Freescale Semiconductor
Application Note

Document Number: AN3530
Rev. 0, 05/2008

Building a USB/IIC/SPI Bridge with ColdFire OTG Module

by: Paolo Alcántara
RTAC Americas

1 Introduction

This document is a reference to get the universal serial bus (USB) module of the Freescale MCF522xx ColdFire microcontroller unit (MCU) to work as a bridge between a PC with Windows OS and two EEPROM memories using SPI or IIC protocol. A basic knowledge about USB, IIC, SPI protocol, Visual Studio© and C-language is assumed. For more information about USB see AN3492.

Contents

© Freescale Semiconductor, Inc., 2008. All rights reserved.

freescale™ semiconductor

2 Getting Started with the ColdFire Example Software

2.1 USB/IIC/SPI Bridge's Features

Figure 1 shows the graphical user interface (GUI) included with this application note. The ColdFire example software performs the following actions:

1. Interchanges data from the M52221DEMO (USB device) board to a PC (USB host) through a USB connection.
2. Read and write 8-bit IIC and SPI memories as individual bytes (Figure 1) or entire memory device content (Figure 2).

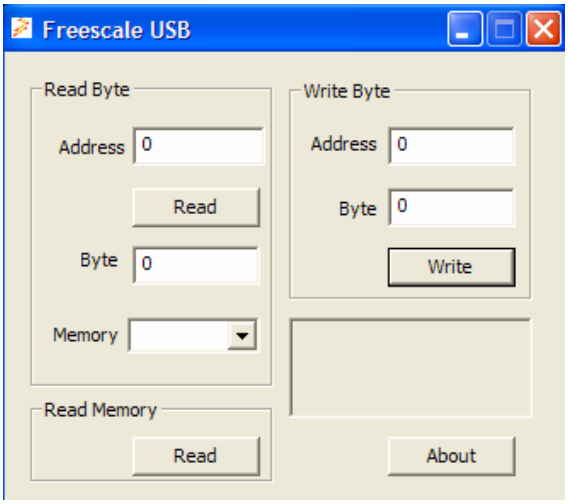

Figure 1. GUI Main Window

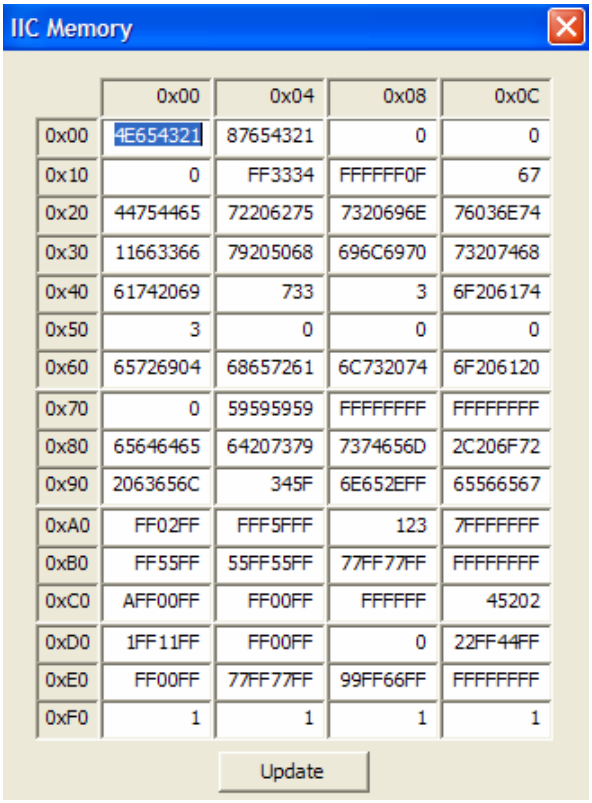

Figure 2. Read Memory

3. Read 256-bytes memories.
4. PC USB driver installation is required one time only.
5. USB connection failures are notified by error alerts.

2.2 Basic Steps to Test the ColdFire Example Software

1. Download and install the CMX stack from www.freescale.com and accept the license of use.
2. Download AN3530SW.zip and unzip it inside the CMXUSB_LITE folder.
3. The following files are included in the Starting with USB folder:
   - USB-IIC-SPI.exe: application running on the PC.
   - Bulkusb.inf: Installation file requested by the PC when the ColdFire board is connected.
   - Bulkusb.sys: Driver file requested when the ColdFire board is connected to the PC, must be in the same folder as bulkusb.inf.
   - Usb_app.elf.s19: flash object image for the MCF52221.
4. Flash usb_app.elf.s19 file into the M52221DEMO board.
5. Reset board and connect the M52221DEMO board to a PC with Windows© OS through a USB cable (host).
6. The message in Figure 3 appears. Select the “Install from a list or specific location” option.

Figure 3. Pop-up Window after Connecting the M52221DEMO Board

NOTE

For a further explanation about this board and how to use it, see the M52221DEMO_UG document at www.freescale.com.

For more details about how to target code to ColdFire boards, look at the *Targeting Coldfire* document inside the installation.

7. Select the "Include this location in the search" option. Browse the folder where the .inf and .sys files are located, as shown in Figure 4. Click next.

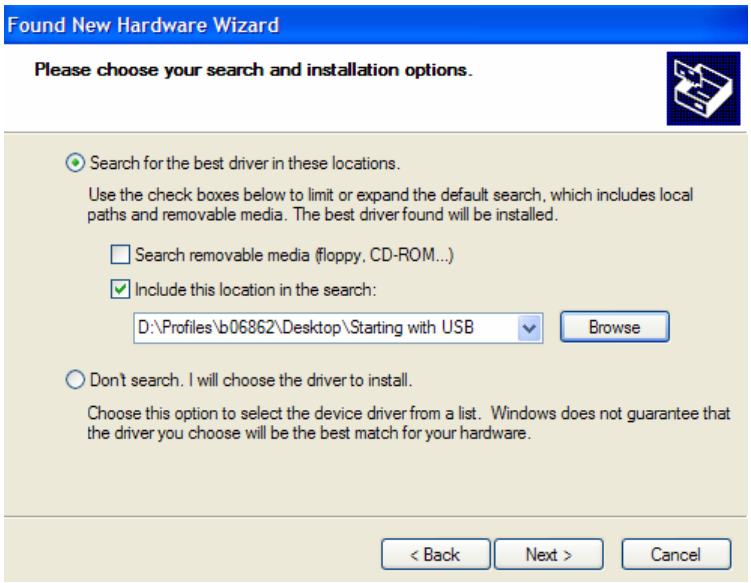

Figure 4. Browsing the .inf and .sys Files

8. A message informing that the installation is finished appears (Figure 5).

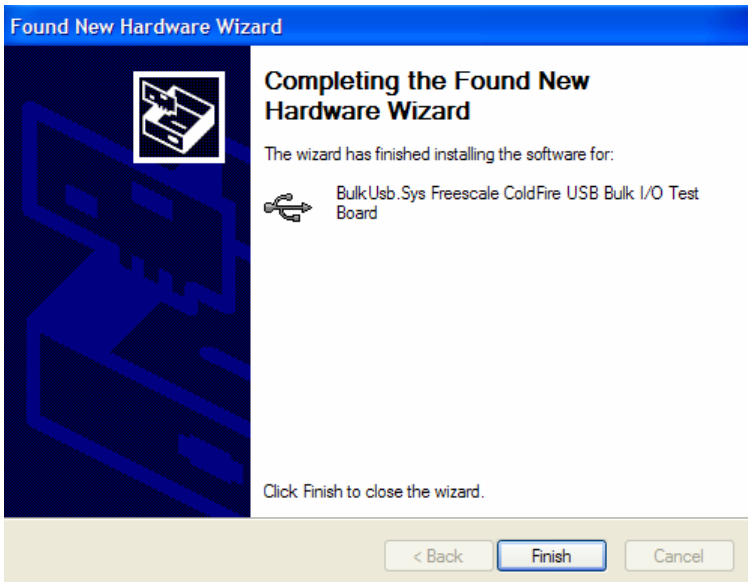

Figure 5. USB Device Installation Completed

9. The USB-IIC-SPI.exe executable file is ready to read or write into the IIC and SPI memories.

NOTE

Figure 11 shows the electrical connections between the memories and the ColdFire board.

3 ColdFire USB Transactions Explanation

3.1 USB Customized Data Packets between PC and ColdFire Board

Figure 6 explains how data is exchanged between the PC and the ColdFire board. According to the USB specifications, all transfers are started from the host, so the ColdFire board is always listening from endpoints 2 and 4. After receiving a specific code, the host initiates the writing/reading instructions to the ColdFire.

For example, to read the entire SPI memory, the host sends two longwords through endpoint 4. The first longword is the READ_MEMORY command (0x500) code and the second longword is the memory command code, SPI_CODE (0x55) in this case. After sending the instructions, the host is set in listening mode and expects 256 bytes from endpoint 3. The ColdFire device sends 256 bytes containing the data read from the SPI memory. This is the case for the other transactions, except form endpoint 2 that doesn't return any data.

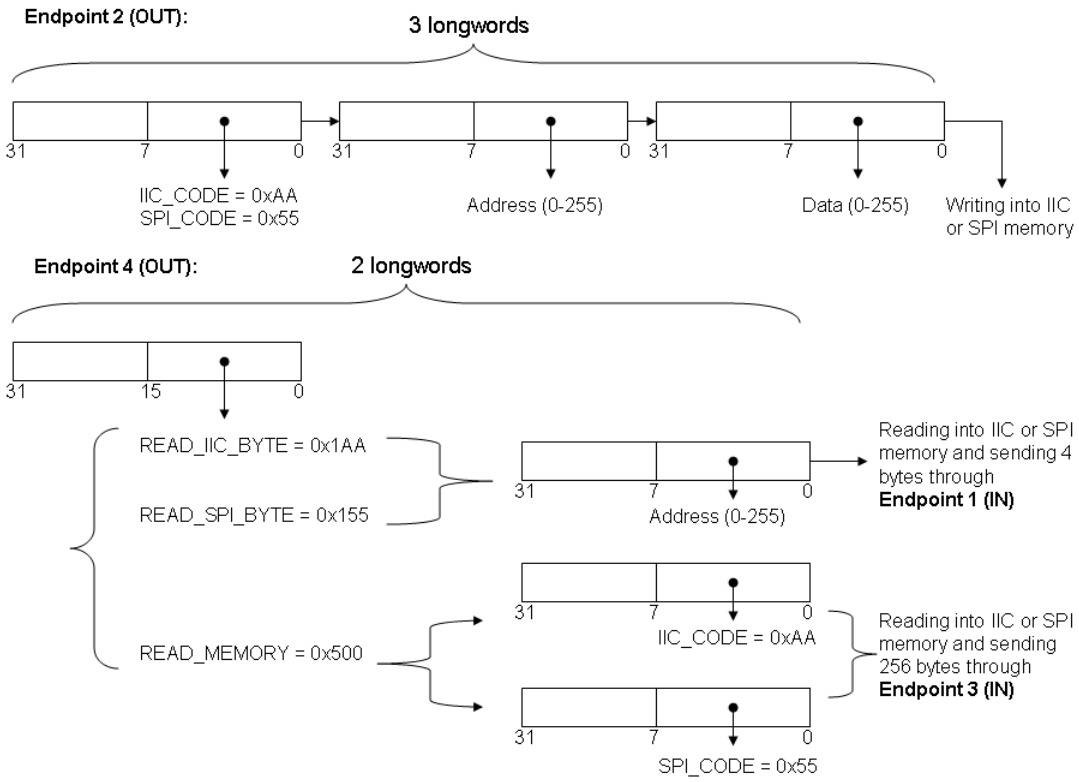

Figure 6. USB Protocol

3.2 Testing Codes Between the Host and the ColdFire Board

The requirements are:

- A PC with Windows XP Professional service pack 2 or Windows XP Professional without service pack.
- A USB protocol analyzer. This ensures the packets arrive in a reliable way.
- A M52221DEMO board rev C.
- Serial EEPROM AT24C02B (IIC protocol) and an AT25020A (SPI protocol). Figure 10 shows the schematic.

4 Using the CMX USB Stack with ColdFire MCF5222x

AN3492 explains the USB and USB CMX stack in detail and can be found in www.freescale.com.

4.1 ColdFire Demo Software Features

The software associated to this application note has the following characteristics:

- Project tested in Codewarrior 6.4v and 7.0v.
- Uses a customized USB class to communicate between the PC and the ColdFire board.
- Some customized packets are sent from the PC to the ColdFire to request an action.
- Uses one control endpoint (default endpoint 0)
- Has four bulk endpoints (2 IN and 2 OUT)
- Always listening to two endpoints. After receiving a message, it is decoded and validated, and then proceeds to send or receive information according to the message receive.
- It doesn't receive power from USB host, but takes it from an external power supply or the USB debugger cable (default configuration) included on the M52221DEMO board.
- The ColdFire board has connections to two kind of memories: IIC memory (part number: 2402) and SPI memory (part number: 2502).
- Two pull-up resistors were included for serial data (SDA) IIC memory and serial output (SO) SPI memory because internal pull-ups in MCF5222x are too weak and these pins have an output collector configuration.
- A delay of 5 ms was included after each IIC memory write because of memory specs.

4.2 First Approach to the ColdFire Board

Figure 7 shows the tree code for MCF5222x with Codewarrior. Table 1 describes some of the files in the project.

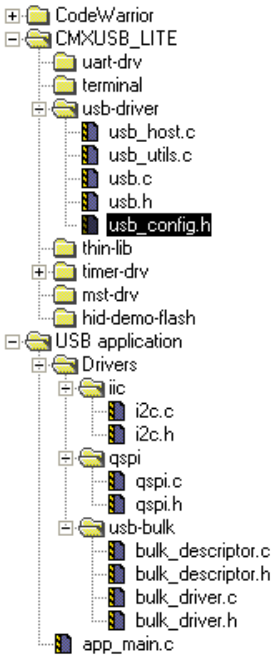

Figure 7. Project in CodeWarrior

Table 1. Files Description

File	Purpose
app_main.c	Makes the USB/IIC/SPI bridge possible.
bulk_driver.c	Contains the USB bulk requests for IN/OUT transactions.
bulk_descriptor.c	USB device descriptor
qspi.c	SPI memory driver
iic.c	IIC memory driver
usb_config.h	CMX USB stack to point to demo software
mcf5222x_vectors.s	Vectors table directs interrupts to the proper exception routine.

4.3 Adding Interrupts in the CodeWarrior Project

USB/IIC/SPI bridge uses interrupts for USB (level 3), SPI (level 2), and IIC (level 1) giving a higher level to USB and going down to SPI and IIC modules.

The USB handler has the highest priority because it needs to drive a fast response from the host to avoid missing messages. Each ISR is mapped in MCF5222x_vectors.s file and defined on its respective driver file as shown in Figure 8.

Figure 8. Steps to add an Interrupt in CodeWarrior

4.4 Changing the USB Device Descriptor

Bulk_descriptor.c contains the device information that the USB host needs. To learn more about device descriptors, see AN3492.

To modify the project for a different application, the following must be considered:

- Customized USB class: this driver can define customized control packets, number of endpoints, and other behaviors depending the programmer needs.
- USB vendor and product ID: these parameters must be equal to the Windows INF file. The operating system (OS) associates the device descriptor to a specific PC-USB driver.
- Size of device descriptor: must be changed if some parameters are added or removed from device descriptor.
- Number, type, size, and direction of endpoints depend on the application.
  - Number defines the amount of pipes created.
  - Type selects from the 4 kinds of endpoints: control, bulk, isochronous and interrupt.
  - Size is the maximum packet length driven by the device.
  - Direction defines if packets come from or go to USB host.

4.5 Device Descriptor

Figure 9 shows the project endpoint map.

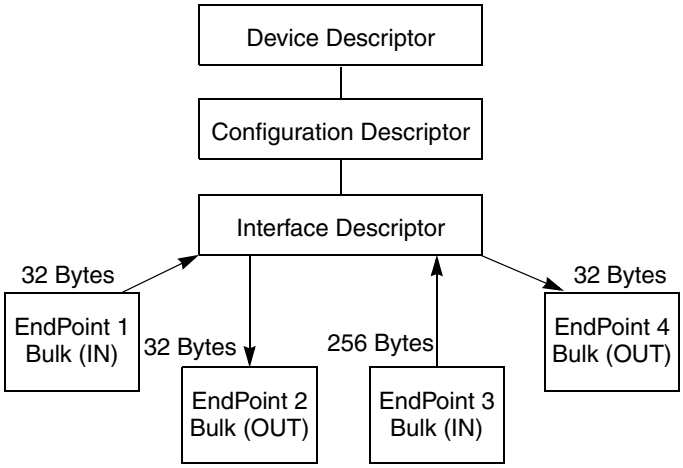

Figure 9. Device Descriptor Map

In this application, the endpoints have the following functions:

- Endpoint 1: sends the byte requested by the endpoint 2.
- Endpoint 2: receives the USB requests to write a specific byte to a specific address and memory (IIC/SPI). If a packet is decoded correctly and is a read memory packet, endpoint 2 sends through endpoint 1 the requested byte. Otherwise, it writes the byte into a specific memory. Endpoint 2 is always listening the USB host.
- Endpoint 3: sends 256 bytes requested by the endpoint 4.

- Endpoint 4: receives commands, depending on the command it performs an action. If it is a request to read entire memory, endpoint 4 sends through endpoint 3 all the information.

4.6 IIC and SPI drivers

Explaining the IIC and SPI modules is beyond the scope of this document. More information is available at www.freescale.com. These modules are managed by interrupts because the USB host needs a fast response. These drivers were adapted to specific memory vendors. To use another memory, some minor changes need to be made.

4.7 USB Device Power

The board was designed to give power when it works as a USB host, but is unable to receive power when it works as a USB device because of board connections. Table 2 describes the jumpers.

Table 2. Jumpers Position

Jumper	Position
5V_SEL	REMOVE
FLT	SET
ILIM	LO (HI/LO)
DPPD	REMOVE
DMPD	REMOVE

If a board needs USB device capability and is not self-powered, it should receive power from pin 1 of the USB receptacle and also needs to be notified in its device descriptor.

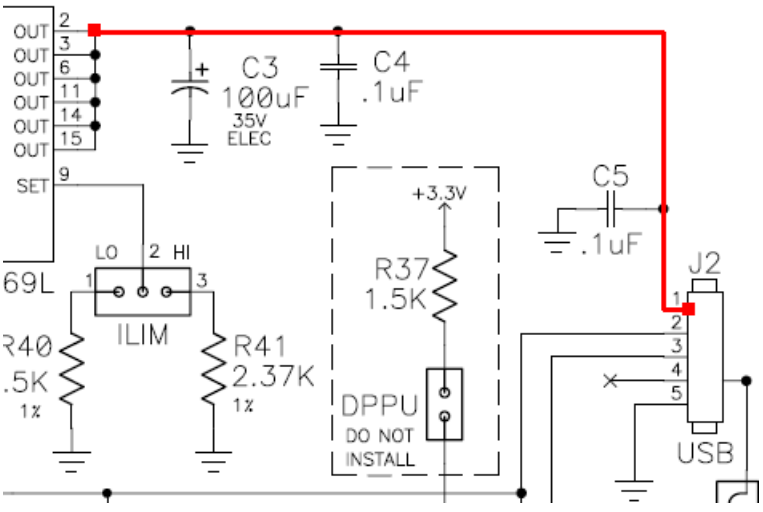

Figure 10. M52221 Schematic

4.8 Electrical Connections

The following schematic is needed to test the software. An IIC 2402 and an SPI 2502, 256 bytes each, are shown on Figure 11. Two pull-up resistances are included because, internally, memories have an open collector configuration.

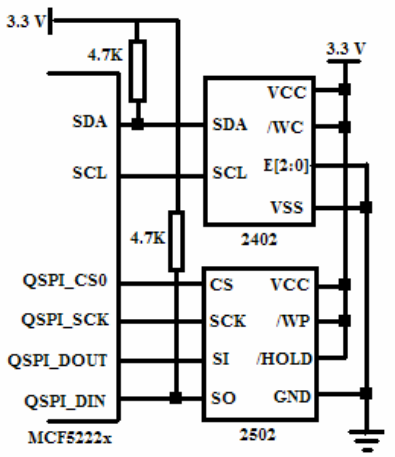

Figure 11. Memories Connection to MCF5222x ColdFire

5 PC USB Host Software

5.1 Project Features

In this project, the PC is the USB Host, and the following characteristics are needed:

- Windows operating system.
- Visual Studio 2005 using Visual C++.
- USB driver from Microsoft Windows Driver Kit (WDK), which can be found at www.microsoft.com. After installing it, read the document C:\WinDDK\6000\relnote.htm. WDK is replacing WinDDK to support the new Windows operating system called Windows Vista. WinDDK is necessary to change or compile the Visual Studio project.
- Bulkusb.sys was built using WinXP x86 Free Build Environment command line found on Programs>Windows Driver Kits>WDK 6000>Build Environments>Windows XP, a command line program, and then using build command plus driver location.

5.2 Generic Driver for a PC Application

The following details must be considered to change the Visual Studio project included with this application note:

- Usb_api.c: includes the creation of all endpoints. It must change if the number of endpoints is changed. It also contains functions to send and receive USB bulk packets. This is the heart of the application.

- Usb_driver.c: this file must not be changed. It includes the low-level communication between Windows USB driver and any device connected to the PC.
- ReadMemoryDlg.cpp, ReadSPImemory.cpp, USB-IIC-SPI.cppm and USB-IIC-SPI-Dlg.cpp files are used to handle GUI and usb_api.c. These files change if the application appearance (GUI: buttons dialogs, etc) needs to look different.
- The INF file (bulkusb.inf in this project) vendor and product ID must be the same as included in the device descriptor file, on the ColdFire board. In this example, the driver uses a VID of 1234 and a PID of 5678.

```
[Manufacturer]
%MfgName%=Microsoft, NTx86, NTia64, NTamd64

[Microsoft.NTx86]
%USB\VID_1234&PID_5678% DeviceDesc%=BULKUSB.Dev, USB\VID_1234&PID_5678

[Microsoft.NTia64]
%USB\VID_1234&PID_5678% DeviceDesc%=BULKUSB.Dev, USB\VID_1234&PID_5678

[Microsoft.NTamd64]
%USB\VID_1234&PID_5678% DeviceDesc%=BULKUSB.Dev, USB\VID_1234&PID_5678
```

Figure 12. PID and VID on Bulkusb.inf File

5.3 Getting the Microsoft USB Host API

The use of Windows Visual Studio requires a license. Otherwise, WDK is free and can be downloaded from Windows web site. USB driver from WDK can be compiled through a DOS command line. Microsoft Visual C++ 2005 Express Edition, a free Microsoft tool, cannot be used with this project because it does not contain some files needed by the USB driver.

These are the functions needed to drive USB host function:

`usb_open_device()`
: Is the USB initialization. Called one time before any USB transaction.

`usb_close_device()`
: Called when the USB device and pipes handles need to be closed.

`usb_write(unsigned char * data, int length )`
: Sends an array of data through an specific endpoint of a specific length. Function declaration includes the number of endpoint, so usb_api.c must be changed to define a new endpoint number.

`int usb_read(unsigned char * data, int length )`
: Receives an array of data through an specific endpoint of a specific length. Function declaration includes the number of endpoint, so usb_api.c must be changed to define a new endpoint number.

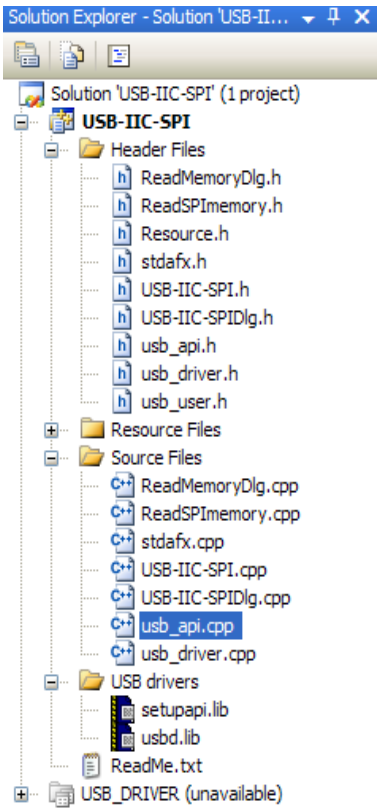

Figure 13. Visual Studio Overview

5.4 Compiling Microsoft USB Host API

The WinDDK must be previously installed to compile the USB-IIC-SPI project. Its size is around 1.6 GB. The default USB driver is stored in the following path:

```
C:\WinDDK\6000\src\usb\bulkusb
```

A copy of this folder is called bulkusb_sys and stored in:

```
C:\WinDDK\6000\src\usb\bulkusb_sys
```

After that, the build command is executed, as explained in Section 5.1, “Project Features”. The build command outputs a SYS file that manages all low-level USB transactions between the host and the device. This and the INF files are requested each time a device is connected to a new PC with Windows OS.

5.5 Starting a Project with Visual Studio and Microsoft USB Host API

The best way to start a new USB application is to use the USB-IIC-SPI project as a starting point because it includes all the necessary files to compile. Otherwise, these steps are useful for the design of a new project using Visual Studio:

1. Go to Files>New>Projec> Visual C++>MFC>MFC Application. A window pops up, select the items as shown in Figure 14.

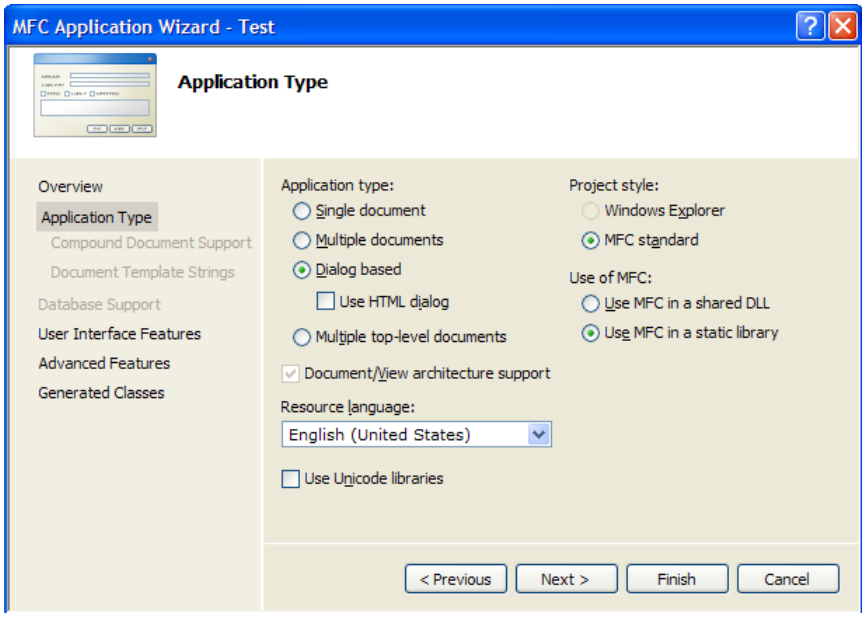

Figure 14. Visual Studio Wizard

2. Selected the project properties as shown in Figure 15.

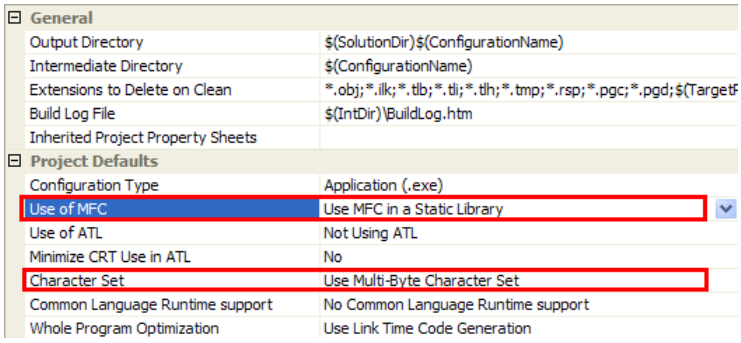

Figure 15. Visual Studio Properties

3. All files from the other project must be added and customized to the project, except ReadMemoryDlg.cpp, ReadSPImemory.cpp, USB-IIC-SPI.cpp, and USB-IIC-SPI-Dlg.cpp because of relativity to the old project. However, they can be used to know how to use WinDDK driver. SYS carpet with USB low-level code should also be added to the project, but not compiled. This helps to avoid some errors during compilation.
4. Finally, the project must be compiled and tested with the device to be connected.

6 Conclusion

The USB module is a versatile module with 16 endpoints that allows the programmer to build robust applications. The software included for Visual Studio and Codewarrior can be used as a starting point for any project that needs communication between a PC and a ColdFire. With a few changes, the Windows application can be used with other Freescale processors like 8/16-bit MCUs, Power Architecture, and i.MX processors.

THIS PAGE IS INTENTIONALLY BLANK

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 010 5879 8000
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Document Number: AN3530
Rev. 0
05/2008

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including “Typicals”, must be validated for each customer application by customer’s technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

RoHS-compliant and/or Pb-free versions of Freescale products have the functionality and electrical characteristics as their non-RoHS-compliant and/or non-Pb-free counterparts. For further information, see http://www.freescale.com or contact your Freescale sales representative.

For information on Freescale’s Environmental Products program, go to http://www.freescale.com/epp.

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org

© Freescale Semiconductor, Inc. 2008. All rights reserved.

freescale™
semiconductor