Interfacing the LM12458 Data Acquisition System to the MC68HC11AX Microcontroller

National Semiconductor
Application Note 947
Nelson Caldani
July 1994

TABLE OF CONTENTS

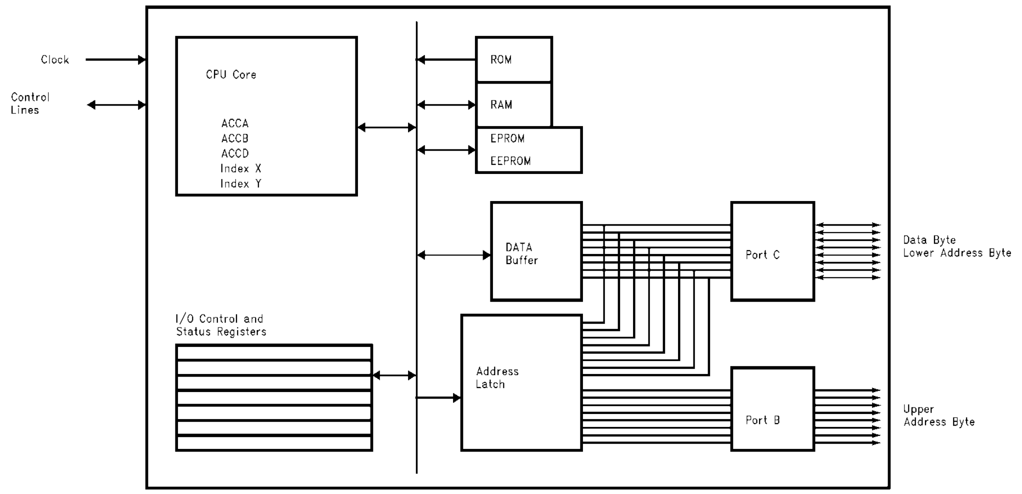
1.0 INTRODUCTION

This application note shows how to interface Motorola's MC68HC11 family of microcontrollers to the LM12458 parallel data acquisition system (DAS). It is intended to give the reader a better understanding of the DAS and how it can be integrated in a μP/μC system. There is a brief description of the MC68HC11Ax and the LM12458. An example is provided that will illustrate how to use the DAS in a system where a specific sampling rate is desired, single-ended and differential measurements of multiple channels need to be made, and alarm conditions must be monitored. The application note provides a detailed circuit schematic showing the interface circuitry of the DAS to the 68HC11 accompanied with the appropriate assembly code necessary to program the DAS to fulfill the specific requirements of the example.

2.0 GENERAL OVERVIEW OF THE MC68HC11AX AND THE LM12458 DAS

2.1 MC68HC11Ax

The 68HC11 is an 8-bit microcontroller with a 16-bit address range. It has two user modes of operation: single-chip and expanded-multiplexed mode. In single-chip mode, the 68HC11 has no external data bus or address bus. It communicates with external peripherals using its I/O ports. In expanded-multiplexed mode, the 68HC11 uses its 8-bit I/O ports B and C as address and data buses. Port B is the upper address byte. The lower address byte and the data bus are multiplexed together on port C. An address strobe (AS) generated by the 68HC11 indicates when the lower address byte is on port C. This signal may be used with external latches to demultiplex the lower address byte and data bus. In a read or write cycle the address is first placed on ports B and C and the AS is asserted. After a certain time period has elapsed, data is placed on port C. Refer to the manual of the 68HC11 for detailed timing diagrams. The 68HC11 has a R/$\overline{\text{W}}$ control signal that indicates the direction of data flow with respect to the μC. This signal in conjunction with the E clock may be used to generate read ($\overline{\text{RD}}$) and write ($\overline{\text{WR}}$) signals going to the DAS and other peripherals. The 68HC11 has two 8-bit accumulators, ACCA and ACCB and a 16-bit accumulator ACCD that are used for manipulating data. ACCD is a cascade of ACCA and ACCB. When using ACCD to read from or write to memory, data is not transferred in a single 16-bit word but a succession of two 8-bit bytes. Data is first transferred via ACCA, followed by ACCB. *Figure 1* shows a block diagram of the MC68HC11Ax in expanded multiplexed mode.



TL/H/12050–1

FIGURE 1. Block Diagram of the MC68HC11Ax in Expanded Multiplexed Mode

2.2 LM12458 DAS

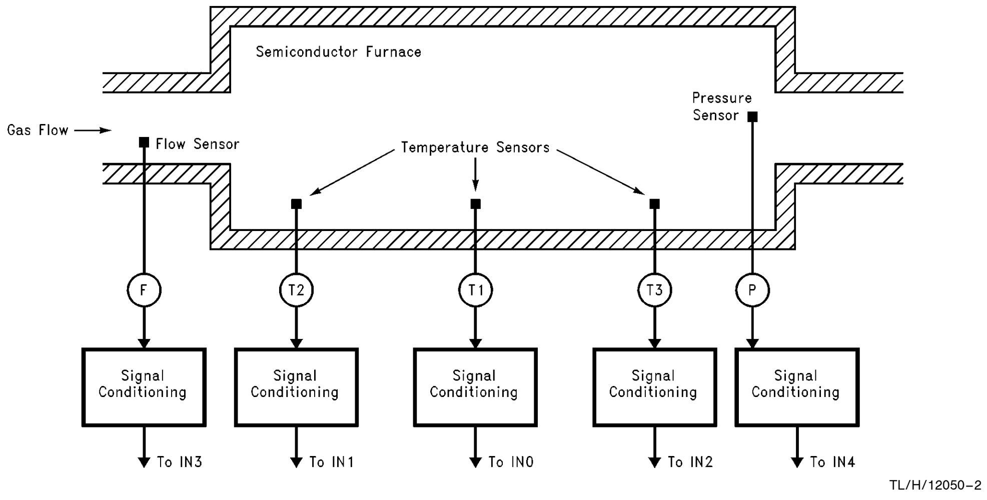
The LM12458 is a 12-bit + sign parallel ADC that has an Instruction RAM, Configuration register, FIFO, Timer register, Interrupt Enable and Status register, watchdog detection circuitry, and an eight channel fully differential multiplexer. The DAS has three modes of operation: 12-bit + sign conversion, 8-bit + sign conversion and an 8-bit watchdog comparison mode.

The Instruction RAM has eight 48-bit wide programmable instruction registers (INSTR0–INSTR7). These instructions are used to set the mode of the DAS, the acquisition time, channels that are to be converted and the watchdog limits. The watchdog detects when a particular channel has passed a programmed limit set in the Instruction RAM. Refer to section 2.1 of the data sheet for a detailed description of the Instruction RAM and the watchdog feature. The MUX channel assignments are also set in the Instruction RAM. Table I in section 2.3 of the data sheet gives the MUX address channel assignments. The Configuration register is the control "panel" of the DAS. Its bits start/stop instruction execution, reset the DAS, perform a full calibration and auto-zero. Refer to section 2.2 of the data sheet for a detailed description of the configuration register. The FIFO has thirty-two 16-bit wide locations. The FIFO stores the conversion data performed by the ADC. Refer to section 3.0 of the data sheet for a detailed description of the FIFO. The DAS has an internal Timer used to delay execution of an instruction by a related amount of clock cycles programmed in the Timer register. Refer to section 2.7 of the data sheet for a detailed description of the timer. The Interrupt Enable register can set up eight different sources that will cause interrupts to occur and the Interrupt Status register indicates the sources of these interrupts. Refer to section 2.4 of the data sheet for a detailed description of the Interrupt Enable register and the Interrupt Status register.

3.0 SYSTEM EXAMPLE

3.1 Furnace Description and System Requirements

A high temperature semiconductor furnace will be the source of the analog signals going to the inputs of the DAS. Refer to *Figure 2* for a detailed setup of the furnace and the signals from the furnace that need to be digitized.



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

FIGURE 2. Signal Measurement Arrangements of the Semiconductor Furnace

The DAS will measure:

- T1 at IN0 with 12-bit resolution. (single-ended)
- T1−T2 with 12-bit + sign resolution. T2 is connected to IN1. (differential)
- T1−T3 with 12-bit + sign resolution. T3 is connected to IN2. (differential)
- The gas flow, F at IN3 with 8-bit resolution and an extended acquisition. (single-ended)
- The pressure, P at IN4 with 8-bit resolution. (single-ended)

The DAS will also be used to monitor three alarm conditions.

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

The following assumptions are made about the furnace system:

- All the analog signals to be measured by the DAS are conditioned so that their voltage levels are within the range 0V−2.5V.
- The output of the flow sensor signal conditioning circuit has a 600Ω source impedance.
- The DAS reference voltage is 2.5V. ($V_{REF}+ = 2.5V$ and $V_{REF}- =$ AGND)
- The DAS is to achieve a throughput rate of 50 Hz per signal measured.

The DAS will be programmed as follows:

- Seven instructions, INSTR0 through INSTR6 of the Instruction RAM will be used.

  INSTR0: Watchdog mode for the gas flow, F, with the acquisition factor D = 1. The timer bit is also set.

  INSTR1: Watchdog mode for the gas pressure P.

  INSTR2: Measure T1 at IN0 with 12-bit resolution. $V_{IN}+ =$ IN0 and $V_{IN}- =$ AGND.

  INTSR3: Measure T1−T2 with 12-bit + sign resolution. T2 is connected to IN1. $V_{IN}+ =$ IN0 and $V_{IN}- =$ IN1.

  INSTR4: Measure T1−T3 with 12-bit + sign resolution. T3 is connected to IN2. $V_{IN}+ =$ IN0 and $V_{IN}- =$ IN2.

  INSTR5: Measure the gas flow, F at IN3 with 8-bit resolution and the acquisition factor D = 1. $V_{IN}+ =$ IN3 and $V_{IN}- =$ AGND.

  INSTR6: Measure the pressure, P at IN4 with 8-bit resolution. $V_{IN}+ =$ IN4 and $V_{IN}- =$ AGND.

- The DAS loops back to INSTR0 after executing INSTR6. Each loop is called an instruction loop.
- Each instruction loop generates five conversion results. The DAS is programmed to interrupt the 68HC11 when the FIFO has five conversions stored in it. This interrupt will occur after execution of INSTR6 in every instruction loop at which time the FIFO will be read by the 68HC11.
- The DAS will not be stopped when the FIFO is read. The FIFO is read during the time delay added between INSTR6 and INSTR0.

Since the flow sensor circuitry has an output impedance of 600Ω, it requires additional acquisition time. From the data sheet, the acquisition factor D in 8-bit mode is:

$D = 0.36 * R_S * f_{CLK}$

$D = 0.36 * 0.600 * 5 = 1.08$

$\mathbf{D = 1}$

3.2 DAS Registers

The following registers contain the necessary values to satisfy the system requirements.

Instruction Register Definition for RP = 00

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

INSTR0: Watchdog Mode for Flow, D = 1, Timer Enabled
$V_{IN}+ = $ IN3, $V_{IN}- = $ AGND

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

INSTR1: Watchdog Mode for Pressure
$V_{IN}+ = $ IN4, $V_{IN}- = $ AGND

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

INSTR2: Measuring T1, Single-Ended, 12-Bit
$V_{IN}+ = $ IN0 (T1), $V_{IN}- = $ AGND

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

INSTR3: Measuring T1−T2, Differential, 12-Bit + Sign
$V_{IN}+ = $ IN0 (T1), $V_{IN}- = $ IN1 (T2)

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

INSTR4: Measuring T1−T3, Single-Ended, 12-Bit + Sign
$V_{IN}+ = $ IN0 (T1) $V_{IN}^{-} = $ IN2 (T3)

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

INSTR5: Measuring Flow, Single-Ended, 8-Bit, D = 1
$V_{IN}+ = $ IN3, $V_{IN}- = $ AGND

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

INSTR6: Measuring Pressure, Single-Ended, 8-Bit, Loop Bit Set
$V_{IN}+ = $ IN4,$V_{IN}- = $ AGND

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

3.2 DAS Registers (Continued)

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							

INSTR0, Limit #1, RP = 01

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

INSTR0, Limit #2, RP = 10

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

INSTR1, Limit #1, RP = 01

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

INSTR1, Limit #2, RP = 10

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

Interrupt Enable Register Definition

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

Watchdog Limit, FIFO Full (Conv = 5), and Auto-Zero Interrupts Enabled

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	1	0	1	0	1

Configuration Register Definition

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	Standby	Full Cal	Auto Zero	Reset	Start

Configuration Register Start Command with SYNC Out

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

3.3 Determining the Throughput Rate

The following table gives the number of clock cycles each instruction takes to execute. The total number of clock cycles may then be used to determine how long it will take for all the instructions to be executed. This will aid in calculating the throughput rate of the DAS.

Instruction Number	State 0	State 1	State 7	State 6	State 4	State 5	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	2			21	25
5	1	1	6	5	1	5	19
6	1	1	2	5	1	5	15
						Total	**253**

Since the timer will be used, an additional 2 clock cycles need to be added to the total 253.

The total time it takes to execute a full instruction loop with a clock frequency (f_{CLK}) of 5 MHz is:

$(253 + 2) * 200$ ns $= 51\ \mu$s

In addition to the instruction loop execution time, there is some delay time added at the end of the instruction loop for each chip select made to the DAS when reading the five conversions stored in the FIFO. Whenever chip select is asserted, all clock dependent activities of the DAS come to a halt. This delay is the total amount of time that chip select has gone low when reading the FIFO. Each chip select cycle is 500 ns long (refer to the timing diagrams). There are five conversions, two bytes each, that will be read. This gives a total of ten chip selects (one chip select per byte). Ten chip selects at 500 ns per chip select gives a total delay time of 5 μs, so the time required to complete an instruction loop and read the conversion results is 51 μs + 5 μs = 56 μs. The desired throughput rate is 50 Hz (a sample per channel every 20 ms). This implies that a delay time between instruction loops must be added. The delay time is equal to:

20 ms $-$ 56 μs $=$ 19.944 ms

This time delay may be inserted between instruction loops by setting the timer bit in INSTR0. The amount of the time delay is related to the value stored in the timer register. The following shows how to calculate the value to put in the timer register to give a delay of 19.944 ms:

The number of 5 MHz clock cycles in 19.944 ms is:
19.944 ms * 5 MHz = 99720 clock cycles

The timer increments every 2^5 (32) clock cycles. Therefore, the value to be placed in the timer register is:

$99720/32 = (3116.25)_{base10}$
$= (\mathbf{0C2C})_{base16}$

The 0.25 fraction is lost when converting $(3116.25)_{base10}$ to hexadecimal. This results in decreasing the delay time by eight 5 MHz clock cycles (8*200 ns = **1.6 μs**) to 19.9424 ms. So the actual throughput rate is:

19.9424 ms + 56 μs = 19.9984 ms

$$\frac{1}{19.9984\ \text{ms}} = \mathbf{50.004\ Hz}$$

4.0 SYSTEM OVERVIEW

4.1 How to Communicate with the DAS

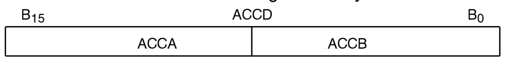
In developing the software and circuitry to program the DAS for the furnace example, Motorola's EVM development system for emulating the MC68HC11A8 was used. In this application note it is assumed that the reader is already familiar with the architecture and instruction set of the MC68HC11A8. The reader should also be familiar with the LM12(H)454/8 DAS by having previously read the data sheet. In this design the μC's E_{clock} is 2 MHz and the DAS clock is running at 5 MHz.

The DAS can be selected to run on an 8-bit or a 16-bit data bus. Since we are using an 8-bit μC the DAS must be configured to operate in 8-bit mode. This is done by hardwiring the "BW" pin to a logic 1. The internal registers of the DAS are all 16 bits long, so in order to read from and write to a 16-bit register while in 8-bit mode, data transfer must take place in two cycles. In fact when in 8-bit mode the internal registers are byte addressable. The 68HC11 architecture and instruction set facilitates this transfer of data in two ways. One way is using standard 8-bit reads and writes via the 8-bit accumulators. For example, a 16-bit write to the configuration register:

```
LDAA #DATA1   ;DATA1 is the least significant
              ;byte (LSB).
STAA CONFIG1  ;CONFIG1 = Address of LSB of
              ;the configuration register.
LDAA #DATA2   ;DATA2 is the most significant
              ;byte (MSB).
STAA CONFIG2  ;CONFIG2 = Address of MSB of
              ;the configuration register.
```

Notice that it takes 4 instructions to do this.

Another way of achieving the same results with half the number of instructions is to use the 68HC11's 16-bit double accumulator D. This accumulator is an actual cascading of ACCA and ACCB. ACCA is the most significant byte of ACCD and ACCB is the least significant byte.



When using ACCD the instruction set emulates that of a 16-bit processor. The above example becomes:

```
LDD #DATA   ;DATA is 16 bits.
STD CONFIG  ;Address of configuration
            ;register.
```

This is not a true single 16-bit transfer of data but a succession of 8-bit ones. When using ACCD the user should notice that data is first transferred from ACCA followed by data from ACCB. This plays an important role in determining the data that is sent to and received from the DAS. To demonstrate this let's look at an example that will write #00FFh to the configuration register located at address 0110h. The lower byte at address 0110h should receive #FFh and the upper byte at address 0111h should receive #00h as shown below:

0111	0110
B_{15} … B_8	B_7 … B_0
00	FF

Configuration Register (Desired Result)

The most intuitive way to do this is:

```
LDD #00FFh
STD 0110h
```

but this will result in:

0111	0110
B_{15} … B_8	B_7 … B_0
FF	00

Configuration Register (Wrong Approach)

This routine is incorrect because the high-byte (#00h) is written to address 0110h instead of 0111h and the low-byte (#FFh) is written to address 0111h instead of 0110h. The reason for this is that the contents of ACCA are written first followed by those of ACCB. The correct way to do this is to swap the two bytes in the first instruction when writing the code.

```
LDD #FF00h
STD 0110h
```

resulting in:

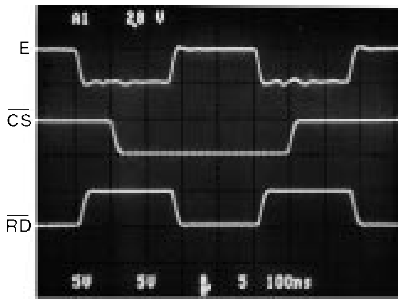
0111	0110
B_{15} … B_8	B_7 … B_0
00	FF

Configuration Register (Correct Approach)

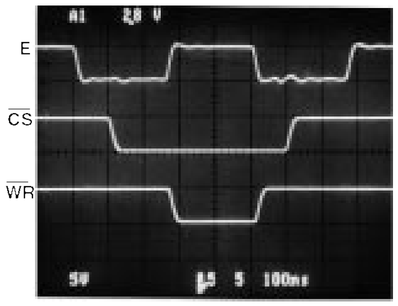
When using ACCD to write data to the DAS, make sure that the ACCD's upper-byte contains the data's lower-byte and its lower-byte contains the data's high-byte. Using ACCD to read from the FIFO and store data in memory results in having the upper and lower bytes of the converted data swapped. This does not cause any problems if the data from memory is interpreted correctly. However, if the user does not want to store the data in this swapped format, ACCD should not be used to write to memory. After the data has been read from the FIFO with the ACCD, accumulators A and B should be used to write the data to memory, not ACCD. Specifically, the data in ACCB should be written to the lower address in memory and the contents of ACCA to the higher address. Now when ACCD is used to read conversion results that are stored in memory, it will have the upper and lower bytes of the data in a more logical format.

4.2 Hardware Description of the Interface Circuitry

The DAS (U7) is seen by the 68HC11 (U1) (Refer to *Figure 4*) as an I/O device placed at a certain location in memory. The interface consists of standard address decoding circuitry using U2, U3, and U4 and demultiplexing of the R/$\overline{W}$ signal line with U5 and U6. U2 is an 8-bit latch that is used to demultiplex the lower address byte from the data byte by using the address strobe signal AS. U4 is an 8-bit comparator that compares the upper address byte to the 8-bit value preset by S1. The value set by S1 determines what upper address byte the DAS is placed at. In this example it is set to 01h. The output of U4 enables U3. U3 is a three bit decoder used to generate the chip select of the DAS. ICs U5 and U6 create the $\overline{RD}$ and $\overline{WR}$ signals to the DAS. Refer to *Figure 3* for the timing diagrams of a read and write cycle. Jumpers J1 and J2 connect the $V_{ref}+$ and $V_{ref}-$ of the DAS to an external source or the on-board 5V supply. Header HDR1 connects the analog inputs to the DAS. The filter capacitors used for VA+, VD+, $V_{ref}+$ $V_{ref}-$, and V_{refout} should be as close to their respective pins as possible. Refer to section 5.0 of the data sheet for power supply and ground considerations.



TL/H/12050–3



TL/H/12050–4

FIGURE 3. Timing Diagrams Showing the E Clock, $\overline{\text{CS}}$, $\overline{\text{RD}}$ and $\overline{\text{WR}}$ Signals

The signals in *Figure 3* were generated by executing the following program:

```
Loop     LDAA $0100     ;Reads the DAS
         STAA $0100     ;Writes to the DAS
         BRA   Loop
```

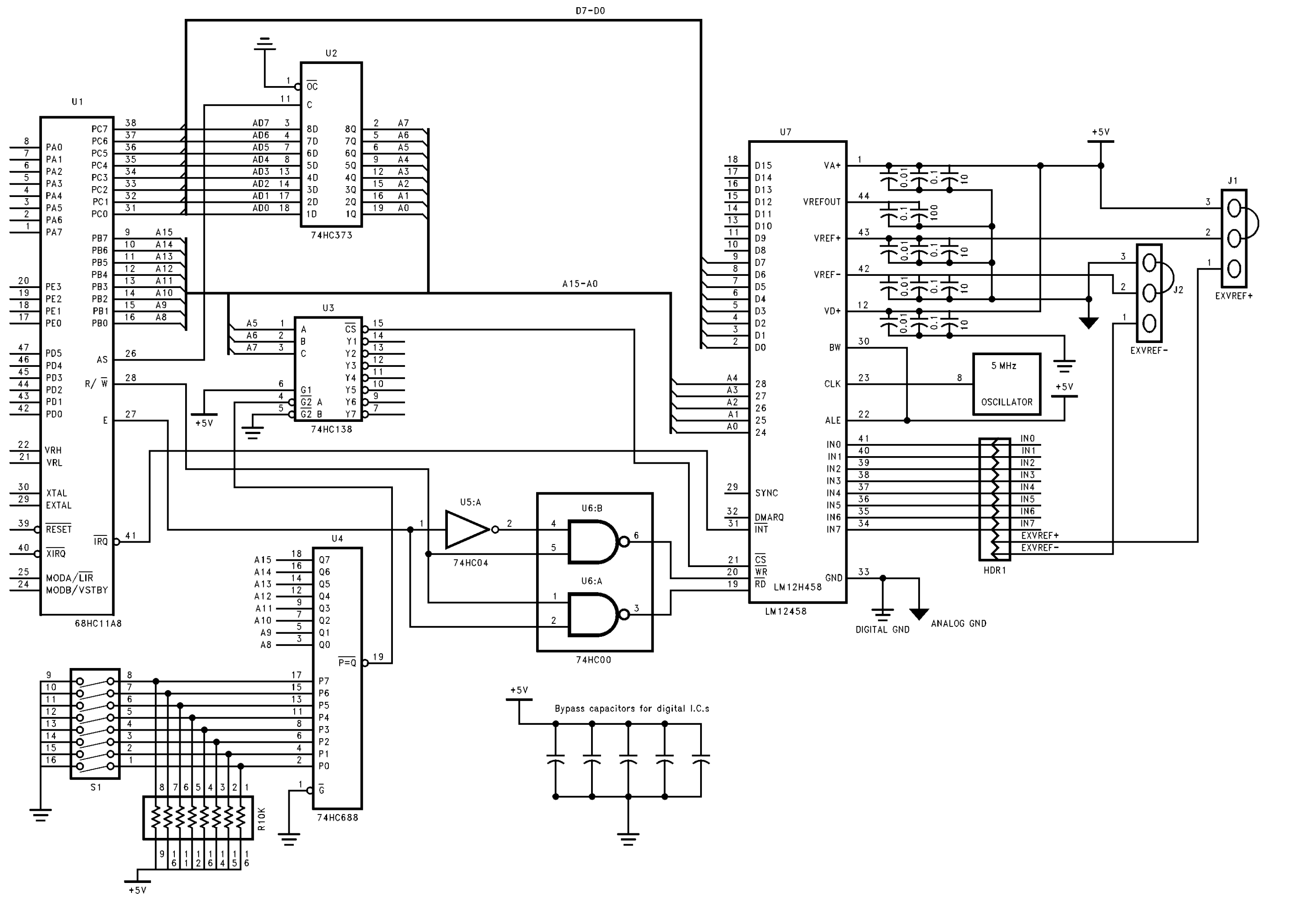
In order to access all the DAS registers, at least twenty seven consecutive memory locations are needed. In this particular example the DAS is placed at address locations 0100h to 011Fh. This address range allows all the DAS registers to be accessed, which are located at addresses 0100h to 011Bh. (Refer to the data sheet for specific addresses of registers.) By selecting the appropriate settings on the dip switch S1, the user can choose the upper adress byte the DAS is placed at.

4.3 Description of the Software

In the first part of the program (*Figure 5*), the DAS is initialized to meet the specifications needed for the furnace example. After the DAS has been initialized and a full auto-calibration is completed, the DAS is reset and instruction execution is started by writing a logic one in the configuration's start bit. The 68HC11 waits for an interrupt. When an interrupt is generated by the DAS, program execution jumps to the interrupt service subroutine. The actions taken by the interrupt service routine are based on which interrupt bits are set in the Interrupt Status Register. The FIFO__FUL interrupt is the only interrupt that is fully serviced by this program. It is important to note that the Interrupt Status register should not be polled because every time $\overline{\text{CS}}$ is pulled low all internal clock-dependent activities of the DAS stop, e.g. the A/D converter, sequencer and timer. Conditions that need to be serviced must be done on an interrupt basis.

When a FIFO full interrupt occurs, the DAS is programmed to empty the five conversions stored in the FIFO starting at the address pointed to by the conversion pointer CONV__POINT. The CONV__POINT holds the address of where data is to begin being stored in memory. In this example, the conversion pointer is initialized to point to address 00h. After reading the five conversions stored in the FIFO and putting them in memory, program execution resumes at the main program following the WAI instruction. This 68HC11 instruction puts the μC in an idle state and waits for an interrupt to occur, at which time program execution will jump to the interrupt service routine. Index register X counts the number of data bytes read from the FIFO and stored in memory. It is also used as an offset to the CONV__POINT to give the address of where the data byte read from the FIFO is to be stored. The value in X is compared against the value CONV__BYTES. When it is greater than CONV__BYTES, program execution is halted. If it is less than CONV__BYTES, the 68HC11 jumps back to the WAI instruction and waits for another FIFO full interrupt to occur. In this example CONV__BYTES is equal to sixty. Since the circuit that was used for this example contains no external memory, the on board microcontroller RAM was used. If external memory is needed the software and board can be easily modified. A memory component can be placed at a certain location in the memory map by using one of the seven available signals from the 74H138 to generate the chip select. The conversion pointer, CONV__POINT needs to be initialized with the beginning address of where converted data is to be stored in memory and CONV__BYTES needs to be set to twice the desired number of conversions. For example, if 8192 samples are to be taken and are to begin being stored at address 1000h, CONV__BYTES should be set to 16384 and CONV__POINT to 1000h.

The DAS__LIMIT interrupt subroutine is intended to service watchdog conditions. In this furnace example no provisions have been made to deal with a watchdog interrupt except that of acknowledging its occurrence and returning to the main program. The manner in which a watchdog condition is handled is system-dependent and must be user-defined. The purpose of the FUL__CAL subroutine is to indicate when the DAS has completed its full auto-calibration cycle. The only action taken in the FUL__CAL interrupt subroutine is a return to the main program.



TL/H/12050–5

FIGURE 4. Schematic Showing the Interface Circuitry

```
INSTR0          EQU     $0100
INSTR1          EQU     $0102
INSTR2          EQU     $0104
INSTR3          EQU     $0106
INSTR4          EQU     $0108
INSTR5          EQU     $010A
INSTR6          EQU     $010C
CONFIG          EQU     $0110
INT_EN          EQU     $0112
INT_STAT        EQU     $0114
TIMER           EQU     $0116
FIFO            EQU     $0118
LMT_STAT        EQU     $011A
FLOW_MAX        EQU     $0FFF           ;User defined limit.
FLOW_MIN        EQU     $0000           ;User defined limit.
PRES_MAX        EQU     $00FF           ;User defined limit.
PRES_MIN        EQU     $1000           ;User defined limit.
TIMER_SET       EQU     $2C0C
CONV_POINT      EQU     $0000           ;User defined. Conversions will begin being stored at
*                                       ;this address.
CONV_BYTES      EQU     $003C           :User defined. This example it is decimal 60.
STAT_HOLD       EQU     $00E0
FIFO_CNT        EQU     $1E
*
**********************************************************************************************
**********************************************************************************************
*
                        ORG $E000
                        CLI             Unmasks interrupt #IRQ.
**********************************************************************************************
**********************  MAIN PROGRAM  *********************************************************
**********************************************************************************************
                        LDY #$0000
                        LDX #$0000
                        STX CONV_POINT       Memory location used to store contents of the interrupt status register.
*
*
*       **************  Initialization of the DAS ***************
*
                        LDD #$0100      Resets the DAS AND SETS THE RAM POINTER TO 00
                        STD CONFIG
*
                        LDD #$0C2A      Instruction 0 is used to set watchdog mode for
                        STD INSTR0      the gas flow measurement with an acquisition delay time set.
```

TL/H/12050–6

FIGURE 5. 68HC11 Assembly Program Listing

```
*
                    LDD #$1008      Instruction 1 used to set watchdog mode for
                    STD INSTR1      pressure measurements.
*
                    LDD #$0000      Initializes instruction 2 in the INSTRUCTION RAM to
                    STD INSTR2      measure T1 single-ended in 12-bit + sign mode.
*
*
                    LDD #$2000      Instruction 3 measures T1-T2 in 12-bit + sign mode
                    STD INSTR3      T2 is at IN1.
*
                    LDD #$4000      Instruction 4 measures T1-T3 in 8-bit mode.
                    STD INSTR4      T3 is at IN2
*
                    LDD #$6024      Instruction 5 measures gas flow in 8-bit mode
                    STD INSTR5      at IN3 with an acquisition delay time set (D=2).
*
                    LDD #1104       Instruction 6 measures pressure in 8-bit mode
                    STD INSTR6      at IN4.
*
                    LDD #$0001      Sets the RAM Pointer to 01.
                    STD CONFIG
*
                    LDD #FLOW_MAX   Sets the watchdog upper limit for maximum flow.
                    STD INSTR0      Limit #1.
*
                    LDD #PRES_MAX   Sets the watchdog upper limit for maximum pressure.
                    STD INSTR1      Limit #1.
*
                    LDD #0002       Sets the RAM Pointer to 10.
                    STD CONFIG
*
                    LDD #FLOW_MIN   Sets the lower watchdog flow limit.
                    STD INSTR0
*
                    LDD #PRES_MIN   Sets the lower watchdog pressure limit.
                    STD INSTR1
*
                    LDD #TIMER_SET  Configures the timer to a preset value so that a
                    STD TIMER       50Hz throughput rate is achieved.
*
                    LDD #$1328      Enabling FIFO full (=5 CONV.),watchdog and
                    STD INT_EN      auto-cal interrupts.
*
```

TL/H/12050–7

FIGURE 5. 68HC11 Assembly Program Listing (Continued)

```
                        LDD INT_STAT    Reads interrupt status register to clear it.
*
                        LDD #$0800      Starts a full auto-calibration cycle.
                        STD CONFIG      An interrupt will be generated when done.
*
*
                        WAI            Waits for interrupt to occur. When an interrupt is generated
*                                      program control resumes at the interrupt service routine.
*
*       *************** STARTS CONVERSION ******************
*
                        LDD #$0200      Resets the DAS
                        STD CONFIG
*
                        CLI             Enables 68HC11 interrupt.
                        LDD #$8100      Starts conversion with sync out.
                        STD CONFIG
*
*
CONV                    WAI                     Waits for interrupt to occur. When an interrupt is generated
*                                               program control resumes at the interrupt service routine.
*
                        CPX #CONV_BYTES         Compares number of conversion bytes (value stored in register X)
                        BLT CONV                to decimal number 60. Two bytes per conversion.
                        STOP                    Program execution is halted.
*
**********************  END OF MAIN PROGRAM  *******************************************************
*
*     **************** INTERRUPT SERVICE ROUTINE *****************
*
*
*
IRQ                     LDAA INT_STAT   Reads the lower byte of the interrupt status register
                        STAA STAT_HOLD  and stores the result at address STAT_HOLD.
*
                        LDAA STAT_HOLD
                        ANDA #$04       Checks bit 2 to determine if FIFO full was
                        BGT FIFO_FUL    the source of the interrupt. If so, continue at FIFO_FUL subroutine.
*
*
                        LDAA STAT_HOLD
                        ANDA #$01       Checks bit 0 to determine if watchdog limit was the
                        BGT DAS_LIMIT   source of the interrupt. If so, resume program at
*                                       DAS_LIMIT subroutine.
```

TL/H/12050–8

FIGURE 5. 68HC11 Assembly Program Listing (Continued)

```
*
                        ANDA #$10        Checks bit 4 to determine if full cal limit caused interrupt.
                        BGT FUL_CAL      If so, continue program control at Ful_Cal subroutine.
*
                        RTI
*
*      **************** READ_FIFO SUBROUTINE WILL TAKE CONVERSIONS ****************************************
*      **************** STORED IN FIFO AND PLACE THEM IN A MEMORY LOCATION.********************************
*
FIFO_FUL                LDY #$0000               Index register Y used as a local counter.
*
LOOP                    LDD FIFO                 Loads the contents of the FIFO in ACCD and increments
*                                                the FIFO pointer.
                        STD CONV_POINT,X         X is used as a global counter and holds the total number of conv bytes.
*                                                Stores the result read from the FIFO at address CONV_POINT + X.
                        INX                      Increments index register X by two for every conversion read.
                        INX                      pointer to the next memory location where data is to be stored.
                        INY
                        INY
                        CPY #$03
                        BLE LOOP                 Escape from loop when Y counter exceeds 3.
                        RTI
*      *****************END OF READ_FIFO SUBROUTINE *******************************************************
*
*
*      **************** DAS_LIMIT SUBROUTINE IS USER DEFINED AND MUST***************************
*      **************** TAKE THE APPROPRIATE MEASURES WHEN WATCHDOG INTERRUPTS******************
*      **************** ARE GENERATED.                                          ******************
*
DAS_LIMIT
*                                The user should check to see whether the interrupt was caused
*                                by pressure or gas flow exceeding their limits and then act
*                                accordingly.
*
                        RTI
*      *****************END OF DAS_LIMIT SUBROUTINE ********************************************
*
*
*      *****************FULL-CAL ROUTINE*********************************************************
FUL_CAL                 RTI
*****************************************************************************************************
*
*      ***************** END OF INTERRUPT SERVICE ROUTINE ****************************************************
```

TL/H/12050–9

FIGURE 5. 68HC11 Assembly Program Listing (Continued)

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