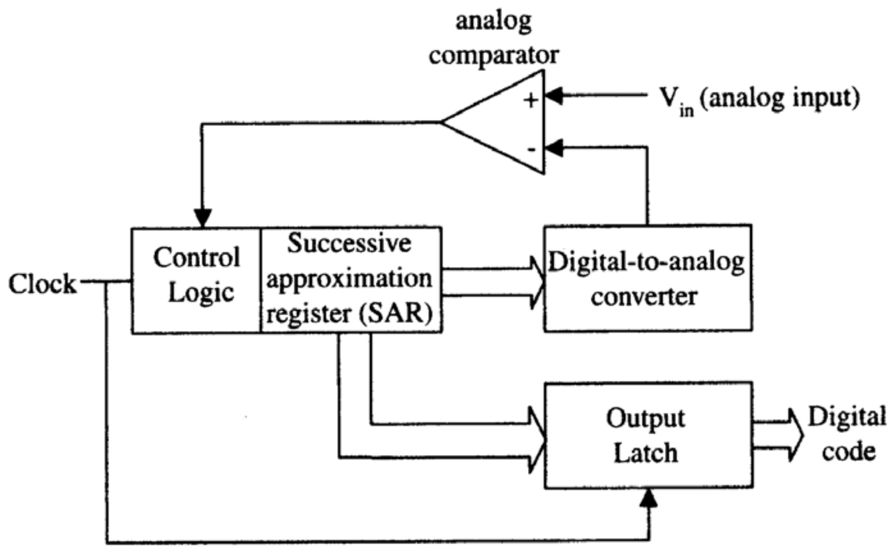
Analog to Digital Converters

The MC9S12 uses a Successive Approximation Register (SAR) Type A/D converter. A Block Diagram is shown. A Successive Approximation Register A/D converter uses an intelligent scheme to determine the input voltage.



Operation

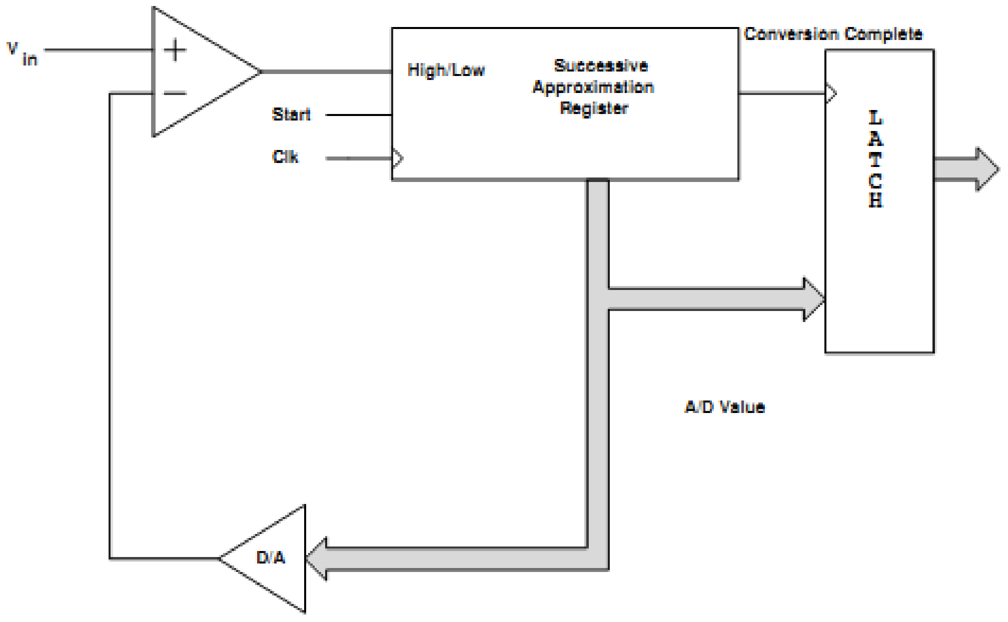
The control logic circuit first loads a digital value half way between V_{RH} and V_{RL} into the SAR register. This digital value is converted to an analog value by the D/A converter and compared the analog V_{IN} voltage.

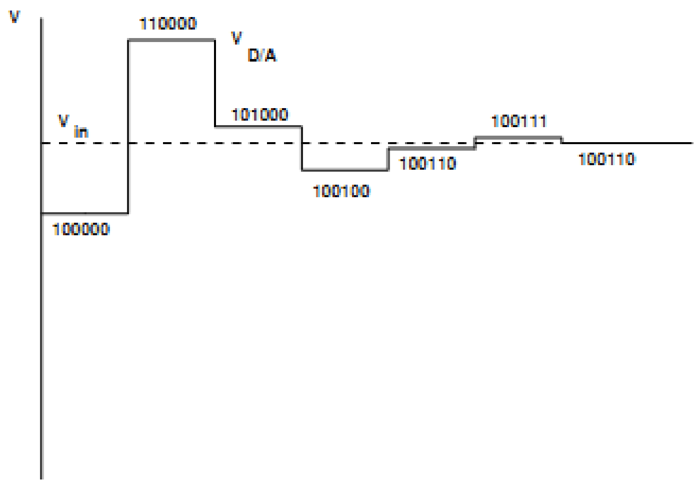
- If the V_{IN} input voltage is greater than the output of the D/A converter (V_{IN} is in the upper half of the range), the control logic sets the most significant bit of the digital output value.

- If the V_{IN} input voltage is less than the output of the D/A converter (V_{IN} is in the lower half of the range), the control logic clears the most significant bit of the digital output value.

The first clock cycle eliminates half of the possible values. On the next clock cycle, the A/D converter tries a new digital value in the middle of the remaining possible values. The second clock cycle allows the converter to determine the second most significant bit of the result.

Each successive clock cycle reduces the range another factor of two. For a N-bit SAR A/D converter, it takes N clock cycles to determine the value of the input voltage



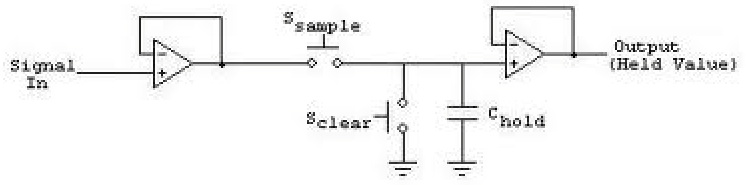


A/D Parameters

Number of bits – N The more bits the better the resolution and the longer the time required to do a conversion.

Conversion Time - A/D converters take time to do a conversion. Conversion time = number of bits (N) x period of clock (T). If the time to do a conversion is long enough that the input signal will change significantly during the conversion process, a sample and hold circuit is commonly used.

Sample and Hold Circuit



When the S_{Sample} switch closes, the capacitor C_{hold} charges up to the value of the input signal voltage. This voltage is a constant and is independent of variations in the input voltage.

Low Reference Voltage - $V_{L,REF}$ - usually 0 Volts

High Reference Voltage – $V_{H,REF}$ - usually set to the supply voltage V_{CC}.

Converter Range = $V_{H,REF}$ - $V_{L,REF}$ = 5 V – 0 V = 5 V

Ratiometric A/D Converters are converters where :

A 0 Volt analog input is converted to a digital value of 0
A V_{CC} analog input is converted to a digital value of $2^n - 1$
An analog voltage of A volts is converted to a digital output value of:
Digital Output = Analog Input x (2^n - 1)

Calculating an Analog Input Voltage for a given Digital Output value

$V_{Analog} = V_{L,REF}$ + (range x Digital Input)/ 2^n - 1

Example Calculations

A 12 bit A/D converter has $V_{L,REF}$ = 0 Volts and $V_{H,REF}$ = 5 V.

What analog voltage values correspond to the digital outputs of 100, 800 and 2400?

Range 5 V – 0 V = 5 V

Digital Value of 100
Analog voltage V = 0 V + (100 x 5 V) / (2^{12} – 1) = 0.12 V

Digital Value of 800
Analog voltage V = 0 V + (800 x 5 V) / (2^{12} – 1) = 0.98 V

Digital Value of 2400

Analog voltage $V = 0\text{ V} + (2400 \times 5\text{ V}) / (2^{12} - 1) = 2.93\text{ V}$

An 8 bit A/D converter has $V_{L,REF} = 0$ Volts and $V_{H,REF} = 4.90$ V.

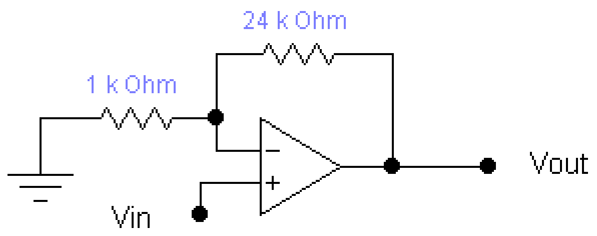
What is the digital output value for an input of 3.5 V?

Digital O/P = $3.5/4.90(2^8 - 1) = 182$ rounded off to nearest digital value

Voltage Scaling Circuits

Some transducers do not have a voltage range that exactly matches the range of the A/D converter.

Example: A sensor has an output range of 0 to 200 mV. Design an Op Amp circuit to scale this range to 0 to + 5 V



The gain of the Op Amp amplifier must be

$A_V = 5\text{ V}/200\text{ mV} = 25$

For a non-inverting amplifier gain is

$Av = 1 + R_F/R_1$ so

$R_F = 24\text{ k}\Omega \quad R_1 = 1\text{ k}\Omega$

HCS12 A/D Converter

The HCS12 family of micro-controllers has one or two 8 channel, 10 bit A/D converters. The MC9S12DP256B derivative has two 8 channel, 8/10 bit A/D converters with a Sample and Hold circuit. Each A/D has eight analog inputs with pins that are numbered AN0 – AN 15. Pins AN0 – AN7 are used by converter AD0 and pins AD8 – AN15 are used by converter AD1.

The analog MUX selects one of the three internal or one of the external signal sources for conversion. The sample selected is amplified and then presented to the capacitor in the Sample and Hold circuit.

A Sample and Hold circuit is used to latch rapidly changing inputs signals. The sample process has two stages. The sample amplifier pre-charges the sample capacitor for 2 clock cycles before it gets connected to the input voltage for a programmable 2, 4, 8 or 16 clock cycles.

The A/D uses a successive-approximation method to perform the conversion. An 8 bit conversion takes 6 µsec while a 10 bit conversion requires 7 µsec. The maximum clock conversion frequency is 2 MHz. This is derived from the E Clock by a programmable pre-scale value.

Block Diagram of HCS12 A/D Converter

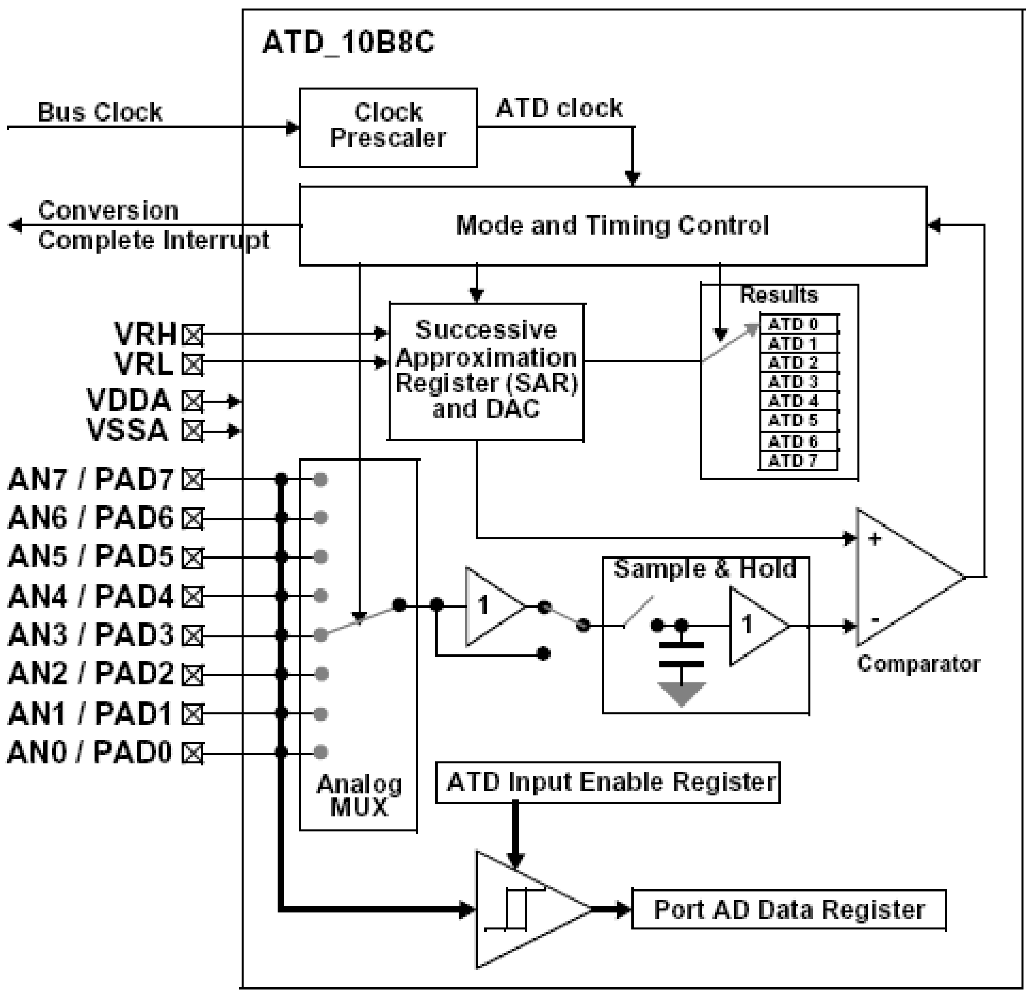


Figure 1. ATD 10B8C Block Diagram

Programming the HCS12 A/D Converter

This handout was derived from notes by (S. Tewksbury)
http://stewks.ece.stevens-tech.edu/CpE390-Main/NotesProgramming/Peripherals/

These notes have been provided to summarize the setup and operation of the HCS12's A/D (analog-to-digital) conversion system. The HCS12 has 8 analog inputs, each of which can be converted to digital and read by your program. Associated with operation of the A/D converter are several registers (control registers that need to be properly set, status registers that are used to determine the state of the A/D converter, and data registers containing the 8-bit digital numbers provided by the input analog channels.

The register addresses are valid for the Freescale MC9S12DP256B derivative.

1. Setting Up the A/D Input Enable Register - ATD0DIEN ($008D)

This register allows the user to enable a PORT ADx pin as a digital input or A/D use.

7	6	5	4	3	2	1	0
IEN7	IEN6	IEN5	IEN4	IEN3	IEN2	IEN1	IEN0

0 – disable digital input buffer to PTADx

1 – enable digital input buffer to PTADx

For A/D operation all bits must be disabled - set to a 0

2. Setting Up the A/D Control Registers

Eight-bit control registers are used to "control" the A/D system. There are a total of six A/D control registers involved for each of the two A/D converters (ATD0 and ATD1) , each being accessed at the memory location given.

1. ATD0CTL0 ($0080): This is unlikely to be used by you (it allows a conversion to be aborted). You can ignore it.
2. ATD0CTL1 ($0081): This is unlikely to be used by you (used for special test conditions). You can ignore it.
3. **ATD0CTL2** ($0082): This is used to turn on the A/D (it is normally initialized to OFF to conserve power).
4. **ATD0CTL3** ($0083): This is used to determine the number of conversions per sequence.
5. **ATD0CTL4** ($0084): This controls the timing of the A/D conversions. You will need to set it.
6. **ATD0CTL5** ($0085): This is used to select the conversion modes and start an A/D conversion.

ATD0CTL2 Settings – address $0082

The register is illustrated below. All entries are set to 0 on reset (when A/D is turned on)

Bit	7	6	5	4	3	2	1	0
Name	**ADPU**	AFFC	AWAI	ETRIGLE	ETRIGP	ETRIGE	ASCIE	ASCIF
Reset Value	0	0	0	0	0	0	0	0
Suggested Value	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

The fields are as follows:

- ADPU: ADPU = 1, turns A/D on. This must be done before other registers are set. When board is reset, the bit is set to 0 (A/D off). IMPORTANT: YOU MUST WAIT 100 MICROSECONDS AFTER TURNING THE A/D ON BEFORE DOING ANYTHING ELSE WITH THE A/D SYSTEM.
- AFFC: Fast flag clear when set to 1. You need not worry about this.
- AWAI: Turns A/D off when S12 is set to low power WAIT state. Value = 1, halt during Wait mode; value = 0, continue to run in Wait Mode. You need not worry about this.
- *ETRIGLE, ETRIGP, ETRIGE:* These control the use of an external clock to operate the A/D. ETRIGE = 1, external trigger enabled; ETRIGE=0, external trigger disabled. ETRGLE and ETRIGP set the sensitivity and the polarity of the external trigger.

ETRIGLE	ETRIGP	External Trigger Sensitivity
0	0	Falling Edge
0	1	Rising Edge
1	0	Low level
1	1	High level

- ASCIE: Enables interrupts if set to 1 (we will be using polling so this should be set to 0, the reset value).
- ASCIF: A "conversion complete" flag used for interrupts. You need not worry about this.

ATD0CTL3 Settings – address $0083

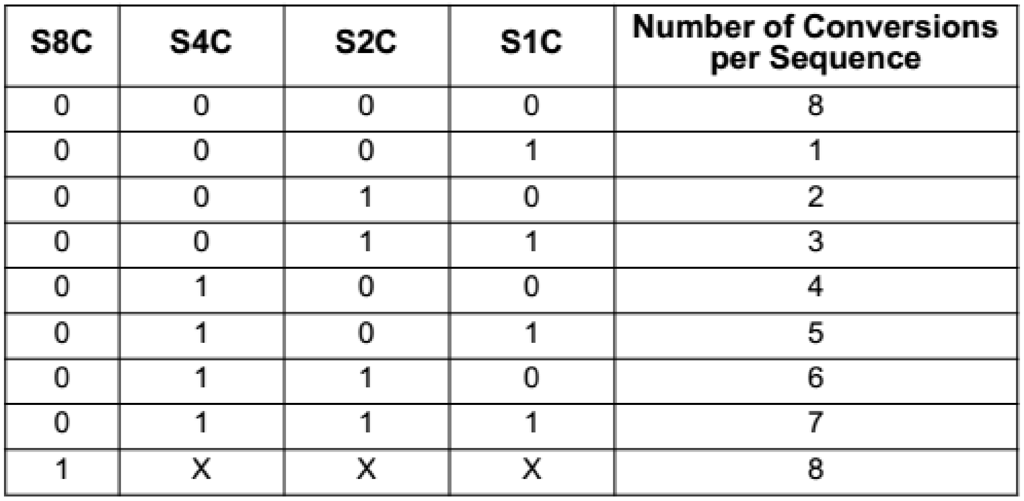
The register is illustrated below. All entries are set to zero on reset (when A/D is turned on)

Bit	7	6	5	4	3	2	1	0
Name	-	S8C	S4C	S2C	S1C	FIFO	FRZ1	FRZ2
Reset Value	0	0	0	0	0	0	0	0
Suggested Value	0	X	**X**	**X**	**X**	**0**	**0**	**0**

X – can be 0 or 1 as required

The fields are as follows:

- *S8C - S1C:* Conversion Sequence Length. Number of conversions per sequence (between 1 and 8) equals the binary value associated with this number (special case - 0000 and 1xxx both give a length of 8.



S8C	S4C	S2C	S1C	Number of Conversions per Sequence
0	0	0	0	8
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	X	X	X	8

- *FIFO:* Concerns the resetting of the sequence - not considered further here
- *FRZ1, FRZ2:* Background debug freeze enables - not considered further here..

ATD0CTL4 Settings – address $0084

The register is illustrated below. All entries are set to zero on reset (when A/D is turned on)

Bit	7	6	5	4	3	2	1	0
Name	SRES8	SMP1	SMP0	PRS4	PRS3	PRS2	PRS1	PRS0
Reset Value	0	0	0	0	0	0	0	0
Suggested Value	X	**0**	**0**	**0**	**0**	**1**	**0**	**1**

The fields are as follows:

- *SRES8:* A/D resolution (8-bit or 10-bit).

  10-bit = 0 8-bit = 1

- SMP1/SMP0: These set the length of the second phase of the conversion time: The common time for applications is 2 A/D conversion clock periods

SMP1	**SMP0**	**Length of 2nd phase of Sample Time**
0	0	2 A/D conversion clock periods
0	1	4 A/D conversion clock periods
1	0	8 A/D conversion clock periods
1	1	16 A/D conversion clock periods

- PRS4 - PRS0 is a 5-bit binary number defining the ATD clock frequency in terms of the HCS12 E-clock (the system HCS12 clock). The ATD clock rate is derived from the E-clock rate by

  $$\text{ATD clock} = \frac{\text{E - clock}}{(\text{PR}+1)*2}$$

  where PR is the 5-bit Pre-Scale value given by the PRS4 – PRS0 values.
  The ATD clock frequency has a minimum value of 500 kHz and a maximum value of 2 MHz. A prescale value of 00101 will divide the E Clock by a factor of 12.

  $$\text{ATD clock} = \frac{24\ \text{MHz}}{(5+1)*2} = 2\ \text{MHz}$$

  The largest pre-scale value is 23. The ATD clock using this value is 500 kHz. There are other pre scale values between 5 and 23 that will work also.

  $$\text{ATD clock} = \frac{24\ \text{MHz}}{(23+1)*2} = 0.5\ \text{MHz}$$

2. Starting an A/D Conversion (Write to ATD0CTL5 address $0085)

All entries are set to zero on reset (when A/D is turned on). Each time you want to perform an A/D conversion, you write to this register. The entries need not be changed - the simple act of writing, even if writing the same thing that is already in the register, starts the conversion.

Bit	7	6	5	4	3	2	1	0
Name	DJM	DSGN	SCAN	MULT	-	CC	CB	CA
Reset Value	0	0	0	0	0	0	0	0
Suggest Value	X	**X**	**X**	**X**	**0**	**X**	**X**	**X**

The A/D can operate either on a single input analog channel or can sequence through the 8 input analog channels (converting one after another = sequential conversions). You will likely be using the "single conversion sequence" for a single input channel.

- *DJM:* Result Register Data Justification. Registers are 16-bits long. The issue is whether an 8-bit result goes into the lower or upper half of the Result Register.

  DJM = 0, left justifies data in results register.

  DJM = 1, right justifies data in result registers.

- *DSGN*: Result Register Data Signed or Unsigned.

  DSGN = 0, unsigned data.

  DSGN = 1, signed data

Note: All signed data conversions must be left justified

- *SCAN:* Continuous Conversion Sequence Mode. Determines whether you take only one A/D sample or whether you have the A/D running at a continuous clock rate.

  SCAN = 0, taking a single sample.

  SCAN = 1, taking continuous samples

- *MULT:* Multi-Channel Sample Mode. Selects between taking samples from only a single input channel or taking samples from multiple channels in sequence.

  MULT = 0, sample only one channel.

  MULT = 1, sample across several channels.

In single channel mode the ATD sequence controller samples only from the specified analog input channel for an entire conversion sequence. The analog channel is selected by channel selection code (control bits CC/CB/CA located in ATDCTL5)

If multi-channel, bits S8C, S4C, S2C, and S1C in ATDCTL3 register specify how many channels (1 to 8 channels) and bits CA, CB, and CC above specifies the first channel in the sequence.

- *Bit 3:* Unused.
- CC, CB, CA. In multi-channel mode the ATD sequence controller samples across channels. The number of channels sampled is determined by the sequence length value (S8C, S4C, S2C, S1C). The first analog channel examined is determined by channel selection code (CC, CB, CA control bits); subsequent channels sampled in the sequence are determined by incrementing the channel selection code. In the case of multi-channel scans (MULT=1), this selection code represents the first channel to be examined in the conversion sequence. Subsequent channels are determined by incrementing channel selection code; selection codes that reach the maximum value wrap around to the minimum value.

CC	CB	CA	Analog Input Channel
0	0	0	AN0
0	0	1	AN1
0	1	0	AN2
0	1	1	AN3
1	0	0	AN4
1	0	1	AN5
1	1	0	AN6
1	1	1	AN7

3. When Has an A/D Conversion Process been Completed

The S12 A/D conversion system contains a **16 bit status register** ATD0STAT at memory locations $0086 and $0087. These are ATD0STAT0 at location $0086 and ATD0STAT1 at location $0077. These registers provide the status of the A/D system (namely when a started conversion has completed. The two registers are shown below.

The bit **SCF** is the **S**ingle **C**hannel **F**lag in ATD0STAT0 indicates that the conversion for a single channel conversion has been completed and the data is ready to be read from the A/D system. After starting the A/D conversion, your program will "poll" this bit to determine when the conversion has been completed and, when set to 1, your program can read the data.

ATD0STAT0 (Memory Location $0086)								
Bit	7	6	5	4	3	2	1	0
Name	SCF	0	ETORF	FiFOR	0	CC2	CC1	CC0
Value at conversion start	0	0	0	0	0	0	0	0
Value when conversion done	1	0	0	0	0	0	0	0

- *SCF:* Sequence Complete Flag. SCF = 1, conversion sequence completed; SCF = 0, conversion sequence not completed.
- *ETORF:* External Trigger Overrun Flag. We will not discuss this.
- *FiFOR:* FIFO Overrun Flag: We will not discuss this.
- *CC2, CC1, CC0:* These 3 read-only bits are the binary value of the conversion counter. The conversion counter points to the result register that will receive the result of the current conversion. CC2=1, CC1=1, CC0=0 indicates that the result of the current conversion will be in ATD Result Register 6.

ATD0STAT1 (Memory Location $0087)								
Bit	7	6	5	4	3	2	1	0
Name	CCF7	CCF6	CCF5	CCF4	CCF3	CCF2	CCF1	CCF0
Value at conversion start	0	0	0	0	0	0	0	0
Value when conversion done	x	x	x	x	x	x	x	x

ATD0STAT1 is used when the A/D converter is used in the channel sequence mode (sequencing through the various channels). CCFx is set to *1* when the conversion for channel x has completed. We are setting up for single channel conversions (not sequencing through the input channels). Therefore, we will not be using the ATD0STAT1 status information.

4. Reading the Data Obtained in the A/D Conversion

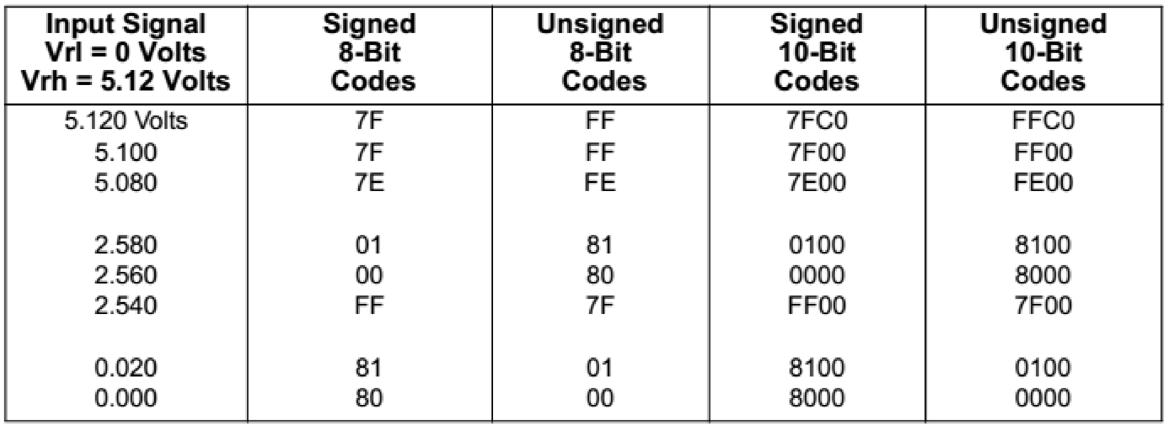
The S12 has eight input analog signal channels (channels 0 through 7). After an A/D conversion of the signal on one of the channels has completed, the results are available in the corresponding A/D Data Register (also called A/D Result Register). Each of these registers is a 16-bit register (to allow for 10-bit A/D conversions). The registers holding data for the various input analog channels are as follows.

Analog Input Channel	A/D Register Holding Conversion Result	Memory Address of Register Holding Result
0	ATD0DR0	$0090
1	ATD0DR1	$0092
2	ATD0DR2	$0094
3	ATD0DR3	$0096
4	ATD0DR4	$0098
5	ATD0DR5	$009A
6	ATD0DR6	$009C
7	ATD0DR7	$009E

5. Programming Steps

When writing your program, you will be turning on the A/D converter by writing (using the suggestions above) *%10000000* to ATD0CTL2 at memory location $0082. Next, you would need to insert the delay loop to wait 100 microseconds. Then you can proceed to set up the other control registers, when ready write to ATDCTL5 to start a conversion, enter another program loop reading the SCF bit of the ATD0STAT0 status register (exiting when that bit is set to 1 signifying that the conversion is complete, and then reading the result register(s) holding the data.

6. Sample of Expected D/A Output Results



Input Signal Vrl = 0 Volts Vrh = 5.12 Volts	Signed 8-Bit Codes	Unsigned 8-Bit Codes	Signed 10-Bit Codes	Unsigned 10-Bit Codes
5.120 Volts	7F	FF	7FC0	FFC0
5.100	7F	FF	7F00	FF00
5.080	7E	FE	7E00	FE00
2.580	01	81	0100	8100
2.560	00	80	0000	8000
2.540	FF	7F	FF00	7F00
0.020	81	01	8100	0100
0.000	80	00	8000	0000

All results based on left justified, signed and unsigned ATD output values.

Sample Calculations

10 bit unsigned conversion for V_{IN} = 4.0 V
Digital O/P = (FFC0/5.12) * 4 = C7CE

10 bit signed conversion for V_{IN} = 4.0 V
Digital O/P = (7FC0/5.12) * 4 = 63CE

10 bit unsigned conversion for V_{IN} = 1.0 V
Digital O/P = (FFC0/5.12) * 1 = 3FF0

10 bit signed conversion for V_{IN} = 1.0 V
Digital O/P = (7FC0/5.12) * 1 + 8000 = 98F4

A/D Programming Exercises

Determine the values for configuring ATD0CTL2, ATD0CTL3, ATD0CTL4 and ATD0CTL5 to achieve:

a) 8 bit conversion, single channel conversion, 2 samples/conversion sequence, sampling the signal from the temperature sensor input on the Demo Board, unsigned data is stored flush right.

b) 10 bit conversion, multiple channel conversion, 2 samples/conversion sequence, sampling the signal from the potentiometer and temperature sensor inputs, signed data is stored flush left.

Answers:

Part a

```
admask2:    equ   %10000000   ;ADPU = 1 Enable A/D
admask3:    equ   %00010000   ;do 2 conversions
admask4:    equ   %10000101   ;8 bit conversion, 2 A/D conversion cycles
admask5:    equ   %10000001   ;DJM=1,unsigned,single scan/channel, channel 1
```

Part b

```
admask2:    equ   %10000000   ;ADPU = 1 Enable A/D
admask3:    equ   %00010000   ;do 2 conversions
admask4:    equ   %00000101   ;10 bit conversion,2 A/D conversion cycles
admask5:    equ   %01010001   ;DJM=0, signed, multiple scans channels 1and 2
```

Pre-scale values PRS4 to PRS0 and SMP0, SMP1 parameters are unchanged.

Sample Code for programming the A/D Converter

Assembler Code - see A/D Lab

C Code

```
//***************************************
//        A/D Converter in C
//****************************************

#include "derivative.h"          // derivative-specific definitions
#include <hidef.h>               // common defines and macros
#define PAD0 0                   // A/D input 02
#define C0F 0x1                  // end of conversion
#define OC0 0x1                  // select O/P compare 0
void wait100usec(void);

void main()
{
        unsigned int result[4];        //initialize an array of 4 for results
        ATD0DIEN &= (~PAD0);           //initialize PORTAD0 for A/D use
        ATD0CTL2 = 0x80;               // power on ATD
        wait100usec();                  // delay for 100 usec
        ATD0CTL3 = 0x20;               // set x conversions
        ATD0CTL4 = 0x85;               // y bit conversion, pre-scale = 00101
        ATD0CTL5 = 0x82;               // start A/D conversion
        while (!(ATD0STAT0 & 0x80));       // wait for conversion to finish
           result[0] = ATD0DR0;        // store results
           result[1] = ATD0DR1;
           result[2] = ATD0DR2;
           result[3] = ATD0DR3;

asm("swi");
}

//  You will learn how this delay function works at a later time

void wait100usec(void)
{
  TSCR1 = 0x90;
  TSCR2 = 0x0;
  TIOS  |= OC0;
  TC0 = TCNT + 2400;
  while (!(TFLG1 & C0F));
}
```

Questions

1. How many conversions are being done?

2. Is the converter programmed for single or continuous conversions, unsigned or signed data, left or right justified?

3. What channel is being used?

Application Note
Written by David Lloyd
Computer Engineering Program
Humber College