Analog/Digital Converters

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

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

- 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?

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

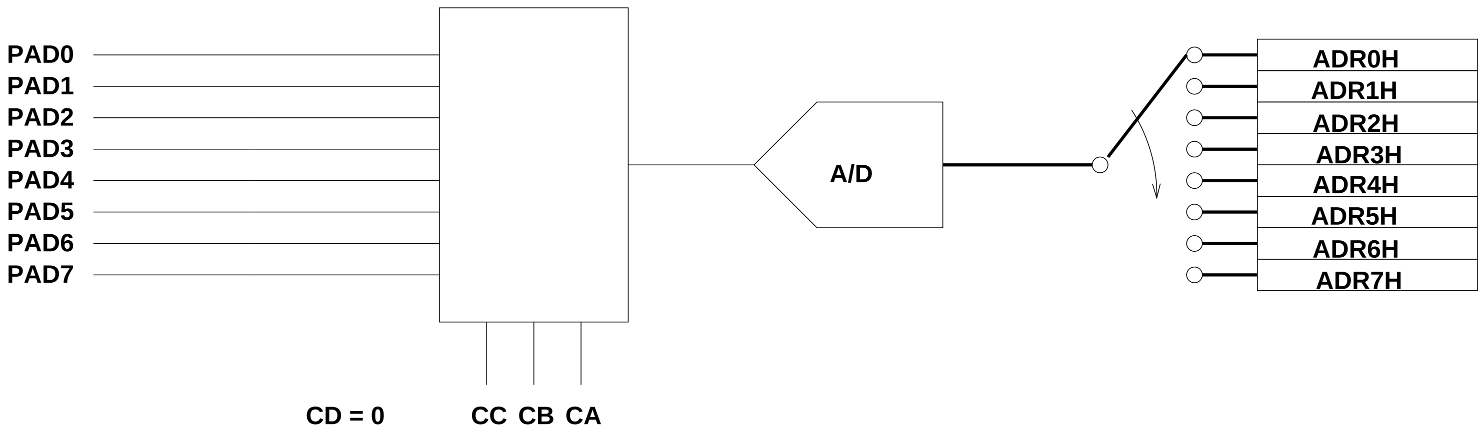
The HC12 Analog/Digital Converter

- The HC12 has a 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 HC12 for the reference voltages V_{RL} and V_{RH}
  - In normal operation 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 must do a sequence of eight conversions at a time.
  - The HC12 has a mode where you can do 4 conversions at a time. We will not discuss this mode.
- You can chose to make eight conversion of a single input channel, or one conversion of all eight input channels.
- The results of the eight conversions are stored in the registers `ADR0` through `ADR7`
  - In eight-bit mode the outputs are stored in eight-bit registers `ADR0H` through `ADR7H`
  - In 10-bit mode the outputs are stored in 16-bit registers `ADR0` through `ADR7`. The data is left-justified in these registers, so the results need to be shifted 6 bits to the left to put the results in standard form
- To program the HC12 A/D converter you need to set up the A/D control registers `ATDCTL2`, `ATDCTL4` and `ATDCTL5`
  - The registers `ATDCTL0`, `ATDCTL1` and `ATDCTL3` are used for factory test, and not normally used in normal operation.

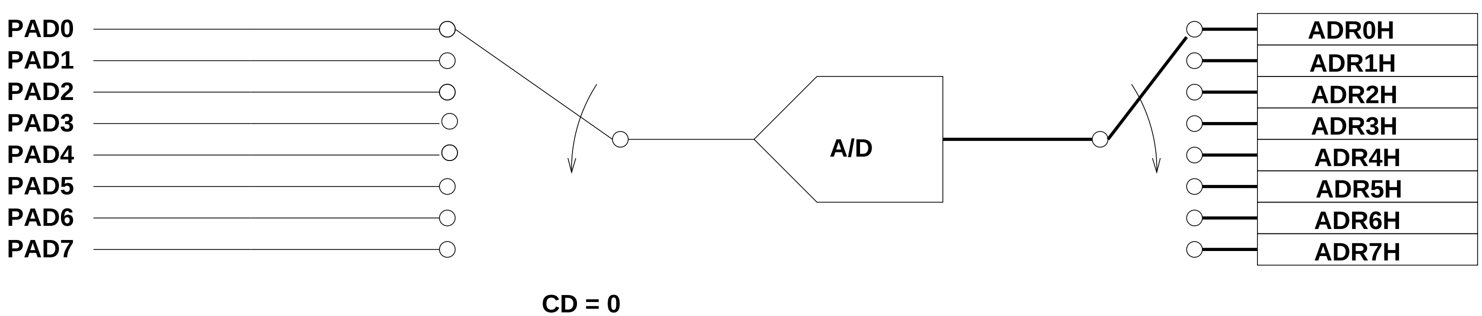
HC12 A/D Converter Setup

S8CM = 1 (8 Channel Mode)

MULT = 0



MULT = 1



Registers for the HC12 A/D Converter

ATDCTL2	**ADPU**	AFFC	ASWAI	0	0	0	**ASCIE**	**ASCIF**
ATDCTL4	**S10BM**	SMP1	SMP0	PRS4	PRS3	PRS2	PRS1	PRS0
ATDCTL5	0	**S8CM**	**SCAN**	**MULT**	**CD**	**CC**	**CB**	**CA**
ATDSTAT	**SCF**	0	0	0	0	CC2	CC1	CC0
	CCF7	CCF6	CCF5	CCF4	CCF3	CCF2	CCF1	CCF0

To Use A/D Converter:

ADPU = 1 (Power up A/D)
S8CM = 1 (8 Conversion Mode)
CD = 0 (Used for factory test only)
S10BM = 0 (8 Bit Mode − dat in ATRxH, or ATRx bits 15−8)
S10BM = 1 (10 Bit Mode − data in ATRx bits 15−6)

SCAN = 0 => Do 8 conversions, then stop
SCAN = 1 => Convert continuously

ATDCTL4 = 0x01 (2 MHz AD clock, 18 cycles per conversion, 8 bit mode)
ATDCTL4 = 0x81 (2 MHz AD clock, 20 cycles per conversion, 10 bit mode)
Other values of ATDCTL4 will not work, or will result in slower operation of A/D

After writing to ATDCTL5, SCF flag cleared and conversions start
After 8 conversions done, SCF flag set (4 conversions if S8CM == 0)

USING THE HC12 A/D CONVERTER

1. Power up A/D Converter (`ADPU = 1` in `ATDCTL2`)
2. Set up `ATDCTL4`
   - For 8-bit mode write `0x01` to `ATDCTL4`
   - For 10-bit mode write `0x81` to `ATDCTL4`
   - Other values of `ATDCTL4` either will not work or will result in slower A/D conversion rates
3. Select 8-channel mode (`S8CM = 1` in `ATDCTL5`)
4. Set `CD = 0` in `ATDCTL5` (`CD = 1` for factory test only)
5. Select `MULT` in `ATDCTL5`:
   - `MULT = 0`: Convert one channel eight times
     – Choose channel to convert with `CC, CB, CA` of `ATDCTL5`.
   - `MULT = 1`: Convert eight channels
6. Select `SCAN` in `ATDCTL5`:
   - `SCAN = 0`: Convert eight samples, then stop
   - `SCAN = 1`: Convert continuously
7. After writing to `ATDCTL5`, the A/D converter starts, and the `SCF` bit is cleared. After eight conversions are complete, the `SCF` flag in `ATDSTAT` is set.
   - You can read the results of 8-bit conversions in `ADR[0-7]H`.
   - You can read the results of 10-bit conversions in `ADR[0-7]`.
8. If `SCAN = 0`, you need to write to `ATDCTL5` to start a new sequence. If `SCAN = 1`, the conversions continue automatically, and you can read new values in `ADR[0-7]H`.
9. To get an interrupt after eight conversions are completed, set `ASCIE` bit of `ATDCTL2`. After eight conversions, the `ASCIF` bit in `ATDCTL2` will be set, and an interrupt will be generated.
10. With 8 MHz E-clock and `ATDCTL4 = 0x01`, it takes 9 μs to make one conversion, 72 μs to make eight conversions.

11. On HC12 EVBU, AD channels 0 and 1 are used to determine start-up program (D-Bug12, EEPROM or bootloader). Do not use AD channels 0 or 1 unless absolutely necessary (you need 7 or 8 channels). If you do need AD channels 0 and/or 1, power up EVBU, then remove the jumpers which selected the start-up mode.

12.
$$\texttt{ADRx}[15..6] = \frac{V_{in} - V_{RL}}{V_{RH} - V_{RL}} \times 1024$$

Normally, V_{RL} = 0 V, and V_{RH} = 5 V, so

$$\texttt{ADRx}[15..6] = \frac{V_{in}}{5\text{ V}} \times 1024$$

Example: `ADR0[15..6] = 448` => V_{in} = 2.19 V

13. To use 10-bit result, set `ATDCTL4 = 0x81` (Gives 2 MHz AD clock with 8 MHz E-clock, 10-bit mode),and add the following to `hc12.h`:

```
#define ADR0   (* (volatile unsigned int *)(_BASE+0x70))
#define ADR1   (* (volatile unsigned int *)(_BASE+0x72))
.
.
```

14. You can get more accuracy by averaging multiple conversions. If you need only one channel, set `MULT = 0`, then average all eight result registers:

```
int avg;

avg = ((ADR0>>6) + (ADR1>>6)
     + (ADR2>>6) + (ADR3>>6)
     + (ADR4>>6) + (ADR5>>6)
     + (ADR6>>6) + (ADR7>>6)) >> 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 <hc12b32.h>

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

main()
{
    ATDCTL2 = 0x80;  /* Power up A/D, no interrupts */
    ATDCTL4 = 0x01;  /* 9 us/conversion, 8-bit mode */
    ATDCTL5 = 0x64;  /* 0 1 1 0 0 1 0 0
                          | | | \____/
                          | | |    |
                          | | |    \___  Bit 4 of Port AD
                          | | \________  Mult = 0 => one channel only
                          | \__________  Scan = 1 => continuous conversion
                          \____________  S8CM => do eight conversions
                     */
/**************************************************************************/

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

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

    while (TRUE)
    {
        if (ADR0H > SET_POINT)
            PORTA &= ~0x01;
        else
            PORTA |= 0x01;
    }
}
```

```
/* Set up for 10-bit, multi-channel, scan mode.
 * Save values in variables
 */

#include <hc1232.h>

/* Define AD result registers for 10 bit mode */
#define ADR0    (* (volatile unsigned int *) 0x70)
#define ADR1    (* (volatile unsigned int *) 0x72)
#define ADR2    (* (volatile unsigned int *) 0x74)
#define ADR3    (* (volatile unsigned int *) 0x76)
#define ADR4    (* (volatile unsigned int *) 0x78)
#define ADR5    (* (volatile unsigned int *) 0x7a)
#define ADR6    (* (volatile unsigned int *) 0x7c)
#define ADR7    (* (volatile unsigned int *) 0x7e)

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

    ATDCTL2 = 0x80;  /* Power up A/D, no interrupts */
    ATDCTL4 = 0x81;  /* 10 us/conversion, 10-bit mode */
    ATDCTL5 = 0x64;  /* 0 1 0 0 0 0 0 0
                          | | | \____/
                          | | |    |
                          | | |    \___  CD = 0; others don't care
                          | | \_________  Mult = 1 => multiple channels
                          | \___________  Scan = 0 => one set of conversions
                          \____________  S8CM => do eight conversions
                    */

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

    while ((ATDSTAT & 0x8000) == 0 ) ;   /* Wait for conversion to finish */
    ch[0] = ADR0 >> 6;
    ch[1] = ADR1 >> 6;
    ch[2] = ADR2 >> 6;
    ch[3] = ADR3 >> 6;
    ch[4] = ADR4 >> 6;
    ch[5] = ADR5 >> 6;
    ch[6] = ADR6 >> 6;
    ch[7] = ADR7 >> 6;
}
```