Embedded Programming in C

55:036

Embedded Systems and Systems Software

Plusses and Minuses of High-level languages for Embedded Programming

- Plusses
  - Easier syntax (usually)
  - Lots of libraries and drivers
- Minuses
  - Loss of efficiency
  - Loss of direct control at the hardware level
    - e.g. typically can't count instruction cycles
    - Some C compilers (e.g.CCS) have problems coping with the PIC's Harvard Architecture

PIC C Compilers

- Microchip C18 (MCC)
- Hi-Tech
- CCS
- Etc.

PIC C Compilers

- Microchip C18 (MCC)
- Hi-Tech
- CCS
- Etc.

This is the compiler we will use.

A student version (valid for 60 days) can be downloaded from: www.microchip.com

PIC C Compilers

- Unfortunately, the various PIC C Compilers are not compatible.
- Some differences are quite significant
  - Different naming conventions
  - Different constructs for bit-level access to ports and other registers
  - Different directives and built-in functions
  - Different constructs for interrupts and ISRs
  - etc.
- Porting C code from one compiler to another can be a non-trivial task

C18 Compiler Documentation

- C18 C Compiler: Getting Started
  - Pay Special Attention to Chapters 3 and 4.
- C18 C Compiler User's Guide
- C18 C Compiler Libraries
- Documentation is linked on the Class Web Site.

C18 Integer Data Types

TYPE-SPECIFIER	SIZE
char	8-bit (signed by default)
signed char	8-bit (signed)
unsigned char	8-bit (unsigned)
int	16 bit (signed)
unsigned int	16-bit (unsigned)
short	Same as int
unsigned short	Same as unsigned int
short long	24-bit (signed)
unsigned short long	24-bit (unsigned)
long	32-bit (signed)
unsigned long	32-bit (unsigned)

Note: Multiple-byte data is stored in "little endian" form

C18 Floating Point Data Types

TYPE-SPECIFIER	SIZE
float	32-bit (IEEE format)
double	Same as float

C18 Data Types—Constant Strings

- Constant strings can be stored in program memory:

  const rom char str[] = {'H', 'i', '\0'};
- All strings declared without the keyword *rom* are stored in data memory:

  const char str[] = {'H', 'i', '\0'};
- Pointers to program memory (rom) strings and pointers to data memory strings are not compatible since they point to different address spaces.

Note: Different versions of the strcpy function

```
/*
* Copy string s2 in data memory to string s1 in data memory
*/
char *strcpy (auto char *s1, auto const char *s2);
/*
* Copy string s2 in program memory to string s1 in data
* memory
*/
char *strcpypgm2ram (auto char *s1, auto const rom char *s2);
/*
```

C18 Example: LCD_Routines.h

```
#ifndef __LCDROUTINES_H
#define __LCDROUTINES_H

#define CMD  0
#define CH   1

/************************************************************************
*  Function Prototypes for LCD_Routines
************************************************************************/
void lcd_write_nibble(char);
void lcd_write_byte( char, char);
void lcd_init(void);
void lcd_gotoxy( char, char);
void lcd_putc( char c);
void lcd_ram_puts(const char *str, char len);
void lcd_rom_puts(const char rom *str, char len);

#endif
```

C18 Example: LCD_Routines.c

```
////////////////////////////////////////////////////////////////////////////
///                      LCD_Routines.c
///
///  lcd_init()   Must be called before any other function.
///
///  lcd_write_nibble(c)  write high-order 4 bits of c to LCD
///
///  lcd_write_byte(mode, c) write byte to LCD. Mode = CMD/CH
///
///  lcd_putc(c)  Will display c at the next position of the LCD.
///                    The following have special meaning:
///                      \f  Clear display
///                      \n  Go to start of second line
///                      \b  Move back one position
///
///  lcd_gotoxy(x,y)  Set write position on LCD (upper left is 0,0)
///
///  lcd_ram_puts(str,len) write data memory string str to LCD
///
///  lcd_rom_  puts(str, len) write program memory string str to LCD
///
////////////////////////////////////////////////////////////////////////////
```