MSP430 microcontroller basics: Information for code files

John H. Davies

June 25, 2008

Code files

The archive contains the code files used for all the example programs in the book. This includes the C language (`.c`), assembly language (`.s43`) and a few header (`.h`) files. It does *not* include the project or workspace files nor the standard header files provided by the development environment.

The files are organized into directories by chapters of the book. Most programs require only a single file of source code but many of those that use the LCD on the TI MSP430FG4618/F2013 Experimenter's Board also need `LCDutils.c`, `LCDutils.h` and `uinttobcds.s43`. There are copies of these in the directory for each chapter where they are used. I added extra functions to the files in later chapters and these are named `LCDutils2.c`, `LCDutils2.h` and `uinttobcds2.s43` to keep them distinct.

The files were read into the main text when the book was produced using LaTeX. Their contents should therefore match the book exactly, unless something terrible has gone wrong! The rest of this note describes some problems that might arise when you use the files on your computer.

IAR Embedded Workbench

The programs were all developed with IAR Embedded Workbench for TI MSP430 (EW430), using versions up to 4.10. There will almost certainly be a newer version by the time that you read this but this should not affect the code files. (On the other hand, the format of project and workspace often changes when a major new version of EW430 is released.) The only problems that I have found arise from minor errors in the header files for particular devices, and many of these can be traced back to data sheets. The bugs may well have been corrected by the time that you read this.

DMA controller in the FG461x – The definition of `DMA0SZ` is missing from `io430xG46x.h` so I added one to listing 9.13.

USCI_B module in the FG461x – Both UCB0I2COA and UCB0I2CSA are listed incorrectly under *Peripherals with byte access* in the data sheet for the FG461x; they should have word access. This error is repeated in `io430xG46x.h`, where these registers are defined as bytes (8 bits) rather than words (16 bits). I edited `io430xG46x.h` to change `char` to `short` in both definitions.

USI module in the F20x3 – The bit to prevent the automatic clearing of USIIFG is called USIIFGCC in the user's guide but USIFGCC (missing an 'I') in the EW430 header file.

C language with other development environments

It's always hard to predict the problems that might arise if you use a different development environment but here are some possible problems. The User's Guide for Code Composer Essentials (CCE) contains a section on migrating from EW430 to CCE. A major new version of CCE will be released about the same time as this book so I can't be more specific.

- I have used bit fields extensively because I think that they are clearer than masks and less prone to errors when writing. For example, this permits `TACTL_bit.TAIFG = 0` instead of `TACTL &= ~TAIFG`. However, bit fields will work only if they are defined in the header file. You will have to replace the fields with masks if fields are not available.
- I have often specified the size of integers with keywords like `uint8_t`. These are defined in `stdint.h`, which is not always available. You will have to used `unsigned char` and similar definitions instead.
- Several intrinsic functions, such as `__enable_interrupt()`, are needed in most programs. They will almost certainly be available but the names might be different.
- Interrupt service routes require extensions to standard C and their definition may therefore vary between development environments.
- You must read the manuals carefully if you wish to mix C and assembly languages, as I did for the conversion between binary values and BCD. The details rely critically

on the convention used to call a function – which registers are used to pass arguments and return values, which must be preserved, and so on.

Assembly language with other development environments

Programs in assembly language will require substantial rewriting in many other development environments. Some do not permit absolute assembly language at all and the names of the segments for relocatable assembly language are often quite different. For example, code goes into the `CODE` segment in EW430 but `.text` in CCE.

Disclaimer

The use of these files is governed by the disclaimer printed in the book.