Minimum System)

same address. They are differentiated by the setting of the two-bit RAM pointer in the Configuration Register.

The interface should provide the address, data and control signals to the DAS with any needed address decoder to generate an appropriate chip-select signal for the DAS. It should also provide the proper timing relationship between ALE, $\overline{CS}$, $\overline{RD}$, $\overline{WR}$, address bus and data bus to satisfy the DAS timing requirements, as specified in the data sheet.

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 the processor's interrupt input ($\overline{INT0}$ or $\overline{INT1}$ of the 80C51), as indicated in *Figures 3, 4* and *5*, or interrupt arbitration logic can be used in systems that have many I/O devices.

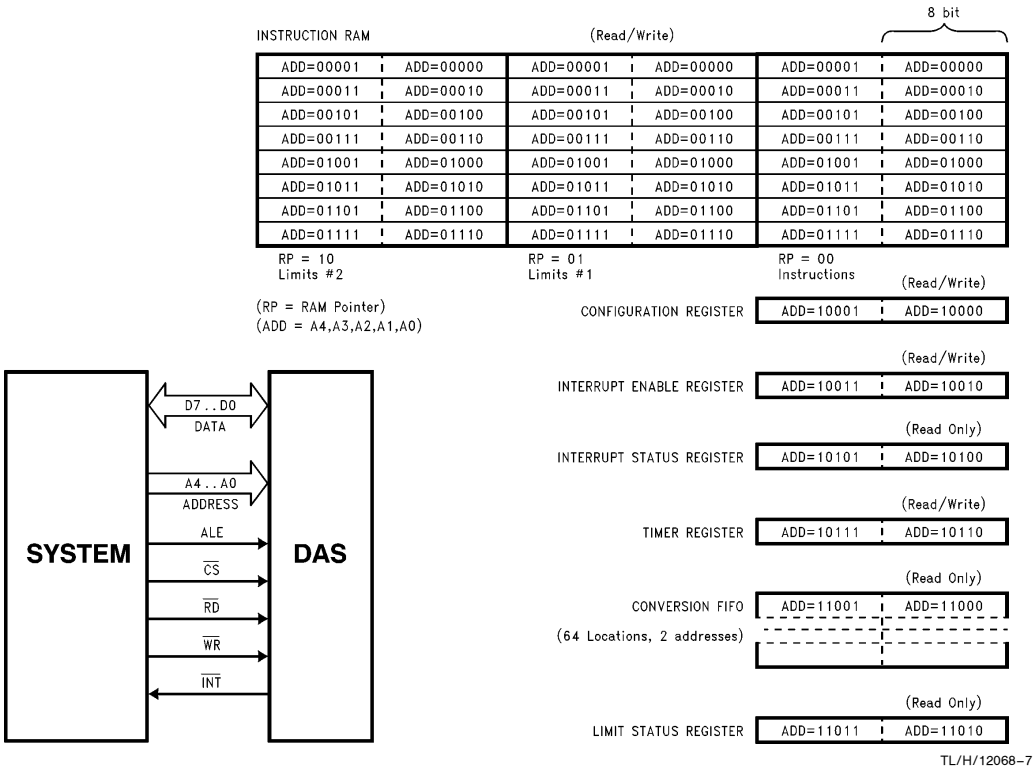


FIGURE 6. DAS Registers, Address Assignments, Interface Buses and Control Signals for 8-Bit Bus Width

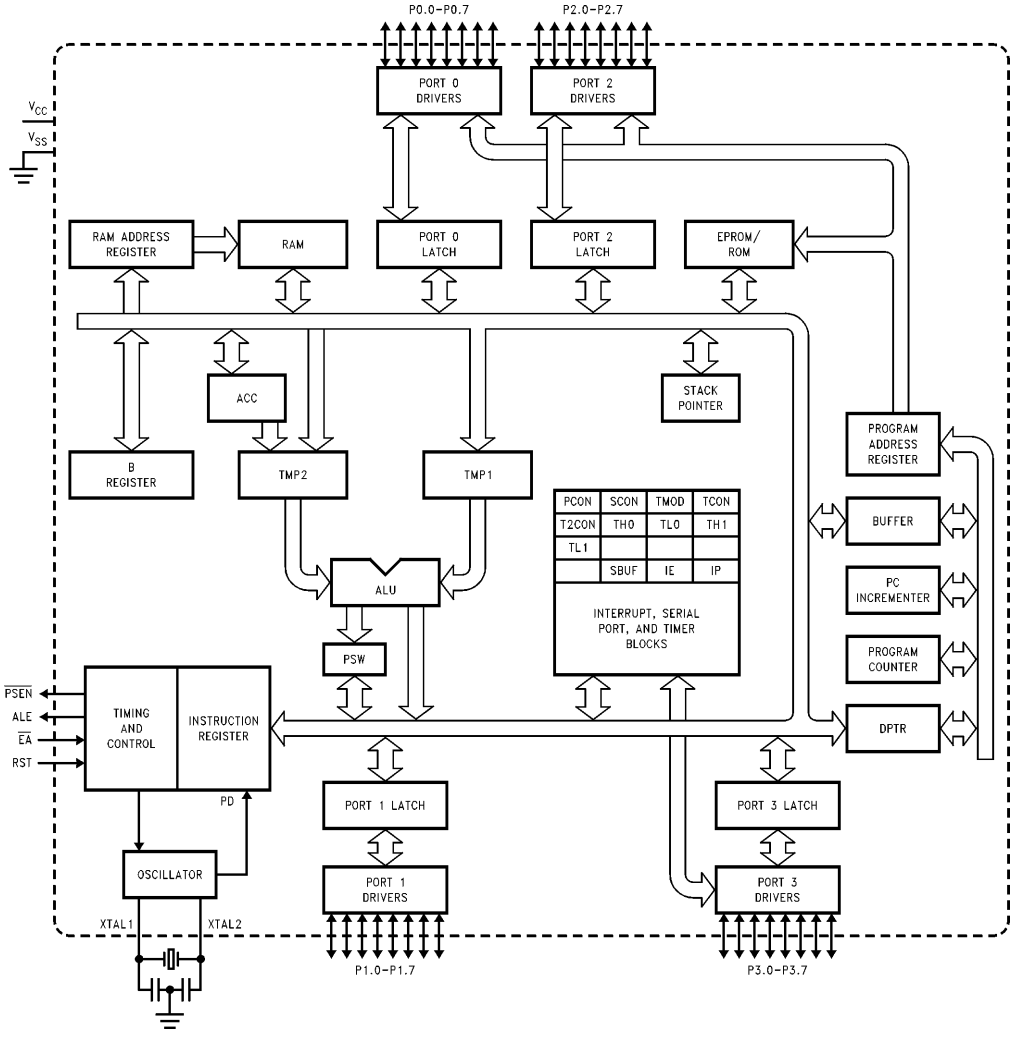
3.0 THE 80C51 MICROCONTROLLER FAMILY

The architecture of the 80C51 family of microcontrollers has been optimized for sequential real time control applications. The family includes versions that have RAM, ROM or EPROM, or CPU only. The majority of the devices in the family can address up to 64 kbytes of program memory, plus 64 kbytes of data memory. Features of these microcontrollers include:

- 8-bit CPU
- Single-bit logic capabilities
- 32 bi-directional, individually addressable I/O lines
- On-board data RAM
- 64 kbytes program memory address space
- Full duplex UART
- On-chip clock oscillator
- On-chip program memory
- 5-source interrupt structure with 2 priority levels
- 16-bit timer/counters
- 64 kbytes data memory address space
- Broad choice of packages

Versions are available that operate at very low supply levels (down to 1.5V), incorporate I^2C (serial bus) interface, have extended I/O, watchdog timer, up to 3 counter/timers, A/D converters, analog comparators, CAN (Control Area Network) bus interface, DMA to on-chip RAM, EEPROM, and security features.

The basic architecture of the 80C51 is shown in *Figure 7*.



TL/H/12068–8

FIGURE 7. 80C51 Architecture

3.1 80C51 Memory Organization

All 80C51 derivatives have separate address space for program and data memory, as shown in *Figure 8*. This logical separation of data and program memory space allows both memory areas to be addressed with 8-bit addresses. However, 16-bit data memory addresses can be generated by using the 16-bit DPTR register.

Although up to 64 kbytes can be addressed, on-board program memory of the 80C51 itself is 4 kbytes, with zero to 32 kbytes available in various family derivatives. The read strobe for external program memory is the $\overline{\text{PSEN}}$ (Program Store Enable). The entire 64k program memory area can be addressed off-chip. So, in the 80C51, with 4 kbytes of ROM (or the 87C51 with 4 kbytes of EPROM), we can have 64 kbytes of program memory off-chip, plus 4 kbytes on-chip. Care must be exercised, however, in switching between on-board and external program memory. The $\overline{\text{EA}}$ (External Access) pin is used to determine whether external or internal program memory is being used. Note that only those derivatives with more than 128 bytes of RAM have the indirectly addressable data memory above address 7FH.

Data Memory (RAM) occupies a separate address space from Program Memory. In the 80C51, the lower 128 bytes of data memory are on-chip, and up to 64 kbytes of external RAM can be addressed. Derivatives have from zero to 512 bytes of RAM.

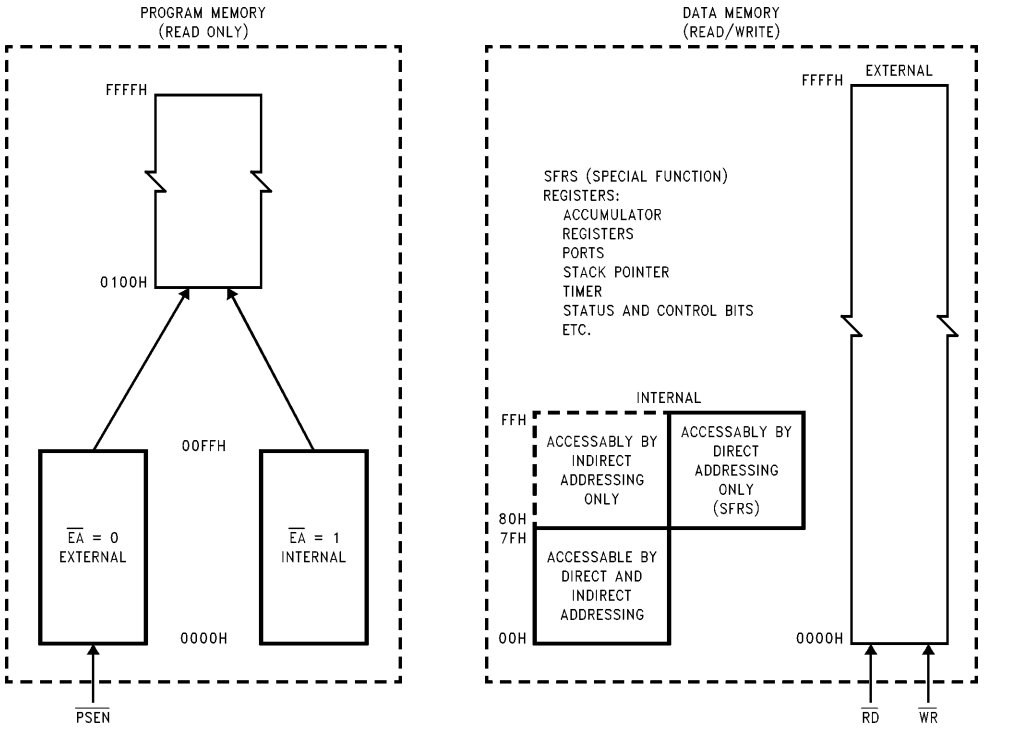


FIGURE 8. 80C51 Memory Structure

3.1.1 Program Memory

The lower portion of Program Memory contains the locations from which execution begins upon reset or the occurrence of an interrupt. After reset, the CPU begins execution from location 0000H, and each interrupt is assigned a location from which execution begins upon invocation of that interrupt. There are 3 bytes allowed for reset programming, so these should be used to cause a jump to the start of the program, which should be above the interrupt vector locations. Eight bytes are allowed for each interrupt execution location. These lower bytes are assigned as follows:

0000H	Reset
0003H	Interrupt 0
000BH	Interrupt 1
0013H	Interrupt 2
001BH	Interrupt 3
0023H	Interrupt 4

An interrupt causes the CPU to begin executing from the above indicated location. If any given interrupt is not being used, its service location is available as general purpose Program Memory. These service locations are located at 8 byte intervals. If a given interrupt service routine is short enough, it can be located within these 8 bytes. Longer service routines can use a jump instruction to skip over subsequent interrupt service locations that are in use.

The lowest 64 bytes to 4 kbytes (whatever on-chip ROM is present) of Program Memory can be either in the on-chip ROM or in external ROM. This selection is made by connecting the $\overline{EA}$ pin either to V_{CC} or V_{SS} (Ground). With this pin tied high, program fetches from addresses 0000H through 0FFFH are from internal ROM, while program fetches from 10000H through FFFFH are directed to external ROM. With the $\overline{EA}$ pin tied low, all program fetches are from external ROM.

A general hardware configuration to execute from external Program Memory is shown in *Figure 9*. Note that 16 I/O lines are used for bus functions during external Program Memory access. Port 0 (P0) is a multiplexed address/data bus, and carries the low byte of the Program Counter (PCL) as an address, and then goes into a float state awaiting the arrival of the code byte from the Program Memory. While PCL is valid on Port 0, the ALE (Address Latch Enable) signal clocks this byte into an address latch. Meanwhile, Port 2 (P2) carries the high byte of the Program Counter (PCH), $\overline{PSEN}$ strobes the Program Memory, and the code byte is read into the microcontroller.

External Program Memory addresses are always 16 bits wide, even though the actual amount of Program Memory may not require the full 16 bits. External program execution sacrifices two of the 8-bit ports, P0 and P2, to the function of accessing Program Memory.

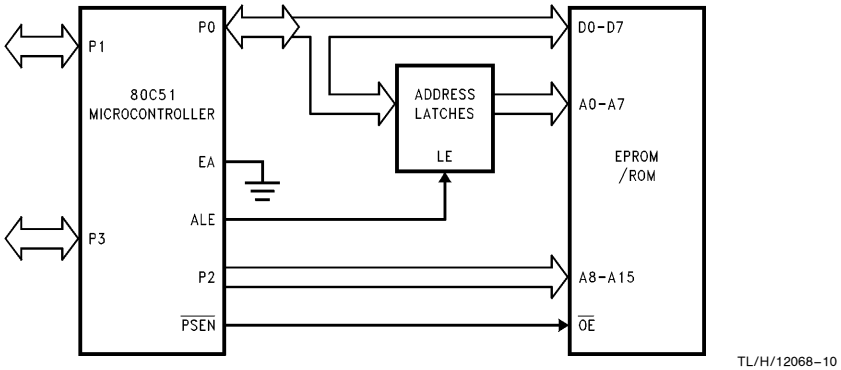


FIGURE 9. A General Hardware Configuration to Execute from External Program Memory

3.1.2 Data Memory (RAM)

A complete discussion of 80C51 data memory would include both internal and external RAM, but the discussion here will be limited to internal RAM. Also, since this discussion of data memory is centered around the 80C51,this discussion is limited to those derivatives which, like the 80C51, have 128 bytes of RAM.

The Lower 128 bytes of RAM are shown mapped in *Figure 10*. The SFRs (Special Function Registers) are mapped to the 128 bytes of memory space immediately above the first 128 bytes, starting at address 80H. Those devices with more than 128 bytes of RAM share this second 128 bytes with the SFR memory space, but the SFRs can only be **directly** accessed, while memory in this space can only be **indirectly** accessed.

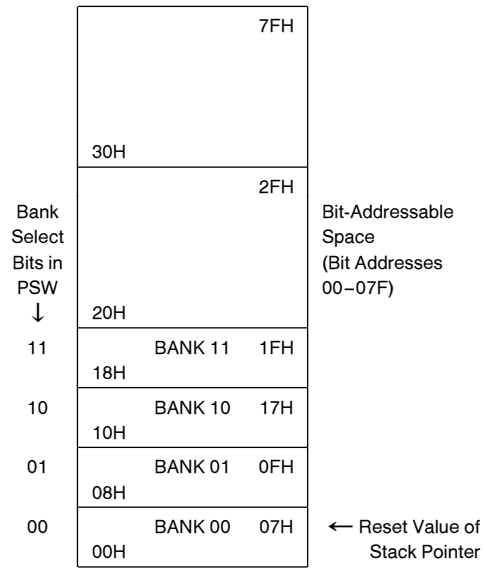


FIGURE 10. Lower 128 Bytes of Internal RAM

Figure 11 details the 80C51 RAM organization. Care must be taken not to use RAM locations that correspond to the register banks that will be used. Each of these register banks contains eight registers, referred to as Register R0 through R7 in program instructions. These registers can be used for relative addressing, as counters, and for temporary data storage. Two bits in the PSW (Program Status Word) select which register bank is in use.

The next 16 bytes above the register banks is a block of bit-addressable memory space. If the program uses only Register Bank 00, then memory locations 08H through 1FH also may be used for data storage. In the program detailed here, data memory locations 40H through 5DH are used to store the conversion results read from the DAS. Should more RAM be required than is available in the 80C51 for a given application, a member of the 80C51 family with more RAM should be selected (the 80CE558 has 1024 bytes of RAM), or up to 64 kbytes of external RAM may be used. *Figure 11* shows RAM organization as used by the example program we will develop here.

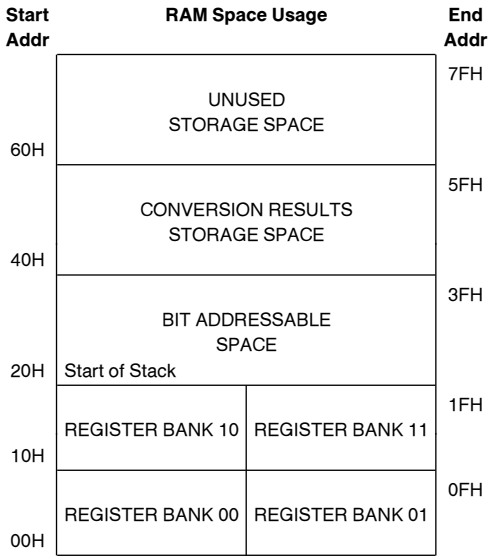


FIGURE 11. 80C51 RAM Organization as Used by Example Program

3.2 80C51 Hardware Description

The 80C51 maps 21 registers onto the upper portion of Data Memory (RAM) in what is called SFR Memory. Those registers with the three least significant address bits equal to zero (addresses x0H and x8H) are bit addressable. *Figure 12* indicates how these Special Function Registers are mapped onto RAM. Note that, while the SFR memory map of *Figure 12* has many ''holes'' in it, these are not usable data storage locations as nothing is located in these areas. The SFRs are not actually part of memory, but are mapped onto memory.

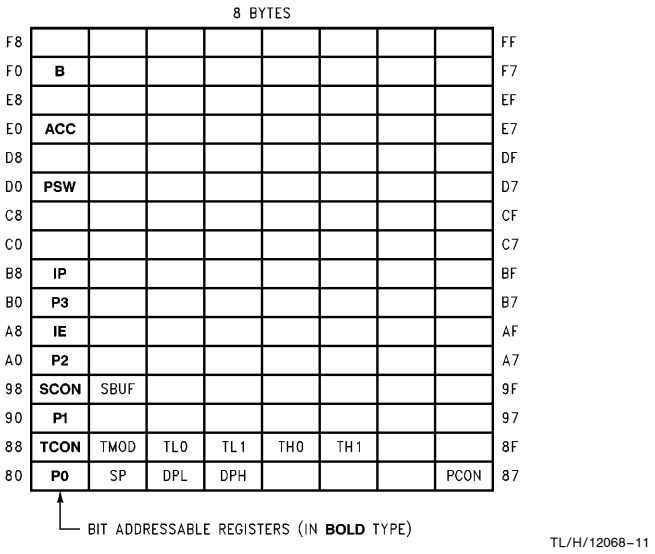


Figure 12. 80C51 Special Function Register (SFR) Map

Most members of the 80C51 family use a dynamic CPU, meaning that node capacitances are used for temporary, dynamic storage within the CPU. For this reason, the CPU can ''forget'' what it is doing if the clock rate is too slow. Dynamic nodes are used to reduce transistor count, decrease die size, and provide a more economical device. The on-chip RAM, however, is fully static; it is only the CPU that is dynamic. Minimum clock frequency is individual product dependent, but is 3.5 MHz for most of the dynamic CPU devices in the family.

The 80(C)31 is ROMless, while the 80(C)51 contains a mask programmable ROM, and the 87C51 has EPROM Program Memory. For the derivatives of the family, the 80Cxx devices are ROMless, the 83Cxx devices have mask programmable ROM, and the 87Cxx devices have EPROM Program Memory. Many of the EPROM versions are available as OTP. The 80C51 family **Special Function Registers** are described below.

The **ACC** is the **Accumulator**, also referred to as, simply, **A**.

The **B Register** is used during multiply and divide operations. For other instructions it can be used as a scratch pad register.

The **Program Status Word (PSW)** register contains program status information, including the carry flag, an auxiliary carry flag (for BCD operations), an overflow flag, two user-defined flags, two register bank select bits, and a parity flag.

The **Stack Pointer (SP)** register indicates the last location to hold information saved with the PUSH instruction, and can reside anywhere in on-chip RAM. It is initialized to 07H after reset, causing the stack to begin at 08H, which is where Register Bank 01 is located. If it is desired to have the stack at some other location, it should be initialized early in the program and before a PUSH instruction is executed.

The **Data Pointer (DPTR)** consists of a high byte (DPH) and a low byte (DPL). Its intended function is to hold a 16-bit address and can be manipulated as a 16-bit register (DPTR) or as two independent 8-bit registers (DPH and DPL).

Ports 0 to 3 are the SFR latches of the respective ports. When used as an input, the external state of the port pin will be held in the port SFR.

The **Serial Data Buffer (SBUF)** is used for serial data transfer (UART operations).

The **Timer Registers** are in pairs (TH0, TL0 and TH1, TL1) that form 16-bit counting registers for Timer/Counters 0 and 1, respectively.

The **Interrupt Priority (IP)** register is used to define high/low priority for each of the five interrupt sources. The other three bits are reserved and should not be written to. The five priority sources have preset priority (see the 80C51 data sheet).

The **Interrupt Enable (IE)** register is used to enable or mask the interrupt sources, as desired.

The **Timer Mode Control (TMOD)** register is used to set the counter/timers to count or time, enables the counters/timers, and selects one of 4 operating modes. Each counter/timer can be set independently of the other.

The **Counter/Timer Control (TCON)** register carries the counter/timer overflow flags, run control bits, interrupt flags and control bits.

The **Serial Port Control (SCON)** register contains enabling bits and mode control bits for serial communications.

The **Power Control (PCON)** register is mainly intended to put the 80C51 into one of its power saving modes, but also contains a Double Baud rate bit for use with serial communications, two general purpose flags and three reserved bits.

3.3 OTHER 80C51 FEATURES

Other features of the 80C51 family include Power-On Reset capability, two power saving modes of operation (Power Down and Idle), ONCE (ON-Circuit Emulation) mode, and an on-chip oscillator.

A **Power-On Reset** may be obtained upon application of power by connecting the RST pin to V_{CC} through a 10 μF capacitor, and to V_{SS} through an 8.2k resistor, as long as the V_{CC} rise time is no longer than 1 ms and the oscillator start-up time is no longer than 10 ms. The CMOS devices do not require the 8.2k resistor, but its presence does no harm.

The 80C51 family has two **Power-Saving Modes** of operation. The **Power Down** mode is invoked when the PD bit in the PCON register is set. In this mode the oscillator is disabled, but the contents of the on-chip RAM and SFRs are maintained and the port pins send out the values held by their respective SFRs. The ALE and $\overline{PSEN}$ outputs are held low.

V_{CC} can be reduced to as low as 2V only after the **Power Down** mode is invoked, but V_{CC} must be restored to operating levels before this mode is terminated. The only recovery from Power Down is a hardware reset, which redefines all the SFRs, but does not change on-chip RAM.

The **Idle** mode is invoked when the IDL bit in the PCON register is set. In this mode the on-chip oscillator continues to run, but the clock signal to the CPU is gated off. The Serial Port, Interrupt, and Timer blocks continue to be clocked, and the CPU status is preserved in its entirety. The ALE and $\overline{PSEN}$ pins hold at logic high levels.

The Idle mode can be terminated in two ways. Activation of any enabled interrupt will clear the IDL bit and cause the Interrupt to be serviced. Following the RETI (Return from Interrupt) command, the next instruction to be executed will be the one following the instruction that put the device into idle. The other way to terminate the Idle mode is with a hardware reset. At this time the CPU will resume execution from where it left off (instruction following the one that set the Idle mode). Three NOP instructions are recommended following the instruction that invokes the idle mode.

The **ONCE (ON-Circuit Emulation)** mode eases testing and debugging of systems without having to remove the processor from the circuit. This mode is invoked by pulling the ALE pin low while the device is in reset and $\overline{PSEN}$ is high, then holding ALE low as RST is deactivated. In this mode, the Port 0 pins float in a high impedance state and the ALE and $\overline{PSEN}$ outputs are weakly pulled high and the oscillator remains active. While in this mode, an emulator or test CPU can be used to drive the circuit. Normal operation is restored after a normal reset is applied.

The **On-Chip Oscillator** in the NMOS members of this family is a single stage linear inverter, with the crystal used in its fundamental response mode in parallel resonance with external capacitance. The external capacitors are not critical, 20 pF–30 pF being suitable with good quality crystals. A ceramic resonator can be used in place of the crystal, with the associated capacitors being, typically, about 50% higher in value.

The On-Chip Oscillator in the CMOS members of this family is very similar to that of the NMOS members, but there are some important differences.

One difference is that the CMOS oscillators can be turned off under software control (by writing a 1 to the PD bit in the PCON Register). Another is that the internal circuitry is clocked by the signal at EXTAL1 in the CMOS versions, and by the signal at XTAL2 in the NMOS versions.

4.0 THE 80C51 HARDWARE INTERFACE

In this section we will detail the development of the interface circuit and software for using the 80C51 (MCS™-51) family of microcontrollers with the DAS. The 80C51 family is available in many versions suitable for a wide variety of applications. The reader is encouraged to refer to the many 80C51 suppliers for derivatives available and for complete information and specifications. Where reference is here made to the 80C51, all products in the family are included, including both the NMOS and CMOS versions of them, unless otherwise noted or indicated.

Since there are many different devices in the 80C51 family with differing pin configurations, no attempt is made on schematics in this section to show connections to all pins of the microcontroller. Device pin numbers shown for the microcontroller are those for the 80C51; other members of the 80C51 family may have different pin numbers for the same function.

Internal program memory will be used in these examples so the 80C51's $\overline{EA}$ pin is tied high, and no external program memory is used. The DAS is mapped to the lower 28 bytes of external data memory.

Interrupt handling is accomplished by the 80C51's vectored interrupt scheme. There are five possible sources of interrupts, any or all of which are maskable. The external interrupts, $\overline{INT0}$ and $\overline{INT1}$, can be programmed to be either level sensitive or transition activated. Transition activation is used in this example because the interrupt flag is cleared when the service routine is vectored to only when transition activation is used freeing our programming from the need to clear this flag. Using transition activation would be a disadvantage only if we enabled more than one interrupt source because the 80C51 interrupt flag is cleared as program control is transferred to the service routine, preventing us from determining what generated the interrupt. In this example, however, only $\overline{INT0}$ (where the DAS' $\overline{INT}$ output is connected) is enabled, so we know that any interrupt must have been generated by the DAS.

Two different interface circuits are presented in *Figures 13* and *14*. The circuit of *Figure 13* uses complete address decoding with the external address latches, as would be needed if the processor would be accessing other devices and/or external memory. A circuit similar to that of *Figure 14* would be used where only a few peripherals and/or a small amount of other external memory would be used such that no more than 64 kbytes of external memory (mapped) space were needed. If the DAS were the only device in the circuit, the address decoder could be eliminated and the $\overline{CS}$ input to the DAS could be connected directly to one of the higher order address pins (ADD5, ADD6, or ADD7). Of course, there are many other address schemes that could be used. These are shown only as examples.

The circuits of *Figures 13* and *14* also show a method of using the internal oscillator of the processor to drive the clock input of the DAS. When this is done, the DAS cannot function when the 80C51 is in the power down mode since the oscillator does not function in this mode. The Idle mode of power saving may be used to reduce power consumption while the DAS is running, however.

Because the 80C51 is an 8-bit machine with an 8-bit wide data bus, we will only be able to transfer 8 bits of information to and from the DAS' 16-bit registers at a time, so the DAS' ''BW'' pin (pin 20) will be tied high and data lines D8 through D15 will not be used.

4.1 Complete Address Decoding

Figure 13 details the complete address decoding used to generate the DAS' $\overline{CS}$ signal. The DAS is accessed as memory mapped I/O at the beginning of the external address range (0000H to 001FH), and 8-bit data access is selected for the DAS (''BW'' pin tied high). External address latch **U3** (74HC573) is used to latch the lower address byte from the 80C51's multiplexed 16-bit address/8-bit data lines. An 8-bit magnitude comparator, **U5** (74HC688), decodes the high order address byte (A15. . .A8) by comparing it with the logic input from the address range selected by the jumpers on header JP1, set for 00H in *Figure 13*. When the address is within the correct range, the output of the magnitude comparator enables the 3-to-8 line decoder, **U4** (74HC138). Output Y0 of **U4** is the $\overline{CS}$ signal for the DAS. The processor address lines A0 through A4 are directly connected to the respective DAS address inputs. With this circuit we must use 16-bit (DPTR) addressing, even if we are not using the high address byte.

The $\overline{INT}$ output of the DAS drives the $\overline{INT0}$ input of the 80C51 processor, allowing the DAS to request service when needed. The selection of this input is arbitrary in this case; we could have selected interrupt input $\overline{INT1}$.

4.2 Simple and Minimal Address Decoding

The circuit of *Figure 14* does not use the external address latch of the 8-bit magnitude comparator, or the address setting jumpers. The 3-to-8 line decoder is still used and is permanently enabled. The Y0 output of this decoder still drives the DAS' $\overline{CS}$ input. However, the ALE output of the 80C51 directly drives the DAS' ALE input. The ALE signal latches address and $\overline{CS}$ lines into the DAS at the start of any data transfer cycle. The DAS is still accessed with the same address range of the circuit of *Figure 13*, and all bits of the higher order byte of the address are ''don't cares''—the DAS will be accessed regardless of the value of the high order address byte.

Outputs Y1 through Y7 of the 3-to-8 line decoder (74HC138) can still be used to access other peripherals, but those devices should have internal address latches for the address and $\overline{CS}$ line, as does the DAS.

If the DAS is the only peripheral/memory, or there are no more than two additional devices to be addressed, one of the high order address bits of the least significant address byte (ADD5, 6 or 7) can be used to directly drive the $\overline{CS}$ input of the DAS, eliminating the 3-to-8 Line Decoder entirely, as shown in *Figure 15*.

4.3 Timing Analysis

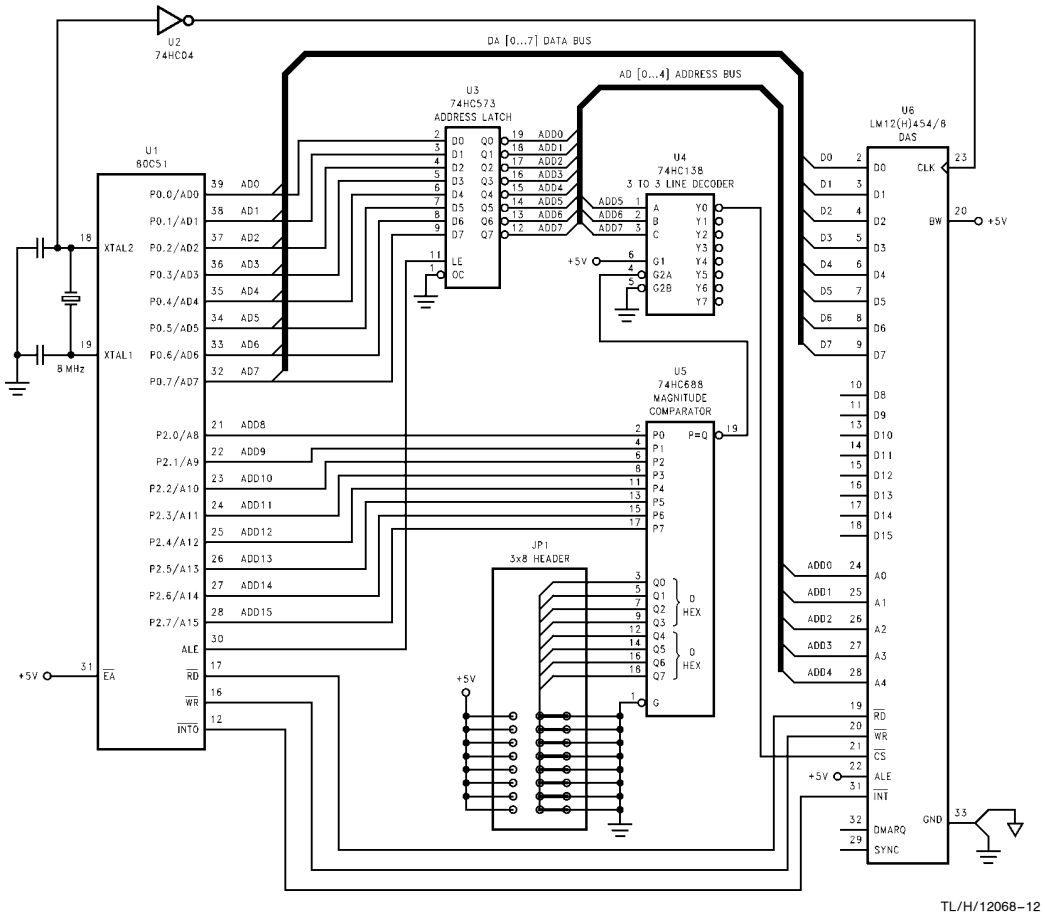
The user should perform a timing analysis as part of the interface hardware design to ensure proper interaction between the processor and the DAS. Each new hardware design should include a comparison of processor timing specifications vs DAS timing requirements. Any timing problems need to be addressed through hardware design or software techniques.

4.3.1 Complete Address Decoding

A study of the switching characteristics of the DAS and the 80C51 with the complete address decoding of *Figure 13* shows that the write cycle timing is more critical than is the read cycle timing.

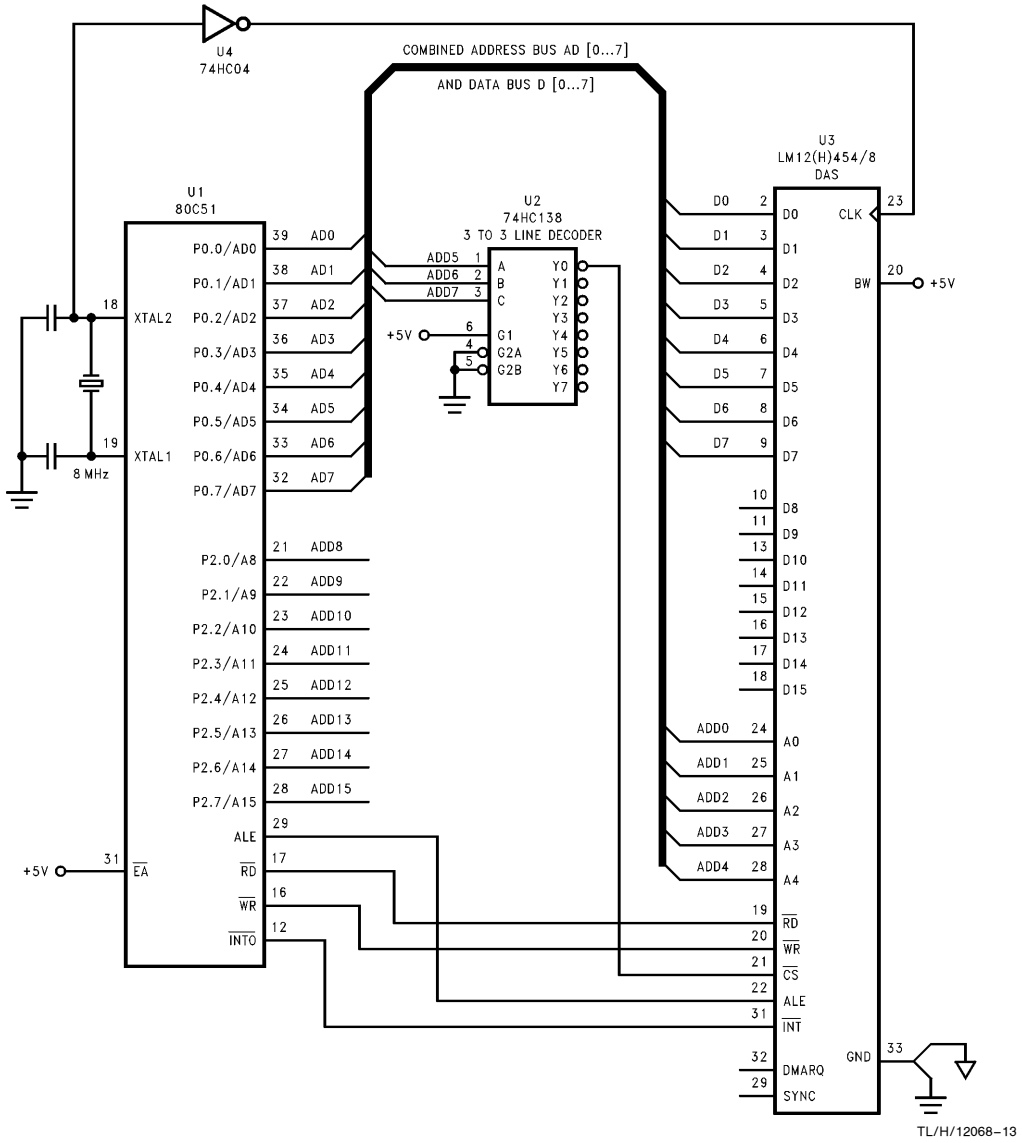
The closest to critical timing for the complete address decoding circuit of *Figure 13* is the Data Valid to $\overline{WR}$ setup time, which has a specification of 40 ns minimum. This requirement is met as long as the processor clock does not exceed 16.6 MHz. Although versions of the 80C51 are available with clock speeds up to 33 MHz, we used a clock speed of 8 MHz in this application because we wanted to use the 80C51 clock to drive the DAS, affording a more economical approach. The LM12H454/8 has a maximum 8 MHz clock specification.

The Data Valid to $\overline{WR}$ setup time is not critical here because the margin between the 40 ns minimum DAS requirement and the 105 ns provided by the 80C51 operating at 8 MHz is quite adequate. Maximum propagation delays considered for the logic devices were those specified at 4.5V supply, 50 pF load and $-40°C$ to $+85°C$ temperature range.



TL/H/12068–12

FIGURE 13. DAS/80C51 Microcontroller Interface (Complete Address Decoding)



TL/H/12068–13

FIGURE 14. DAS/80C51 Microcontroller Interface (Simple Address Decoding)

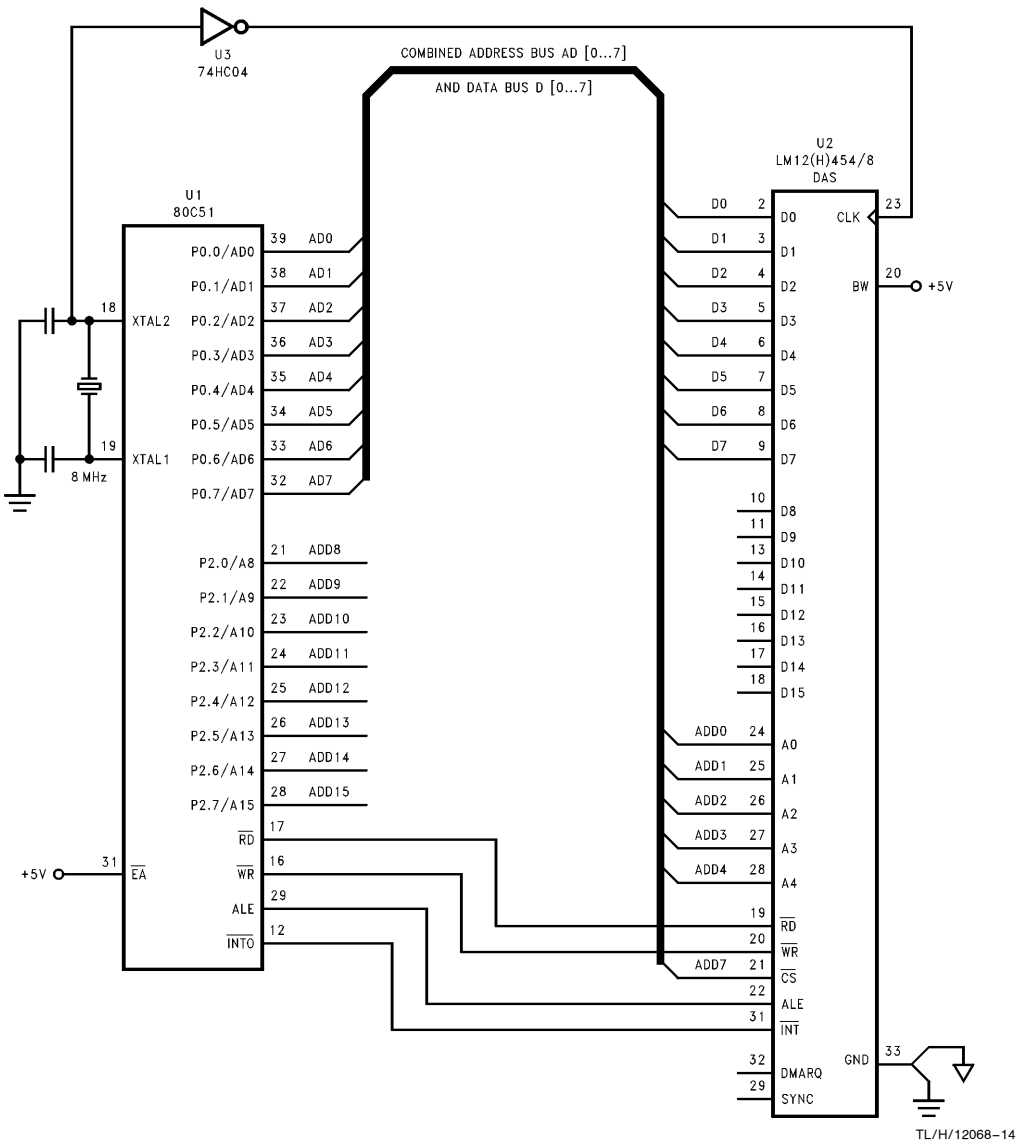


FIGURE 15. DAS/80C51 Microcontroller Interface (Minimal Address Decoding)

4.3.2 Simple and Minimal Address Decoding

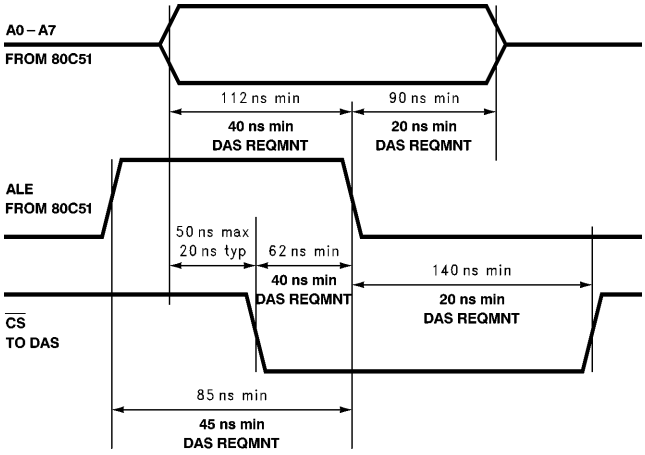
A study of the switching characteristics of the DAS and the 80C51 with the simple address decoding of *Figure 14*, as well as minimal address decoding of *Figure 15* with the 3-to-8 Line Decoder eliminated, shows that the write cycle timing is again more critical than the read cycle timing.

Here the $\overline{CS}$ low to ALE low time is the limiting specification, with 62 ns minimum provided by the 80C51, against a minimum 40 ns required by the DAS. This timing relationship is shown in *Figure 16*. Again, maximum propagation delays considered for the logic devices were those specified at 4.5V supply, 50 pF load and −40°C to +85°C temperature range.

5.0 A System Example: A Semiconductor Furnace

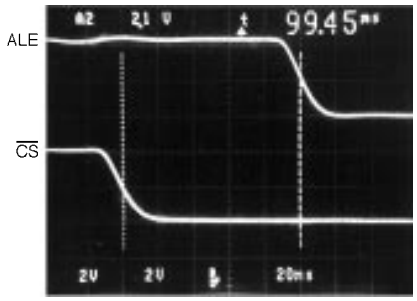
In this example the DAS measures the inputs from five sensors in a semiconductor furnace. This example will use one of the circuits of *Figures 13* through *15* as the data acquisition and control system. We will define the system requirements and, based upon these requirements, the DAS programming will be specified and a typical assembly routine for the 80C51 will be presented for DAS initialization and data capture.

Figure 17 diagrams a typical semiconductor furnace arrangement with sensors to measure gas flow, chamber pressure, and three temperature sensors to measure furnace temperature at each end and the middle of the furnace.



TL/H/12068–15

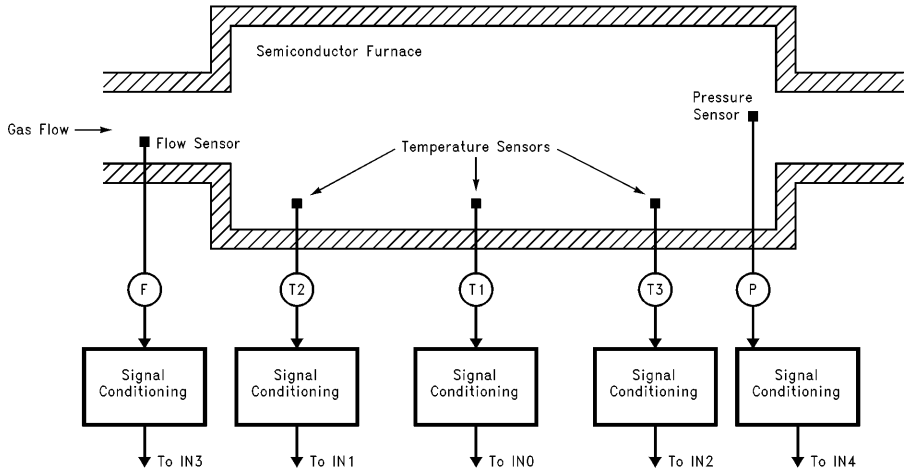
a. Timing Requirements



TL/H/12068–16

b. Actual Timing $\overline{CS}$ Fall to ALE Fall

FIGURE 16. DAS/80C51 Interface Timing (Simple Address Decoding)



TL/H/12068–17

FIGURE 17. Diagram of a Typical Measurement Arrangement in a Semiconductor Furnace

5.1 Assumptions and System Requirements

The following assumptions are made for this system:

- All of the sensor signals are conditioned (gain and offset adjusted) to provide voltage levels within the 0V–2.5V range 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 a 600Ω source impedance.
- The output of the temperature and pressure signal conditioners have output impedances of less than 10Ω.
- The DAS reference voltage is 2.5V ($V_{REF}+ = 2.5V$ and $V_{REF}- = 0V$ = AGND)
- Any of the circuits of *Figure 13* through *Figure 15* may be used for the furnace measurement and monitoring system.
- An approximate throughput rate of 50 Hz is desired for each set of measurement results, providing a set of measurement data about every 20 ms; because of the slowly changing nature of the input signals, precisely controlled throughput rate is not essential for proper system performance.

Control of the furnace requires measurements of the following:

- Temperature T1 at furnace center with 12-bit resolution.
- Temperature T1 relative to T2 with 12-bit + sign resolution.
- Temperature T1 relative to T3 with 12-bit + sign resolution.
- Gas flow, F, with 8-bit resolution and longer acquisition time.
- Chamber pressure, P, with 8-bit resolution.

We also need to monitor the system for three conditions that should produce an alarm:

- Gas flow, F, drops below a minimum limit.
- Gas flow, F, exceeds a maximum limit.
- Pressure, P, exceeds a maximum limit.

5.2 DAS Setup and Register Programming

The conditioned sensor outputs are assigned to the DAS inputs as follows:

- IN0: T1
- IN1: T2
- IN2: T3
- IN3: F
- IN4: P
- IN5: Not used—grounded
- IN6: Not used—grounded
- IN7: Not used—grounded

Seven DAS instructions are needed for measurement and limit monitoring of the system. Five instructions are needed to perform conversions for data collection, and two instructions are used to perform the ''watchdog'' function for comparison of gas flow and pressure against their respective limits.

The following procedures are used for system operation and DAS programming:

- A delay is added before the first instruction (#0) to provide an approximate 50 Hz throughput rate.
- The seven instructions are executed in sequence from 0 to 6 with no added delay between instructions.

- After execution of the last instruction (#6), the DAS loops back to instruction #0 and continues until told to stop by the 80C51 host processor. Each loop is called an instruction loop.
- Each instruction loop generates 5 conversion results. When the FIFO contains 30 data points (6 sets of 5), the DAS generates a FIFO Full interrupt to the 80C51.
- Start bit is not cleared when reading the FIFO. The reading adds extra delay after each six instruction loops. (See Section 2.3 and Appendix B for a discussion on maintaining precise time intervals between successive readings at the same input.

Since the input from the flow sensor has a source impedance greater than 60Ω (recall that it is 600Ω), it requires additional acquisition time for proper settling. The formula from paragraph 2.1 of the DAS data sheet, where Bits 12–15 are discussed, allows us to determine the number to be stored in bits 12 through 15 to insure proper input settling prior to start of conversion. Because we are using 8-bit resolution to measure flow rate, this formula is

$$D = 0.36 \times R_S(k\Omega) \times f_{CLK}(MHz).$$

Substituting the 600Ω (0.6 kΩ) source impedance and 8 MHz clock frequency,

$$D = 0.36 \times 0.6 \times 8 = 1.728$$

and we round up to a value of 2, or 0010 binary.

Now we are ready to specify the contents of the DAS registers.

INSTRUCTION REGISTER

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	$V_{IN}-$			$V_{IN}+$			Pause	Loop

Instruction #0: Measuring T1 (IN0), Single Ended, 12-Bit Resolution, Timer Enabled

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 (IN0) − T2 (IN1), Single Ended, 12-Bit Resolution

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 (IN0) − T3 (IN2), Differential Mode, 12-Bit + Sign

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 (IN3), Single Ended, 8-Bit Resolution, D = 2

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 (IN3), Single Ended, D = 2

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 (IN4), Single Ended, 8-Bit Resolution

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 (IN4), Single Ended, Loop Bit Enabled

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

Instructions #4 and #6 also have limit values; instruction #4 has two limits and instruction #6 has one. These limits are referred to as F__MIN, F__MAX and 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
Address to generate INT1:		0, not used
Programmed number of results in FIFO for INT0:		30 (11110 binary)

Interrupt Enable Register:

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
Number of FIFO results for INT0					Address for INT1			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:

— We will not perform an auto zero or calibration before each conversion (D2 = 0).

— Bits D13 through D15 of the accumulated data will contain the instruction number that is associated with that data (D5 = 0).

— In this example, the Sync bit (D7) is programmed as an input (''0''), but is not used to start conversions.

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 determine the total sequence execution time. Table II shows the number of clock cycles for each instruction. Please see the data sheet, Section 4.0 (Sequencer) for the discussion of the states and their duration.

TABLE II. Instruction Execution Times

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	6			21	29
4	1	1	6	5	1	5	19
5	1	1	2			21	29
6	1	1	2	5	1	5	19
						TOTAL:	253

State 2, Calibration, not used
State 3, Run Timer, Being determined here

There is a total of 253 clock cycles, plus a fixed 2 clock cycle Timer delay, resulting in 255 clock cycles, yielding a time delay of

$$255 \div 8\ \text{MHz} = 31.875\ \mu\text{s}$$

To get the needed timer delay, we must subtract this figure from the 20 ms desired between the beginning of successive execution cycles.

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

A single timer count is 32 clock cycles or, at 8 MHz clock frequency,

$$32 \div 8\ \text{MHz} = 4\ \mu\text{s}$$

The desired timer delay is then

$$19.968\ \text{ms} \div 4\ \mu\text{s} = 4992 = 1380\text{H}.$$

Timer Register

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

Now we can program the 80C51 controller to interact with the DAS.

5.3 Microcontroller Programming

External write operations are used to tell the DAS what to do and external read operations are used to gather information from the DAS. Both of these are accomplished with the 80C51 through the MOVX (Move External) instruction. Internal data transfers use the MOV instruction. The general form of these instructions is

```
MOV     [destination], [source]
MOVX    [destination], [source]
```

The DPTR (Data Pointer) Register is generally used as a 16-bit address, but we can just as easily use R0 or R1 as an 8-bit address register or pointer as long as we realize that the high address byte at Port 2 will be at some unknown value unless we assign it a value. Moving information to the Data Pointer requires a 3-byte instruction, whereas moving information to R0 or R1 requires a two byte instruction. The Data Pointer was chosen in this example so that the program could be used with all three example circuits (*Figures 13, 14* and *15*).

Writing data to the DAS would, generally, follow the following example. The DAS__REG__ADD would be an 8-bit address for this example.

```
MOV     DPH, #0                 ;Set DPTR high byte to zero
MOV     DPL, #DAS__REG__ADD     ;Set address pointer
MOV     A, #LOW__BYTE__DATA     ;low byte data to ACC (Accumulator)
MOVX    @DPTR, A                ;Write data low byte to DAS
INC     DPTR                    ;Increment Data Pointer
MOV     A, #HIGH__BYTE__DATA    ;high byte data to accumulator
MOVX    @DPTR, A                ;Write data high byte to DAS
```

Reading data from the DAS is very similar to writing to it.

```
MOV     DPH, #0                 ;Clear DPTR high byte
MOV     DPL, #DAS__REG__ADD     ;Set address pointer for read address
MOVX    A, @DPTR                ;Read data low byte from DAS to ACC
MOV     [destination__1],A      ;Move data from ACC to memory
INC     DPTR                    ;Increment Data Pointer
MOVX    A, @DPTR                ;Read data high byte from DAS to ACC
MOV     [destination__2],A      ;Move data from ACC to memory
```

Paragraph 3.0 of the data sheet cautions that, when reading the FIFO, the lower byte of data (A0 = 0) should be read first, followed by reading of the upper byte (A0 = 1), in order to prevent a loss of the lower data byte. This is adhered to in this application example.

5.4 80C51 Assembly Routine for the Semiconductor Furnace Example

The program listing of *Figure 18* is the 80C51 assembly routine for the semiconductor furnace example. The program only contains the DAS initialization routine and the DAS interrupt service routine. A complete program for the application would have all the data manipulation and control functions included.

The routine closely follows the flowcharts of *Figures 2a* and *2b* and is well commented. However, the DAS' START bit is not cleared at the start of the interrupt service routine even though the flow chart indicates that it is.

The DAS initialization routine saves program memory and execution time by incrementing the Data Pointer (DPTR) to step through the DAS register addresses.

The interrupt service routine uses the JB (Jump if Bit set) instruction to test the state of the interrupt status bits.

The READ__FIFO routine reads the FIFO contents and stores them in a specified memory block. The size of the block and number of FIFO locations being read is programmable. This example reads 30 16-bit locations. This routine takes 5 lines of assembly code and 11 bytes of program memory.

```
DAS_8051        DAS-80C51 Sample Program Ver 1.1

                  1   ;*************************************************************
                  2   ;*                                                           *
                  3   ;*  AN 80C51 ASSEMBLY ROUTINE FOR THE SEMICONDUCTOR FURNACE  *
                  4   ;*                    APPLICATION EXAMPLE                    *
                  5   ;*                                                           *
                  6   ;*  BY:    Nicholas Gray                                     *
                  7   ;*  DATE:  May 20, 1994                                      *
                  8   ;*************************************************************
                  9   $TITLE(DAS-80C51 Sample Program Ver 1.1)
                 10   $LIST
                 11   $PAGEWIDTH(110)
                 12   $PAGING
                 13   $PAGELENGTH(55)
                 14   $PRINT(DAS_8051.LST)
                 15   $DEBUG
                 16   $OBJECT
                 17   $SYMBOLS
                 18   $MOD51
                 19   ;***** DAS REGISTERS / VARIABLES / CONSTANTS  SYMBOLIC DEFINITIONS *****
 0000            20             INSTR0     EQU    00H      ;DAS INSTRUCTION REGISTER ADDRESSES
 0002            21             INSTR1     EQU    02H      ;DEFINE READ/WRITE REGISTER ADDRESSES
 0004            22             INSTR2     EQU    04H
 0006            23             INSTR3     EQU    06H
 0008            24             INSTR4     EQU    08H
 000A            25             INSTR5     EQU    0AH
 000C            26             INSTR6     EQU    0CH
 000E            27             INSTR7     EQU    0EH
                 28
 0010            29             CONFIG     EQU    10H      ;DAS CONFIGURATION REGISTER ADDRESS, R/W
 0012            30             INTEN      EQU    12H      ;DAS INTERRUPT ENABLE REG. ADDRESS, R/W
 0014            31             INSTAT     EQU    14H      ;DAS INTERRUPT STATUS REG. ADDRESS, R
 0016            32             TIMER      EQU    16H      ;DAS TIMER REG. ADDRESS, R/W
 0018            33             FIFO       EQU    18H      ;DAS FIFO ADDRESS, R/W
 0040            34             DAS_RESULT EQU    40H      ;START ADD. OF TOP 60 LOCATIONS OF 8051
                 35                                        ;ON-CHIP RAM TO STORE CONVERSION RESULTS
 001E            36             FIFO_CNT   EQU    30       ;NUMBER OF RESULTS IN FIFO (DECIMAL)
 0013            37             T_SETH     EQU    13H      ;TIMER PRESET VALUE, HIGH BYTE
 0080            38             T_SETL     EQU    80H      ;TIMER PRESET VALUE, LOW BYTE
 00FF            39             F_MAX      EQU    0FFH     ;HIGH GAS FLOW LIMIT
 0000            40             F_MIN      EQU    00       ;LOW GAS FLOW LIMIT
 00FF            41             P_MAX      EQU    0FFH     ;HIGH PRESSURE LIMIT
 00D1            42             CAL_FLG    BIT    0D1H     ;PSW BIT 2 FOR CALIBRATION FLAG
                 43
                 44
                 45   ;***** BEGINNING OF PROGRAM *****
 0000            46             ORG        00H
 0000 020071     47             LJMP       START            ;JUMP BECAUSE INTERRUPT SERVICE
                 48                                        ;VECTORS START AT 0003H
                 49
 0003            50             ORG        03H              ;DAS INTERRUPT SERVICE VECTOR
 0003 802B       51             SJMP       DASIS            ;JUMP AROUND OTHER
                 52                                        ;INTERRUPT SERVICE VECTORS
```

TL/H/12068–18

FIGURE 18. 80C51 Assembly Program Listing

```
                    53    ;***** DAS INTERRUPT SERVICE ROUTINE *****
0030                54              ORG       30H
0030 C0E0           55    DASIS:    PUSH      ACC             ;SAVE ACCUMULATOR
0032 C0D0           56              PUSH      PSW             ;SAVE PROGRAM STATUS WORD
0034 C082           57              PUSH      DPL             ;SAVE DPTR
0036 C083           58              PUSH      DPH
0038 D2D3           59              SETB      RS0             ;GO TO REGISTER BANK 01
003A 758300         60              MOV       DPH,#0          ;CLEAR DPTR HIGH BYTE
003D 758214         61              MOV       DPL,#INSTAT     ;INTERRUPT STATUS REG ADDR TO R0
0040 E0             62              MOVX      A,@DPTR         ;READ INTERRUPT STATUS REG
0041 20E216         63              JB        ACC.2,READ_FIFO ;JUMP IF BIT 2 IS SET
0044 20E008         64              JB        ACC.0,DAS_LIMIT ;JUMP IF BIT 0 IS SET
0047 758210         65              MOV       DPL,#CONFIG     ;IF NONE OF ABOVE IS SET,
004A 7402           66              MOV       A,#02H          ;  CAL MUST BE DONE; RESET DAS
004C F0             67              MOVX      @DPTR,A
004D 8017           68              SJMP      FIN             ;GO RESTORE REGISTERS, RETURN
                    69
                    70    ;***** SERVICE ROUTINE - WATCHDOG LIMIT *****
004F                71    DAS_LIMIT:
004F 75821B         72              MOV       DPL,#01BH       ;READ LIMIT STATUS REG LIMIT #1
0052 E0             73              MOVX      A,@DPTR
0053 F5F0           74              MOV       B,A             ;LIMIT #1 TO "B" REGISTER
0055 0582           75              INC       DPL             ;READ LIMIT STATUS REG LIMIT #2
0057 E0             76              MOVX      A,@DPTR
                    77
                    78              ;THIS REST OF THIS SERVICE ROUTINE SHOULD TEST THE NECESSARY BITS
                    79              ;OF REGISTERS "A" AND "B", DETERMINE AND TAKE THE PROPER ACTION
                    80              ;BASED UPON WHICH BIT IS SET.
                    81
0058 800C           82              SJMP      FIN             ;RESTORE REGISTERS & RETURN
                    83
                    84    ;***** SERVICE ROUTINE - READ_FIFO *****
005A 758218         85    READ_FIFO: MOV      DPL,#FIFO       ;LOAD DAS FIFO ADDRESS
005D 7940           86              MOV       R1,#DAS_RESULT  ;LOAD START ADDDRESS FOR MEMORY LOAD
005F 7A3C           87              MOV       R2,#2*FIFO_CNT  ;SET # OF BYTES TO BE READ
0061 E0             88    RF2:      MOVX      A,@DPTR         ;READ DATA FROM DAS' FIFO
0062 F7             89              MOV       @R1,A           ;STORE FIFO DATA
0063 09             90              INC       R1              ;INCREMENT R1 FOR NEXT LOCATION
0064 DAFB           91              DJNZ      R2,RF2          ;DECREMENT COUNTER AND JUMP IF NOT ZERO
0066 C2D1           92    FIN:      CLR       CAL_FLG         ;CLEAR CALIBRATION FLAG
0068 D083           93              POP       DPH             ;RECOVER DPTR
006A D082           94              POP       DPL
006C D0D0           95              POP       PSW             ;RECOVER PSW, ALSO BACK TO REG BANK 00
006E D0E0           96              POP       ACC             ;RECOVER ACCUMULATOR
0070 32             97              RETI                      ;RETURN FROM INTERRUPT
                    98    ;***** PROCESSOR INITIALIZATION *****
0071 7580FF         99    START:    MOV       P0,#0FFH        ;PRESET PORT0 FOR DAS ADDRESSING
0074 75A0FF        100              MOV       P2,#0FFH        ;PRESET PORT2 FOR DAS ADDRESSING
0077 E4            101              CLR       A
0078 F5A8          102              MOV       IE,A            ;DISABLE ALL INTERRUPTS
007A F5D0          103              MOV       PSW,A           ;CLEAR PROGRAM STATUS WORD
007C F583          104              MOV       DPH,A           ;CLEAR DPTR HIGH BYTE
007E 75811F        105              MOV       SP,#01Fh        ;INITIALIZE STACK POINTER
0081 D289          106              SETB      IE0             ;SET INT0 FOR TRANSITION ACTIVATION
0083 C2D3          107              CLR       RS0             ;SELECT REGISTER BANK 0
0085 C2D4          108              CLR       RS1
```

TL/H/12068–19

FIGURE 18. 80C51 Assembly Program Listing (Continued)

```
                    109   ;     ***** INITIALIZE DAS REGISTERS *****
                    110   ;***** SET CONFIGURATION REGISTER *****
0087 758210         111             MOV       DPL,#CONFIG     ;LOAD CONFIG REG ADDR LO BYTE TO R0
008A 7402           112             MOV       A,#02H
008C F0             113             MOVX      @DPTR,A         ;RESET DAS
008D A3             114             INC       DPTR            ;CONFIG REG ADDR HI BYTE
008E E4             115             CLR       A
008F F0             116             MOVX      @DPTR,A         ;SET RAM POINTER TO 0, RP = 00
                    117   ;***** LOAD THE INSTRUCTIONS *****
                    118   ;INSTRUCTION 0 - MEASURE T1, SINGLE ENDED, 12-BIT + SIGN, TIMER ENABLED
0090 758200         119             MOV       DPL,#INSTR0     ;FIRST INSTRUCTION ADDRESS TO DPTR
0093 F0             120             MOVX      @DPTR,A         ;SET TO READ CHANNEL 0
0094 A3             121             INC       DPTR            ;2ND HALF OF INSTR0 ADDRESS
0095 7402           122             MOV       A,#02H          ;LOAD ACC WITH "02"
0097 F0             123             MOVX      @DPTR,A         ;WRITE A "02" TO AWAIT TIMER
                    124   ;INSTRUCTION 1 - MEASURE T1-T2, DIFFERENTIAL MODE, 12-BIT + SIGN
0098 A3             125             INC       DPTR            ;INSTRUCTION 1 ADDRESS
0099 C4             126             SWAP      A               ;ACC IS NOW "20"
009A F0             127             MOVX      @DPTR,A         ;SET TO READ CH0-CH1 DIFF
009B A3             128             INC       DPTR            ;2ND HALF OF INSTR1 ADDRESS
009C E4             129             CLR       A
009D F0             130             MOVX      @DPTR,A         ;WRITE A "00" (12-BIT READ)
                    131   ;INSTRUCTION 2 - MEASURE T1-T3, DIFFERENTIAL MODE, 12-BIT + SIGN
009E A3             132             INC       DPTR            ;INSTRUCTION 2 ADDRESS
009F 7440           133             MOV       A,#040H
00A1 F0             134             MOVX      @DPTR,A         ;SET TO READ CH2-CH0 DIFF
00A2 A3             135             INC       DPTR            ;2ND HALF OF INSTR2 ADDRESS
00A3 E4             136             CLR       A
00A4 F0             137             MOVX      @DPTR,A         ;WRITE A "00" (12-BIT READ)
                    138   ;INSTRUCTION 3 - MEASURE F, SINGLE ENDED, 8-BIT + SIGN
00A5 A3             139             INC       DPTR            ;INSTRUCTION 3 ADDRESS
00A6 740C           140             MOV       A,#0CH
00A8 F0             141             MOVX      @DPTR,A         ;SET TO READ CHANNEL 3
00A9 A3             142             INC       DPTR            ;2ND HALF OF INSTR3 ADDRESS
00AA 7424           143             MOV       A,#24H
00AC F0             144             MOVX      @DPTR,A         ;8-BIT READ W/ACQ TIME DELAY
                    145   ;INSTRUCTION 4 - WATCHDOG MODE, F, SINGLE ENDED
00AD A3             146             INC       DPTR            ;INSTRUCTION 4 ADDRESS
00AE 740C           147             MOV       A,#0CH
00B0 F0             148             MOVX      @DPTR,A         ;SET TO READ CHANNEL 3
00B1 A3             149             INC       DPTR            ;2ND HALF OF INSTR4 ADDRESS
00B2 7428           150             MOV       A,#28H
00B4 F0             151             MOVX      @DPTR,A         ;SET WATCHDOG MODE W/ACQ DELAY
                    152   ;INSTRUCTION 5 - MEASURE P, SINGLE ENDED, 8-BIT
00B5 A3             153             INC       DPTR            ;INSTRUCTION 5 ADDRESS
00B6 7410           154             MOV       A,#10H
00B8 F0             155             MOVX      @DPTR,A         ;SET TO READ CHANNEL 4
00B9 A3             156             INC       DPTR            ;2ND HALF OF INSTR5 ADDRESS
00BA 7404           157             MOV       A,#04H
00BC F0             158             MOVX      @DPTR,A         ;8-BIT READ W/NO ACQ DELAY
                    159   ;INSTRUCTION 6 - WATCHDOG MODE, P, SINGLE ENDED, LOOP BIT ENABLED
00BD A3             160             INC       DPTR            ;INSTRUCTION 6 ADDRESS
00BE 7411           161             MOV       A,#11H
00C0 F0             162             MOVX      @DPTR,A         ;READ CH4, SET LOOP BIT
00C1 A3             163             INC       DPTR            ;2ND HALF OF INSTR6 ADDRESS
00C2 7408           164             MOV       A,#08H
00C4 F0             165             MOVX      @DPTR,A         ;WATCHDOG MODE
```

TL/H/12068–20

FIGURE 18. 80C51 Assembly Program Listing (Continued)

```
                    166   ;***** SET WATCHDOG LIMITS *****
                    167   ;SET GAS FLOW (F) HIGH LIMIT
00C5 851182         168             MOV     DPL,CONFIG+1    ;CONFIG REG HIGH BYTE ADDR
00C8 7401           169             MOV     A,#01H          ;SET RAM POINTER TO 01
00CA F0             170             MOVX    @DPTR,A         ;WRITE RAM POINTER
00CB 758208         171             MOV     DPL,#INSTR4     ;INSTR4 HIGH LIMIT ADDRESS
00CE 74FF           172             MOV     A,#F_MAX        ;INSTR4 (GAS FLOW) HIGH LIMIT
00D0 F0             173             MOVX    @DPTR,A         ;WRITE F HIGH LIMIT
00D1 A3             174             INC     DPTR            ;2ND HALF OF LIMIT ADDRESS
00D2 7402           175             MOV     A,#02H          ;SET FOR MAX LIMIT
00D4 F0             176             MOVX    @DPTR,A         ;WRITE IT
                    177   ;SET PRESSURE (P) HIGH LIMIT
00D5 75820C         178             MOV     DPL,#INSTR6     ;INSTR6 HIGH LIMIT ADDRESS
00D8 74FF           179             MOV     A,#P_MAX        ;INSTR6 (PRESSURE) HIGH LIMIT
00DA F0             180             MOVX    @DPTR,A         ;WRITE P HIGH LIMIT
00DB A3             181             INC     DPTR            ;2ND HALF OF LIMIT ADDRESS
00DC 7402           182             MOV     A,#02H          ;SET FOR MAX LIMIT
00DE F0             183             MOVX    @DPTR,A         ;WRITE IT
                    184   ;SET GAS FLOW (F) LOW LIMIT
00DF 758210         185             MOV     DPL,#CONFIG     ;CONF REG LOW BYTE ADDR
00E2 7410           186             MOV     A,#10H          ;SET RAM POINTER TO 10
00E4 F0             187             MOVX    @DPTR,A         ;WRITE RAM POINTER
00E5 758208         188             MOV     DPL,#INSTR4     ;INSTR4 LOW LIMIT ADDRESS
00E8 7400           189             MOV     A,#F_MIN        ;INSTR4 (GAS FLOW) LOW LIMIT
00EA F0             190             MOVX    @DPTR,A         ;WRITE FLOW LOW LIMIT
00EB A3             191             INC     DPTR            ;2ND HALF OF LIMIT ADDRESS
00EC E4             192             CLR     A               ;SET FOR MIN LIMIT
00ED F0             193             MOVX    @DPTR,A         ;WRITE IT
                    194   ;SET PRESSURE (P) LOW LIMIT
00EE 75820C         195             MOV     DPL,#INSTR6     ;INSTR6 LOW LIMIT ADDRESS
00F1 F0             196             MOVX    @DPTR,A         ;WRITE INSTR6 (P) LOW LIMIT OF 0
00F2 A3             197             INC     DPTR            ;2ND HALF OF LIMIT ADDRESS
00F3 04             198             INC     A               ;ACC=1 (SIGN BIT)
00F4 F0             199             MOVX    @DPTR,A         ;SET FOR MIN LIMIT
                    200   ;***** ENABLE INTERRUPTS *****
00F5 758212         201             MOV     DPL,#INTEN      ;INTERRUPT ENABLE REG ADDR
00F8 7415           202             MOV     A,#15H          ;ENABLE INTERRUPTS #0, 2, 4
00FA F0             203             MOVX    @DPTR,A
00FB A3             204             INC     DPTR            ;2ND HALF OF REG ADDRESS
00FC 741E           205             MOV     A,#FIFO_CNT     ;MOVE FIFO COUNT TO ACC
00FE 23             206             RL      A               ;ROTATE TO MSBs
00FF 23             207             RL      A
0100 23             208             RL      A               ;A = F0H
0101 F0             209             MOVX    @DPTR,A         ;INTERRUPT AFTER 30 MEASUREMENTS
                    210   ;***** INITIALIZE TIMER *****
0102 758216         211             MOV     DPL,#TIMER      ;LOAD DAS TIMER REG ADDRESS
0105 7480           212             MOV     A,#T_SETL       ;LOAD TIMER LOW BYTE VALUE
0107 F0             213             MOVX    @DPTR,A         ;SET TIMER LOW BYTE VALUE
0108 A3             214             INC     DPTR            ;2ND HALF OF ADDRESS
0109 7413           215             MOV     A,#T_SETH       ;LOAD TIMER HIGH BYTE VALUE
010B F0             216             MOVX    @DPTR,A         ;SET TIMER LOW BYTE VALUE
                    217   ;***** ENABLE PROCESSOR INTERRUPT *****
010C 75A881         218             MOV     IE,#81H         ;ENABLE INT0, MASK OTHERS
```

TL/H/12068–21

FIGURE 18. 80C51 Assembly Program Listing (Continued)

```
                    219    ;***** DAS FULL CALIBRATION *****
010F D2D1           220               SETB        CAL_FLG         ;SET CALIBRATION FLAG
0111 758210         221               MOV         DPL,#CONFIG     ;CONFIG REG LOW BYTE ADDRESS
0114 7408           222               MOV         A,#08H          ;START DAS FULL CALIBRATION
0116 F0             223               MOVX        @DPTR,A
0117 30D1FD         224               JNB         CAL_FLG,$       ;WAIT FOR CAL_FLG TO BE CLEARED DURING
                    225                                           ;INTERRUPT IN SERVICE ROUTINE
                    226    ;***** BEGIN CONVERSIONS *****
011A 7401           227               MOV         A,#01H          ;START BIT SET TO 1
011C 900010         228               MOV         DPTR,#CONFIG    ;LOAD CONGIFURATION REGISTER ADDRESS
011F F0             229               MOVX        @DPTR,A         ;START CONVERSIONS
                    230      ;THE FOLLOWING WOULD NORMALLY BE REPLACED BY WHATEVER ROUTINES ARE NECESSARY
                    231      ;FOR THE SYSTEM.  THE ACTUAL PROGRAM MIGHT CONTINUE ON.
                    232      ;
0120 80FE           233               SJMP        $               ;WAIT FOR INTERRUPT
                    234    END

VERSION 1.2i ASSEMBLY COMPLETE, 0 ERRORS FOUND

ACC. . . . . . . . . . . . . . . D ADDR  00E0H  PREDEFINED
B. . . . . . . . . . . . . . . . D ADDR  00F0H  PREDEFINED
CAL_FLG. . . . . . . . . . . . . B ADDR  00D1H
CONFIG . . . . . . . . . . . . .   NUMB  0010H
DASIS. . . . . . . . . . . . . . C ADDR  0030H
DAS_LIMIT. . . . . . . . . . . . C ADDR  004FH
DAS_RESULT . . . . . . . . . . .   NUMB  0040H
DPH. . . . . . . . . . . . . . . D ADDR  0083H  PREDEFINED
DPL. . . . . . . . . . . . . . . D ADDR  0082H  PREDEFINED
FIFO . . . . . . . . . . . . . .   NUMB  0018H
FIFO_CNT . . . . . . . . . . . .   NUMB  001EH
FIN. . . . . . . . . . . . . . . C ADDR  0066H
F_MAX. . . . . . . . . . . . . .   NUMB  00FFH
F_MIN. . . . . . . . . . . . . .   NUMB  0000H
IE . . . . . . . . . . . . . . . D ADDR  00A8H  PREDEFINED
IE0. . . . . . . . . . . . . . . B ADDR  0089H  PREDEFINED
INSTAT . . . . . . . . . . . . .   NUMB  0014H
INSTR0 . . . . . . . . . . . . .   NUMB  0000H
INSTR1 . . . . . . . . . . . . .   NUMB  0002H  NOT USED
INSTR2 . . . . . . . . . . . . .   NUMB  0004H  NOT USED
INSTR3 . . . . . . . . . . . . .   NUMB  0006H  NOT USED
INSTR4 . . . . . . . . . . . . .   NUMB  0008H
INSTR5 . . . . . . . . . . . . .   NUMB  000AH  NOT USED
INSTR6 . . . . . . . . . . . . .   NUMB  000CH
INSTR7 . . . . . . . . . . . . .   NUMB  000EH  NOT USED
INTEN. . . . . . . . . . . . . .   NUMB  0012H
P0 . . . . . . . . . . . . . . . D ADDR  0080H  PREDEFINED
P2 . . . . . . . . . . . . . . . D ADDR  00A0H  PREDEFINED
PSW. . . . . . . . . . . . . . . D ADDR  00D0H  PREDEFINED
P_MAX. . . . . . . . . . . . . .   NUMB  00FFH
READ_FIFO. . . . . . . . . . . . C ADDR  005AH
RF2. . . . . . . . . . . . . . . C ADDR  0061H
RS0. . . . . . . . . . . . . . . B ADDR  00D3H  PREDEFINED
RS1. . . . . . . . . . . . . . . B ADDR  00D4H  PREDEFINED
SP . . . . . . . . . . . . . . . D ADDR  0081H  PREDEFINED
START. . . . . . . . . . . . . . C ADDR  0071H
TIMER. . . . . . . . . . . . . .   NUMB  0016H
T_SETH . . . . . . . . . . . . .   NUMB  0013H
T_SETL . . . . . . . . . . . . .   NUMB  0080H
```

TL/H/12068–22

FIGURE 18. 80C51 Assembly Program Listing (Continued)

APPENDIX A:

Register Bit Assignments and Programmer's Notes

The following is an information summary concerning the various registers of the DAS, together with forms for programmer's notes to aid in setting instructions and in providing design documentation.

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 instruction execution. "1" Starts instruction execution.

D1: RESET: "1" stops the sequencer (resets the start bit), resets instruction pointer in the Interrupt Status Register, resets all interrupt flags, and clears the conversion FIFO. Automatically resets itself after two clock cycles.

D2: AUTO-ZERO: "1" causes a long auto-zero calibration cycle to be performed.

D3: FULL CALIBRATION: "1" causes a full calibration cycle to be performed.

D4: STANDBY: "1" puts the DAS into low-power standby mode. "0" returns the DAS to operational status identical to that caused by setting the RESET bit. The Instruction RAM can be accessed through the Read/Write operations while in the Standby mode.

D5: CHANNEL MASK: "0" causes bits 13 through 15 of the conversion result to be the instruction number associated with the conversion result in bits 0 through 12. "1" causes bits 13 through 15 to hold the extended sign bit.

D6: A/Z EACH: "1" causes a short auto-zero cycle to be performed before each conversion.

D7: SYNC I/O: "0" causes Sync pin to be an input; "1" causes it to be an output.

D9–D8: RAM POINTER: Selects the section of the Instruction RAM:

00 = Instructions, 01 = Limits #1, 10 = Limits #2.

D10: TEST: This bit is used for production testing and must be kept at "0" for normal operation.

D11: DIAGNOSTIC: "1", along with a correctly chosen instruction, allows verification that the DAS' ADC is performing correctly. See paragraph 2.2 of the data sheet for details. Available only in the LM12(H)458.

D15–D12: DON'T CARE

PROGRAMMER'S NOTES:

Configuration Register: Address: Symbol:

Note:

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
X	X	X	X												

Hexadecimal value:

Configuration Register: Address: Symbol:

Note:

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
X	X	X	X												

Hexadecimal value:

Configuration Register: Address: Symbol:

Note:

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
X	X	X	X												

Hexadecimal value:

INSTRUCTION RAM (Read/Write):

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

D0: Loop: ''0'' causes flow to the next instruction, ''1'' causes loop back to instruction #0.

D1: PAUSE: ''0'' causes no action. ''1'' causes sequencer to stop before executing current instruction and to reset Start bit in Configuration Register to ''0''. Writing a ''1'' to the Start bit restarts 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'' causes operation with internal timing. ''1'' causes S/H and conversion timing to be controlled by the SYNC input pin. If this bit is high, bit D7 of Configuration register must be high to prevent the DAS from getting hung up.

D9: TIMER: ''0'' causes no delay; ''1'' causes a halt to instruction execution until the timer counts down to zero.

D10: 8/12: ''0'' causes 12-bit + sign conversion, ''1'' causes 8-bit + sign conversion.

D11: WATCHDOG: ''0'' causes no comparison; ''1'' causes watchdog comparison.

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
For 12-bit + sign: $D = 0.45 \times R_S(k\Omega) \times f_{CLK}$ (MHz)
For 8-bit + sign: $D = 0.36 \times R_S\ (k\Omega) \times f_{CLK}$ (MHz)
where D = Content of D15–D12, R_S = input source resistance

PROGRAMMER's NOTES:

Instruction #0: Address: Symbol:

Note:

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

Hexadecimal value:

Instruction #1: Address: Symbol:

Note:

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

Hexadecimal value:

Instruction #2: Address: Symbol:

Note:

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

Hexadecimal value:

Instruction #3: Address: Symbol:

Note:

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

Hexadecimal value:

Instruction #4: Address: Symbol:

Note:

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

Hexadecimal value:

Instruction #5: Address: Symbol:

Note:

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

Hexadecimal value:

Instruction #6: Address: Symbol:

Note:

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

Hexadecimal value:

Instruction #7: Address: Symbol:

Note:

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

Hexadecimal value:

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

D8: >/<: High or low limit determination; "0" = inputs lower than limit generate an interrupt, "1" = inputs higher than limit generate an 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
Don't Care						> / <	Sign	Limit							
X	X	X	X	X	X										

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
Don't Care						> / <	Sign	Limit							
X	X	X	X	X	X										

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
Don't Care						> / <	Sign	Limit							
X	X	X	X	X	X										

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
Don't Care						> / <	Sign	Limit							
X	X	X	X	X	X										

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
Don't Care						> / <	Sign	Limit							
X	X	X	X	X	X										

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
Don't Care						> / <	Sign	Limit							
X	X	X	X	X	X										

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
Don't Care						>/<	Sign	Limit							
X	X	X	X	X	X										

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
Don't Care						>/<	Sign	Limit							
X	X	X	X	X	X										

Hexadecimal value:

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

D8: >/<: High or low limit determination; ''0'' = inputs lower than limit generate an interrupt, ''1'' = inputs higher than limit generate an 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
Don't Care						>/<	Sign	Limit							
X	X	X	X	X	X										

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
Don't Care						>/<	Sign	Limit							
X	X	X	X	X	X										

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
Don't Care						>/<	Sign	Limit							
X	X	X	X	X	X										

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
Don't Care						> / <	Sign	Limit							
X	X	X	X	X	X										

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
Don't Care						> / <	Sign	Limit							
X	X	X	X	X	X										

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
Don't Care						> / <	Sign	Limit							
X	X	X	X	X	X										

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
Don't Care						> / <	Sign	Limit							
X	X	X	X	X	X										

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
Don't Care						> / <	Sign	Limit							
X	X	X	X	X	X										

Hexadecimal value:

INTERRUPT ENABLE REGISTER (Read/Write):

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
Number of conversion results in FIFO to Generate Interrupt INT0					Instruction Number that Generates Interrupt INT1			INT7 Standby	INT6 Power	INT5 Pause	INT4 Cal	INT3 A/Z	INT2 Result	INT1 Instr	INT0 W/D

D0: INT0: Generate interrupt when a watchdog limit is passed

D1: INT1: Generate interrupt when the programmed instruction (D10–D8) is reached

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

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

D4: INT4: Generate interrupt when a full calibration cycle is completed

D5: INT5: Generate interrupt when a pause condition is encountered

D6: INT6: Generate interrupt when low power supply is detected

D7: INT7: Generate interrupt when the DAS is returned from standby and is ready

D10–D8: Programmable instruction number to generate an interrupt when that instruction is reached. When this instruction is reached, D1 is set high.

D15–D11: Programmable number of conversion results in the FIFO to generate an interrupt. When this instruction is reached, D2 is set high.

Interrupt Enable Register: Address: Symbol:

Note:

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
Number of conversion results in FIFO to Generate Interrupt INT0					Instruction Number that Generates Interrupt INT1			INT7 Standby	INT6 Power	INT5 Pause	INT4 Cal	INT3 A/Z	INT2 Result	INT1 Instr	INT0 W/D

Hexadecimal value:

Interrupt Enable Register: Address: Symbol:

Note:

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
Number of conversion results in FIFO to Generate Interrupt INT0					Instruction Number that Generates Interrupt INT1			INT7 Standby	INT6 Power	INT5 Pause	INT4 Cal	INT3 A/Z	INT2 Result	INT1 Instr	INT0 W/D

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															

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

Delay = 32 × N + 2

PROGRAMMER'S NOTES:
Timer Register: Address: Symbol:

Note:

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

Hexadecimal value:

Timer Register: Address: Symbol:

Note:

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

Hexadecimal value:

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											

Hexadecimal value:

D11–D0: CONVERSION RESULTS:
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 (Channel Mask) of Configuration Register

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 Standby	INST6 Power	INST5 Pause	INST4 Cal	INST3 A/Z	INST2 Result	INST1 Instr	INST0 W/D

Bits #0 through 7 are interrupt flags that will be set to ''1'' when the conditions indicated below occur. The bits are set to ''1'' whether or not the interrupt is enabled in the Interrupt Enable register. The bits reset to ''0'' when the register is read, or when the device is reset through the Configuration register.

D0: INST0: set to 1 when a watchdog limit is passed

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

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

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

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

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

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

D7: INST7: set to 1 when the DAS is returned from standby and is ready

D10–D8: Holds the instruction number being executed or, (during a Pause or Timer delay) will next be executed

D15–D11: Holds the current number of conversion results present in the FIFO

FIFO Register: Address: Symbol:

Note:

APPENDIX B:

CRITICAL TIMING CONSIDERATIONS

The solution presented in this application note does not insure equal time delays between successive sets of data. The controller could be interrupted by a higher priority interrupt, delaying servicing of the DAS' interrupt. Also, the time from interrupt request until that interrupt is serviced can vary because the interrupt is not serviced until completion of the controller instruction currently being executed. Sometimes these delays are unacceptable, as in DSP applications, or whenever an FFT is to be performed on the data.

To insure that the time interval between the start of successive data acquisitions is constant, we can use the SYNC input to the DAS. To do this, the I/O bit of the Configuration Register (bit 7) must be set to a ''0'', configuring the SYNC pin as an input and making it possible to synchronize the start of conversion to an external event. The SYNC bit of the Instruction RAM (bit 8) of the first instruction can now be set to a ''1'' causing the sequencer to suspend operation at the end of the internal S/H's acquisition cycle and to wait (in the ''sample'' mode of the S/H) until a rising edge appears at the SYNC pin of the DAS, at which time the S/H acquires the input signal (goes into the ''hold'' mode) and the ADC begins performing a conversion on the next rising edge of the clock. The falling edge at the SYNC input is totally insignificant, and there is no need to synchronize the SYNC signal with the clock. Of course, the stability of the SYNC input signal will determine the repeatability of the time intervals between sets of data.

We must ascertain that there is sufficient time to retrieve the data from the DAS while the DAS is awaiting another rising edge at the SYNC pin. Bringing the $\overline{CS}$ line low to access the DAS (reading from or writing to it) will cause the clock to be internally gated off, affecting the timing of successive data gathering if acquisition, conversion, or a comparison is in progress. In the example of this application note there should be no problem as there is a very long 20 ms between successive SYNC rising edges. In some applications, however, this may not be the case and we need to be sure that we are not accessing the DAS during the SYNC input rise or while it is doing anything but awaiting the next rise of the SYNC input. Ensuring proper timing will entail an analysis of the time needed for the DAS to go through one sequencer cycle and the time needed for the processor or controller to retrieve information from the DAS. Since, in this application, the DAS is performing operations for only about 32 μs out of every 20 ms, there is more than enough time to retrieve information from the DAS.

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