HW and SW Development Tools for the HCS12

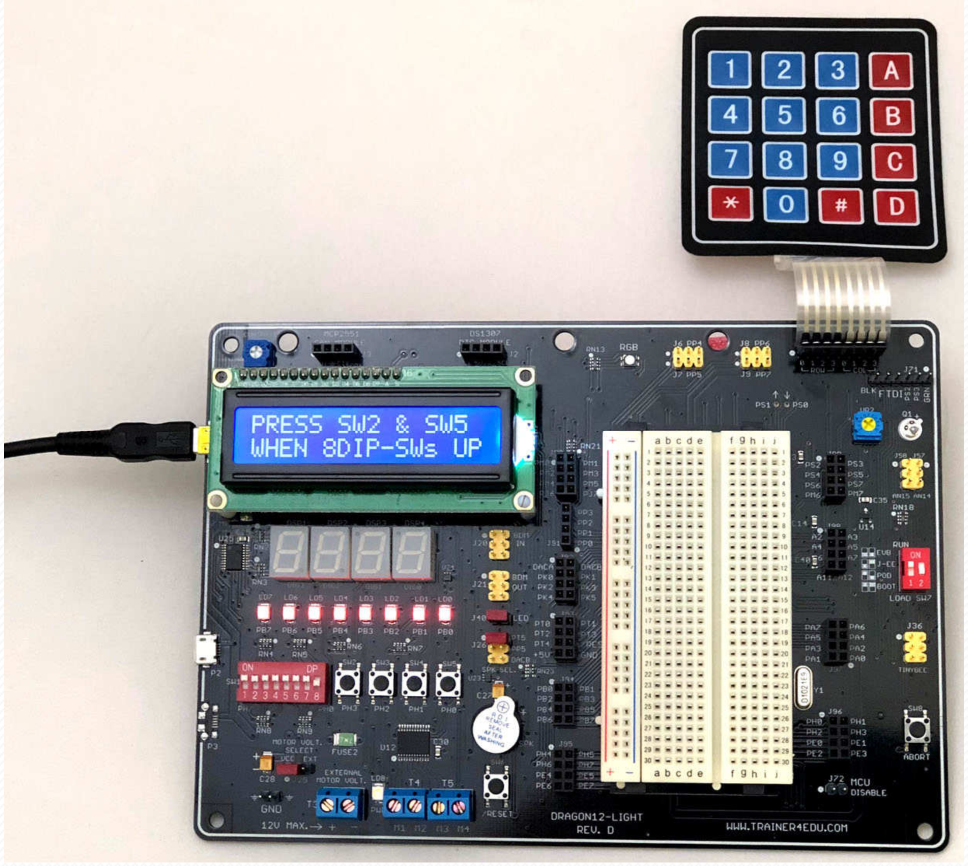
Dr. Abdelaziz Trabelsi
COEN 317 (Fall 2019)

Outline

- HW development tools for the HCS12
- Software development tools for the HCS12
- Using CodeWarrior and Full Chip Simulation mode
- Introduction to parallel I/O port and simple I/O devices

Hardware Development Tools

- Dragon12-Light Board pre-loaded with Freescale Serial Monitor for CodeWarrior.
- The board includes the MC9S12DG256BCPV.
- Dragon12-Light Board Manual contains many information on the components connections to the microprocessor.



Hardware Development Tools

- Among many other components, the Dragon12-Light Board includes:
  - Dual 10-bit DAC for testing SPI interface and generating analog waveforms
  - 4-digit, 7-segment display for learning multiplexing technique
  - 16×2 LCD display module with LED backlight
  - Eight LEDs connected to port B
  - An 8-position DIP switch connected to port H
  - Four pushbutton switches connected to PH0-PH3
  - Speaker driven by timer, DAC, or PWM for alarm, voice and music
  - 4×4 membrane keypad (made of push button switches)
  - Light sensor for home automation applications
  - Temperature sensor for home automation applications
  - RGB color LED

Hardware Development Tools

- The crystal frequency is 8 MHz and this results in a 4 MHz bus speed.
- The bus speed can be boosted to 25 MHz by configuring the PLL.

Software Development Tools

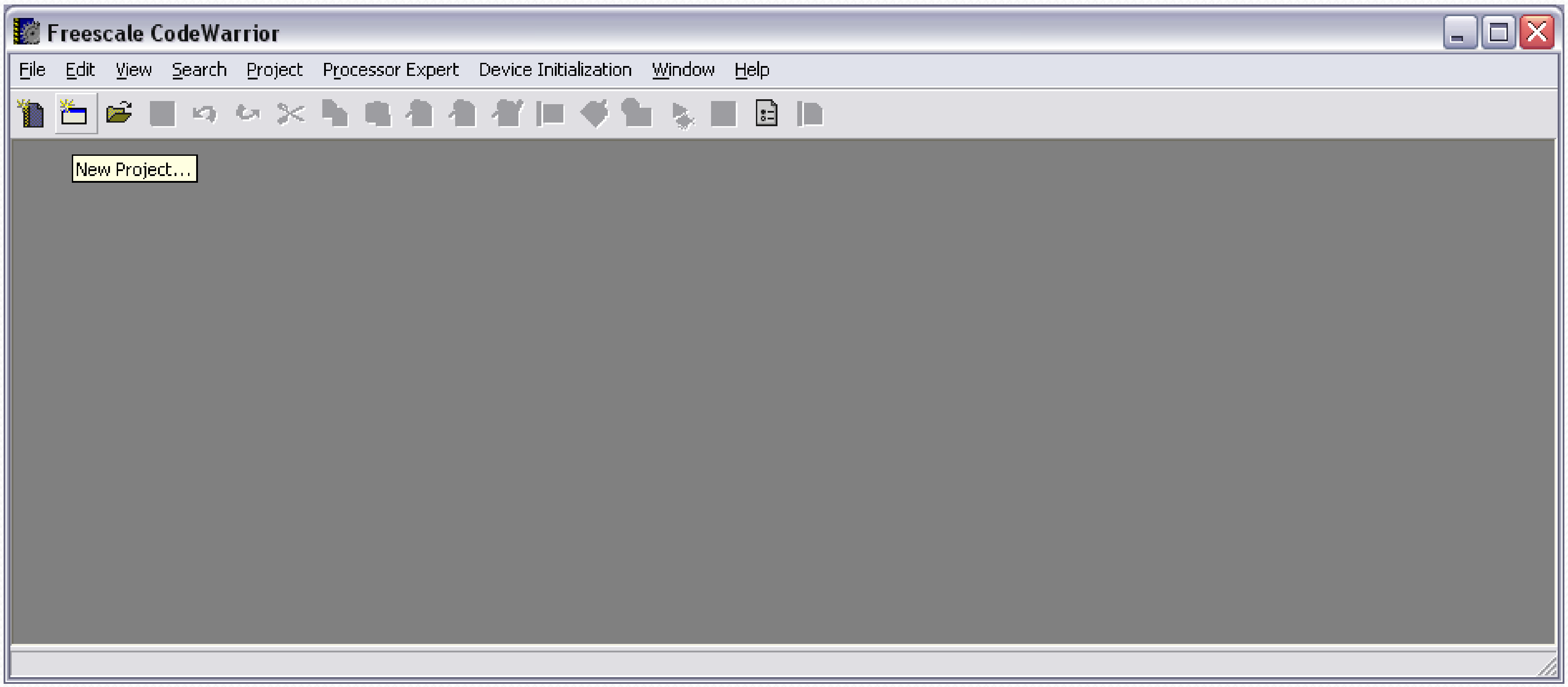
- CodeWarrior IDE for HCS12(X) Microcontrollers (Classic) v. 5.1 SE (Special Edition).
- Projects can be created in purely Assembly Code, in C, or in mixed C/Assembly.
- The software can be used in two modes:
  - Full Chip Simulation:
    - The Board is not required to be connected.
    - The register and memory data are simulated values.
  - HCS12 Serial Monitor:
    - The Board is required to be connected.
    - The registers and memory data are the actual ones.

CodeWarrior for HSC12 Assembly Programing

- Start CodeWarrior for HCS12 and click Create New Project.

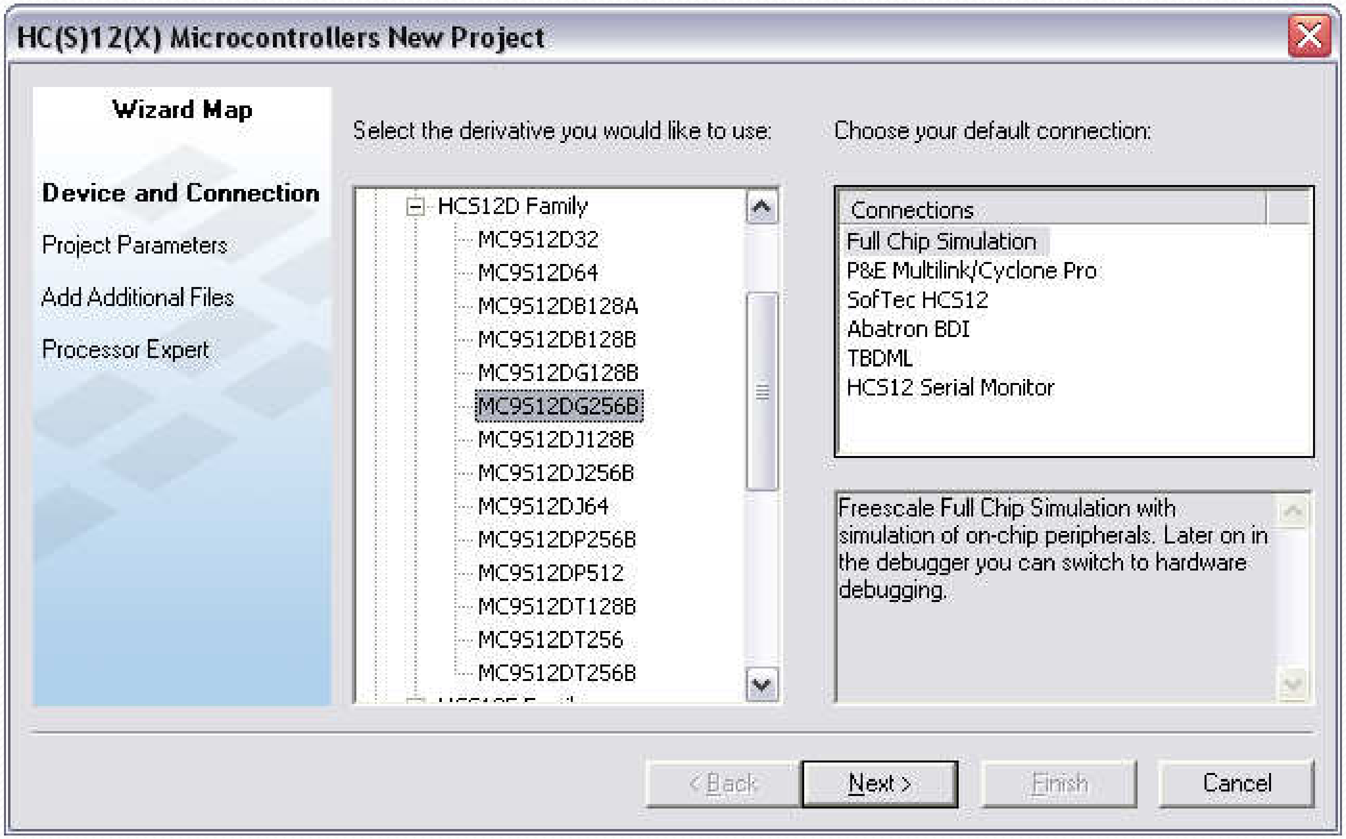
CodeWarrior for HSC12 Assembly Programing

- If CodeWarrior is already running, you may select new project from the file menu or click the icon on the tool bar.



CodeWarrior for HSC12 Assembly Programing

- Select the MCU to program as well as your connection.
  - Choose MC9S12DG256B and Full Chip Simulation (click Next).
  - You can change your connection at any time after creating a project.

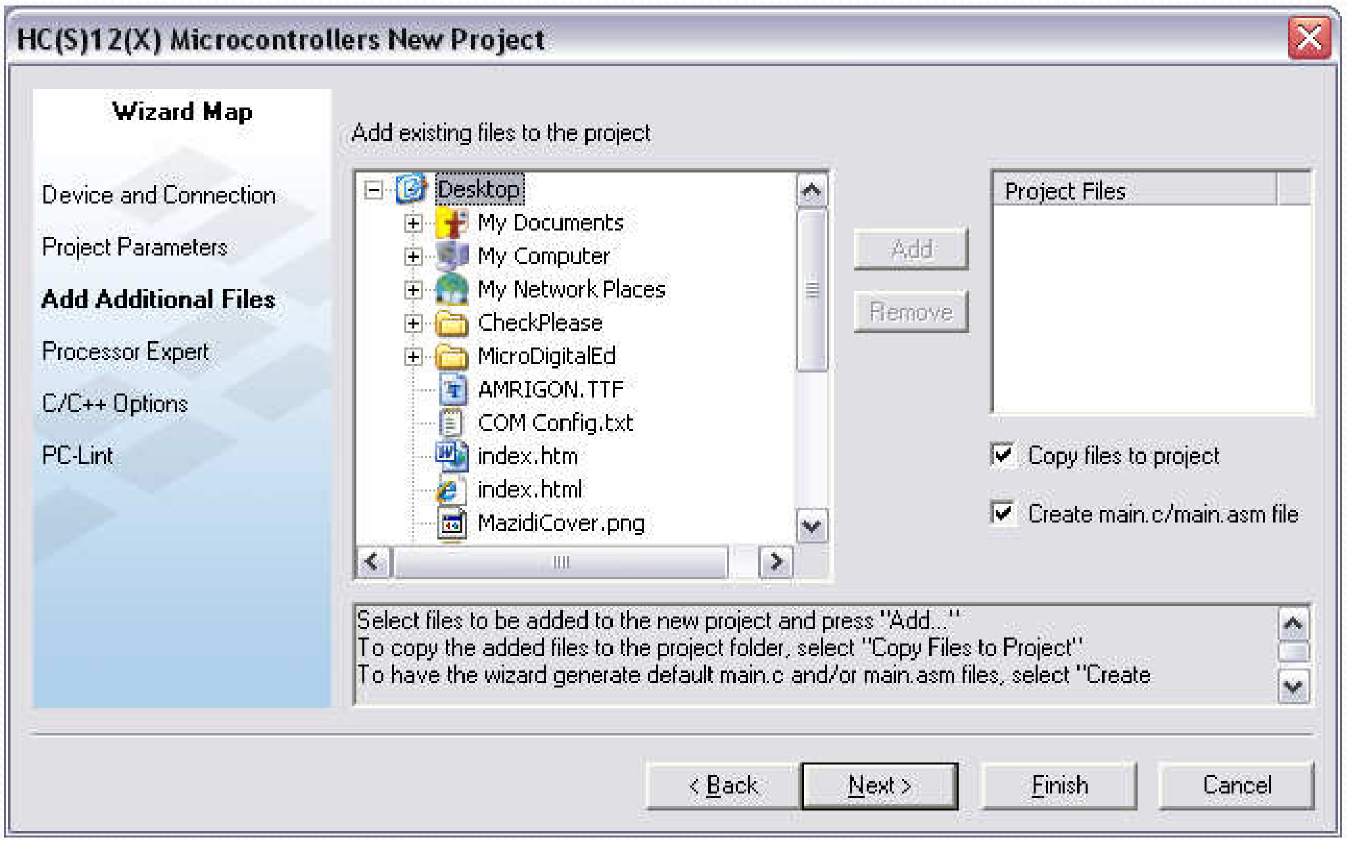


CodeWarrior for HSC12 Assembly Programing

- Set the location where the project is stored and name the project.
- Choose the language(s) you will be using
  - **Unselect** the “Relocatable assembly” then **select** “Absolute assembly” (click Next).

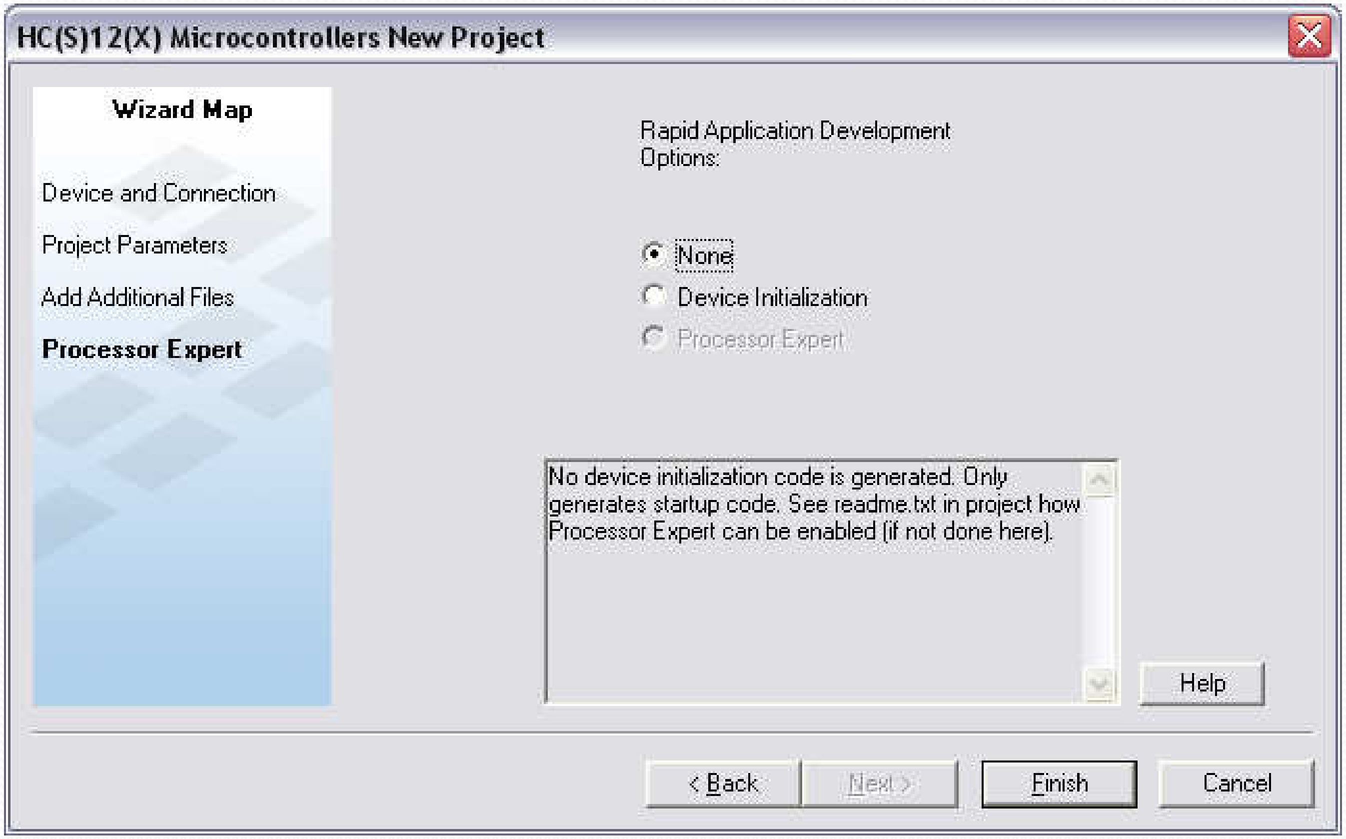
CodeWarrior for HSC12 Assembly Programing

- Choose any files your project will use.
  - In this lecture, no need for any additional files (click Next).
  - You can also add files to your project after it has been created.



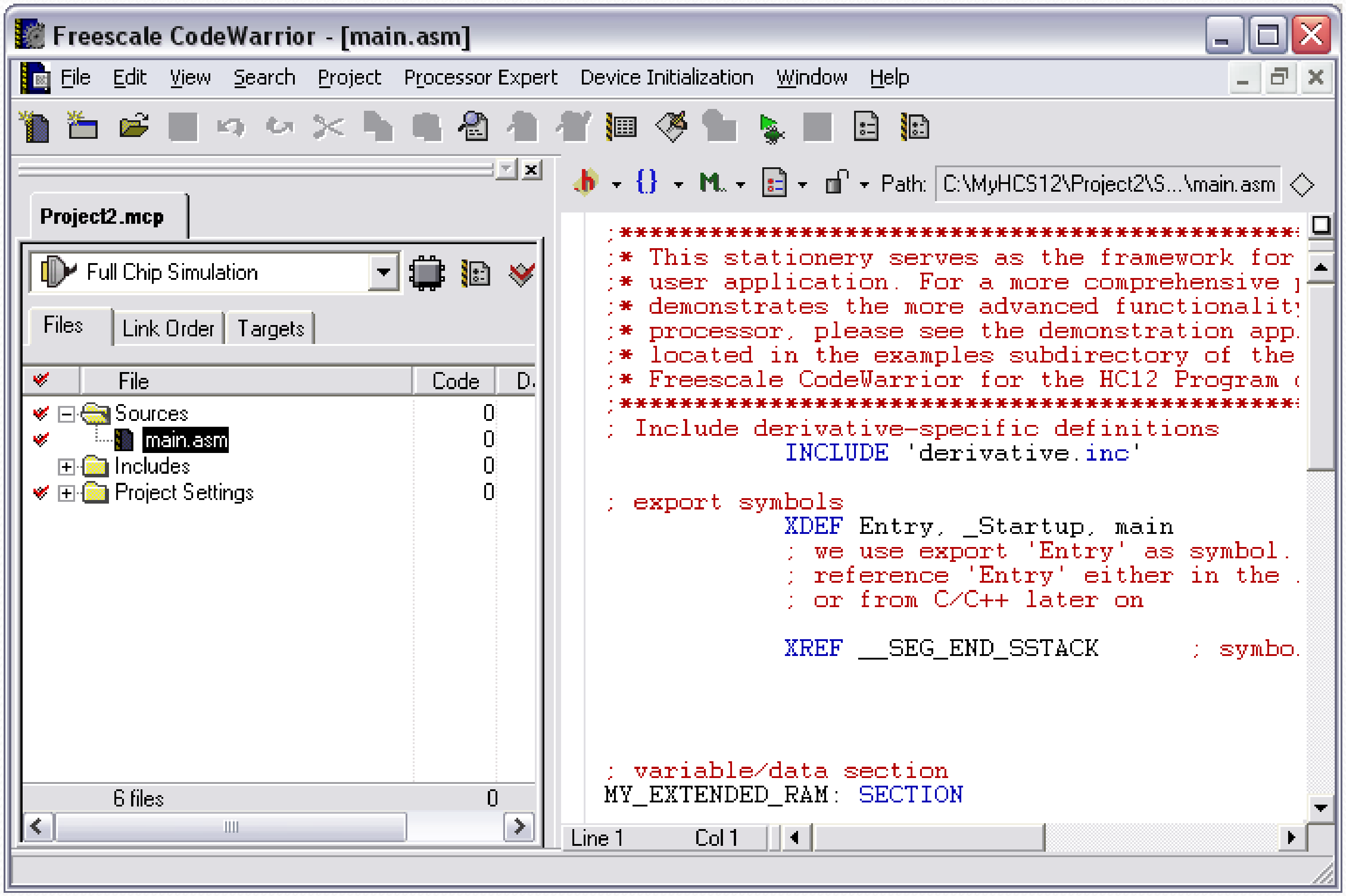
CodeWarrior for HSC12 Assembly Programing

- Choose ‘**None**’ for Rapid Application Development (click Finish).



CodeWarrior for HSC12 Assembly Programing

- Expand the **Sources** folder created by CodeWarrior and double click on main.asm

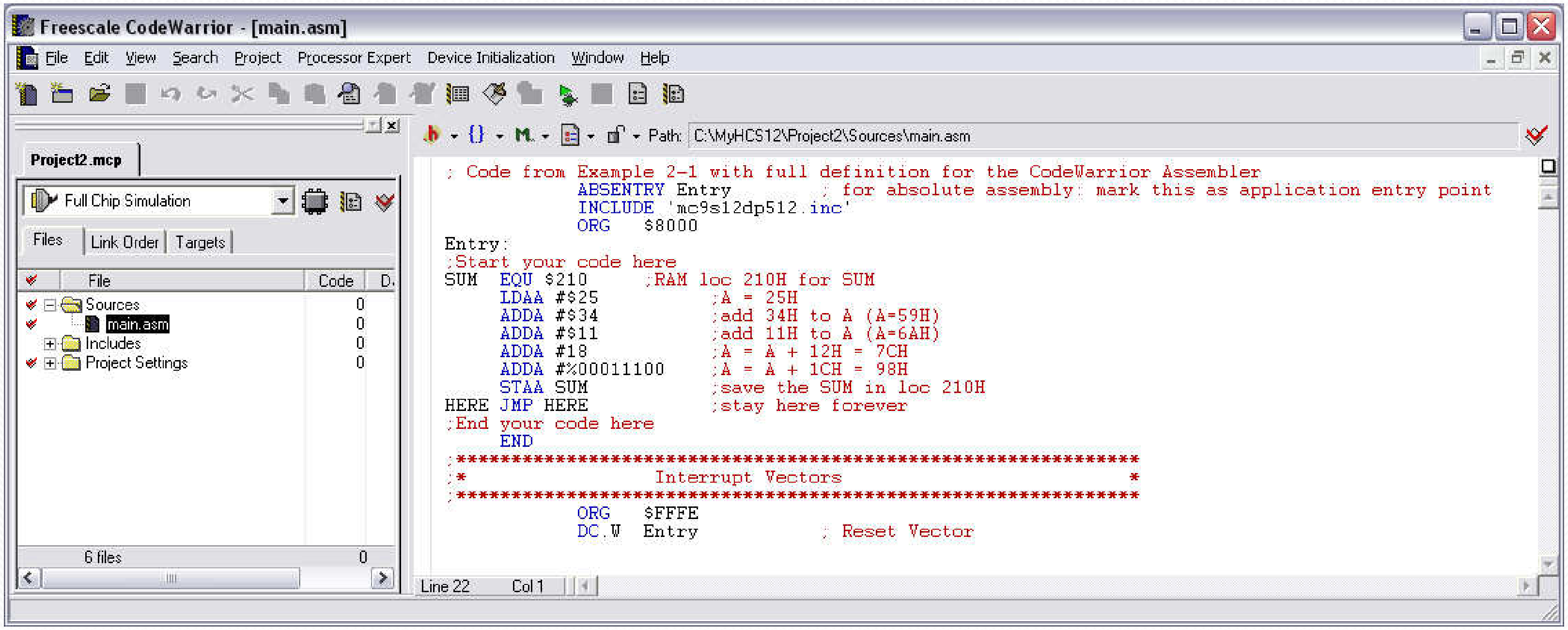


CodeWarrior for HSC12 Assembly Programing

- Minimal requirements for an Assembly program are as follows:
  - ABSENTRY Entry declaration at the top (i.e., application entry point for absolute assembly).
  - INCLUDE file that defines registers of the MCU.
  - ORG declaration above the Entry label.
  - ORG $FFFE and DS.W Entry declaration below the program for reset vector.

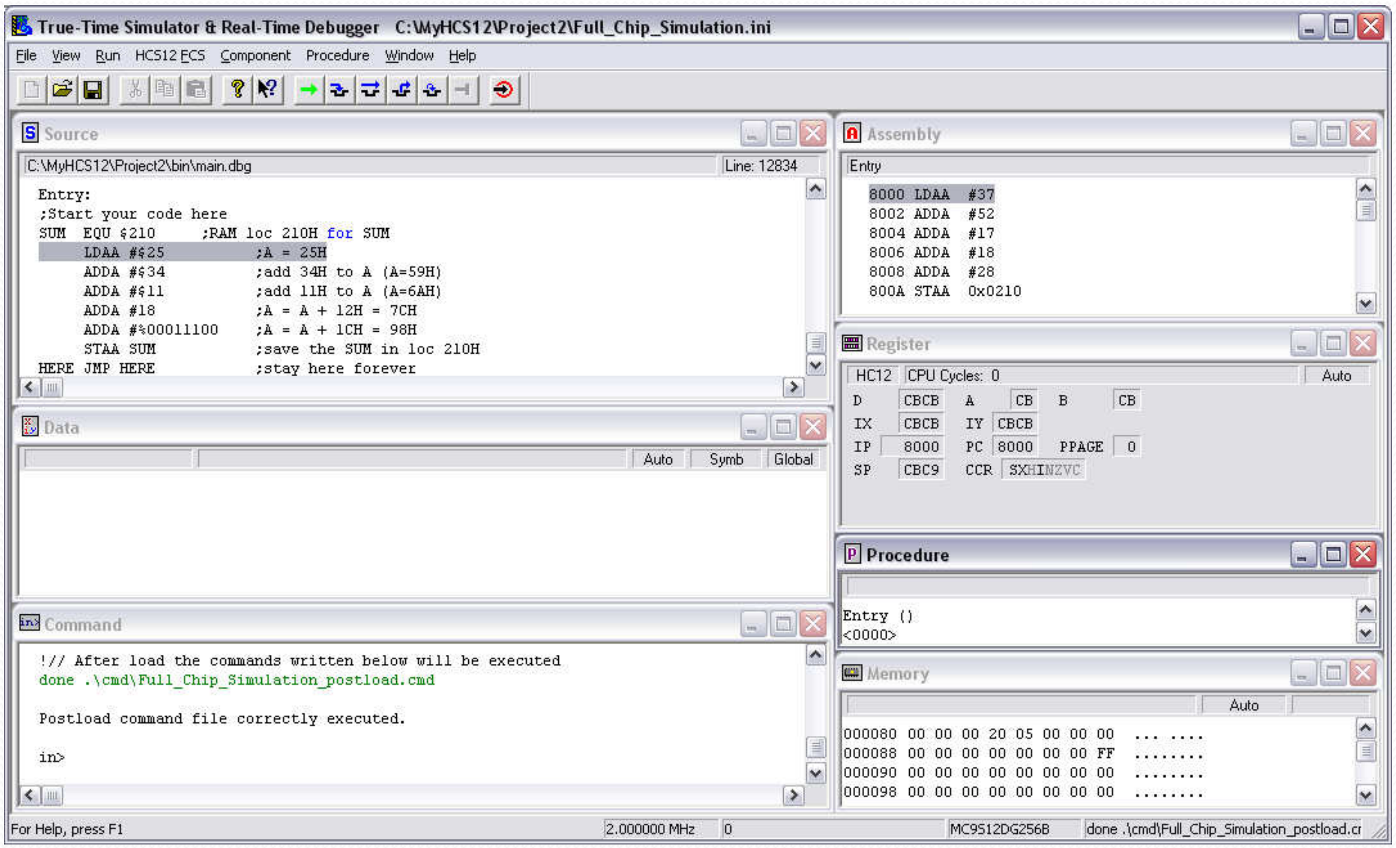
CodeWarrior for HSC12 Assembly Programing

- Note that the book does not define Assembly code this way.
  - This is for compatibility with a variety of Assemblers for the HCS12.
  - You can use the code as template for the examples in the book by replacing the code between Entry: and END.



CodeWarrior for HSC12 Assembly Programing

- Copy the program “Toggling_LEDs_on_PORTB” provided on Moodle course page and paste it into the main.asm program.
- Full Chip Simulation should be selected in the drop-down box
  - Click Make icon (F7) Click Debug (F5).



CodeWarrior for HSC12 Assembly Programing

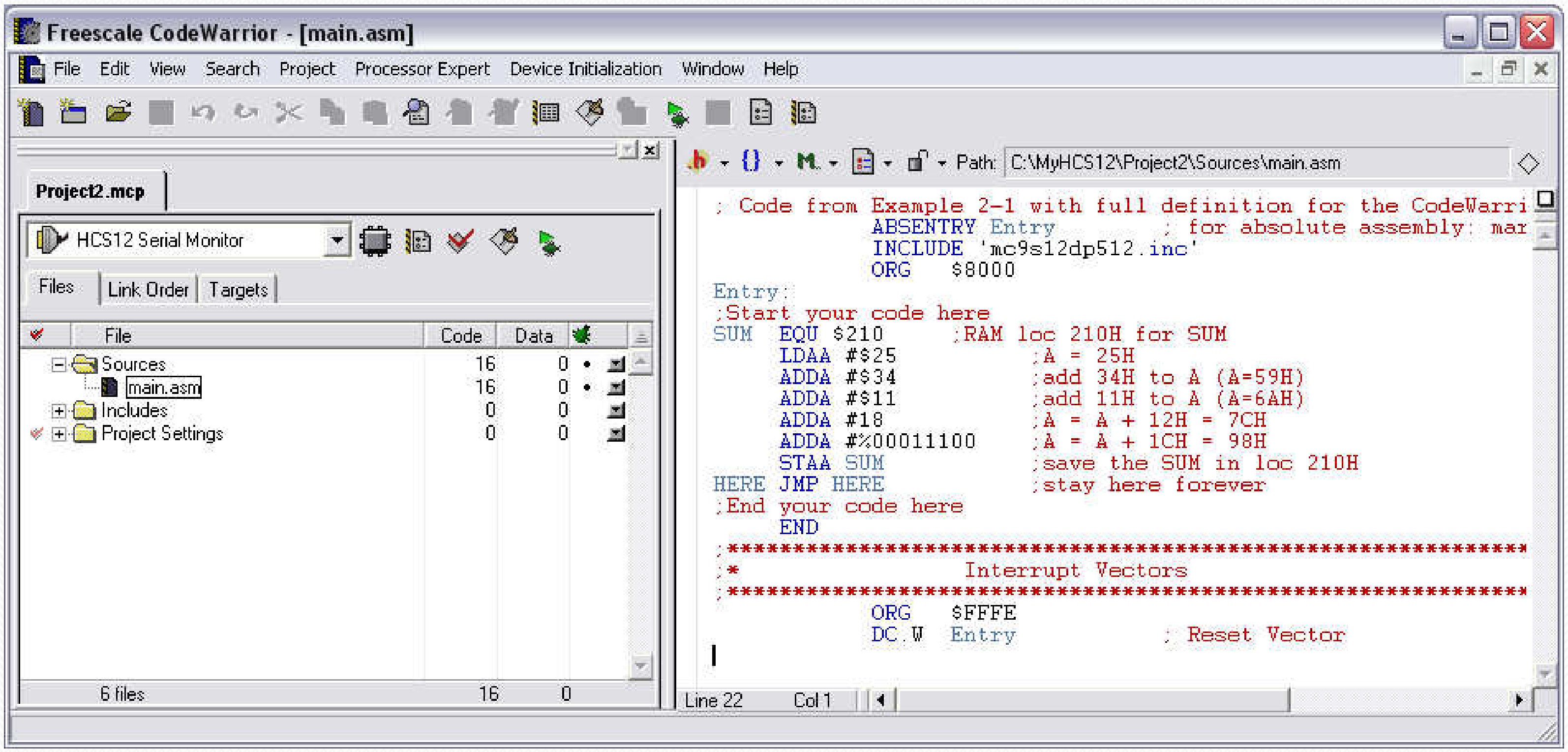
- After clicking F5 (Debug) CodeWarrior will give you “True Time Simulator & Real Time Debugger” screen.
- Use F11 to single step through the program or use blue icons to debug the program.
- To examine the contents of memory locations, click on Memory window and you will see Memory at the top (next to Component).
- Click on Memory and drop-down menu gives you the options
  - See “CodeWarrior_Debugger_HC12” file provided on Moodle course page for more details.

CodeWarrior for HSC12 Assembly Programing

- Compiling, downloading and executing a program for Dragon12-Light Board
  - In the drop-down where it shows Full Chip Simulation choose HCS12 Serial Monitor.
  - Connect your Dragon12 board to the x86 PC USB port and power up the board.
  - Press the RESET button.
  - Then Press F7 (make), F5 (Debug) to Download (make sure you choose the right COM port), and F5 (run) to execute the program.
  - You will see the LEDs toggle on and off.

CodeWarrior for HSC12 Assembly Programing

- Always close the True-time Simulation window, RESET the board, and make sure you are in HCS12 Serial Monitor mode before you download and execute any program.



Parallel I/O Port and Simple I/O Devices

- HCS12 device may have from 48 to 144 pins arranged in 3 to 12 I/O Ports.
  - An I/O port consists of a set of I/O pins and the registers required to control its operation.
    - Each I/O port has a data direction register (DDRx, x is the port name)
  - An I/O pin can be configured for input (0) or output (1).
  - An I/O pin usually serves multiple functions.
    - When it is not used as a peripheral function, an I/O pin can be used as a GP I/O pin.

Parallel I/O Port and Simple I/O Devices

- I/O REGISTERS
  - **Device User Guide** (provided on Moodle course page) shows the detailed register map (Section 1.6) of the MC9S12DG256 from $0000 to $03FF (1KB).
  - Some of the registers which are linked to the most common components in the Dragon12-Light Board will be used.
  - The mc9s12dg256.inc file contains all the numeric equivalent of the port numbers as well as memory positions.

Parallel I/O Port and Simple I/O Devices

- I/O REGISTERS
  - PORTA
    - GPIO register located at $0000.
    - Connected to the keypad on the Board.
    - If the keypad is used, PORTA must be configured so that the bits 7 to 4 are outputs and the bits 3 to 0 are inputs.
      - This is done via **DDRA** register located at $0002.
    - For Keypad: DDRA $\leftarrow$ $F0 = 1111000.
  - PORTB
    - GPIO register located at $0001.
    - Connected to the LEDs on the Board.
    - If LEDs are used, PORTB must be configured so that all the bits are outputs.
      - This is done via **DDRB** register located at $0003.
    - For LEDs and the 7-segment displays: DDRB $\leftarrow$ $FF.

Parallel I/O Port and Simple I/O Devices

- I/O REGISTERS
  - PORTP
    - GPIO register located at \$0258.
    - On the Board, the bits 3 to 0 are connected to the cathodes of the 7-segment displays; while the bits 4 to 6 are connected to the RGB LED.
    - If used with the 7-segment display and the RGB LED, PTP must be configured so that the bits are outputs.
      - This is done via **DDRP** register located at \$025A.
    - For the 7-segment displays and RGB LED: DDRP ← \$FF.
  - PORTH
    - GPIO register located at \$0260.
    - Connected to DIP Switch and Push Buttons (last 4 bits) on the Board.
    - To use the DIP Switch and the Push Buttons, we must configure PTH so that the bits are inputs.
      - This is done via **DDRH** register located at \$0262.
    - For DIP Switches: DDRH ← \$00.

Parallel I/O Port and Simple I/O Devices

- I/O REGISTERS
  - PORTK
    - GPIO register located at $0032.
    - Connected to the LCD controller on the Board.
    - If the LCD Controller is used, PORTK must be configured so that some bits are outputs and other inputs.
      - This is done via **DDRK** register located at $0033.

Parallel I/O Port and Simple I/O Devices

- I/O REGISTERS

Number of pins available in each paralle port

Port Name	No. of Pins	Pin Name
A	8	PA7~PA0
B	8	PB7~PB0
E	8	PE7~PE0
H	8	PH7~PH0
J	4	PJ7~PJ0
K	7	PK4~PK0
M	8	PM7~PM0
P	8	PP7~PP0
S	8	PS3~PS0
T	8	PT7~PT0
PAD1, PAD0	16	PAD15~PAD0
L	8	PL7~PL0
U	8	PU7~PU0
V	8	PV7~PV0
W	8	PW7~PW0

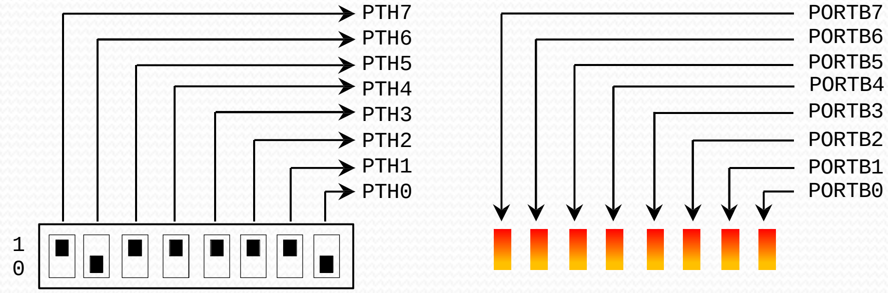
Parallel I/O Port and Simple I/O Devices

- Example 1

Read the DIP switches and output the result to the LEDs.

```
DDRH ← $00
DDRB ← $FF
while (1)
    A ← PORTH
    PORTB ← A
end
```

ASM Code: `unit5a.asm`



Parallel I/O Port and Simple I/O Devices

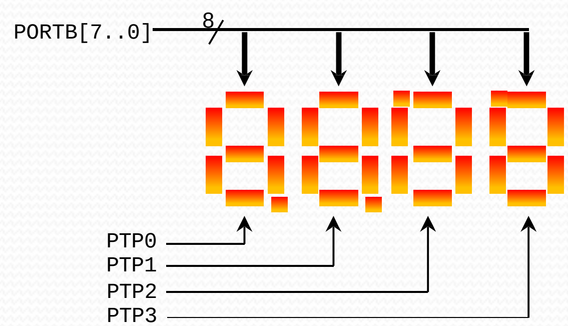
- Example 2

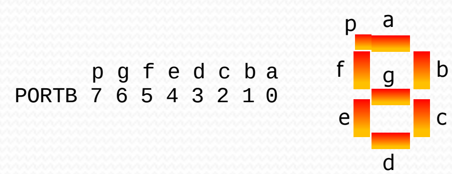
Read 4-bit data from DIP switches (the last 4 LSBs) and display the hexadecimal value on the four 7-segment displays.

```
DDRB ← $FF, DDRP ← $FF
DDRH ← $00
X ← sevsegdata
PTP ← $00
while (1)
     A ← PORTH
     A ← A AND $0F
     Display data on A on
     all 7-seg displays
     B ←  m[[X] + [A]]
     PORTB ← B
end
```

ASM Code: unit5b.asm

	Address	8 bits	
	...		
sevsegdata →	0x1000	$BF	'0'
	0x1001	$86	'1'
	0x1002	$DB	'2'
	0x1003	$CF	'3'
	0x1004	$E6	'4'
	0x1005	$ED	'5'
	0x1006	$FD	'6'
	0x1007	$87	'7'
	0x1008	$FF	'8'
	0x1009	$EF	'9'
	0x100A	$F7	'A'
	0x100B	$FC	'B'
	0x100C	$B9	'C'
	0x100D	$DE	'D'
	0x100E	$F9	'E'
	0x100F	$F1	'F'
	...		



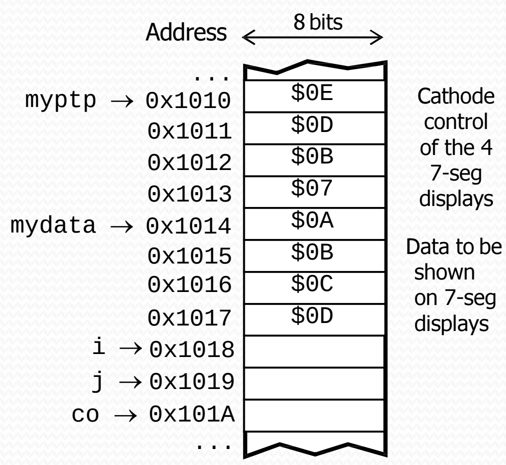


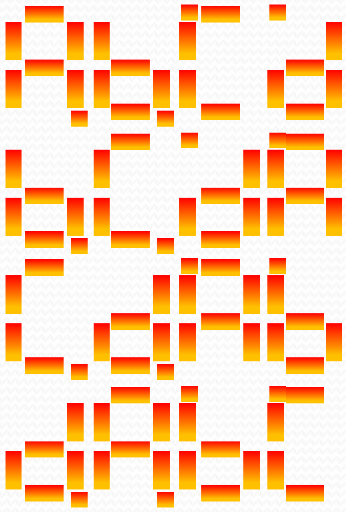
Parallel I/O Port and Simple I/O Devices

- Example 3

Read 4 bytes of data from memory and display the hexadecimal values on the four 7-segment displays. Then, rotate the 4 digits to the left every 1 second. For visual persistence, display each digit for only 1 ms. This delay was computed using a bus speed of 24 MHz (this requires an extra piece of code, this code could not run on the Debugger step by step).

```
X ← sevsegdata
Y ← myptp
co ← 0        ; indicates the 'shift'
while (1)
  for j = 0 to 249
    for i = 0 to 3
      idx ← i + co
      if i+co ≥ 4 then
        idx ← i+co-4
      end
      Display data on
       'idx+mydata' on display 'i'
      Delay 1 ms
    end
  end
  co ← co+1
    if co ≥ 4 then
    co ← 0
    end
end
```





ASM Code: unit5c.asm

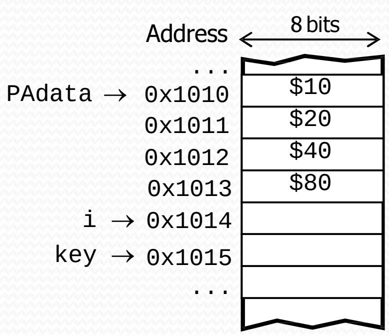
Parallel I/O Port and Simple I/O Devices

- Example 4

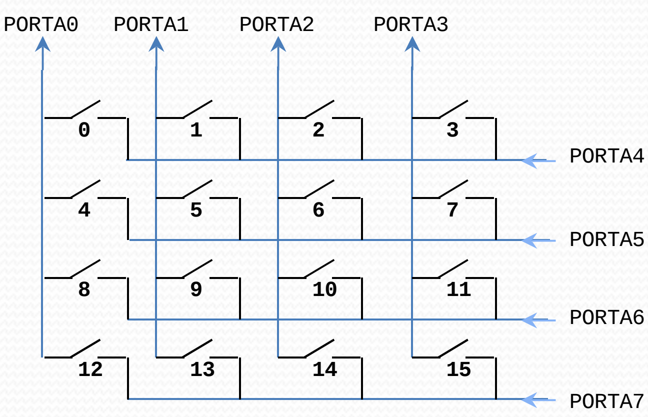
Read the key pad, and display the hexadecimal value on all 7-segment displays.

	Output	Input			
	PORTA[7..4]	PORTA[3..0] = 0001 idx = 0	PORTA[3..0] = 0010 idx = 1	PORTA[3..0] = 0100 idx = 2	PORTA[3..0] = 1000 idx = 3
i=0	0001	KEY0	KEY1	KEY2	KEY3
i=1	0010	KEY4	KEY5	KEY6	KEY7
i=2	0100	KEY8	KEY9	KEY10	KEY11
i=3	1000	KEY12	KEY13	KEY14	KEY15

KEY = 4*i + idx



Row Control for the KeyPad



Parallel I/O Port and Simple I/O Devices

- Example 4 (contd.)

Read the key pad, and display the hexadecimal value on all 7-segment displays.

<table>
<tr><th>Main Routine</th><th>getidx subroutine. Input: A(3..0)</th></tr>
<tr><td><pre>
DDRA ← $F0, DDRB ← $FF, DDRP ← $FF, DDRH ← $00
Ξ.← sevsegdata
PTP ← $00
Ψ.← PAdata
while (1)
   for i = 0 to 3
      PORTA ← [PAdata + i]
      Delay 1 ms
      A ← PORTA ; Read PORTA
      idx = getidx (A(3..0))
      If idx = 15 then
         Display blank on all 7-seg displays
      else
         key ← i*4 + idx ;
         Display hex value of ‘key’ on all 7-seg displays
      end
   end
end
</pre></td><td><pre>
if A(3..0) = 1000
   idx = 3
elsif A(3..0) = 0100
   idx = 2
elsif A(3..0) = 0010
   idx = 1
elsif A(3..0) = 0001
   idx = 0
else
   idx = 15
end if
</pre></td></tr>
</table>

ASM Code: `unit5d.asm`

Parallel I/O Port and Simple I/O Devices

- Example 5

RGB LED control via DIP switch.

PORTP4 = RED LED cathode, PORTP5 = BLUE LED cathode, PORTP6 = GREEN LED cathode.

The anodes of these 3 LEDs are connected to PTM2. All the pins that control the LEDs are in negative logic.

Input: PTH[2..0]. PTH2 = Red, PTH1 = Green, PTH0 = Blue

Output: PTP[6..4]. /R = not (PTP4), /G = not (PTP6), /B = not (PTP5).

```
DDRH ← $00, DDRM ← $FF, PTM ← $00;
PTM2 = 0          ; sets the anodes to ‘1’
while (1)
A ← PTH
A ← A AND $07     ; A stores only the 3 LSBs or PORTH
temp(4) ← A(2), temp(6) ← A(1), temp(5) ← A(0)
; Rewires the bits of PTH[2..0] into PTP[6..4]
PTP ← temp        ; PTP controls the cathodes.
if a bit in PTP is ‘1’ then
  that cathode is ‘0’
end
```

R	G	B	Color
0	0	0	Black
0	0	1	Blue
0	1	0	Green
0	1	1	Cyan
1	0	0	Red
1	0	1	Magenta
1	1	0	Yellow
1	1	1	White

ASM Code: `unit5e.asm`