Appendix

A5

QWIKADDRESS AND QWIKPH UTILITIES

A5.1 OVERVIEW

This appendix describes two utilities of value in debugging code. Both are available from the author's Web site, www.picbook.com. The QwikAddress utility was developed by David Flowers of Georgia Tech to deal with several issues that arise as a result of using the structured assembly preprocessor of Chapter 6. The error file generated by the assembler includes a line number for each error that points back into the .apr file generated by the preprocessor, whereas what is needed is the corresponding line number in the original .asm source file. By dragging the .err file from the folder containing the source file (e.g., C:\Work), into the QwikAddress window, each of the error line numbers is translated back to its source file line number.

Subsequently, when debugging user code with the QwikBug monitor program described in the previous appendix, this QwikAddress utility serves two further roles. When setting a watch variable, the RAM or register address of the variable is needed. When setting a breakpoint, its program memory address is needed. This capability is also of value when using Microchip Technology's ICD2 in-circuit debugger and the MPLAB environment with code generated using structured assembly constructs. In this case, QwikAddress reaches a breakpoint address via the selection of a desired subroutine in one table followed by the switching to a listing of that subroutine in the source file. When the desired line in the subroutine is selected, its address is displayed.

The QwikPH utility was developed by Chris Twigg of Georgia Tech to generate and insert a Program Hierarchy into a source file. The Program Hierarchy lists the relationships between all of the subroutines in the source file. An accurate Program Hierarchy for a large program is difficult to generate manually with complete accuracy. It is helpful in several ways, including the tracking down of the corruption of a variable being used by several subroutines.

A5.2 DAVID FLOWERS' QWIKADDRESS UTILITY

This utility is normally opened at the same time that a folder is opened showing the source and listing files being debugged. Figure A5-1 shows the files generated by the assembler for the template program, P3.asm. Clicking on this and dragging it into the **QwikAddress** window triggers QwikAddress to act upon this file. A watch variable address can be found by selecting **Variables** in the **View** pulldown menu, as shown in Figure A5-2. It may be easier to find the address of a variable using the alphabetized list obtained by selecting **Sort Lists** in the **Sort** pulldown menu, as shown in Figure A5-3. The address of one of the Special Function Registers is found as shown in Figure A5-4.

Obtaining a breakpoint address begins by selecting **Labels** in the **View** pulldown menu, as shown in Figure A5-5. For a specific address within the **Pbutton** subroutine, for example, the **Pbutton** subroutine address is double-clicked in this window, as shown in Figure A5-6. This opens the source file shown in

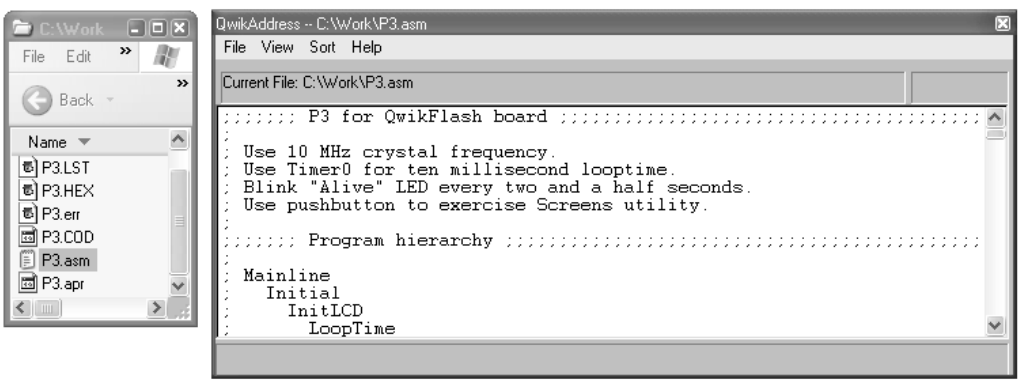

Figure A5-1 Selecting a source file to be operated upon by QwikAddress

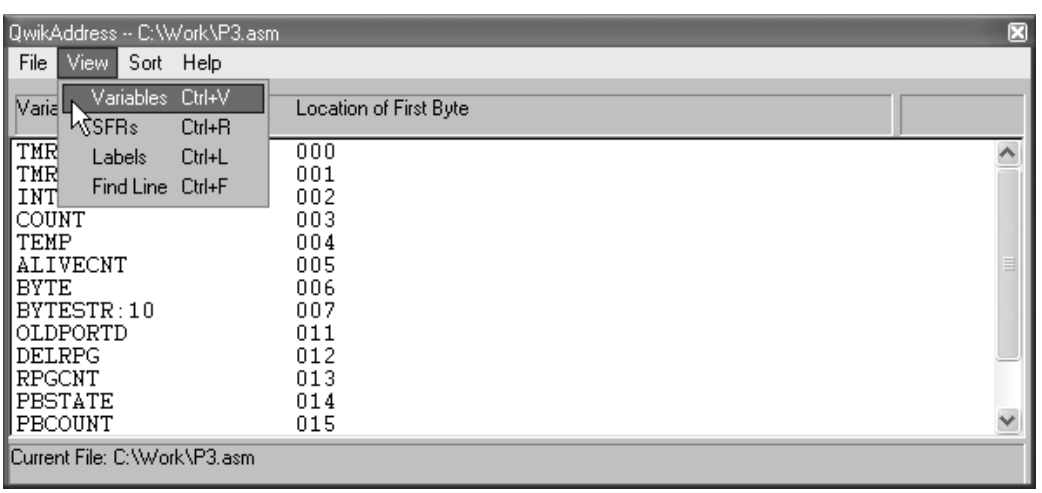

Figure A5-2 The display of all user-defined variables and their addresses

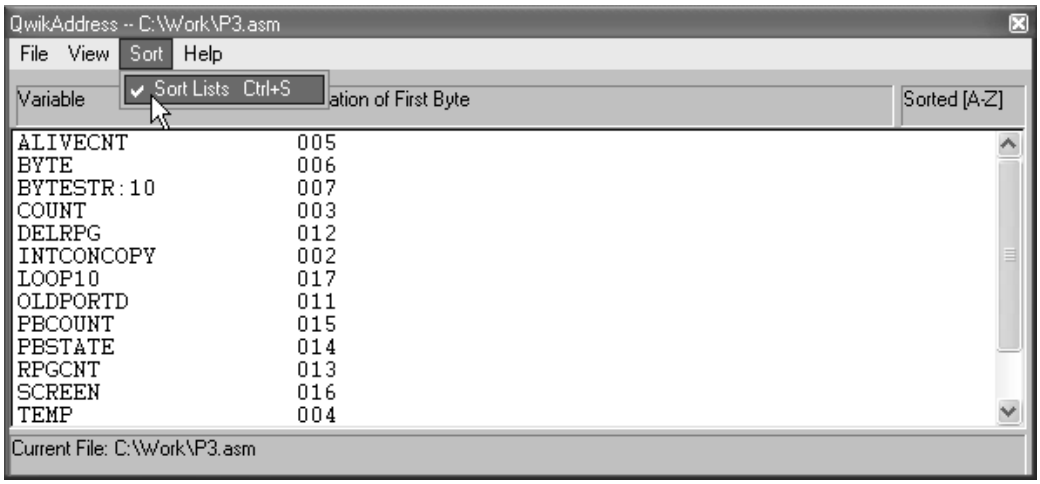

Figure A5-3 Alphabetizing the user-defined variables

Figure A5-4 The display of the Special Function Registers

Figure A5-7, scrolled down to the **Pbutton** subroutine. Further scrolling into this subroutine is shown in Figure A5-8. Double-clicking on the line shown leads to the display of the address of the first instruction generated by the macro, **MOVLF**.

A5.3 CHRIS TWIGG'S QWIKPH UTILITY

This utility opens to a tree of file folders that is navigated to the folder holding the source (.asm) file of interest. Thus, in Figure A5-9, the Work folder with the pathlist C:\Work has been opened. It is shown holding two source files, P3.asm and P4.asm. Before selecting one of these to be operated upon, the

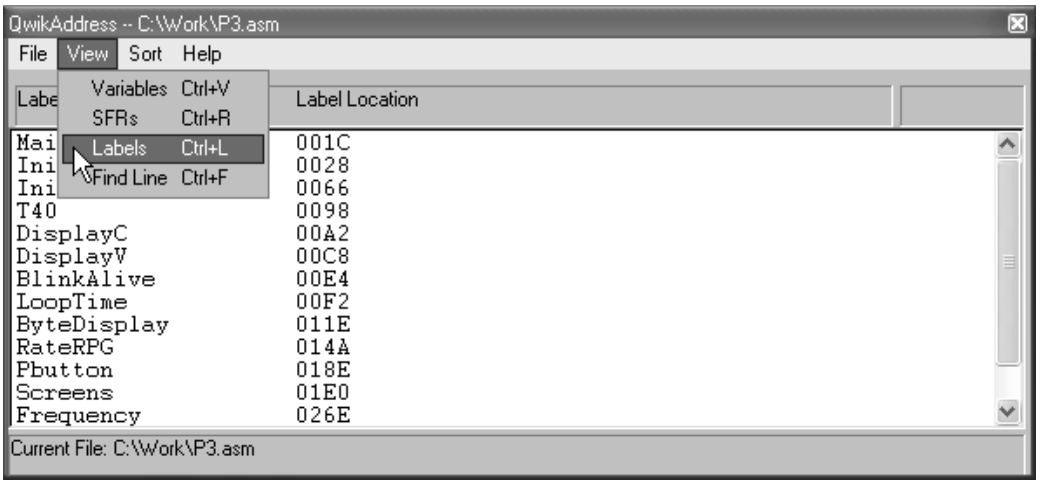

Figure A5-5 Addresses of all subroutines

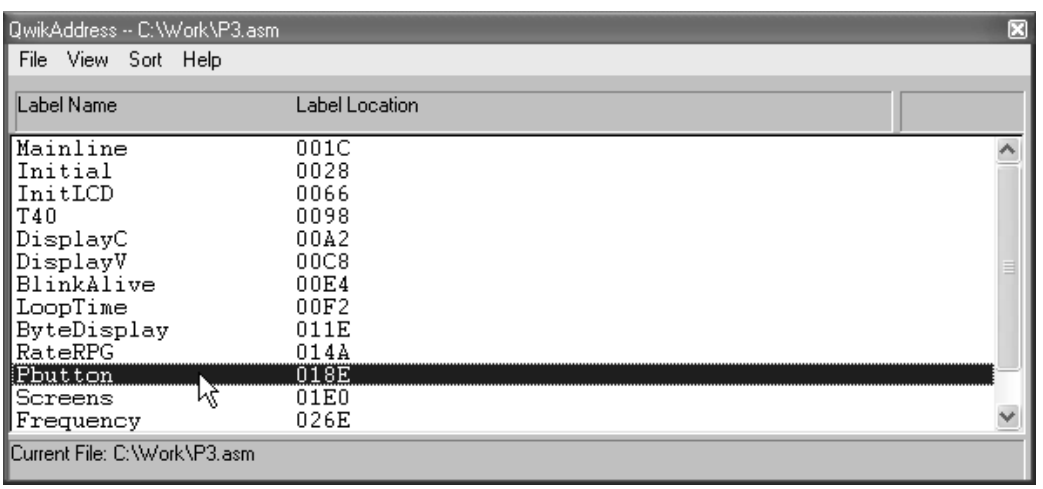

Figure A5-6 Selection of a specific subroutine

Config pulldown menu and the **Special Labels** window can be opened, as shown in Figure A5-10 and A5-11. This window allows a user to name the mainline program and the interrupt service routine(s) with arbitrary names such that the QwikPH utility will handle them appropriately.

The **Config** pulldown menu also permits the choice of having the QwikPH utility write the Program Hierarchy to the clipboard for manual inspection and manual insertion into the source file. Alternatively, by checking the AutoUpdate ASM entry, the Program Hierarchy already in the file will be updated with the new results. As a backup, the original file is first renamed with a numerical suffix and then the utility generates the new .asm file. When using the AutoUpdate ASM feature, the utility looks for "Program Hierarchy" in a comment line. After finding this comment line, QwikPH replaces all lines between

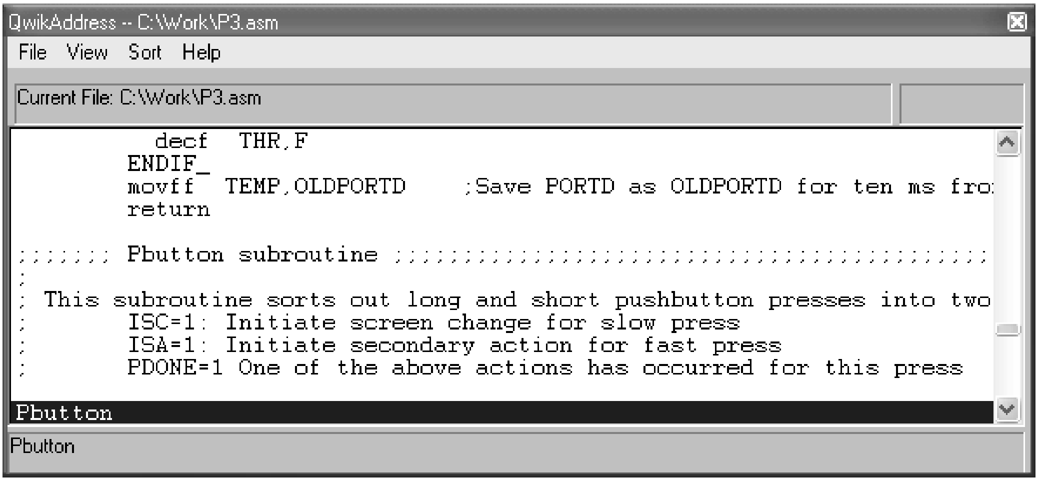

Figure A5-7 Source file display of the selected subroutine

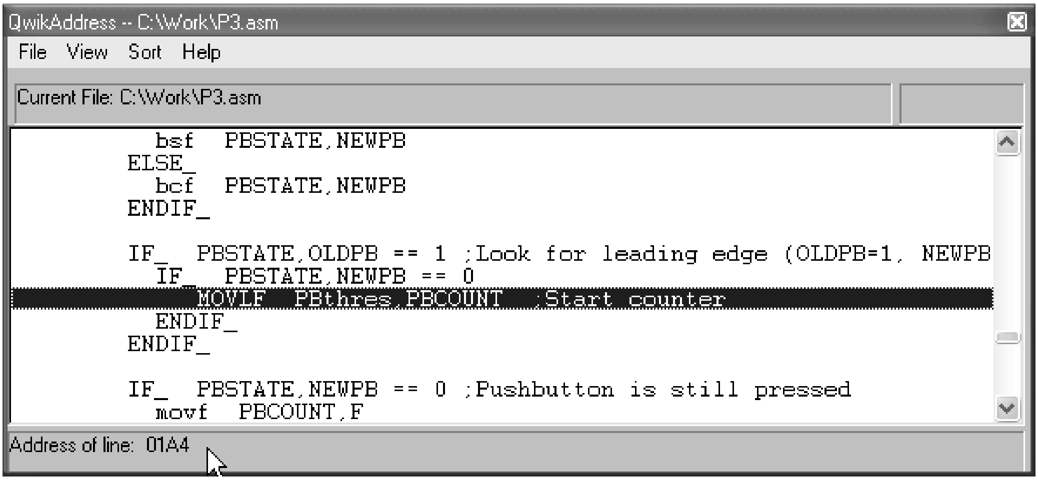

Figure A5-8 Obtaining the address of a specific instruction within the selected subroutine

it and the next comment line or blank line with the newly generated program hierarchy. If QwikPH fails to find this comment line, it will copy the generated hierarchy to the clipboard.

Double-clicking the selected file in the menu initiates the utility's activity. Upon completion, the action taken is written in the status bar of the QwikPH utility, as shown in Figure A5-12. The resulting section of the source file is shown in Figure A5-13.

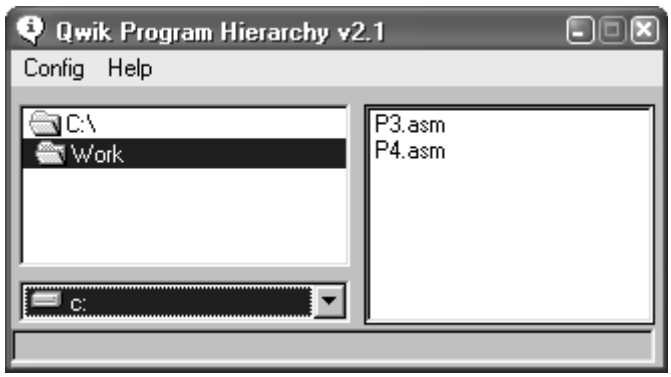

Figure A5-9 Selection of source file for Program Hierarchy generation

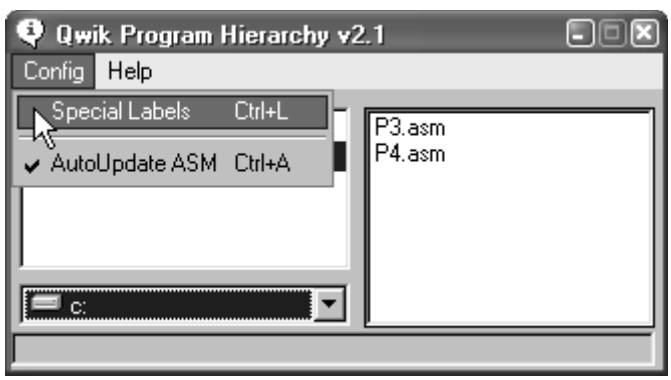

Figure A5-10 Configuration of QwikPH to identify vector labels

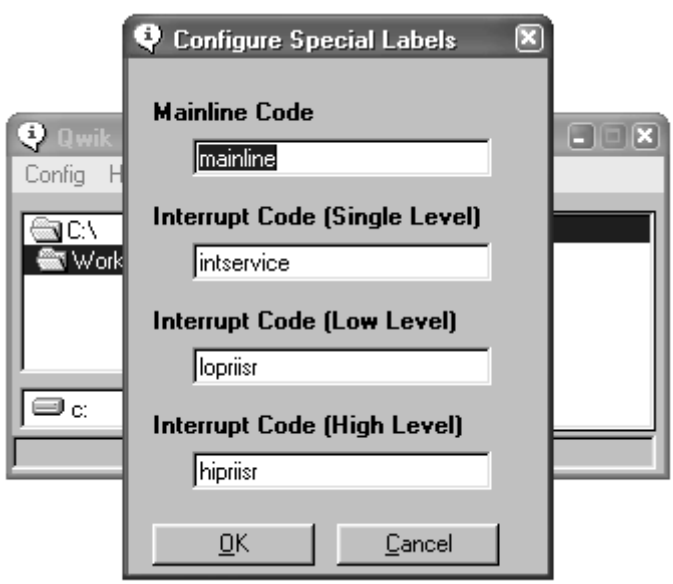

Figure A5-11 Entry of vector label names

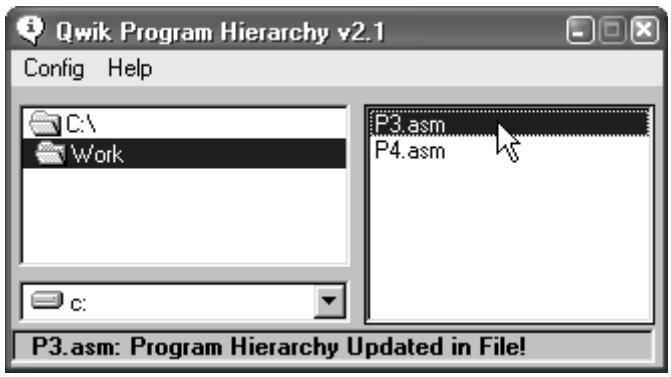

Figure A5-12 Initiating Program Hierarchy generation

```
;;;;;;; P3 for QwikFlashboard ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; Use 1Ø MHz crystal frequency.
; Use TimerØ for ten millisecond looptime.
; Blink "Alive" LED every two and a half seconds.
; Use pushbutton to exercise Screens utility.
;
;;;;;;; Program hierarchy ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; Mainline
;   Initial
;     InitLCD
;       LoopTime
;     DisplayC
;       T4Ø
;   BlinkAlive
;   Pbutton
;   Screens
;     DisplayC
;       T4Ø
;     Frequency
;     Period
;     PWmax
;     RateRPG
;     ByteDisplay
;       DisplayC
;         T4Ø
;       DisplayV
;         T4Ø
;   LoopTime
;
;;;;;;; Assembler directives ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

Figure A5-13 Automatic updating of P3.asm's Program Hierarchy