WISP Tutorial: IAR Embedded Workbench IDE

Hee-Jin Chae
UMass Amherst Computer Science
http://www.rfid-cusp.org/

June 2007 Draft

1 Install the Software

IAR Embedded Workbench IDE can be downloaded from the TI website

http://focus.ti.com/docs/toolsw/folders/print/iar-kickstart.html.

It can be also installed from the MSP430 CD-ROM which comes with the USB programmer (Version 3, Rev. K – Released 01 May 2006), however, it is recommended that the latest version (Version 3, Rev. O – as of June 2007) is installed. This supports up to 4K in size for programs.

Both links on the website (one under "Description" and one under "Support Software") downloads the same version of IAR Embedded Workbench IDE. Once downloaded, execute FET_R4xx.exe and follow the installation procedure.

2 Install the Hardware

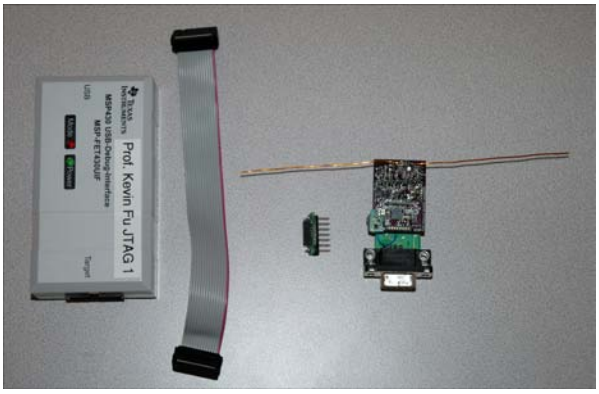
When connecting the Emulation Interface to a PC for the first time, the Hardware Wizard starts automatically and opens the "Found New Hardware Wizard" window. The default location of the driver files is:

C:\Program Files\IAR Systems\Embedded Workbench x.x\430\drivers\TIUSBFET\WinXP

Refer to FET User's Guide (appendix E) for detailed instructions how to install the USB drivers (This can be downloaded from the same website).

3 Getting started & Tips

To familiarize yourself with IAR Embedded Workbench IDE, the User Guide gives a good tutorial on the IDE (Help→MSP430 IAR Embedded Workbench IDE User Guide). This section aims to point out few key points and provides few tips.



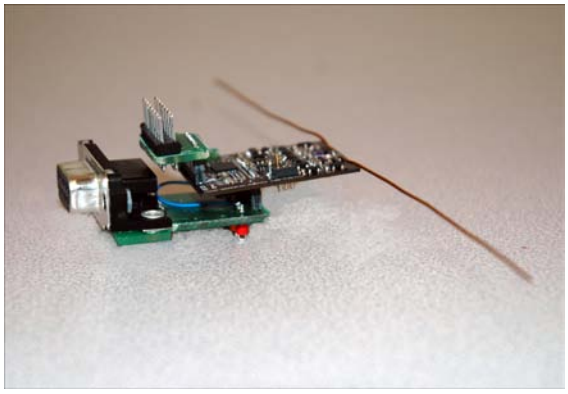


Figure 1: The left picture shows each components: USB FET, JTAG cable, adapter board, and WISP (from the left). The right picture shows adapter board connected to WISP

Tab	Setting
Target	Device: MSP430F1232
Output	Output file: Executable
Library Configuration	Library: CLIB

Table 1: Project option settings

Connecting WISP to USB FET WISP needs a special adapter board to connect to the USB FET. Pin 1 on the JTAG cable (red stripe) goes on the side of the adapter board (green board) that says "1". Connect the adapter board to the WISP by aligning "1" with the "WISP" lettering on the WISP board. Basically, the adapter board should extend away from the WISP (not over the WISP), and the red stripe on the JTAG cable should be on the same side as the "WISP" logo (Refer to Figure 3).

Setting Project Options When setting up a new project, make sure to set the general options to match the processor as in Table 3.

Also, there are two different settings for Debugger (Project→Options →Category: Debugger) When debugging with the hardware, set Driver under Debugger:Setup tab to be "FET Debugger" (Refer to Figure 3 dialog box on the left) and Connection under FET Debugger:Setup tab to be "Texas Instrument USB-IF"(Figure 3 dialog box on the right).

Downloading application to MSP430 Programs are loaded to MSP430 through debugging mode. There is no separate procedure to "just load" program into MSP430. When downloading an application, you can set the options to retain or erase the memory content (Figure 3) – Project→Options→Debugger:FET Debugger→ Download tab. The address range of information memory is from 0x1000 to 0x10FF (256 bytes). This is what we refer to when we say flash memory. WISP's MSP430 has 8KB of main memory (0xE000-0xFFDF).

Changing Stack Size The default for stack and heap size is 80 bytes each, and it is possible to change these two values for different projects (Refer to Figure 3).

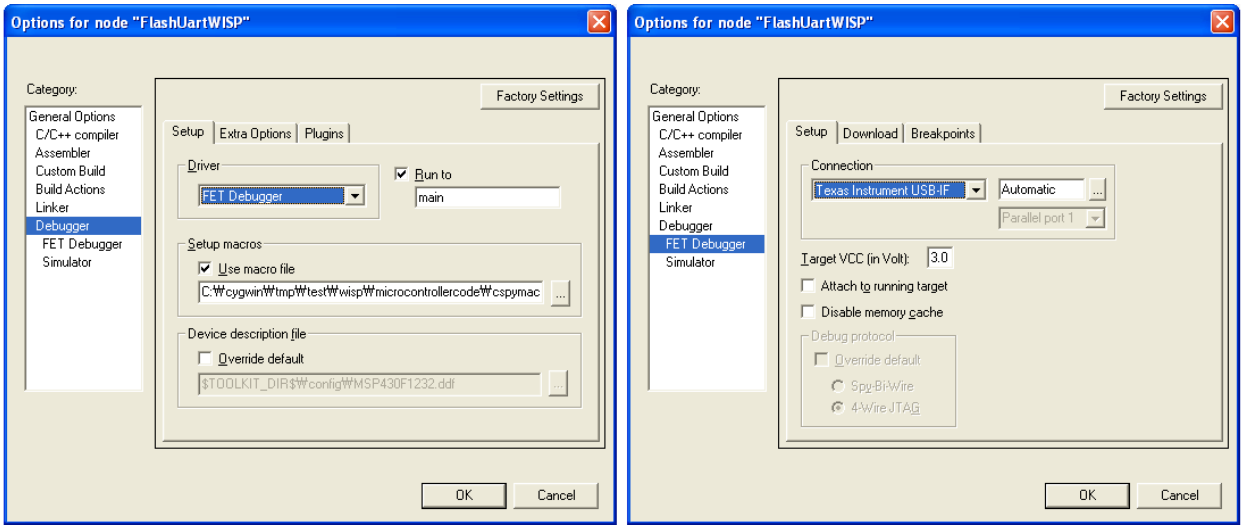


Figure 2: Debugger options

Because of the extremely limited RAM in MSP430(256 bytes), it is important not to overestimate stack and heap size. Moreover, RAM is also used by program variables (Segment `DATA16_I, DATA16_Z, DATA16_N`). Heap is used by `malloc` and `free`. You can output compiler list file which shows useful information (Refer to IAR Embedded Workbench User Guide, page 32 for detailed list of information provided by the list file.) The end of the list file shows the amount of stack, code, and data memory required. Although the amount of stack it displays does not always reflect the actual stack required during the execution, data memory information can be useful in determining how much of RAM can be dedicated for stack and heap.

To utilize this file, check off Project → C/C++ compiler → List(tab) → "Output list file" box. `.lst` file will be created under output folder.

4 Introduction to WISP Programming

This section provides introduction to WISP programming by going through basic flow of the WISP program implemented by Intel(FlashUartWISP). The program is a mix of assembly and C++ code. When transmitting and receiving, assembly code is used to fine tune the timing which may be confusing at first glance. For most of the time, you should not be needed to modify or understand this part completely to run the program.

Figure 4 shows WISP lifecycle. When the device starts up (either from sleep → Port 2 Interrupt or bootup → `main()` (hardware reset)), it is going to check to see if it has enough voltage. If it does not, it will go to LPM(low power mode) 4 and repeat this process – Power Save Mode (LP4). Port 2 Interrupt and `main()` also prepares 64-bit ID and 16-bit CRC to be sent to the reader. This is done by calling `preload_rfid_flashdata()` or `preload_adcdata()` function for flash demo or sensor demo respectively from `wisp_specific_cmds.h` header file – Generate Packet. Otherwise, it will try to receive bits from the reader.

Port 1 Interrupt receives bits. When it is finished, the Timer interrupt fires and the received data is evaluated. You can see the section where the "cmd" character array is checked against

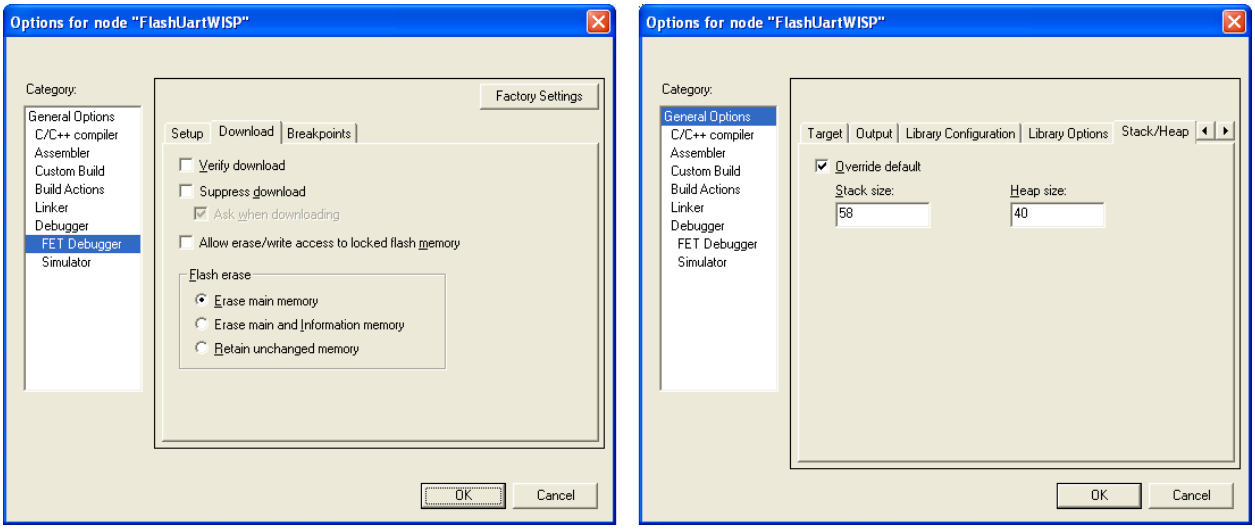


Figure 3: Project options for application download on the left, Stack/heap size change dialog on the right

	WISP	RFID Reader
Bits/Second	2400	115200
Data Bits	8	8
Parity	None	None
Stop Bits	1	1
Flow Control	None	Hardware

Table 2: RS232 settings for WISP and Alien RFID reader

known reader commands. If the command is known, WISP sends bits back to the reader using `sendTagResponse()`, which sends the information in the `reply[]` character array preloaded during Port 2 Interrupt or `main()` – Receive and Transmit.

4.1 Tag-to-Reader Communication

WISP currently runs EPC Gen1 protocol and works with Alien RFID reader that Intel has provided. The RFID reader can communicate with a computer via a standard RS232 serial port, and WISP can also communicate via RS232 if it's equipped with RS232 board. The settings for both communications are defined in Table 4.1.

Note that the scroll lock button on a keyboard may interfere with HyperTerminal when it is enabled.

Using onewisp.exe demo program (Visual Basic), click on the FLASH/UART tab and click the "Enable Demo" radio button. Then, click "Read Data From Flash" to retrieve the data stored in memory. This will send command `get taglist 200` to the reader which will poll the reading range for any valid tags. With `get taglist` command, the reader retrieves the stored tag ID list once. Using the command with an optional integer "n" instructs the command to be repeated "n" times and returns a combined list of tag IDs retrieved from "n" inquiries. For more detailed information on how the reader interacts with the tag, please refer to the ALR-9780,9640 Reader Interface Guide.

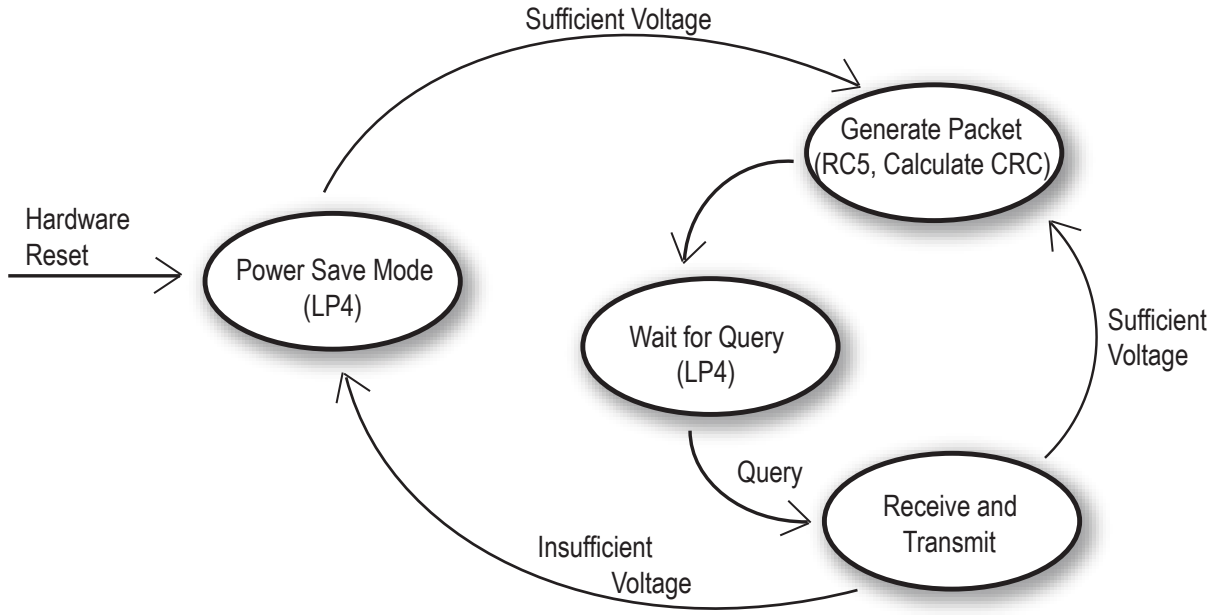


Figure 4: WISP lifecycle

To write data, either input 8 bytes in hex into the "Hex" box, or enter data in ASCII or decimal format and click the appropriate "Load Format" button. This converts the data to hex format. Then, click the "Write Data to Flash" button. The writing process will take 20 to 30 seconds, and completion is notified by a message box. Note that the tag `FF 00 01 02 03 04 05 06` is a generic response to an RFID write command.

You can also use a script or HyperTerminal to read or write. The following is the set of commands to read. It takes a little while for the WISP capacitors to charge, so try `get taglist 200` command couple times before troubleshooting.

```
 set acquiremode = global scroll
set mask = 8,1,0
get taglist 200
get taglist 200
```

Before doing any writes via a scrip or HyperTerminal, first perform a read to charge up the WISP.

```
 set mask = 8,address,data
get taglist 200
```

The address is the byte number times 4, and bytes 0 through 7 are the available choices. For example, "28" for byte 7. The data is a byte represented in hex whose endian has been reversed.

This is necessary because the RFID reader reverses the endian when it transmits. When data is read from WISP, the original, un-reversed endian is reported. For example, "B2" for ASCII character "M".

When the demo program starts up, it constantly sends commands to the reader to constantly poll for valid Tag ID. This command is for getting sensor data (i.e, getting ID of WISP running Sensor WISP application). To enable constant polling of tag ID, check of "Run" first to stop the reader. Click "G Scroll" radio button to set acquire mode to `global scroll`. Then check "Run" again to start polling for tag ID.