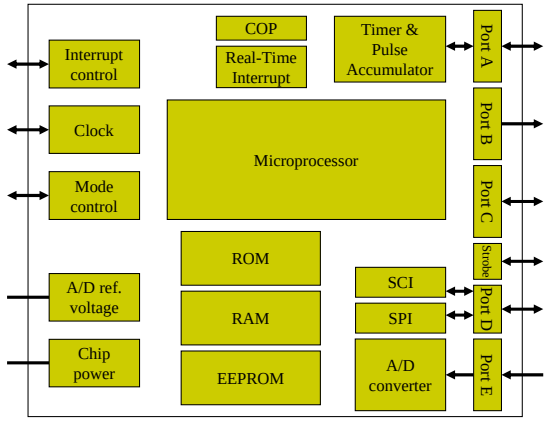
EE 3170 Microcontroller Applications

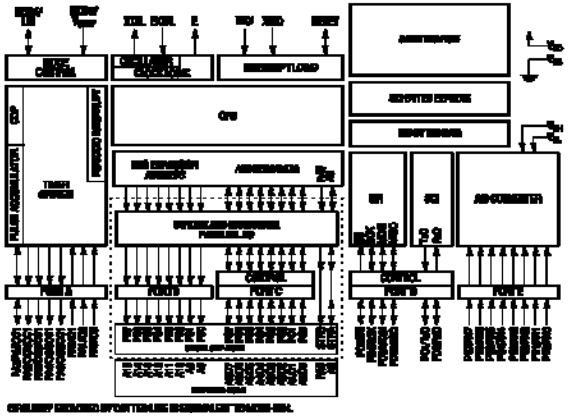
Lecture 14: Advanced 68HC11 Hardware- Part III: Analog-to-Digital Converter
- Miller §7.11

Based on slides for ECE3170 by Profs. Davis, Kieckhafer, Tan, and Cischke

Block Diagram of 68HC11A8



Block Diagram of 68HC11A8



68HC11A8 Components

- Memory
  - RAM
  - ROM
  - EEPROM
- Parallel Input/Output
  - Port B
  - Port C
- Strobe
  - STRA
  - STRB
- Programmable Timer & Pulse Accumulator
  - Port A
- SCI (SPI)
  - Serial Communications (Peripheral) Interface
  - Port D
- Analog-to-Digital Converter
  - Port E

Why Do We Need A/D and D/A Conversion?

- Real world is analog, not digital.
  - Microcontroller must interface to real-world.
  - Efficient D/A and A/D are important.
    - Digital to analog is easy.
    - Analog to digital is harder.

Digital-to-Analog & Analog-to-Digital

- D/A output is easy.
  - We output the digital value.
  - A low-pass (reconstruction) filter smoothes output.
- **A/D** input is harder.
  - We must represent an analog value by one of a set of digital values.

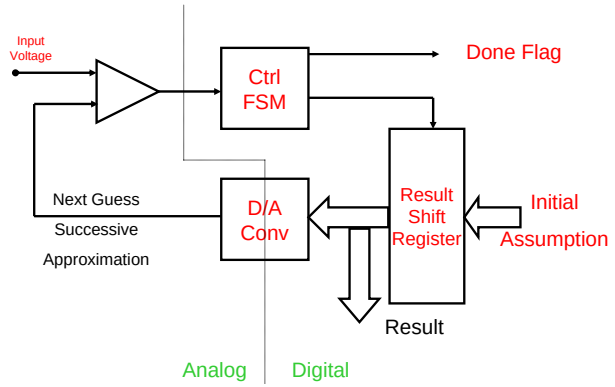
What is the Basic A/D Procedure?

- **Goal**: Find a binary value proportional to an input voltage.
- Typical A/D Algorithm
  - Pass input voltage to analog voltage comparator.
  - Guess the number represented by the voltage.
  - Pass number to D/A converter to get assumed voltage.
  - Pass assumed voltage to other input of comparator.
  - **If** Comparator output = 0
    Then the number guessed was correct (output it)
    Else guess again

Analog-to-Digital Conversion

- Conversion Principles
  - Generate a Binary Number Proportional to a DC Voltage
    - Between Low and High Reference Voltages (Ratiometric)
      - V_{RL} => $00
      - V_{RH} => $FF
  - Converter Hardware
    - Analog Comparator
    - Controller
    - D/A Register
    - D/A Converter
  - Auxiliary Hardware
    - Multiplexer
    - Sample-and-Hold

Basic A/D Algorithm



The Guess Again Step

- The key issue is how we guess the next value.
- Basic methods are
  - Up counter
  - Successive approximation

Up Counter

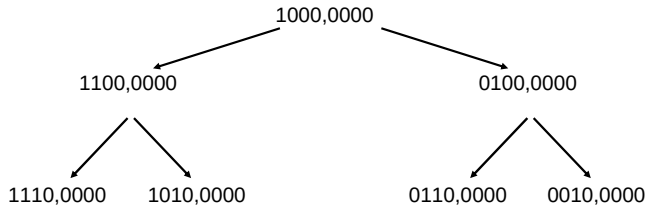
- Simplest method is an Up-counter
  - Done_Flag = 0
  - Number = 0
  - While difference $\neq 0$
    - Increment number
  - Done_Flag = 1
  - Output the Number
- Problem: may have to count to FF.
  - Worst-case time = 2^n cycles where n is the number of bits.

Successive Approximation

- Classic binary search
- Best Worst-Case Time
  - Done_Flag = 0
  - Number = $2^N/2$
  - While difference $\neq 0$
    - If Guess is too low
      Then Number = Number **+** Number/2
      Else Number = Number **-** Number/2
  - Done_Flag = 1
  - Output the Number

Algorithm Progress

- Error is cut in half every step
- Compare against:



What is the 68HC11 A/D Converter?

- **One A/D**: Multiplexer selects among inputs.
- **8 Inputs** on PORTE pins (P7…P0)
- Multiplexer selects the pin to be read.
  - Eight A/D converters would be more expensive.
- **Sample-and-Hold** Unit
  - samples input voltage at beginning of conversion
  - holds voltage throughout conversion
    - problems could occur if voltage allowed to change
- **Converter** uses successive approximation.

68HC11 A/D Converter, cont.

- **Outputs**:
  - Done-Flag = CCF-bit of ADCTL reg @ $1030
  - Four Data Value Registers (8-bits each)
    - ADR1 @ $1031
    - ADR2 @ $1032
    - ADR3 @ $1033
    - ADR4 @ $1034
- **Timing**-- exactly 32 E-clock cycles
  - = 16 μs with 8 MHz crystal

ADC registers

			Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
$1030	Analog-to-Digital Control Status Register (ADCTL)	Read:	CCF		SCAN	MULT	CD	CC	CB	CA
		Write:								
		Reset:	0	0	Indeterminate after reset					
$1031	Analog-to-Digital Results Register 1 (ADR1)	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	Indeterminate after reset							
$1032	Analog-to-Digital Results Register 2 (ADR2)	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	Indeterminate after reset							
$1033	Analog-to-Digital Results Register 3 (ADR3)	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	Indeterminate after reset							
$1034	Analog-to-Digital Results Register 4 (ADR4)	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	Indeterminate after reset							

ADCTL

Address: $1030

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	CCF		SCAN	MULT	CD	CC	CB	CA
Write:								
Reset:	0	0	Indeterminate after reset					

= Unimplemented

CCF — Conversion Complete Flag
This bit is set after an A/D conversion cycle and cleared when ADCTL is written.
Bit 6 — Unimplemented
Always reads 0
SCAN — Continuous Scan Control
0 = Do four conversions and stop
1 = Convert four channels in selected group continuously
MULT — Multiple Channel/Single Channel Control
0 = Convert single channel selected
1 = Convert four channels in selected group

ADCTL Register

CCF — Conversion Complete Flag

- **read-only status indicator**
- **set when all four A/D result registers hold valid conversions**

SCAN — Continuous Scan Control Bit

- **0-- the four requested conversions are performed once**
- **1--conversions are performed continuously; registers updated as data becomes available.**

MULT — Multiple Channel/Single Channel Control Bit

- **0-- four consecutive conversions on a single channel specified.**
- **1-- a conversion on each of four channels**

CD:CA — Channel Selects D:A Bits

- Refer to the **Table.** When a multiple channel mode is selected (MULT = 1), the two least significant channel select bits (CB and CA) have no meaning and the CD and CC bits specify which group of four channels is to be converted.

ADCTL Register, p. 2

CD:CA — Channel Selects D:A
Refer to the following table.

Channel Select Control Bits CD:CC:CB:CA	Channel Signal	Result in ADRx if MULT = 1	Result in ADRx if MULT = 0
0000	AN0	ADR1	ADR[4:1]
0001	AN1	ADR2	ADR[4:1]
0010	AN2	ADR3	ADR[4:1]
0011	AN3	ADR4	ADR[4:1]
0100	AN4	ADR1	ADR[4:1]
0101	AN5	ADR2	ADR[4:1]
0110	AN6	ADR3	ADR[4:1]
0111	AN7	ADR4	ADR[4:1]
10XX	Reserved	—	—
1100	V_{RH}(1)	ADR1	ADR[4:1]
1101	V_{RL}(1)	ADR2	ADR[4:1]
1110	$(V_{RH})/2$(1)	ADR3	ADR[4:1]
1111	Reserved(1)	ADR4	ADR[4:1]

1. Used for factory testing

OPTION Register

$1039 System Configuration Options Register (OPTION)

	ADPU	CSEL	IRQE(1)	DLY(1)	CME		CR1(1)	CR0(1)
Read: Write:								
Reset:	0	0	0	1	0	0	0	0

ADPU — A/D Power-Up Bit

0 = A/D powered down
1 = A/D powered up

CSEL — Clock Select Bit

0 for highest A/D accuracy

IRQE — Configure IRQ for Edge-Sensitive Only Operation

DLY — Enable Oscillator Startup Delay Bit

CME — Clock Monitor Enable Bit

CR[1:0] — COP Timer Rate Select Bits

Initiation and Activation

- A/D initializes with Power off
  - It uses lots of power.
  - To power-up, set **ADPU** bit of OPTION register @ $1039
- Activation
  - Write a value to the ADCTL Register
  - Starts the A/D working.

Single and Multi-channel Modes

- A/D always scans 4 channels in sequence.
  - puts the 4 results into ADR1…ADR4 registers
  - **Single-Channel Mode**
    - just re-reads the same input pin 4 times
  - **Multiple Channel Mode**
    - reads 4 adjacent pins in rapid succession
    - reads either the high-order or low-order 4 pins
- **MULT** bit of ADCTL register selects mode.
- CC, CB, CA bits of ADCTL register select channel(s).

Continuous Scan Option

- Single-Scan Mode:
  - A/D scans its assigned 4 channels and stops
- Continuous Scan Mode
  - A/D continuously scans its assigned 4 channels.
  - ADR1 … ADR4 always contain fresh data on the input voltages
  - No further program intervention required
- SCAN bit of ADCTL selects mode.

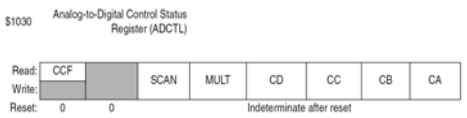
Some Details

- A/D can not cause an interrupt.
  - Conversion time is fixed (32 E-clock cycles).
  - Polling is convenient.
  - Exact time is known.
    - Program counts down.
- To power-up, set ADPU bit of OPTION register @ $1039
- Activation
  - Write a value to the ADCTL Register.
    - Starts A/D working.

Analog-to-Digital Example

- Application
  - Read Analog value on channels AD0 and AD1.
  - Execute one block of code {BIG0} if AD0>AD1;
  - Execute a different block of code {BIG1} if AD1>AD0.

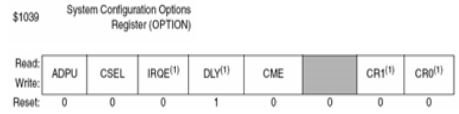
Analog-to-Digital Example



SCAN = 0

MULT = 1

CD:CC = 00



ADPU = 1

CSEL = 0

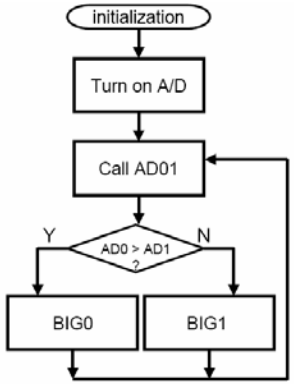
The Code

/Definitions

```
ADCTL     equ $1030
ADR1      equ $1031
OPTION    equ $1039

          org $2000
ADBUF     rmb 4
          * Polled I/O
          * No ISR vector
          org $FFFE
          fdb MAIN
```


More Code

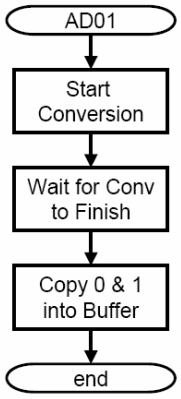



```
INIT    ldaa #%10000000
        staa OPTION

INP     jsr     AD01

        ldaa    ADBUF
        cmp     ADBUF+1
        ble     BIG1
BIG0    nop
        bra     INP
BIG1    nop
        bra     INP
```

Last Code




```
AD01  ldaa   #%10010000
      staa   ADCTL
      ldx    ADCTL
LOOP  brclr  0,X,$80,*
      ldaa   1,X
      staa   ADBUF
      ldaa   2,X
      staa   ADBUF+1
      rts
```