Lecture 23

March 19, 2012

Analog/Digital Converters

- The MC9S12 A/D Converter
- Single Channel vs Multiple Channels
- Single Conversion vs Multiple Conversions
- MC9S12 A/C Registers
- Using the MC9S12 A/D Converter
- A C program to use the MC9S12 A/D Converter

Analog/Digital Converters

- A 10-bit A/D converter is used to convert an input voltage. The reference voltages are $V_{RL} = 0$V and $V_{RH} = 5$V.
  - What is the quantization level of the A/D converter?

$$\Delta V = \frac{V_{RH} - V_{RL}}{2^b - 1} = 4.88 \text{ mV}$$

- What is the dynamic range of the A/D converter?

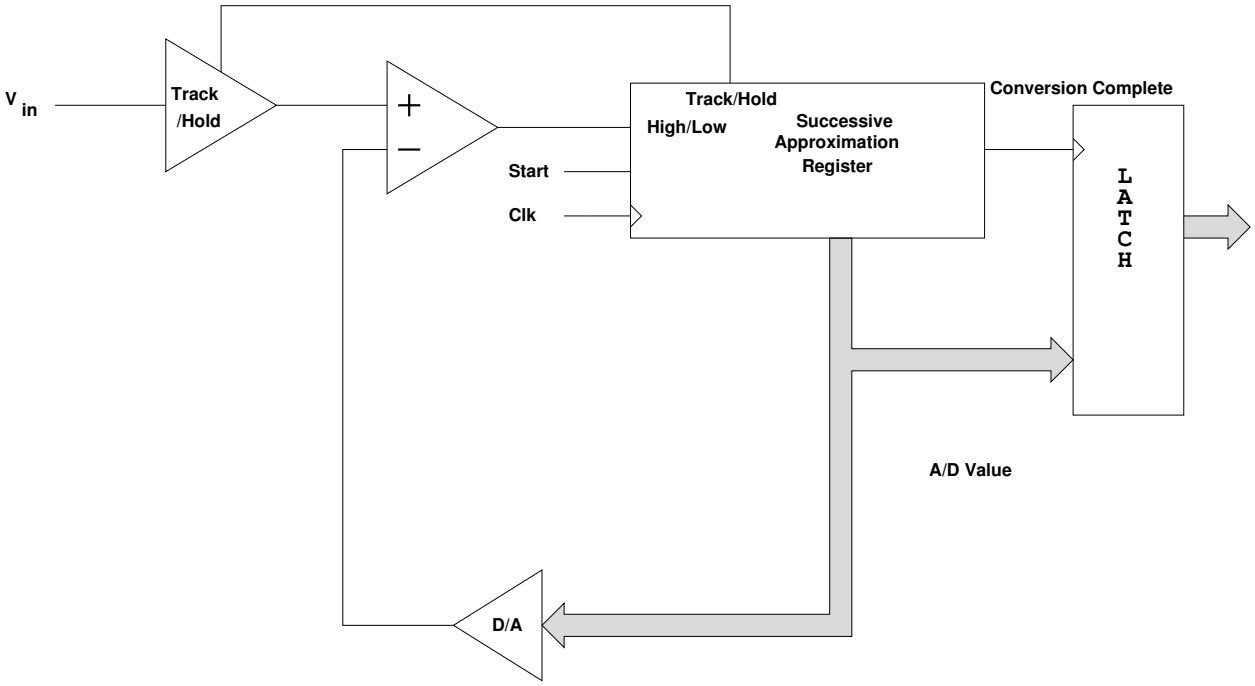
$$\text{DR}_{\text{dB}} = 6.02b = 60.2 \text{ dB}$$

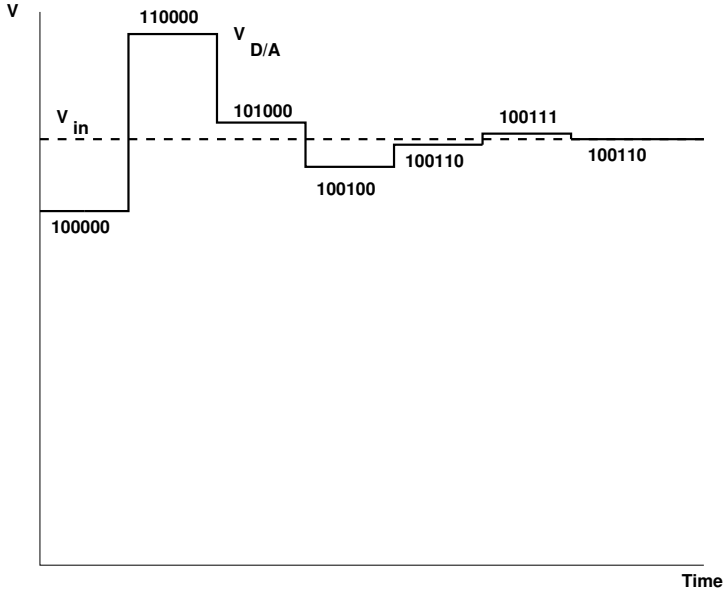
- If the value read from the A/D converter is `0x15a`, what is the input voltage?

$$V_{in} = V_{RL} + \frac{V_{RH} - V_{RL}}{2^b - 1}\text{ADvalue} = 0 \text{ V} + 4.88 \text{ mV} \times 346 = 1.6894 \text{ V}$$

- The MC9S12 has two 10-bit A/D converter (ATD0 and ATD1).
  - Each A/D converter has an 8-channel analog mulitplexer in front of it, so each channel can convert 8 analog inputs (but not at exactly the same time).
- ATD0 uses the eight bits of Port AD0, called PAD00 through PAD07
  - Ports AD0 and AD1 of ATD0 are used by DBug-12 at startup to determine whether to execute DBug-12, or to run code from EEPROM of the bootloader.
- ATD1 uses the eight bits of Port AD1, called PAD08 through PAD15

SUCCESSIVE APPROXIMATION A/D CONVERTER



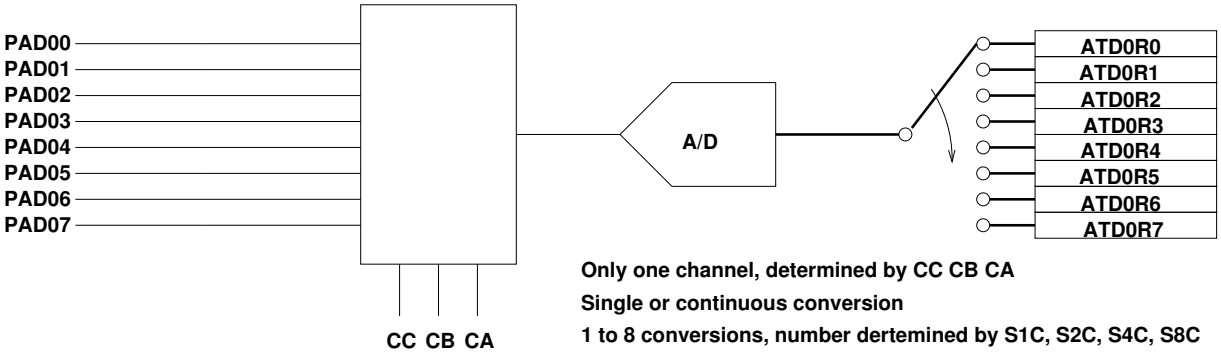


The MC9S12 Analog/Digital Converter

- We will discuss only ATD0. ATD1 is identical.
- ATD0 is an eight-channel 10-bit A/D converter.
  - The A/D converter can also be used in 8-bit mode.
- There are eight inputs to the A/D converter.
- The inputs are fed through a multiplexer to the single A/D converter.
- There are inputs on the MC9S12 for the reference voltages V_{RL} and V_{RH}
  - On the Dragon12 board, $V_{RL} = 0$ V and $V_{RH} = 5$ V.
  - You must have $V_{SS} \leq V_{RL} < V_{RH} \leq V_{DD}$.
  - The accuracy of the A/D converter is guaranteed only for $V_{RH} - V_{RL} = 5$ V.
- When using the A/D converter, you can choose between performing **single** or **continuous conversion** on a **single channel** or `multiple channels`.
- The AD conversion results are stored in the registers `ATD0DR0` through `ATD0DR7`
  - You can choose whether to have the results left-justified or right-justified.
- To program the MC9S12 A/D converter you need to set up the A/D control registers `ATD0CTL2`, `ATD0CTL3`, `ATD0CTL4` and `ATD0CTL5`
- The registers `ATD0CTL0` and `ATD0CTL1` are used for factory test, and not used in normal operation.
- When the AD converter is not used, Port AD0 can be used for general purpose input
  - Register `ATD0DIEN` is used to set up Port AD0 pins for use as a general purpose inputs.
  - The values on the pins are read from `PORTAD0`.

MC9S12 A/D Converter Setup

MULT = 0

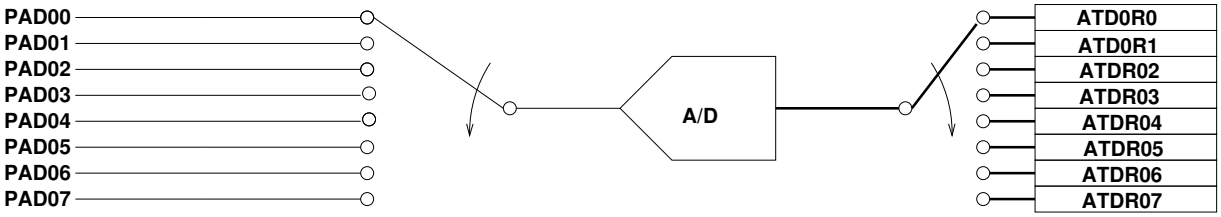


Only one channel, determined by CC CB CA

Single or continuous conversion

1 to 8 conversions, number dertemined by S1C, S2C, S4C, S8C

MULT = 1



Several Channels.

Starting channel determined by CC CB CA

1 to 8 conversions, number determined by S1C, S2C, S4C, S8C

ATD0CTL2	**ADPU**	AFFC	ASWAI	ETRIGLE	ETRIGLP	0	**ASCIE**	**ASCIF**	**0x0082**
ATD0CTL3	0	**S8C**	**S4C**	**S2C**	**S1C**	FIFO	FRZ1	FRZ0	**0x0083**
ATD0CTL4	**SRES8**	SMP1	SMP0	PRS4	PRS3	PRS2	PRS1	PRS0	**0x0084**
ATD0CTL5	**DJM**	**DSGN**	**SCAN**	**MULT**	**0**	**CC**	**CB**	**CA**	**0x0085**
ATD0STAT0	**SCF**	0	ETORF	FIFOR	0	CC2	CC1	CC0	**0x0086**
ATD0STAT1	CCF7	CCF6	CCF5	CCF4	CCF3	CCF2	CCF1	CCF0	**0x008B**

To Use A/D Converter:

ADPU = 1 (Power up A/D)

SCAN = 0 => Single conversion sequence

SCAN = 1 => Convert continuously

S8C, S4C, S2C, S1C: Number of conversions per sequency: 0001 –– 0111 (1 to 7)
0000 or 1xxx (8)

SRES8 = 0 => 10 Bit Mode
SRES8 = 1 => 8 Bit Mode

DJM = 0 => Left justified data in the result registers
DJM = 1 +> Right justified data in the result registers

DSGN = 0 => Unsigned data in the result registers
DSGN = 1 => Signed data representation in the result registers (only for left justified)

ATDCTL4 = 0x85 => 2 MHz AD clock, 12 cycles per conversion, 8 bit mode

ATDCTL4 = 0x05 => 2 MHz AD clock, 14 cycles per conversion, 10 bit mode

Other values of ATDCTL4 will not work, or will result in slower operation of A/D

SCF Flag is set after a sequence of conversions is complete
The SCF Flag is cleared when ATD0CTL5 is written, or by writing a 1 to the SCF bit

After writing to ATD0CTL5, SCF flag cleared and conversions start

USING THE MC9S12 A/D CONVERTER

1. Power up A/D Converter (`ADPU = 1` in `ATD0CTL2`)
2. Select number of conversions per sequence (`S8C S4C S2C S1C in ATD0CTL3`)
   S8C S4C S2C S1C = 0001 to 0111 for 1 to 7 conversions
   S8C S4C S2C S1C = 0000 or 1xxx for 8 conversions
3. Set up `ATD0CTL4`
   - For 8-bit mode write `0x85` to `ATD0CTL4`
   - For 10-bit mode write `0x05` to `ATD0CTL4`
   - Other values of `ATD0CTL4` either will not work or will result in slower A/D conversion rates
4. Select DJM in `ATD0CTL5`
   (a) DJM = 0 `=>` Left justified data in the result registers
   (b) DJM = 1 `=>` Right justified data in the result registers
5. Select DSGN in `ATD0CTL5`
   (a) DSGN = 0 `=>` Unsigned data representation in the result register
   (b) DSGN = 1 `=>` Signed data representation in the result register

   The Available Result Data Formats are shown in the following table:

SRES8	DJM	DSGN	Result Data Format
1	0	0	8-bit/left justified/unsigned - Bits 15-8
1	0	1	8-bit/left justified/signed - Bits 15-8
1	1	X	8-bit/right justified/unsigned - Bits 7-0
0	0	0	10-bit/left justified/unsigned - Bits 15-6
0	0	1	10-bit/left justified/signed - Bits 15-6
0	1	X	10-bit/right justified/unsigned - Bits 9-0

6. Select `MULT` in `ATD0CTL5`:
   - `MULT = 0`: Convert one channel eight the specified number of times
     - Choose channel to convert with `CC, CB, CA` of `ATD0CTL5`.
   - `MULT = 1`: Convert across several channels. `CC, CB, CA` of `ATD0CTL5` is the first channel to be converted
7. Select `SCAN` in `ATD0CTL5`:
   - `SCAN = 0`: Convert one sequence, then stop
   - `SCAN = 1`: Convert continuously
8. After writing to `ATD0CTL5`, the A/D converter starts, and the `SCF` bit is cleared. After a sequence of conversions is completed, the `SCF` flag in `ATD0STAT0` is set.
   - You can read the results in `ATD0DRx [0-7]H`.
9. If `SCAN = 0`, you need to write to `ATD0CTL5` to start a new sequence. If `SCAN = 1`, the conversions continue automatically, and you can read new values in `ADR[0-7]H`.
10. To get an interrupt after the sequence of conversions are completed, set `ASCIE` bit of `ATD0CTL2`. After the sequence of conversions, the `ASCIF` bit in `ATD0CTL2` will be set, and an interrupt will be generated.
11. With 24 MHz bus clock and `ATD0CTL4 = 0x05`, it takes 7 μs to make one conversion, 56 μs to make eight conversions.
12. On MC9S12 EVBU, AD0 channels 0 and 1 are used to determine start-up program (D-Bug12, EEPROM or bootloader). Do not use AD0 channels 0 or 1 unless absolutely necessary (you need more thann 14 A/D channels).
13. $$\texttt{ATD0DRx} = \frac{V_{in} - V_{RL}}{V_{RH} - V_{RL}} \times 1024$$
    Normally, $V_{RL} = 0$ V, and $V_{RH} = 5$ V, so
    $$\texttt{ATD0DRx} = \frac{V_{in}}{5\text{ V}} \times 1024$$
    Example: `ATD0DR0 = 448 =>` $V_{in} = 2.19$ V
14. To use 10-bit result, set `ATD0CTL4 = 0x05` (Gives 2 MHz AD clock with 24 MHz bus clock, 10-bit mode)
15. You can get more accuracy by averaging multiple conversions. If you need only one channel, set `MULT = 0`, set `SC` bits for eight conversions, then average all eight result registers. The following assumes the data was right justified:

```
int avg;

avg = (ATD0DR0 + ATD0DR1
       ATD0DR2 + ATD0DR3
       ATD0DR4 + ATD0DR5
       ATD0DR6 + ATD0DR7) >> 3;
```

```
/* Read temperature from PAD4.  Turn on heater if temp too low,
 * turn off heater if temp too high.  Heater connected to Bit 0
 * of Port A.
 */
#include "hcs12.h"

#define TRUE 1
#define SET_POINT 72    /* Temp at which to turn heater on or off */

main()
{
    ATD0CTL2 = 0x80;   /* Power up A/D, no interrupts */
    ATD0CTL3 = 0x00;   /* Doe eight conversions */
    ATD0CTL4 = 0x85;   /* 8-bit mode */
    ATD0CTL5 = 0xA4;   /* 1 0 1 0 0 1 0 0
                          | | | |   \___/
                          | | | |     |
                          | | | |     \__ Bit 4 of Port AD
                          | | | \________ MULT = 0 => one channel only
                          | | \__________ Scan = 1 => continuous conversion
                          | \____________ DSGN = 0 => unsigned
                          \______________ DJM  = 1 => right justified
                       */
/*****************************************************************************/

    DDRA = 0xff;    /* Make Port A output */
    PORTA = 0x00;   /* Turn off heater */

/*****************************************************************************/

    while (TRUE)
    {
        if (ATD0DR0H > SET_POINT)
            PORTA &= ~BIT0;
        else
            PORTA |= BIT0;
    }
}
```

```
/* Convert signals on Channels AD08 through AD15
 * Set up for 10-bit, multi-channel, mod
 * Do one set of scans
 * Save values in variables
 */

#include "hcs12.h"

main()
{
    unsigned int ch[8];   /* Variable to hold result */

    ATD1CTL2 = 0x80;  /* Power up A/D, no interrupts */
    ATD1CTL3 = 0x40;  /* Do eight conversions */
    ATD1CTL4 = 0x05;  /* 10-bit mode, 7 us/conversion */
    ATD1CTL5 = 0x92;  /* 1 0 0 1 0 0 1 0
                         | | | |   \__/
                         | | | |    |
                         | | | |    \___  First channel = 2
                         | | | \________  MULT = 1 => multiple channels
                         | | \__________  SCAN = 0 => one set of conversions
                         | \____________  DSGN = 0 => unsigned
                         \______________  DJM = 1 => right justified
                      */

/*************************************************************************/


    while ((ATD1STAT0 & BIT7) == 0 ) ;   /* Wait for sequence to finish */
    ch[0] = ATD1DR0;
    ch[1] = ATD1DR1;
    ch[2] = ATD1DR2;
    ch[3] = ATD1DR3;
    ch[4] = ATD1DR4;
    ch[5] = ATD1DR5;
    ch[6] = ATD1DR6;
    ch[7] = ATD1DR7;
}
```