• Topics for Exam 2
• The MC9S12 Pulse Width Modulation System
• Huang Sections 8.10 and 8.11
• PWM_8B8C Block User Guide
• Analog-to-Digital Converters
• Huang Sections 12.1-12.2
  o What is PWM
  o The MC9S12 PWM system
  o Registers used by the PWM system
  o How to set the period for PWM Channel 0
  o How to set the clock for PWM Channel 0
  o Interdependence of clocks for Channels 0 and 1
  o PWM Channels 2 and 3
  o Using the MC9S12 PWM
  o A program to use the MC9S12 PWM
  o Introduction to A/D Converters

**Topics for Exam 2 (March 30 or April 1?)**

1. C Programming

   (a) Setting and clearing bits in registers
   • PORTA = PORTA | 0x02;
   • PORTA = PORTA & ~0x0C;

   (b) Using pointers to access specific memory location or port.
   • *(unsigned char *) 0x0400 = 0xaa;
   • #define PORTX (*(unsigned char *) 0x400)
   PORTX = 0xaa;

2. Interrupts

   (a) Interrupt Vectors (and reset vector)
   • How to set interrupt vectors in C

   (b) How to enable interrupts (specific mask and general mask)

   (c) What happens to stack when you receive an enabled interrupt
(d) What happens when you leave ISR with RTI instruction?

(e) What setup do you need to do before enabling interrupts?

(f) What do you need to do in interrupt service routine (clear source of interrupt, exit with RTI instruction)?

(g) How long (approximately) does it take to service an interrupt?

3. Timer/Counter Subsystem

(a) Enable Timer

(b) Timer Prescaler
   • How to set
   • How it affects frequency of timer clock

(c) Timer Overflow Interrupt

(d) Input Capture

(e) Output Compare

(f) How to enable interrupts in the timer subsystem

(g) How to clear flags in the timer subsystem

(h) Be able to look at registers and determine timer is set up
   • Which channels are being used
   • Which are being used for Input Capture, which for Output Compare
   • How to time differences from Timer count registers

4. Real Time Interrupt

(a) How to enable

(b) How to change rate

(c) How to enable interrupt

(d) How to clear flag
5. Pulse Width Modulation

(a) How to get into 8-bit, left-aligned high-polarity mode

(b) How to set PWM period (frequency)
   - Using Clock Mode 0
   - Using Clock Mode 1

(c) How to set PWM duty cycle

(d) How to enable PWM channel

(e) Be able to look at PWM registers and determine PWM frequency and duty cycle
The MC9S12 Pulse Width Modulation Subsystem

- The MC9S12 PWS subsystem allows you to control up to eight devices by adjusting the percentage of time the output is active.

- We will discuss 8-bit, high polarity, left-aligned modes.

- Different types of devices need different PWM periods.

- The hard part of setting up the PWM subsystem is figuring out how to set up the MC9S12 to get the period you want.

- Once you determine the period in seconds, convert this to clock cycles:

  \[ \text{Period in cycles} = \text{Period in seconds} \times 24,000,000 \\text{cycles/sec} \]

- Once you have period in clock cycles, figure out how to get this value (or close to this value) using the following:

  \[
  \text{Period} = \begin{cases} 
  \text{PWMPER0} \times 2^N & \text{if PCLK0x == 0} \\
  \text{PWMPER0} \times 2^{N+1} \times M & \text{if PCLK0x == 1} 
  \end{cases}
  \]

- Find values of PWMPERx, N and (if using clock mode 1) M.

- Choose PWMPERx to be fairly large (typically 100 or greater).

- For channels 0, 1, 4 and 5, N is set using the PCKA bits of register PWMPRCLK, and M is set by the eight-bit register PWMSCLA.

- For channels 2, 3, 6 and 7, N is set using the PCKB bits of register PWMPRCLK, and M is set by the eight-bit register PWMSCLB.

- For example, to get a 10 ms period on Channel 0:

  Period in cycles = 10ms \times 24,000,000 \text{cycles/sec} = 240,000

  Cannot use clock mode 0. The largest number of cycles possible using clock mode 0 is 255 \times 27 = 32,640

Using clock mode 1:

\[ 240,000 = \text{PWMPER0} \times 2^{N+1} \times M \]
Let $\text{PWMPER0} = 100$. Then we get the following:

\[
\begin{array}{c|c}
N & M \\
0 & 1200 \\
1 & 600 \\
2 & 300 \\
3 & 150 \\
4 & 75 \\
5 & 37.5 \\
6 & 18.75 \\
7 & 9.375 \\
\end{array}
\]

Since $M$ has to be less than 256, we can use $N = 3$ or $N = 4$.
For $N = 3, M = 150$:

- $\text{PWMCLK} = \text{PWMCLK} \mid 0x01$; // Clock mode 1 for Channel 0
- $\text{PWMPRCLK} = (\text{PWMPRCLK} \& \sim 0x4) \mid 0x03$; // $N = 3$ for Channel 0
- $\text{PWMSCLA} = 150$ // $M = 150$ for Channel 0
- $\text{PWMPER0} = 100$;
Interdependence of clocks for Channels 0, 1, 4 and 5

- The clocks for Channels 0, 1, 4 and 5 are interdependent.
- They all use PCKA and PWMSCLA.
- To set the clock for Channel n, you need to set PCKA, PCLKn, PWMSCLA (if PCLKn == 1) and PWMPERn where n = 0, 1, 4 or 5.
PWM Channels 2, 3, 6 and 7

- PWM channels 2, 3, 6 and 7 are similar to PWM channels 0, 1, 4 and 5

- To set the clock for Channel n, you need to set PCKB, PCLKn, PWMSCLB (if PCLKn == 1) and PWMPERn where n = 2, 3, 6 or 7
Using the HCS12 PWM

1. Choose 8-bit mode (PWMCTL = 0x00)

2. Choose high polarity (PWMPOL = 0xFF)

3. Choose left-aligned (PWMCAE = 0x00)

4. Select clock mode in PWMCLK:
   - PCLKn = 0 for $2^N$,
   - PCLKn = 1 for $2^{(N+1)} \times M$,

5. Select N in PWMPRCLK register:
   - PCKA for channels 5, 4, 1, 0;
   - PCKB for channels 7, 6, 3, 2.

6. If PCLKn = 1, select M
   - PWMSCLA = M for channels 5, 4, 1, 0
   - PWMSCLB = M for channels 7, 6, 3, 2.

7. Select PWMPERn, normally between 100 and 255.

8. Enable desired PWM channels: PWME.

9. Select PWMDTYn, normally between 0 and PWMPERn. Then
   
   Duty Cycle $n = \frac{PWMDTYn}{PWMPERn} \times 100\%$

   Change duty cycle to control speed of motor or intensity of light, etc.

10. For 0% duty cycle, choose PWMDTYn = 0x00.
Program to use the MC9S12 PWM System

 /************************************************************************
 * Program to generate 15.6 kHz pulse width modulation
 * on Port P Bits 0 and 1
 * 
 * To get 15.6 kHz: 24,000,000/15,600 = 1538.5
 * 
 * Cannot get exactly 1538.5
 * 
 * Use 1536, which is 2^9 x 3
 * 
 * Lots of ways to set up PWM to achieve this. One way is 2^3 x 192
 * Set PCKA to 3, do not use PWMSCLA, set PWMPER to 192
 *
/*/ 

#include <hdef.h>    /* common defines and macros */
#include "derivative.h" /* derivative-specific definitions */

void main(void)
{
    /* Choose 8-bit mode */
    PWMCTL = 0x00;

    /* Choose left-aligned */
    PWMAEL = 0x00;

    /* Choose high polarity on all channels */
    PWMPOL = 0xFF;

    /* Select clock mode 0 for Channels 1 and 0 (no PWMSCLA) */
    PWMCLK = PWMCLK & ~0x03;

    /* Select PCKA = 3 for Channels 1 and 0 */
    PWMPRCLK = (PWMPRCLK & ~0x4) | 0x03;

    /* Select period of 192 for Channels 1 and 0 */
    PWMPER1 = 192;
    PWMPER0 = 192;

    /* Enable PWM on Channels 1 and 0 */
    PWME = PWME | 0x03;
}
PWMDTY1 = 96;  /* 50% duty cycle on Channel 1 */
PWMDTY0 = 46;  /* 25% duty cycle on Channel 0 */
while (1)
{
    /* Code to adjust duty cycle to meet requirements */
}
}
Analog/Digital Converters

• An Analog-to-Digital (A/D) converter converts an analog voltage into a digital number

• There are a wide variety of methods used for A/D converters
  Examples are:
    – Flash (Parallel)
    – Successive Approximation
    – Sigma-Delta
    – Dual Slope Converter

• A/D converters are classified according to several characteristics
  – Resolution (number of bits) — **typically 8 bits to 24 bits**
  – Speed (number of samples per second) — several samples/sec to several billion samples/sec
  – Accuracy — how much error there is in the conversion.

• High-resolution converters are usually slower than low-resolution converters

• The MC9S12 has two 10-bit successive approximation A/D converters (which can be used in 8-bit mode)

• The MC9S12 uses an analog multiplexer to allow eight input pins to connect to any of the A/D converters.
Comparator

- A comparator is used in many types of A/D converters.
- A comparator is the simplest interface from an analog signal to a digital signal
- A comparator compares two voltage values on its two inputs
- If the voltage on the + input is greater than the voltage on the - input, the output will be a logic high
- If the voltage on the + input is less than the voltage on the - input, the output will be a logic low.

If $V_{in} > V_{ref}$ then $V_{out} = V_{cc}$
If $V_{in} < V_{ref}$ then $V_{out} = 0$
Flash (Parallel) A/D Converter

• A flash A/D converter is the simplest to understand

• A flash A/D converter compares an input voltage to a large number of reference voltages

• An n-bit flash converter uses $2^n-1$ comparators

• The output of the A/D converter is determined by which of the two reference voltages the input signal is between,

• Here is a 3-bit A/D converter
Flash (Parallel) A/D Converter

• A B-bit Flash A/D converter **requires** \(2^B - 1\) **comparators**

• An 8-bit Flash A/D requires 255 comparators

• A 12-bit Flash A/D converter would require 4,095 **comparators**!
  – Cannot integrate 4,095 comparators onto an IC

• Such A/D are available in IC form up to 8-bit and 10-bit

• Flash A/D converters can sample at several billion samples/sec
A/D Converter Resolution and Quantization

• If the voltage input voltage is 3.2516 V, the lowest 5 comparators will be turned on, and the highest 2 comparators will be turned off.

• The output of the 3-bit flash A/D converter will be 5 (101).

• For a 3-bit A/D converter, which has a range from 0 to 5 V, an output of 5 indicates that the input voltage is between 3.125 V and 3.750 V.

• A 3-bit A/D converter with a 5 V input range has a quantization value of 0.625 V.

• The quantization value of an A/D converter can be found by

\[ \Delta V = \frac{V_{RH} - V_{RL}}{2^b} \]

where \( V_{RH} \) is the highest voltage the A/D converter can handle, \( V_{RL} \) is the lowest voltage the A/D converter can handle, and \( b \) is the number of bits of the A/D converter.

• The HC12 has a 10-bit A/D converter. The typical voltage range used for the HC12 A/D is \( V_{RH} = 5 \) V and \( V_{RL} = 0 \) V, so the HC12 has a quantization value of

\[ \Delta V = \frac{5 \text{ V} - 0 \text{ V}}{2^{10}} = 4.88 \text{ mV} \]

• The dynamic range of an A/D converter is given in decibels (dB):

\[ \text{DR(dB)} = 20 \log 2^b = 20 \times b \log2 = 6.02b \]

• A 10-bit A/D converter has a dynamic range of

\[ \text{DR(dB)} = 6.02 \times 10 = 60.2 \text{ dB} \]
A/D Sampling Rate

- The rate at which you sample a signal depends on how rapidly the signal is changing.
- If you sample a signal too slowly, the information about the signal may be inaccurate.
• A 1,050 Hz signal sampled at 500 Hz looks like a 50 Hz signal

• To get full information about a signal you must sample more than twice the highest frequency in the signal

• Practical systems typically use a sampling rate of at least four times the highest frequency in the signal
Digital-to-Analog (D/A) Converters

- Many A/D converters use a D/A converter internally
- A D/A converter converts a digital signal to an analog voltage or current
- To understand how most A/D converters work, it is necessary to understand D/A converters
- The heart of a D/A converter is an inverting op amp circuit
- The output voltage of an inverting op amp circuit is proportional to the input voltage:

\[ V_{\text{out}} = \frac{-R_F}{R_0} V_{R0} \]
Digital-to-Analog (D/A) Converters

• An inverting op amp can produce an output voltage which is a linear combination of several input voltages

\[ V_{\text{out}} = -\frac{R_F}{R_0} V_{R_0} - \frac{R_F}{R_1} V_{R_1} - \frac{R_F}{R_2} V_{R_2} - \frac{R_F}{R_3} V_{R_3} \]
Digital-to-Analog (D/A) Converters

• By using input resistors which scale by factors of 2, a summing op amp can produce an output which follows a binary pattern
Digital-to-Analog (D/A) Converters

- By using switches on the input resistors, a summing op amp can produce an output which is a binary number (representing which switches are closed) times a reference voltage.
Slope A/D Converter

- A simple A/D converter can be constructed with a counter and a D/A converter
- The counter counts from 0 to $2^b - 1$
- The counter drives the input of the D/A converter
- The output of the D/A converter is compared to the input voltage
- When the output of the comparator switches logic level, the generated voltage passed the input voltage
- By latching the output of the counter at this time, the input voltage can be determined (with the accuracy of the quantization value of the converter)

**Problem** with Slope A/D converter: Could take $2^b$ clock cycles to test possible values of reference voltages
SLOPE A/D CONVERTER

\[ \frac{N}{2} \text{ Clock Cycles per Conversion} \]

- \( V_{in} \)
- Counter
- Latch
- CLK
- D/A

Graph:
- \( V_{in} \)
- \( V_{D/A} \)
- Latch Here
- Time
Successive Approximation A/D Converter

• A successive approximation (SA) A/D converter uses an intelligent scheme to determine the input voltage

• It first tries a voltage half way between $V_{RH}$ and $V_{RL}$

• It determines if the signal is in the lower half or the upper half of the voltage range
  – If the input is in the upper half of the range, it sets the most significant bit of the output
  – If the input is in the lower half of the range, it clears the most significant bit of the output

• The first clock cycle eliminates half of the possible values

• On the next clock cycle, the SA A/D tries a voltage in the middle of the remaining possible values

• The second clock cycle allows the SA A/D to determine the second most significant bit of the result

• Each successive clock cycle reduces the range another factor of two

• For a B-bit SA A/D converter, it takes B clock cycles to determine the value of the input voltage
SUCCESSIVE APPROXIMATION A/D CONVERTER

N Clock Cycles per Conversion

V_in

Start

Clk

High/Low

Successive Approximation Register

Conversion Complete

LATCH

A/D Value

D/A

V

110000

V_D/A

V_in

100100

100110

100110

100000

100100

Time
Successive Approximation A/D Converter

• An SA A/D converter can give the wrong output if the voltage changes during a conversion

• An SA A/D converter **needs an input buffer** which holds the input voltage constant during the conversion

• This input buffer is called a **Track/Hold** or **Sample/Hold** circuit

• It usually works by charging a capacitor to the input voltage, then disconnecting the capacitor from the input voltage during conversion

• The voltage on the capacitor remains constant during conversion

• The HC12 has a Track/Hold amplifier built in

• SA A/D converters have resolutions of up to 16 bits

• SA A/D converters have speeds up to several million samples per second