C Language Programming with CodeWarrior

Dr. Abdelaziz Trabelsi
COEN 317 (Fall 2019)

Outline

- Introduction to C
- Data types for HCS12
- C program structure for HCS12
- Guidance on CodeWarrior
- CodeWarrior for HCS12 C programming
- Examples with
  - Assembly Programming
  - Mixed C/Assembly Programming

Introduction to C

- C has gradually replaced assembly language in many microprocessor-based applications.
- A summary of C language constructs that are used in HCS12 programming are covered in this lecture.
- Books on C language
  - Kernighan & Ritchie, “The C Programming Language”, Pearson, 2015.
  - Deitel & Deitel, “C: How to Program”, Prentice Hall, 2015.
  - Kelly & Pohl, “A Book on C: Programming in C”, Addison-Wesley, 1998.

Introduction to C (Contd.)

- A C program consists of functions and variables.
- A function contains statements that specify the operations to be performed.
- Types of statements:
  - Declaration
  - Assignment
  - Function call
  - Control flow
  - Null
- The `main()` function is required in every C program.

Data Types for HCS12

Signed integer (2's compl.)	Unsigned integer	# of bytes
`char`	`unsigned char`	1
`int`	`unsigned int`	2
`long int`	`unsigned long int`	4

C Program Structure for HCS12

- Example of `main()` function:

```
/* common defines and macros: file found in CodeWarrior folder */
#include <hidef.h>
/* derivative-specific definitions (e.g., PORTA, DDRK) */
#include "derivative.h"


/* for malloc, calloc */


#include <stdlib.h>              // standard library in C

/* global variables: for Small Memory Model, they are place
right after the Stack (i.e., starting from $1100). Stack is
empty at $1100*/
unsigned char a, b;              // a, b: unsigned 8-bit numbers

/* Small Memory Model: the program starts at $C000 */

// function declaration
unsigned char myfun (unsigned char val); // Input: 1 byte. Output: 1 byte
```

C Program Structure for HCS12 (Contd.)

- The main function looks like the following:

```
/* body: considered to be a subroutine in the ASM code */
void main(void) {
/* local variables */
unsigned char i;
...
...
}

/* functions*/
unsigned char myfun (unsigned char val)
{
unsigned char temp;
...
...
return temp;            // value of 'temp' is returned
}
```

Guidance on CodeWarrior

- Memory Model
  - Small (Program/Data fit into 64 KB memory space).
- Program and Data Location
  - Program starts at $C000 (16 KB Fixed Flash, Page $3F).
  - By default, data starts at $1100.
  - Stack starts (empty) at $1100 (Stack size = 0x100).
- Debug
  - The main program in C is treated as a subroutine and the variables are local variables stored in the Stack.
  - Windows Data:1 and Data:2 show how variables change in real time.
    - Use Step Over to avoid entering ASM code of every C instruction.
    - Use Step On when you want to execute a C function line by line. Once inside the C function, use Step Over to avoid entering ASM code of every C instruction.

Guidance on CodeWarrior (Contd.)

- Global variables
  - Defined outside main program.
  - Placed starting at address $1100.
  - Instruction loading data is needed to initialize a global variable.
- Constants
  - Do not occupy memory positions (equivalent to EQU).
- Local variables
  - Initializing them is same as to execute a series of instructions to load data on the Stack.
  - Initial values are loaded using normal ASM instructions.

Guidance on CodeWarrior (Contd.)

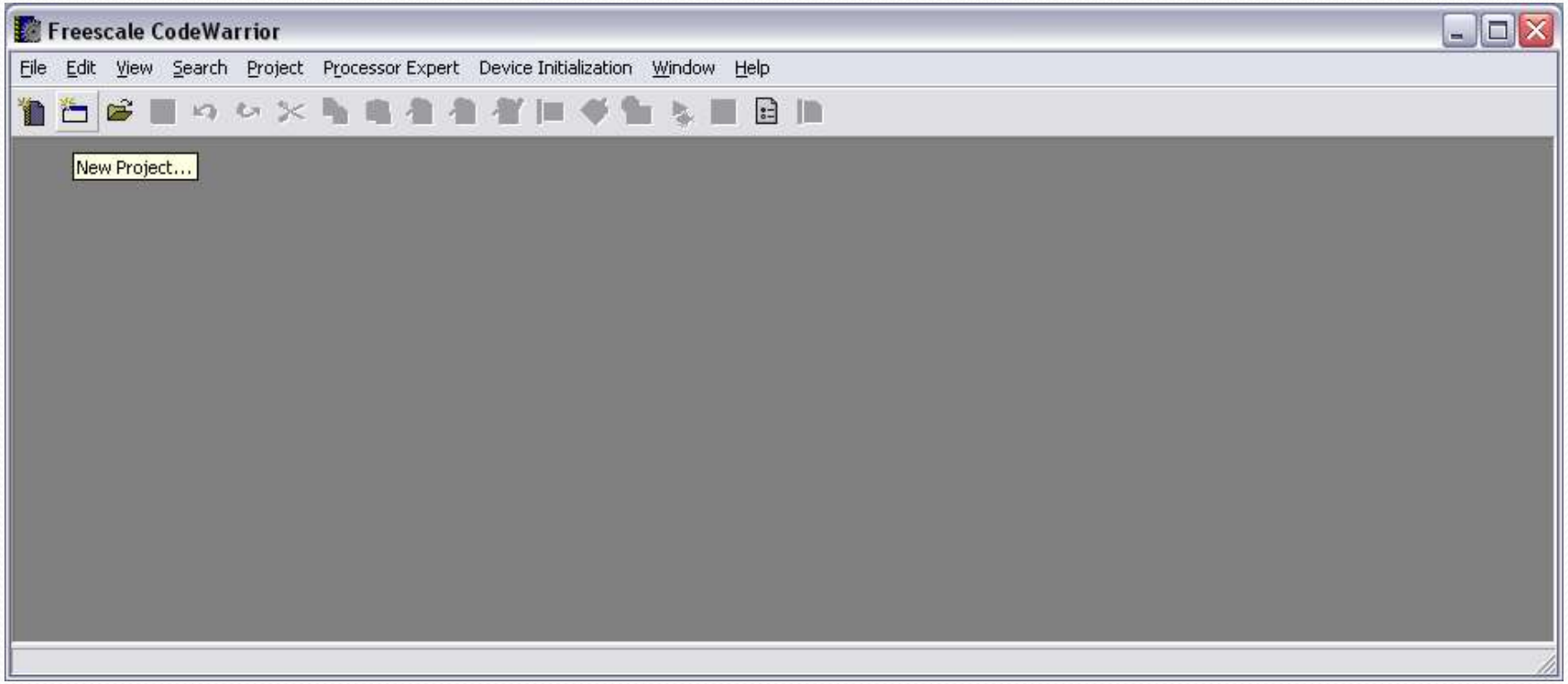
- I/O registers
  - Occupy memory space from $0000 to $03FF.
  - All common names can be used: PORTA, PORTB, PORTP, PORTH, PORTM, DDRA, DDRB, DDRP, DDRH, DDRM, etc.
- Sign-extension
  - In additions and subtractions, no worry about sign extension.
  - If all variables are signed and if we accumulate numbers into a larger data size, sign-extension will be taken care of automatically.

CodeWarrior for HCS12 C Programming

- Start CodeWarrior for HCS12 and click Create New Project.

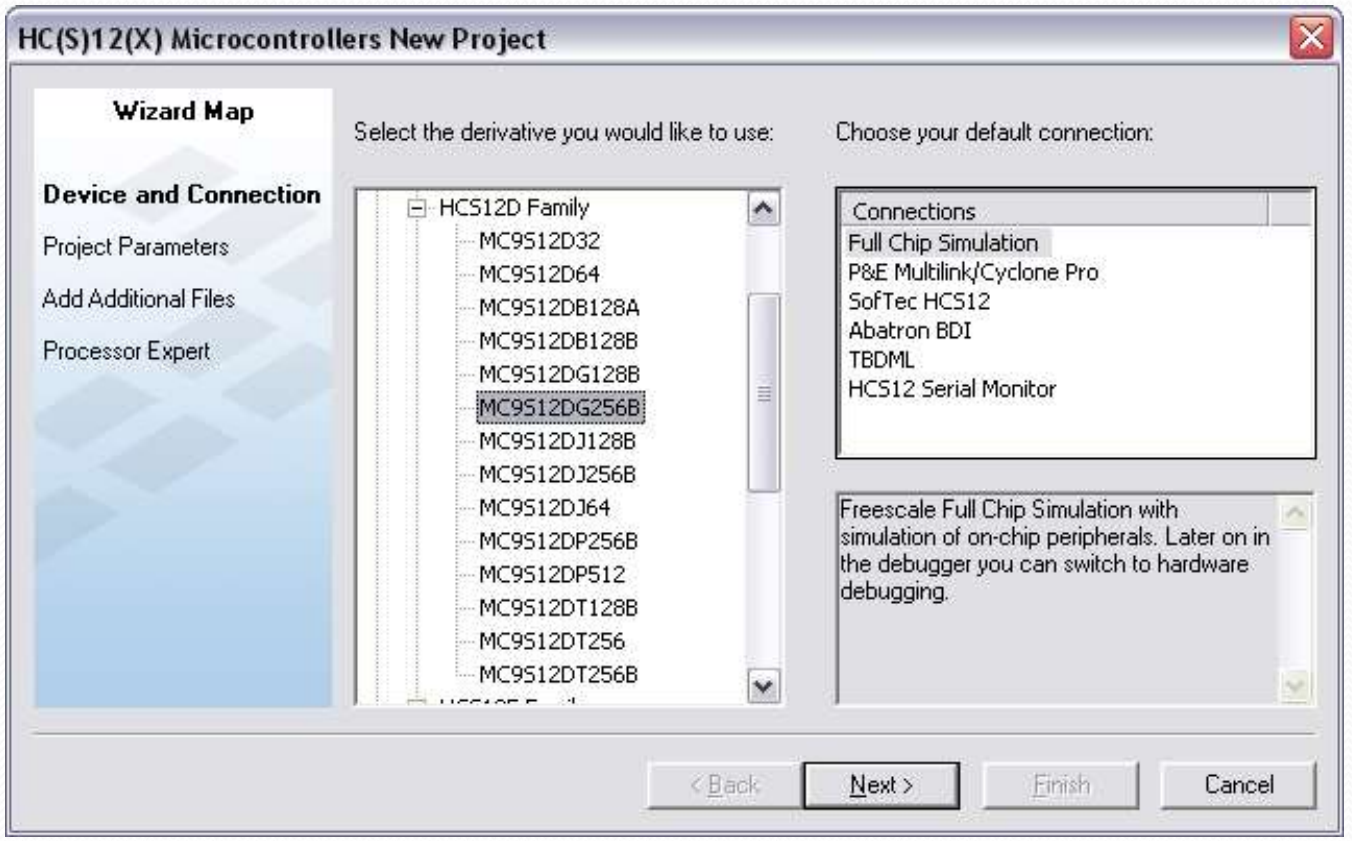
CodeWarrior for HCS12 C Programming

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



CodeWarrior for HCS12 C Programming

- 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.

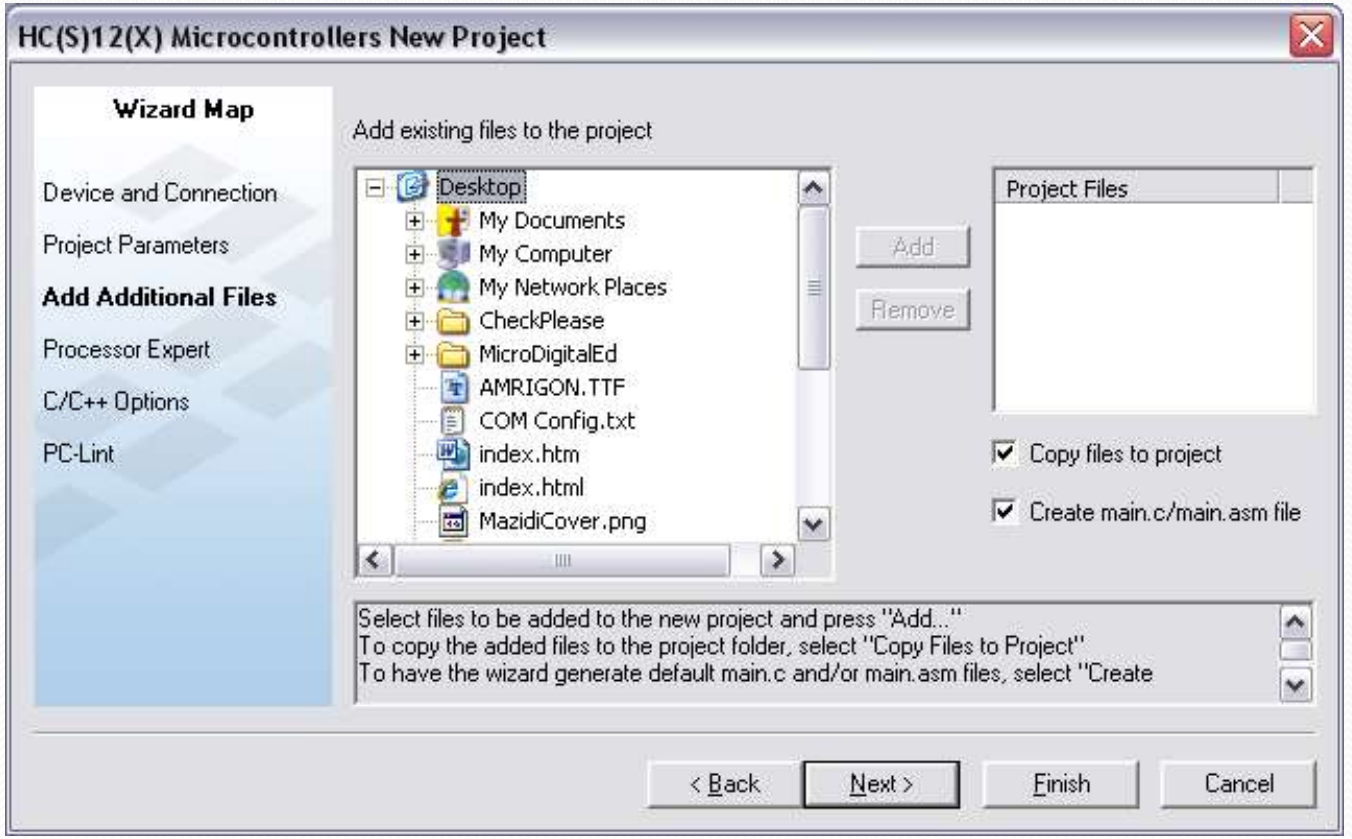


CodeWarrior for HCS12 C Programming

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

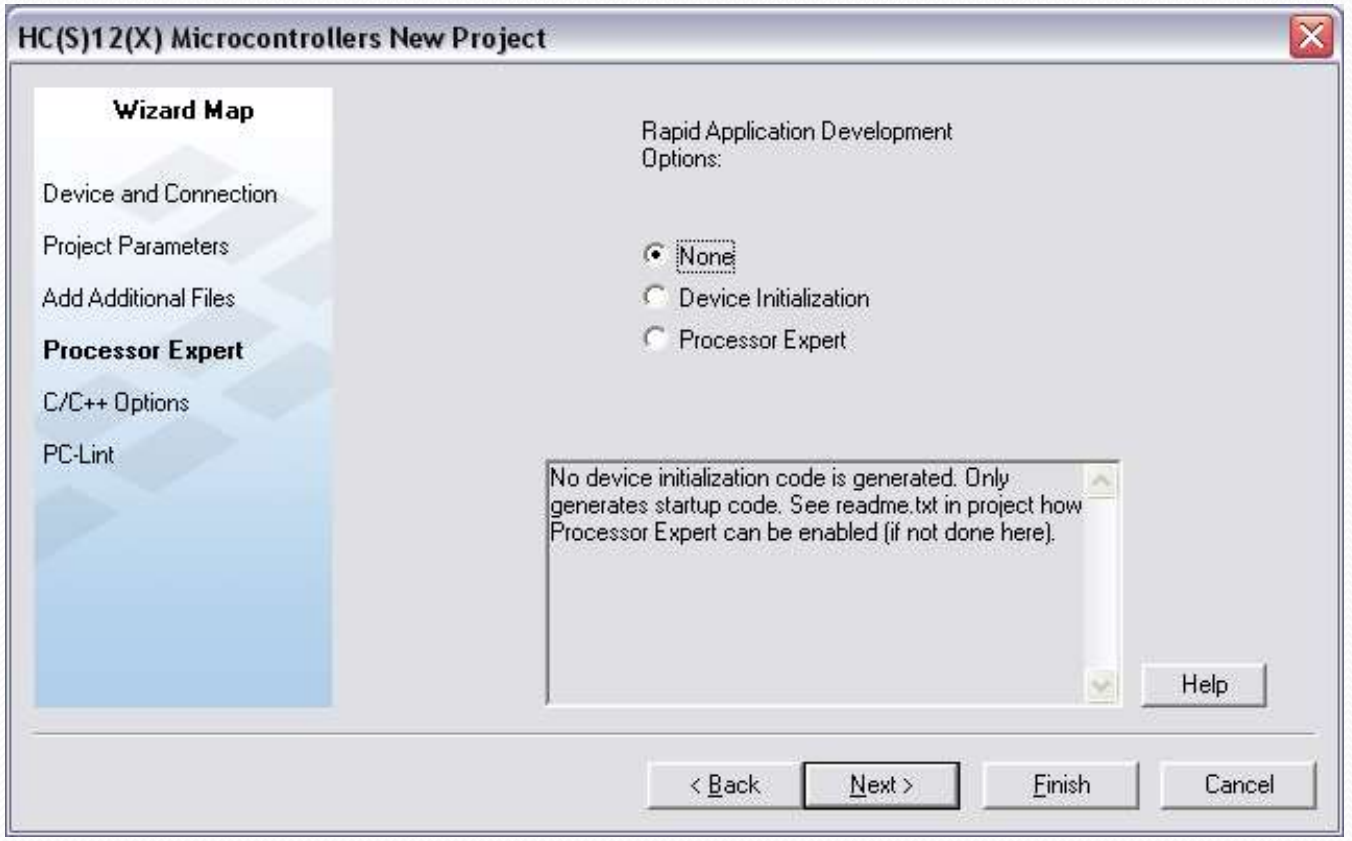
CodeWarrior for HCS12 C Programming

- 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.



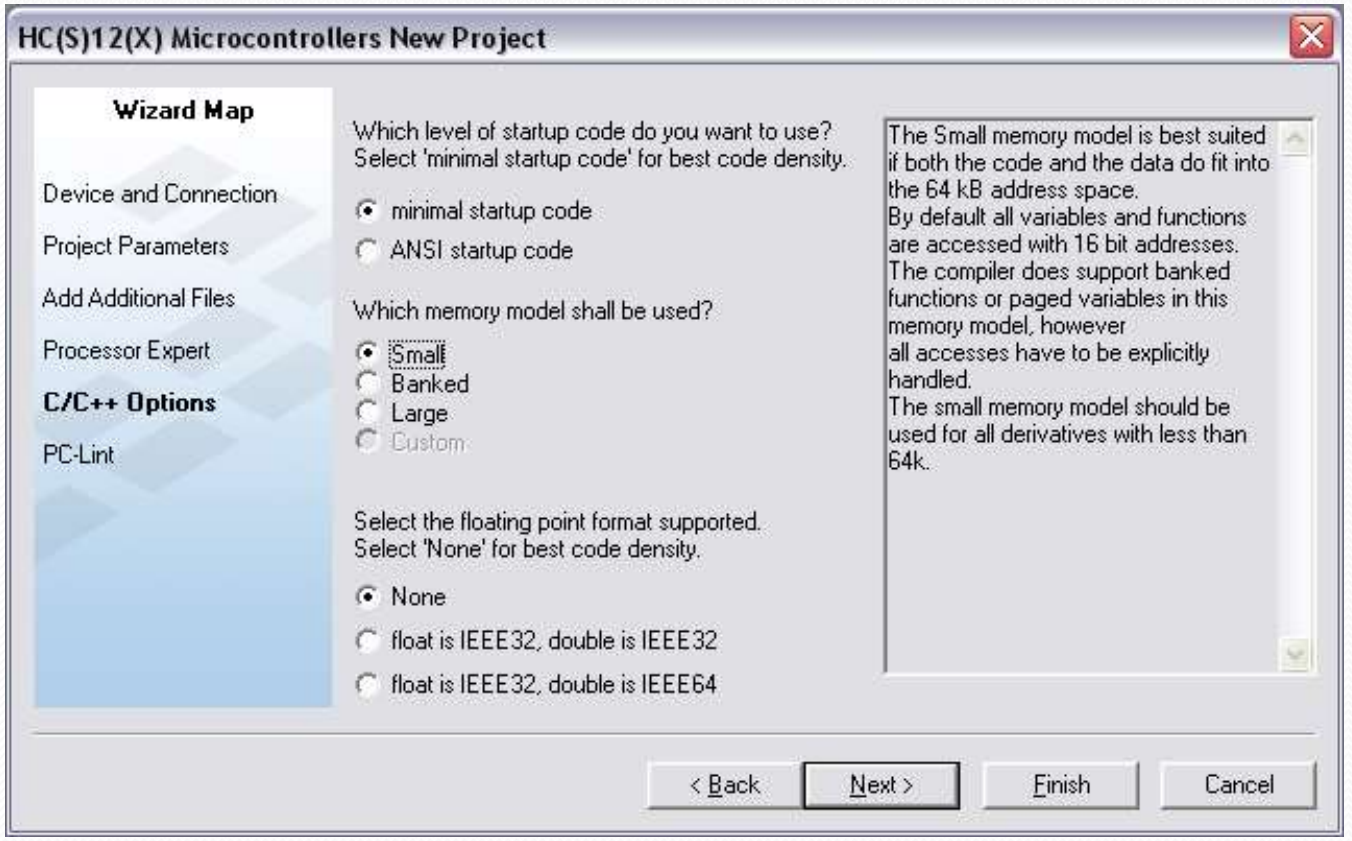
CodeWarrior for HCS12 C Programming

- Choose '**None**' for Rapid Application Development (click Next).



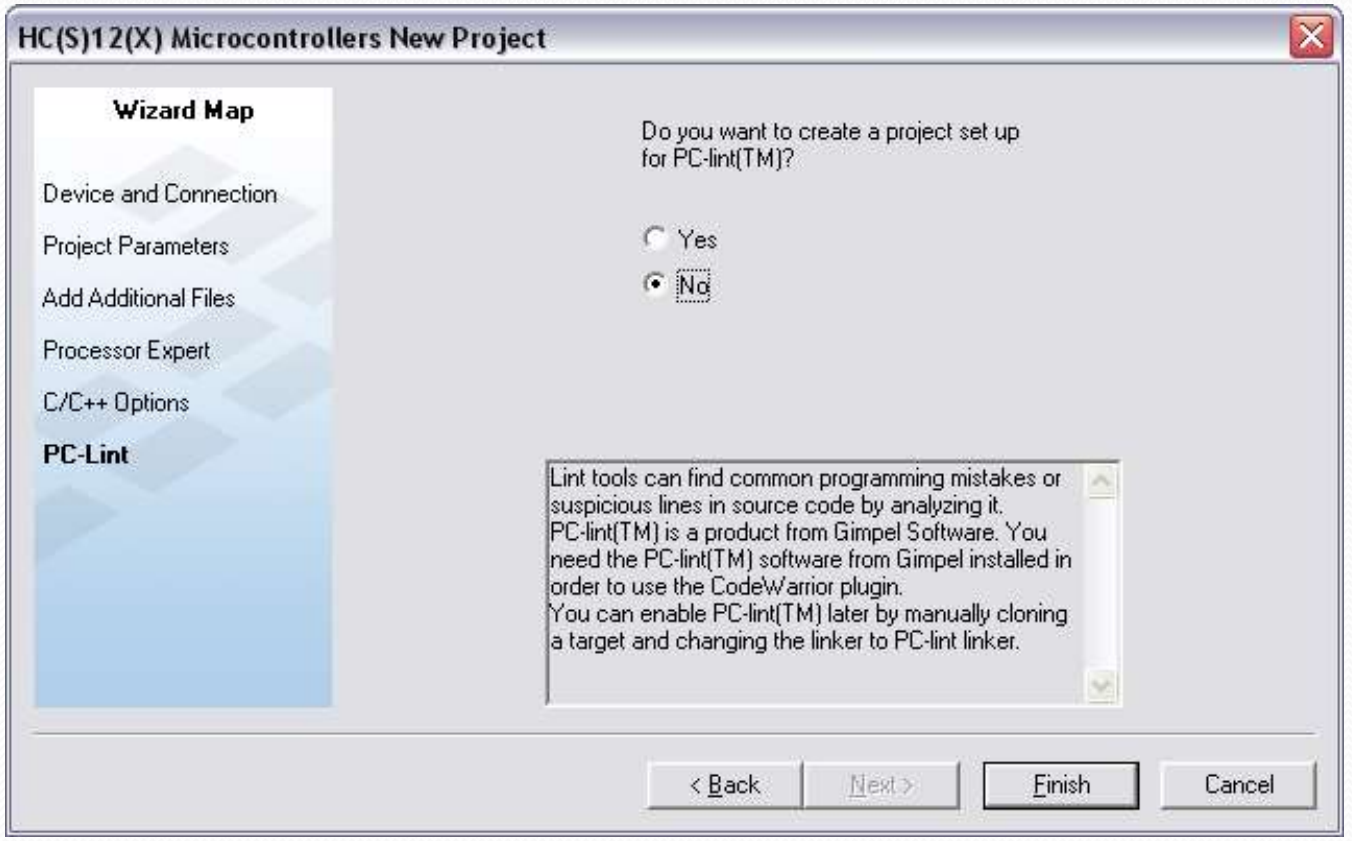
CodeWarrior for HCS12 C Programming

- Choose the start up options shown in the figure (click Next).



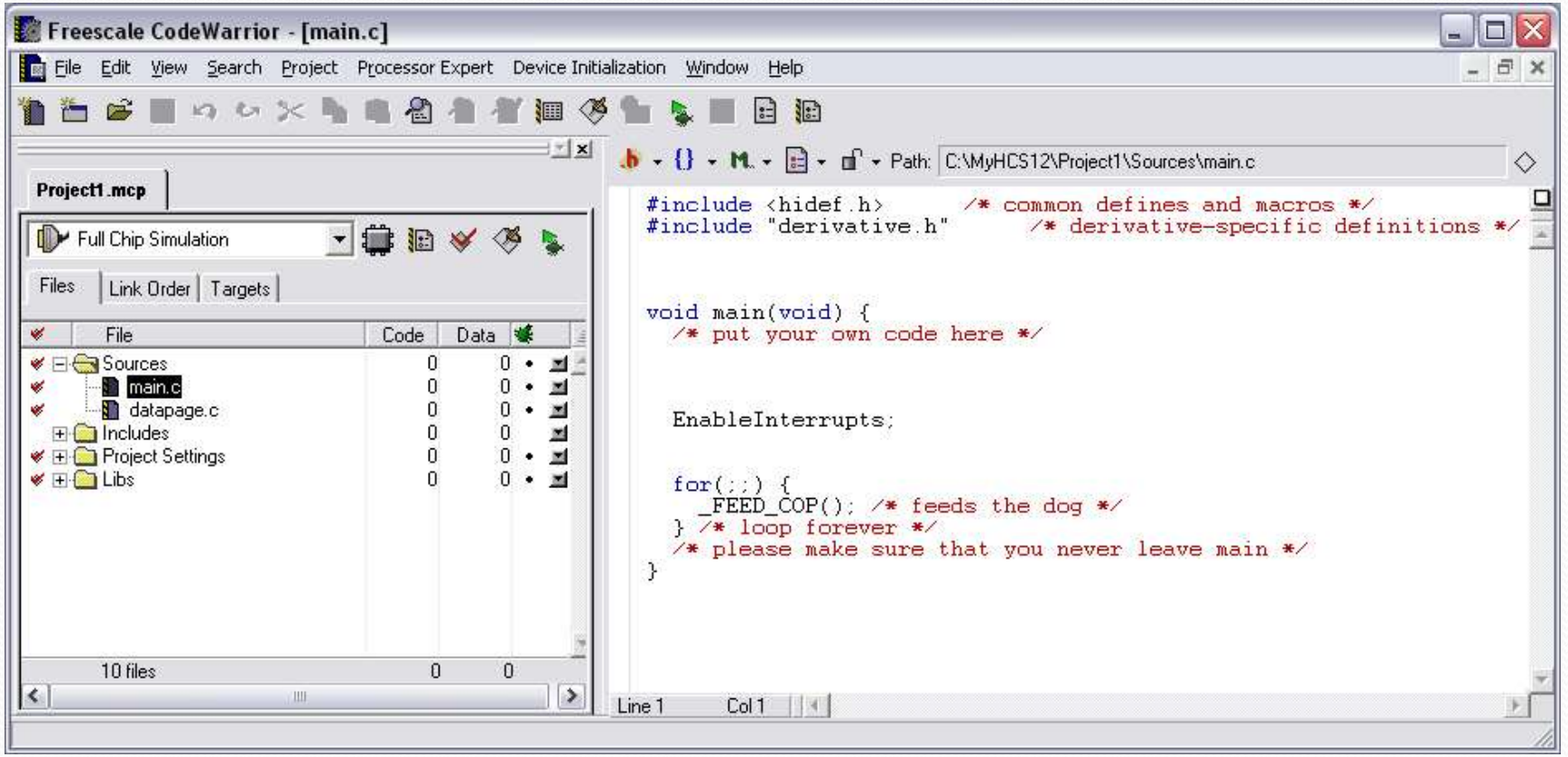
CodeWarrior for HCS12 C Programming

- Choose "**No**" for PC_Lint (click Finish).



CodeWarrior for HCS12 C Programming

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

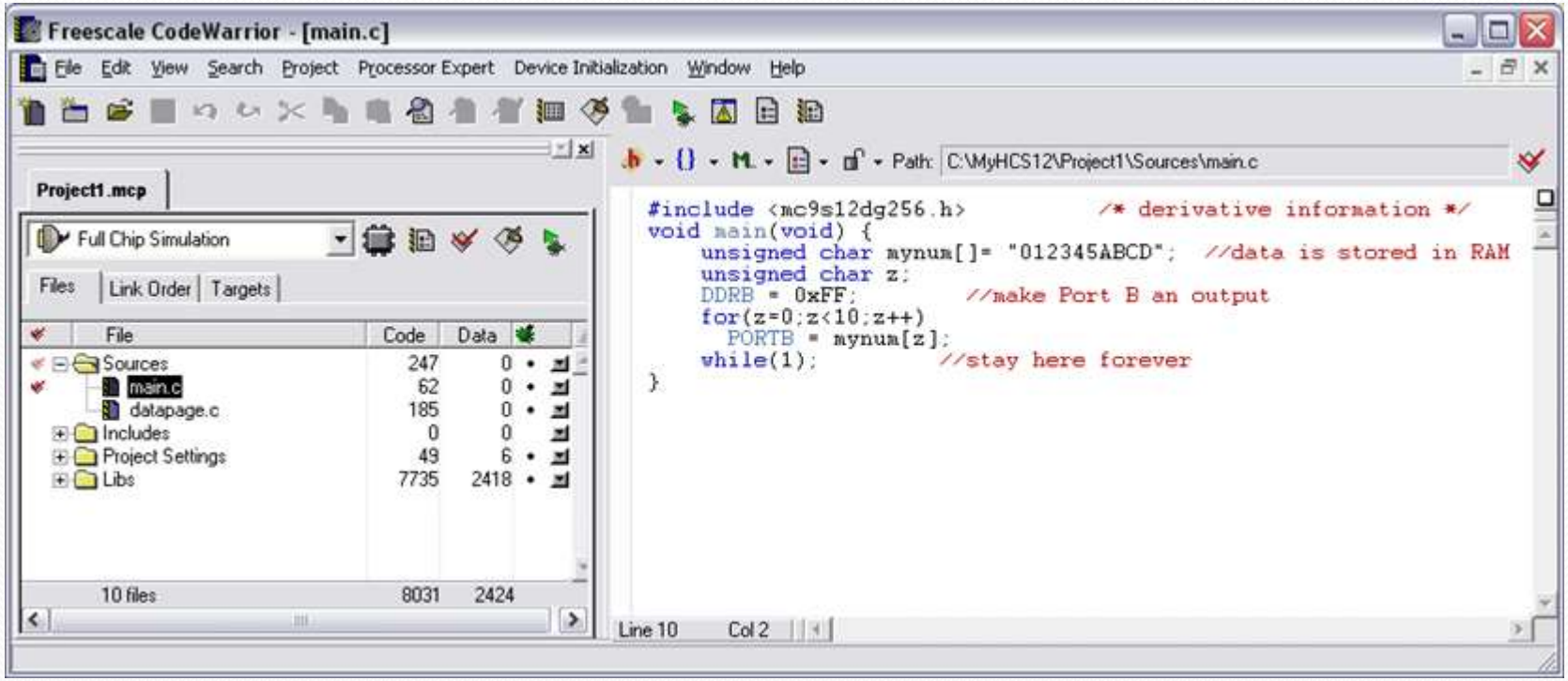


CodeWarrior for HSC12 C Programing

- Minimal requirements for a C:
  - Header file that defines the registers of the MCU.
  - Function `void main(void) {…}` with a blocking function at the bottom or the program should be an infinite loop.

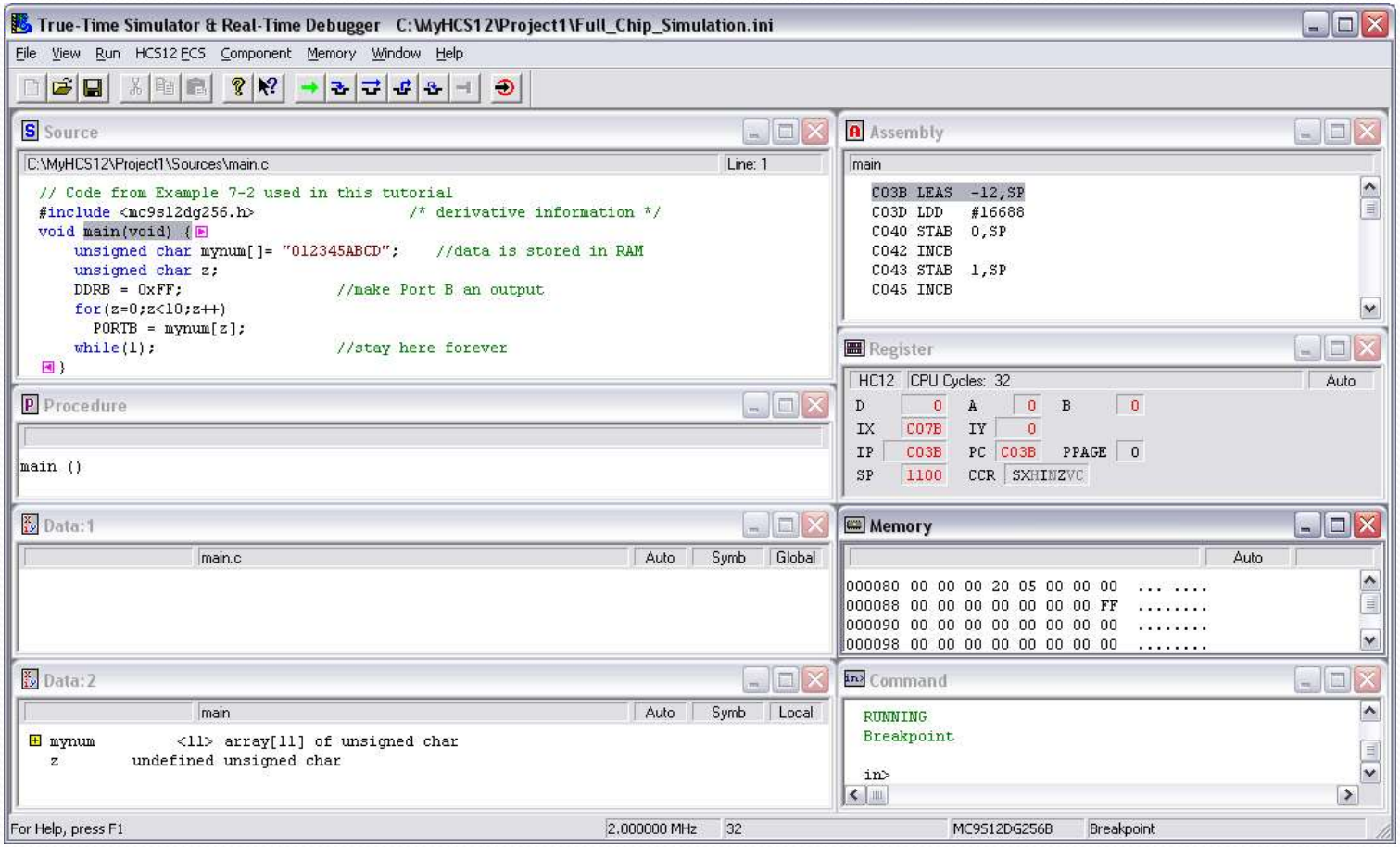
CodeWarrior for HSC12 C Programing

- Replace the template code in main.c program with the program shown below.



CodeWarrior for HSC12 C Programing

- Full Chip Simulation should be selected in the drop-down box
  - Click Make icon (F7) Click Debug (F5).



CodeWarrior for HSC12 C Programing

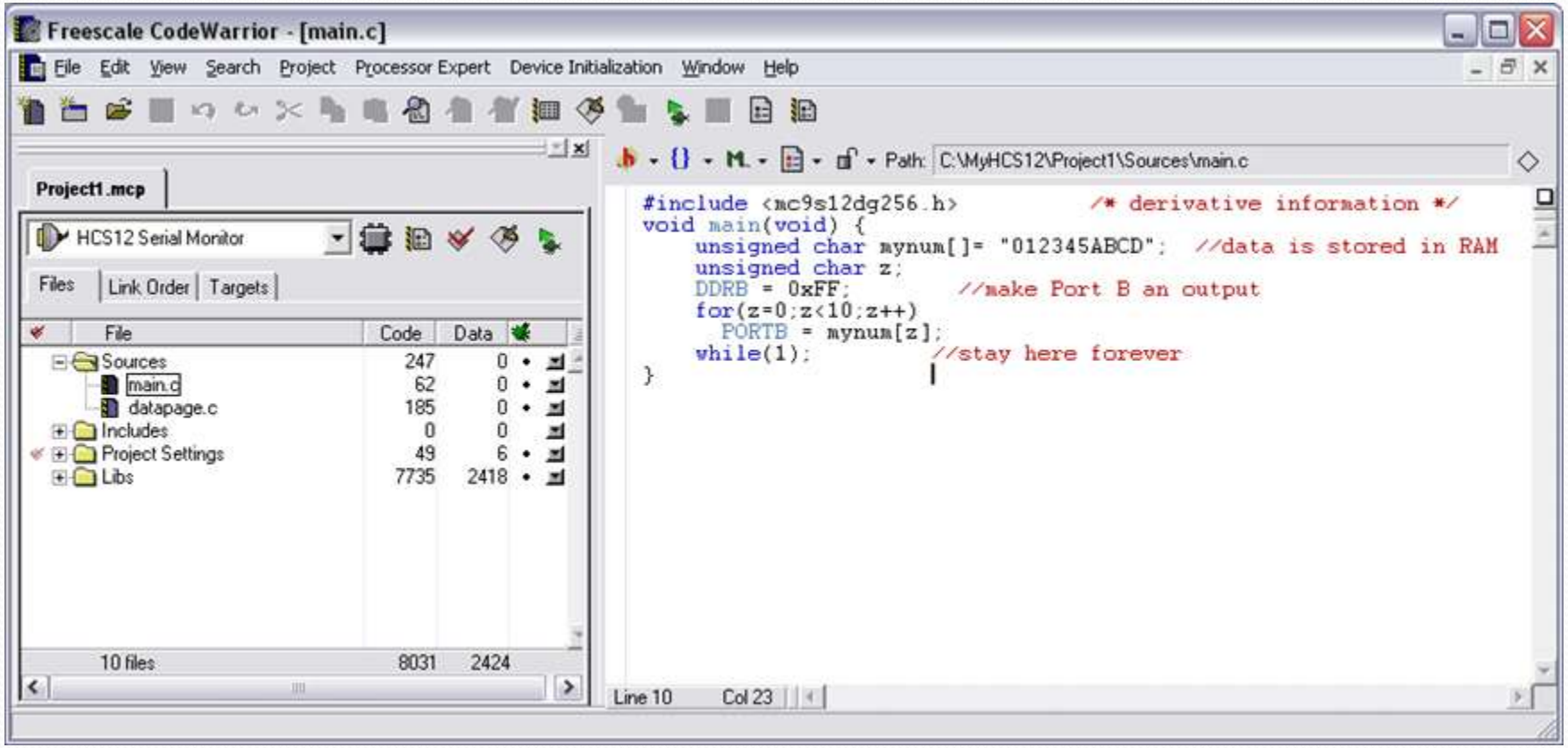
- After clicking F5 (Debug), CodeWarrior will open “True Time Simulator & Real Time Debugger” screen.
- Use F11 to single step through the program or use blue icons to debug the program.
- To examine the contents of memory locations, click on Memory window and you will see Memory at the top (next to 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 C 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.

CodeWarrior for HSC12 C 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.



Assembly Programming

- Example 1

  Calculate the average of an array. Use the DIP Switch to do an OR operation between the computed average and the DIP Switch state. The result must appear on the LEDs. The following constructs are covered:

  - Functions in C (input parameters, output parameters).
  - HCS12 I/O registers.
  - Bit-wise AND operator and OR operator.

  Provided **C Code:** unit6a.c

Assembly Programming (Contd.)

- Example 2

  Given ten (10) 16-bit unsigned numbers in an array, calculate the sum. Then, add the sum to an initial value. The following constructs are covered:

  - A local variable defined as a constant, and a global variable with initial value.
  - A for-loop.
  - Inline assembly instructions:
    - Use CPU registers, I/O registers, and variables in C as memory locations (e.g., `asm("ldd sum"); load value of variable sum onto register D`).
    - You might modify execution of your program by modifying the registers. If you are not sure whether you will affect the execution of your program, you can do: `asm ("pshd"); asm ("ldd sum"); asm ("puld").`

  Provided **C Code:** `unit6b.c`

Assembly Programming (Contd.)

- Example 3

  Given an array of numbers, find the maximum or minimum (depending on DIP Switch bit 0). If the bit is 1, the maximum appears on the LEDs. If the bit is 0, the minimum appears on the LEDs. The following constructs are covered:

  - Functions in C (input array).
  - While statement.

  Provided **C Code:** unit6c.c

Assembly Programming (Contd.)

- Example 4

  Given an array of numbers, calculate the polynomial and store it on another array. The following constructs are covered:

  - Type-casting in C.
  - Functions in C (input array, output array).
  - Pointers.

  Provided **C Code:** unit6d.c

Mixed C/Assembly Programming

- The following topics are covered:
  - Mixed C/Assembly code in CodeWarrior.
  - Creation of header and ASM files.
  - Input parameters to an ASM function: 1 byte (B), 2 bytes (D), 3 bytes (B:X), 4 bytes (X:D)
  - Getting the return value of an ASM function as an accessible value in the C code.

Mixed C/Assembly Programming (Contd.)

- Example 5

  Add a number stored as a constant to another number (structures are used).

  Provided **C Code:** unit6e.c mixasm_6e.h

  Provided **ASM Code:** mixasm_6e.asm

Mixed C/Assembly Programming (Contd.)

- Example 6

  Display a 4-byte number on the 7-segment displays.

  Provided **C Code:** `unit6f.c  mixasm_6f.h`

  Provided **ASM Code:** `mixasm_6f.asm`