C Programming Language
C

• C is better to use than assembly for embedded systems programming.
  – You can program at a higher level of logic than in assembly, so programs are shorter and easier to understand.

• The C compiler will translate the C program into machine code for you.
  – A possible disadvantage is that the resulting machine code may be longer and slower than if you wrote assembly to begin with.

Brooks found that no matter what programming language is chosen, a professional developer will write in average 10 lines of code (LoC) day!
Example C Program

A C program consists of functions and variables.

```
#include "derivative.h"  // Causes the file “derivative.h” to be included here

/* Square the input value */
int square(int a)
{
    int b;
    b = a*a;
    return b;
}

#include <stdio.h>  // Causes the file “stdio.h” to be included here

void main(void)
{
    int x;
    x = 2;
    PTT = square(x);  // output to Port T
}
```

This is a function. It returns a single value.

The “main()” function is required in every C program.

“void” means that it doesn’t return anything

This is a comment

You have to declare a variable before using it

This is a comment
# C Data Types for the HCS12

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
<th>Memory Size</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>char</td>
<td>character</td>
<td>1 byte</td>
<td>-128 to 127</td>
</tr>
<tr>
<td>unsigned char</td>
<td>character</td>
<td>1 byte</td>
<td>0 to 255</td>
</tr>
<tr>
<td>int, short int</td>
<td>integer</td>
<td>2 bytes</td>
<td>-32768 to 32767</td>
</tr>
<tr>
<td>unsigned int</td>
<td>integer</td>
<td>2 bytes</td>
<td>0 to 65535</td>
</tr>
<tr>
<td>long int</td>
<td>extended precision integer</td>
<td>4 bytes</td>
<td>$-2^{31}$ to $2^{31}-1$</td>
</tr>
<tr>
<td>float, double</td>
<td>floating point</td>
<td>4 bytes</td>
<td>About $\pm 10^{38}$</td>
</tr>
</tbody>
</table>

Note:
- Data types are processor-specific
- In our MCU, an “int” is 16 bits
- In a 32-bit microprocessor, an “int” is 32 bits

### Examples of declarations

```c
int i;
char cx, cy;
```

You can optionally initialize the variable when it is declared

```c
unsigned int m = 0; /* declare integer variable m and initialize it to 0 */
char echo = 'y';    /* ASCII code of letter y is assigned to echo */
```
Constants

• Numbers
  – A number can be specified in different bases

• Characters
  – A character in single quotes is represented by the ASCII code for the character
  – A string is a sequence of characters enclosed in double quotes (automatically terminated by NUL, or 0)

• Examples
  1234 No prefix means decimal
  01234 A “0” prefix means octal
  0x4F A “0x” prefix means hex

• Examples
  ‘A’ Has the value 0x41
  “ABCD” Is stored as 0x41, 0x42, 0x43, 0x44, 0x00
Arrays

• An array is a variable with multiple items

• When you declare an array you specify its size
  – Example:
    ```
    int M[5]; /* array M has 5 integer elements */
    ```

• You can refer to a particular element using an integer subscript in square brackets (note - the 1st element has index 0)
  – Example:
    ```
    M[0] = 1; // assign 1 to the first element of M
    ```

• You can initialize an array when it is defined
  – Example:
    ```
    int M[] = {1,2,3,4,5}; // no need to specify size here
    ```
Arithmetic Operators

+  add and unary plus
-  subtract and unary minus
*  multiply
/  divide      -- truncate quotient to integer when both operands are integer
%  modulus (or remainder)        -- cannot be applied to float or double
++  increment (by 1)
--  decrement (by 1)

• Examples
  - What are the values of \( i, j, k, n \) after these statements?

\[
i = 23/12;
j = 23 \% 12;
k = 5;
n = k++;\]
Short hand for operators

• You can combine the assignment operator (=) with another operator

• Examples

  \[ n += k; \quad \text{// same as } n = n + k \]
  \[ x *= 2; \quad \text{// same as } x = x * 2 \]
  \[ i -= 1; \quad \text{// same as } i = i - 1, \text{ also } i-- \]
Relational and Logical Operators

- Relational operators compare the value of two operands
  - If the result of the comparison is true, the value of the expression is 1
  - Otherwise the value of the expression is 0

- Example
  \[(i == j)\]
  This expression has the value 1 if \(i\) equals \(j\); 0 otherwise

- Relational and logical operators
  
  \n  \[==\text{ equal to}\]
  \[!=\text{ not equal to}\]
  \[>\text{ greater than}\]
  \[>=\text{ greater than or equal to}\]
  \[<\text{ less than}\]
  \[<=\text{ less than or equal to}\]
  \[\&\&\text{ and}\]
  \[\mid\mid\text{ or}\]
  \[!\text{ not (one’s complement)}\]
Control Flow

• If-Else Statement
  
  if (expression)
      statement(s)
  else
      statement(s)  -- The else part is optional

• Example
  
  if (a != 0)
      r = b;
  else {
      r = c;
      s = r + 0x0a;
  }  -- Use { } to enclose multiple statements
Control Flow

• While statement
  
  \texttt{while (expression)}  
  \texttt{statement(s)}

  – Example – sum all numbers in an array M

  \begin{verbatim}
  int i, sum;
  int M[] = {12, 34, 56, 78, 90};
  
i = 0;
  sum = 0;
  while (i<5)   {
      sum = sum + M[i];
      i = i + 1;
  }
  \end{verbatim}

  – Example

  \begin{verbatim}
  int_cnt = 5;
  while (int_cnt) ; // do nothing while int_cnt is not 0
  \end{verbatim}

This is an empty statement
Control Flow

- **For-Loop Statement**
  
  \[
  \text{for (expr1; expr2; expr3)} \\
  \quad \text{statement(s)}
  \]

  where
  
  - \( expr1, expr3 \) are assignments or function calls and \( expr2 \) is a relational expression
  
  - \( expr1 \) is performed first, then statement(s) are executed as long as \( expr2 \) is true. \( expr3 \) is performed after each iteration.

- **Example** – sum all numbers in an array \( M \)

```c
int i, sum;
int M[] = {12, 34, 56, 78, 90};

sum = 0;
for (i = 0; i < 5; i++) {
    sum += M[i];
}
```
Write C statements to do ...

- Find the largest value in an array. Array is defined as:
  ```c
  int array[10];
  ```

- Find the length of a character string, defined as:
  ```c
  char mystring[];
  ```
Write C statements to do ...

- Find the largest value in an array. Array is defined as:
  ```c
  int array[10];
  int i, largest;
  largest = array[0];
  for (i=1; i<10; i++)
    if (array[i] > largest)
      largest = array[i];
  ```

- Find the length of a character string, defined as:
  ```c
  char mystring[];
  int i=0, len=0;
  while (mystring[i] != 0)
    len++;
  ```
Example – Gray Code counter

• Problem statement:
  – Continuously count up from 0 through 7, and then wrap around back to 0
  – Display the count as a 3 bit Gray code on Port T
  – Recall a 3 bit Gray code is: 000,001,011,010,110,111,101,100

• Approach:
  – Make a table and initialize it with the Gray codes
  – Have an infinite loop that increments a counter
  – Get the value from the table corresponding to the count, and write it to PortT
  – When the counter reaches 8, reset it back to zero
Initialize Table with Gray codes (note - this can be done at load time, and the table can be stored in ROM)

Set up Port T, bits 0..2, for output
Initialize the counter n=0
Repeat forever
   Get the nth value from the table; ie Table(n)
   Write that value to PortT
   increment n
   if n==8
      n = 0
   end
   (Optional: delay a short time)
end

C

// Initialize Table with Gray codes
char Table[] = {0x00,0x01,0x03,0x02,0x06,0x07,0x05,0x04};
int n = 0;

DDRT = 0x07; //Set up Port T, bits 0..2, for output
while (1) {
   PTT = Table[n];  // Write next value to PortT
   if (++n >= 8)
      n = 0;
   // Optionally delay a short time
}
Summary / Questions

• Why is better to program in C, rather than assembly, if you can?

• When would you ever want to program in assembly?

• How do you declare a variable?