# Debugging EZ-USB FX1/FX2LP Firmware Using the Keil Debugger Environment

## About this document

### Scope and purpose

This application note is a step-by-step beginner's guide to debug EZ-USB™ FX1/FX2LP™ firmware using the Keil™ Debugger. It provides the guidelines to start and stop a debug session, set breakpoints, and step through code.

### Intended audience

This application note is intended for application engineers interested in debug EZ-USB FX1/FX2LP firmware using the Keil Debugger.

## Table of contents

## 1 Introduction

While developing FX1/FX2LP firmware projects, designers must be able to debug code. The firmware debug can be performed by printing debug messages over UART or by performing single-step debugging using the Keil debugger environment. Refer to the application note **AN58009 - Serial (UART) Port Debugging of FX1/FX2LP™ Firmware** for more details on Serial (UART) port debugging. This application note explains FX1/FX2LP firmware debug using the Keil Debugger environment, The Keil tools provided with the EZ-USB FX2LP family of development kits give the ability to debug code using single step, start, stop and other features discussed in this application note. The information provided here has been obtained directly from the Keil help system. Cypress provides two utilities to help customers develop designs. One of them is the "USB Control Center" included with the **EZ-USB FX3 SDK**. To use the Keil debugger, connect a serial port from the host PC (where the firmware project is) to the development board. Use a standard serial cable or USB-to-Serial cable to connect the PC with the development board.

There are two serial ports on the FX2LP development board. They are labeled as "SIO-0" and "SIO-1". Connect the serial cable to one of these two serial ports based on one of the four debug monitor files used.

# 2 Download of the Debug Monitor File

## 2.1 Automatic Download

The debug monitor file is automatically loaded when the development board is connected to the PC if the default EEPROM is used. If so, the green BKPT MONITOR LED located just below the USB connector on the development board indicates the monitor file is loaded and uses SIO-1. The debug monitor file can also be manually downloaded using the “USB Control Center” if the default EEPROM image is not used.

## 2.2 Manual Download

The Cypress USB utility (“USB Control Center”) uses USB to download a monitor file. Plug the **CY3684 EZ-USB® FX2LP Development Kit** board into the PC using a USB cable.

Cypress provides four debug monitor files from which to choose:

- *mon-ext-sio0-c0.hex* – connects to SIO-0, loads to external SRAM memory of the DVK
- *mon-ext-sio1-c0.hex* – connects to SIO-1, loads to external SRAM memory of the DVK
- *mon-int-sio0.hex* – connects to SIO-0, loads to internal memory
- *mon-int-sio1.hex* – connects to SIO-1, loads to internal memory

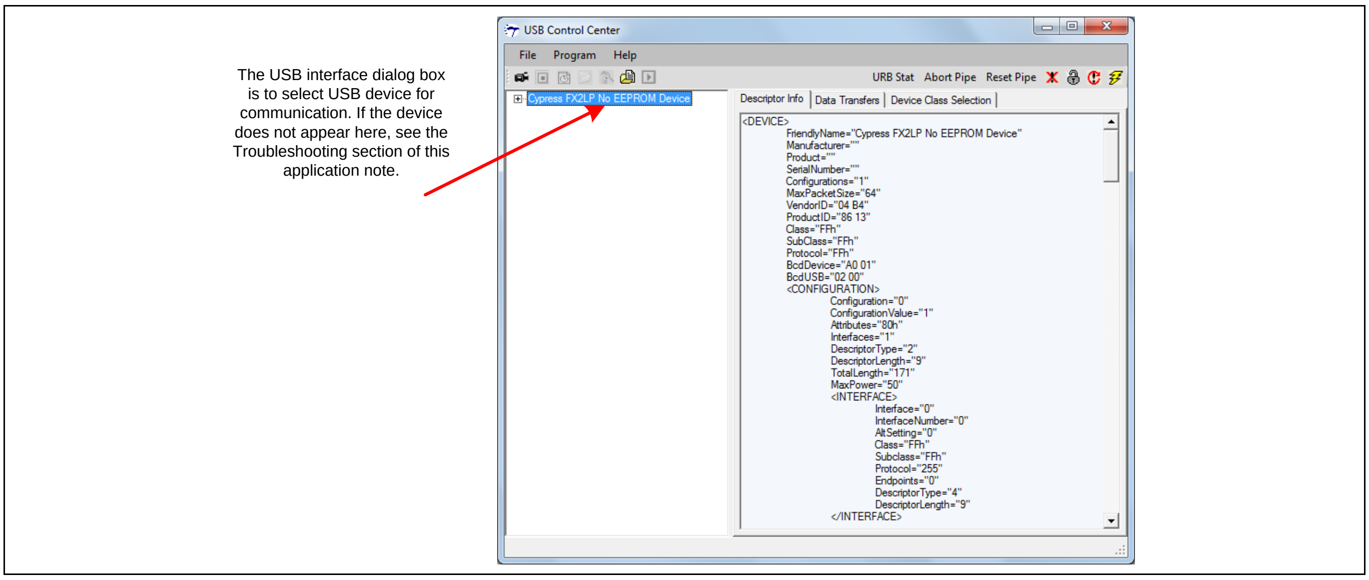
Additional information on the external SRAM memory, you can refer the Chapter 5 Memory of **FX2LP TRM**. It gives you more information on how to use the external memory.

Make sure the EZ-USB development board is plugged into the PC using a USB cable.

Download and install **EZ-USB FX3 SDK**. Launch the USB Control Center:

Go to **Start** > **All Programs** > **Cypress** > **EZ-USB FX3 SDK** > **Cypress USBSuite** > **Control Center**.

The application launches and the following window displays (see **Figure 1**).

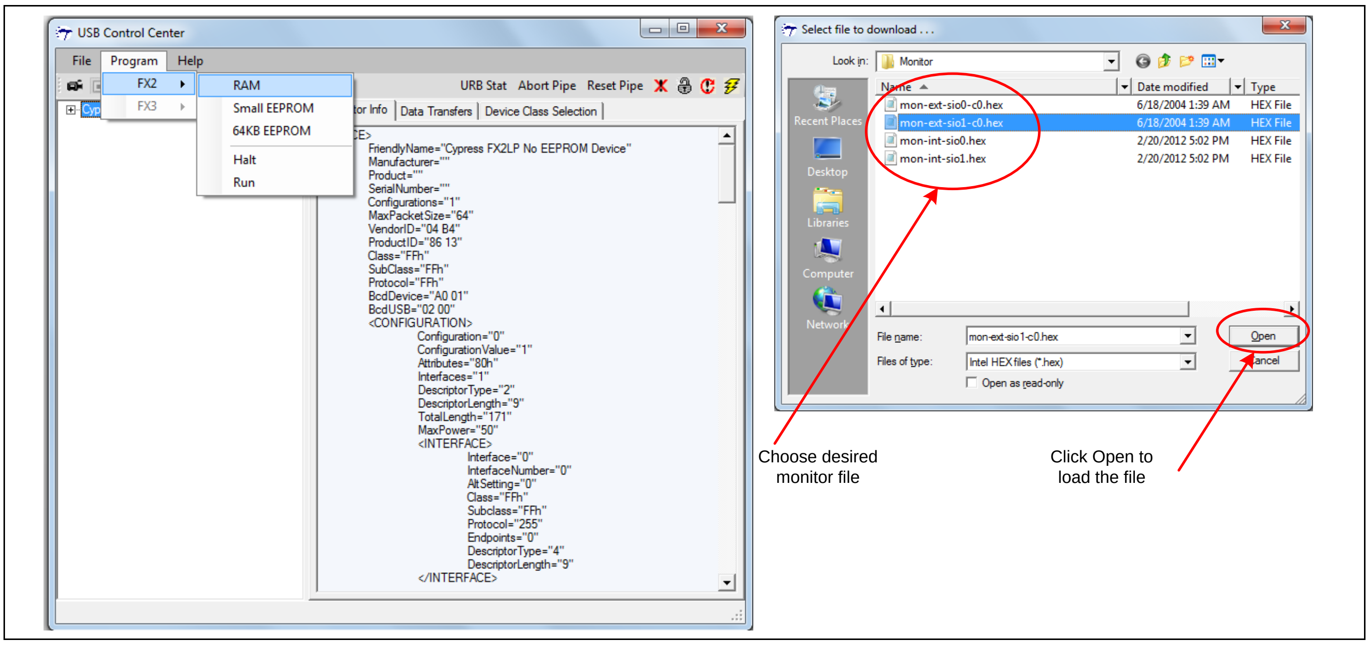


The USB interface dialog box is to select USB device for communication. If the device does not appear here, see the Troubleshooting section of this application note.

```
<DEVICE>
        FriendlyName="Cypress FX2LP No EEPROM Device"
        Manufacturer=""
        Product=""
        SerialNumber=""
        Configurations="1"
        MaxPacketSize="64"
        VendorID="04 B4"
        ProductID="86 13"
        Class="FFh"
        SubClass="FFh"
        Protocol="FFh"
        BcdDevice="A0 01"
        BcdUSB="02 00"
        <CONFIGURATION>
                Configuration="0"
                ConfigurationValue="1"
                Attributes="80h"
                Interfaces="1"
                DescriptorType="2"
                DescriptorLength="9"
                TotalLength="171"
                MaxPower="50"
                <INTERFACE>
                        Interface="0"
                        InterfaceNumber="0"
                        AltSetting="0"
                        Class="FFh"
                        Subclass="FFh"
                        Protocol="255"
                        Endpoints="0"
                        DescriptorType="4"
                        DescriptorLength="9"
                </INTERFACE>
```

**Figure 1 USB Interface Dialog Box – Select Debug Monitor File**

Click **Program** > **FX2** > **RAM**. The Specify Monitor File dialog box displays (see **Figure 2**).



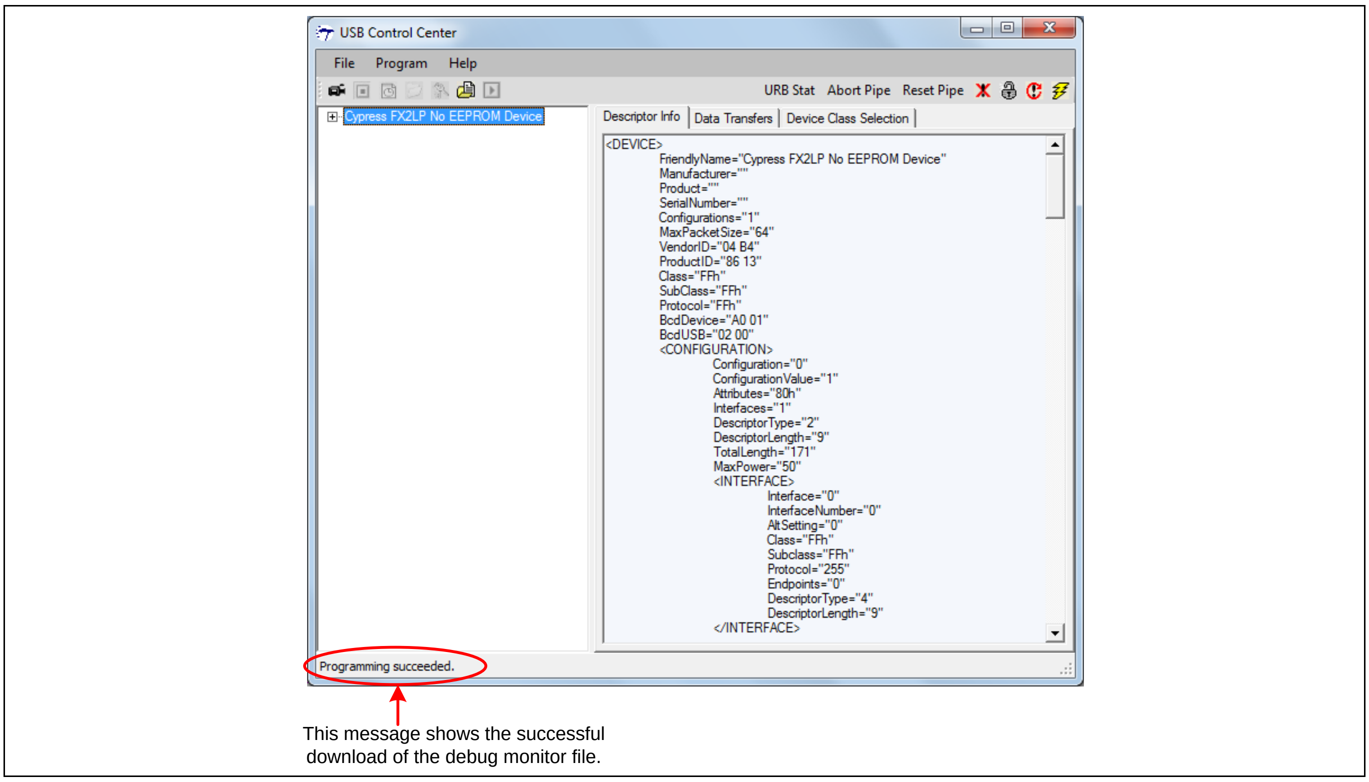
```
E>
    FriendlyName="Cypress FX2LP No EEPROM Device"
    Manufacturer=""
    Product=""
    SerialNumber=""
    Configurations="1"
    MaxPacketSize="64"
    VendorID="04 B4"
    ProductID="86 13"
    Class="FFh"
    SubClass="FFh"
    Protocol="FFh"
    BcdDevice="A0 01"
    BcdUSB="02 00"
    <CONFIGURATION>
        Configuration="0"
        ConfigurationValue="1"
        Attributes="80h"
        Interfaces="1"
        DescriptorType="2"
        DescriptorLength="9"
        TotalLength="171"
        MaxPower="50"
        <INTERFACE>
            Interface="0"
            InterfaceNumber="0"
            AltSetting="0"
            Class="FFh"
            Subclass="FFh"
            Protocol="255"
            Endpoints="0"
            DescriptorType="4"
            DescriptorLength="9"
        </INTERFACE>
```

| Name | Date modified | Type |
|---|---|---|
| mon-ext-sio0-c0.hex | 6/18/2004 1:39 AM | HEX File |
| mon-ext-sio1-c0.hex | 6/18/2004 1:39 AM | HEX File |
| mon-int-sio0.hex | 2/20/2012 5:02 PM | HEX File |
| mon-int-sio1.hex | 2/20/2012 5:02 PM | HEX File |

File name: mon-ext-sio1-c0.hex
Files of type: Intel HEX files (*.hex)

**Figure 2 Specify Monitor File Dialog Box**

If your dialog box does not match, go to
*C:\Cypress\USB\CY3684_EZ-USB_FX2LP_DVK\1.1\Target\Monitor*,
or it can also be found in the folder 'Monitor' attached along with this application note.



```
<DEVICE>
    FriendlyName="Cypress FX2LP No EEPROM Device"
    Manufacturer=""
    Product=""
    SerialNumber=""
    Configurations="1"
    MaxPacketSize="64"
    VendorID="04 B4"
    ProductID="86 13"
    Class="FFh"
    SubClass="FFh"
    Protocol="FFh"
    BcdDevice="A0 01"
    BcdUSB="02 00"
    <CONFIGURATION>
        Configuration="0"
        ConfigurationValue="1"
        Attributes="80h"
        Interfaces="1"
        DescriptorType="2"
        DescriptorLength="9"
        TotalLength="171"
        MaxPower="50"
        <INTERFACE>
            Interface="0"
            InterfaceNumber="0"
            AltSetting="0"
            Class="FFh"
            Subclass="FFh"
            Protocol="255"
            Endpoints="0"
            DescriptorType="4"
            DescriptorLength="9"
        </INTERFACE>
```

Programming succeeded.

**Figure 3 USB Interface Dialog Box – Download Debug Monitor File**

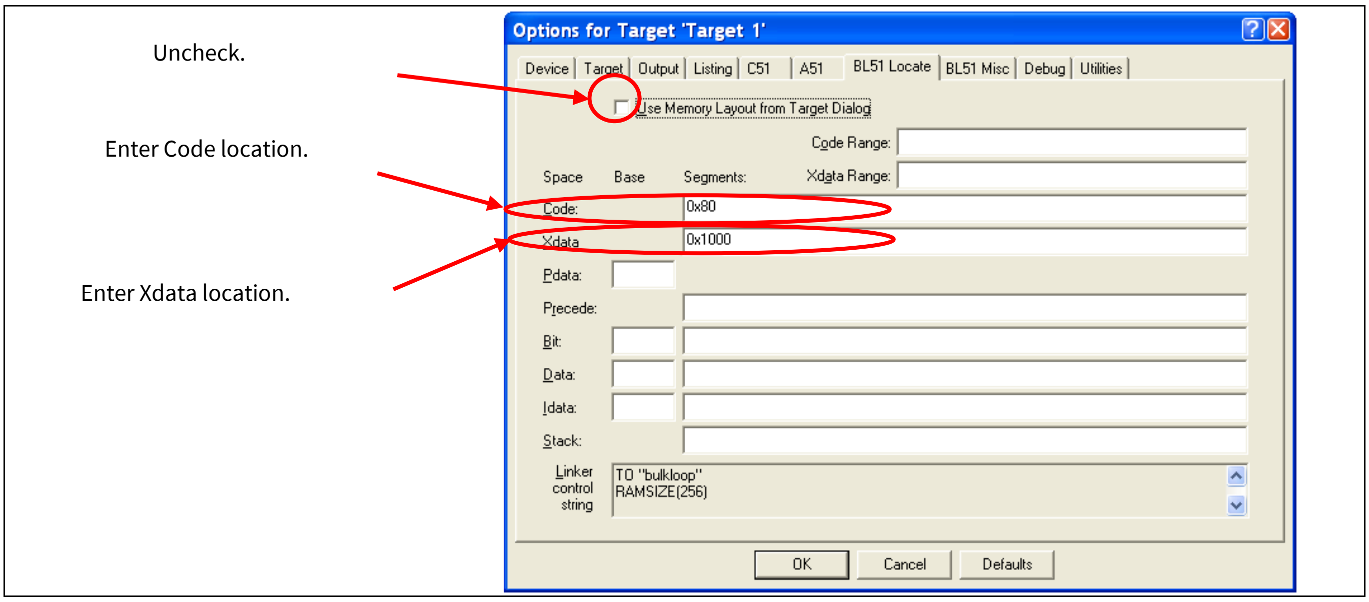
The lit green BKPT MONITOR LED indicates a successful download.

## 3 Debug FX1/FX2™ Using External Monitor Code

For debugging using external monitor code, '*mon-ext-sio0-c0.hex*' and '*mon-ext-sio1-c0.hex*' are used for Serial port 0 and Serial port 1 respectively. When either of these files is chosen, follow the steps below to debug the FX2LP firmware.

### 3.1 Keil Debug Monitor, Debug Settings

1. Launch the Keil µVision2 application:
   Go to **Start** > **All Programs**> **Keil µVision2**.
2. Open the project:
   In the Keil µVision2 application, go to **Project** > **Open Project**.
3. Specify code location and data location in memory:
   Go to **Project** > **Options for Target 'Target 1'**, and select the **BL51 Locate** tab (or go to **Project** > **Components, Environment, Books**…, and select the **Folders/Extensions** tab for Keil µVision3).



☐ Use Memory Layout from Target Dialog

Code: 0x80
Xdata: 0x1000

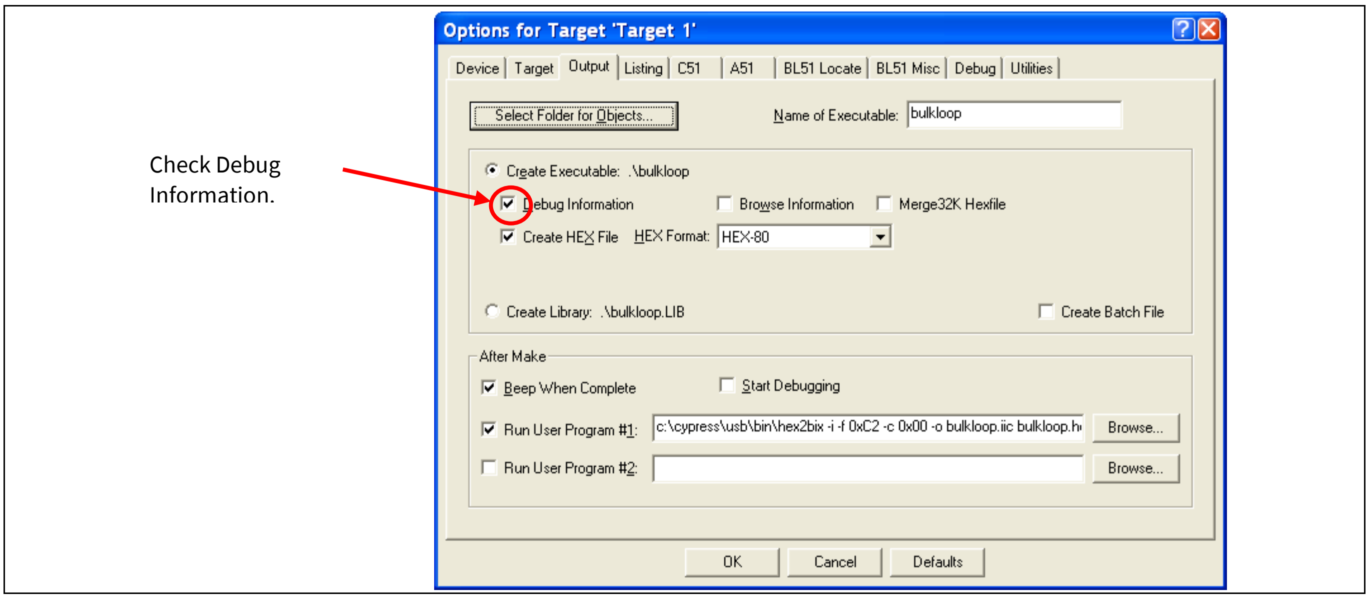
Linker control string:
```
TO "bulkloop"
RAMSIZE(256)
```

**Figure 4 Options for Target 'Target 1' Dialog Box**

Because the FX2LP has an interrupt vector located from 0x00 to 0x7C, it is recommended to set the Code location from 0x80 and the Xdata location from 0x1000. Note that these values are used only when the monitor file is loaded to the external memory. Section **5.2 Specifying the Code and XData Range** explains the Code and Xdata values to be chosen if the monitor file is loaded to internal RAM.

Also, ensure the code and monitor code locations do not overlap. Typically, choose a monitor that resides in external memory and have the code reside in internal memory so that the two do not step over each other.

To enable using breakpoints while debugging, check Debug Information on the Output tab in the Options for Target 'Target 1' dialog box (see **Figure 5**).


Check Debug Information.
Options for Target 'Target 1'
Device | Target | Output | Listing | C51 | A51 | BL51 Locate | BL51 Misc | Debug | Utilities
Select Folder for Objects...
After Make
Browse...
OK
Cancel
Defaults

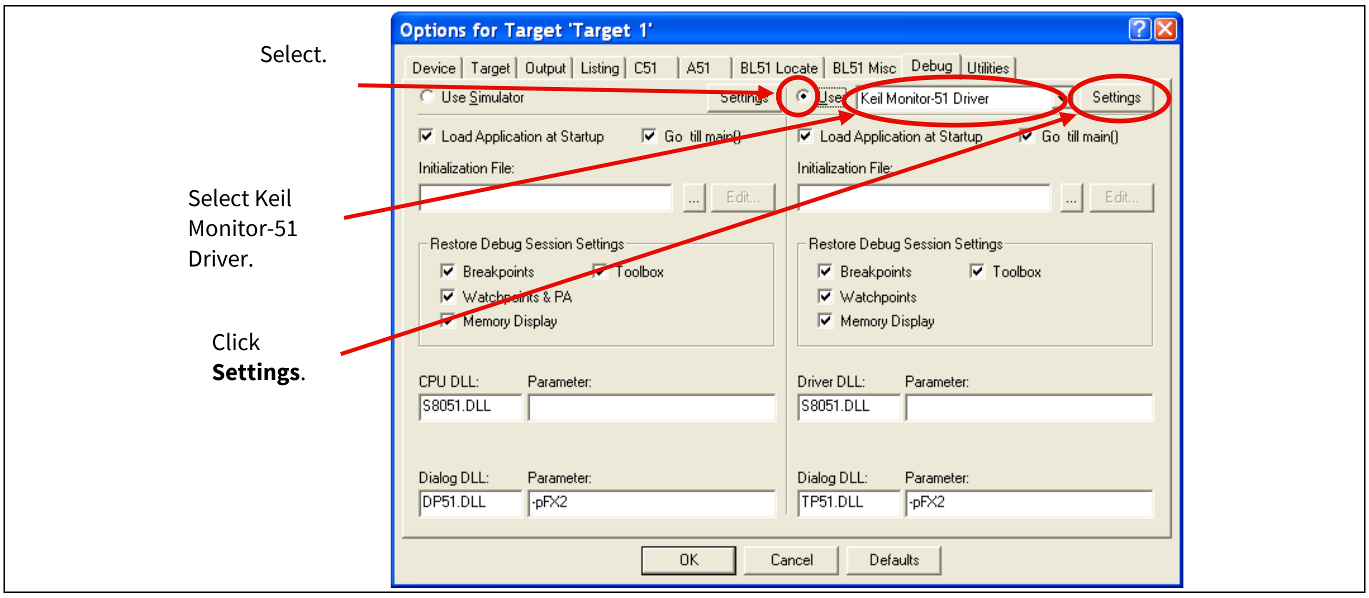
Name of Executable: bulkloop

(●) Create Executable: .\bulkloop
[✓] Debug Information [ ] Browse Information [ ] Merge32K Hexfile
[✓] Create HEX File HEX Format: HEX-80
( ) Create Library: .\bulkloop.LIB [ ] Create Batch File

[✓] Beep When Complete [ ] Start Debugging
[✓] Run User Program #1: c:\cypress\usb\bin\hex2bix -i -f 0xC2 -c 0x00 -o bulkloop.iic bulkloop.he
[ ] Run User Program #2:

**Figure 5 Options for Target 'Target 1' Dialog Box – Debug Breakpoints**

Ensure the correct serial port and baud rate are set. Click the **Debug** tab. The following window displays (see **Figure 6** and **Figure 7**).


Select.
Select Keil Monitor-51 Driver.
Click **Settings**.
Options for Target 'Target 1'
Device | Target | Output | Listing | C51 | A51 | BL51 Locate | BL51 Misc | Debug | Utilities
Settings
Settings
Restore Debug Session Settings
Restore Debug Session Settings
Edit...
Edit...
OK
Cancel
Defaults

( ) Use Simulator | (●) Use: Keil Monitor-51 Driver

Simulator side:
[✓] Load Application at Startup [✓] Go till main()
Initialization File:
[✓] Breakpoints [✓] Toolbox
[✓] Watchpoints & PA
[✓] Memory Display
CPU DLL: S8051.DLL Parameter:
Dialog DLL: DP51.DLL Parameter: -pFX2

Monitor side:
[✓] Load Application at Startup [✓] Go till main()
Initialization File:
[✓] Breakpoints [✓] Toolbox
[✓] Watchpoints
[✓] Memory Display
Driver DLL: S8051.DLL Parameter:
Dialog DLL: TP51.DLL Parameter: -pFX2

**Figure 6 Options for Target 'Target 1' Dialog Box – Debug Settings**

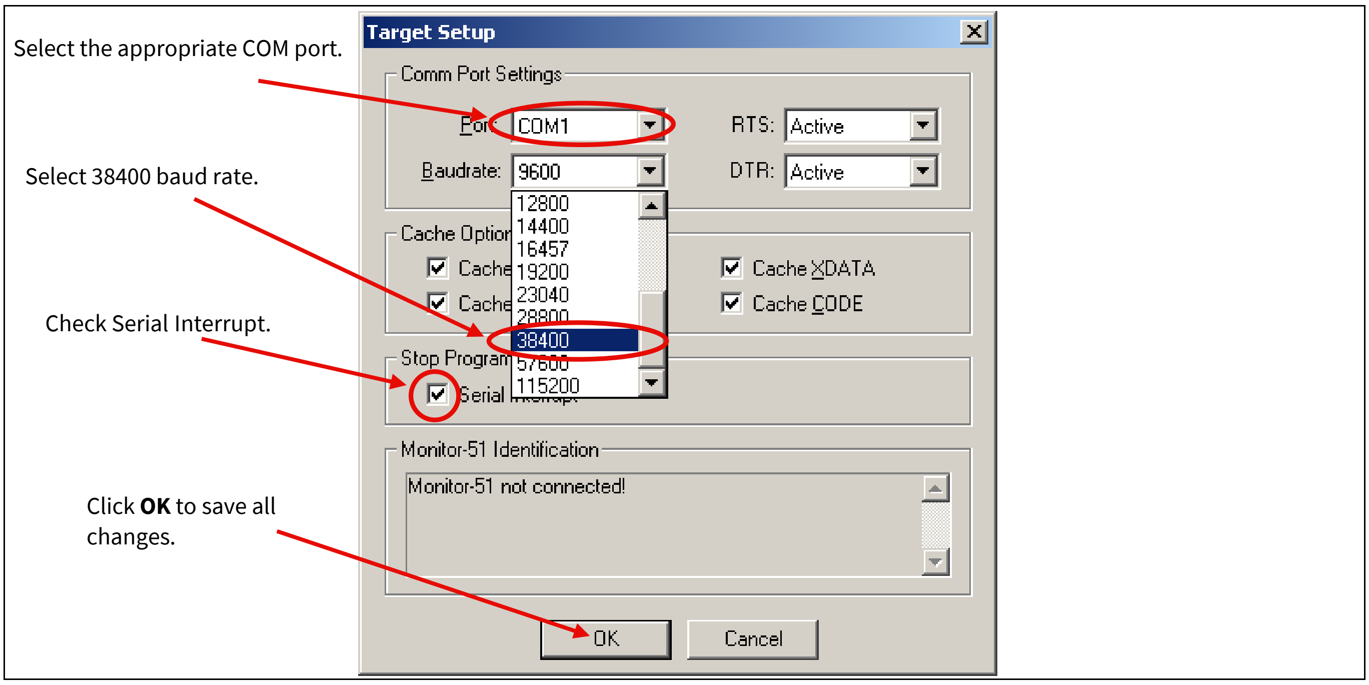


**Figure 7** **Options for Target 'Target 1' Dialog Box – COM Port, Baud Rate Settings**

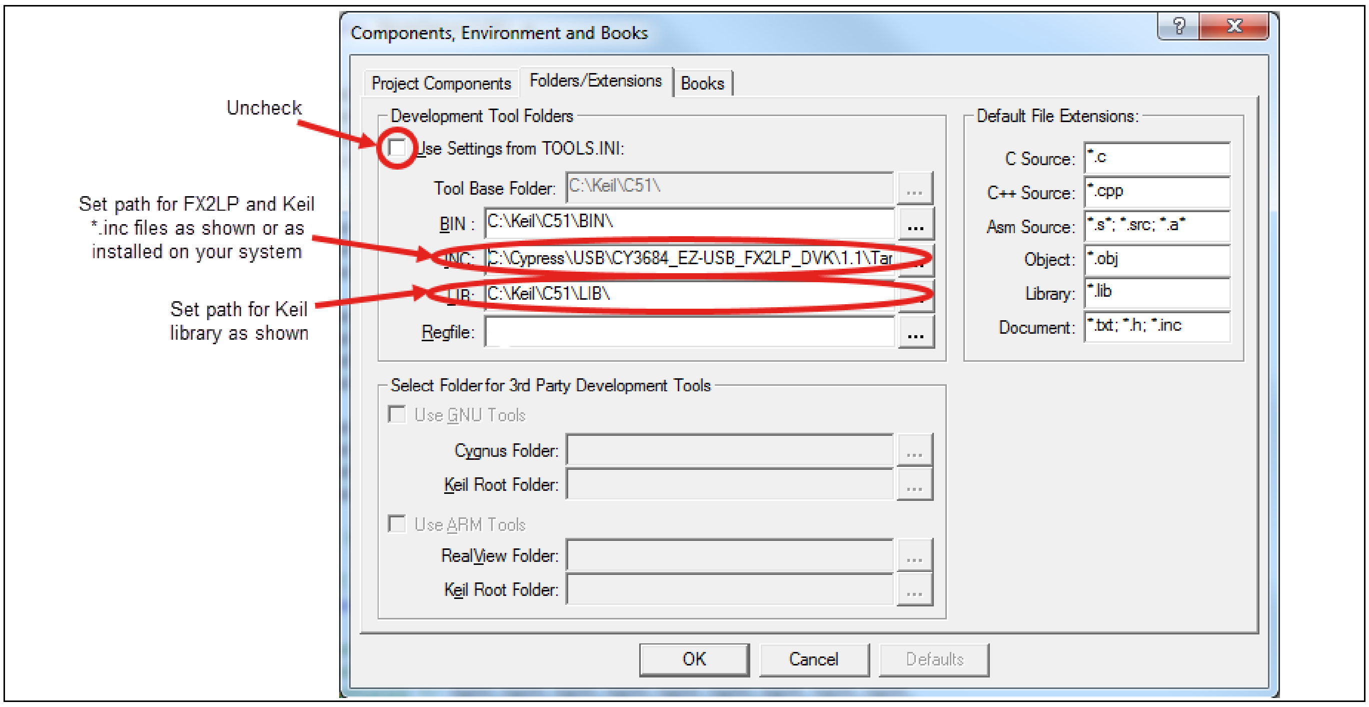
Note that the correct COM port should be selected. When the serial port 'SIO-0' or 'SIO-1' of FX2LP is connected to the PC, it will enumerate in the Device Manager as 'USB Serial Port (COMx)', where x is the port number. Choose this port number in the debug settings. If it does not enumerate as a COM port, install the drivers for 'RS232 9-pin to 9-pin cable' for your OS.

Click **OK** in the Options to save all settings. This returns you to the main window of the Keil µVision2 application. Note that the debug port of FX2LP works only at a baud rate of 38400.

Now all the necessary debug settings are set.

To ensure a project build and allow debugging, specify the environment folder settings according to the path of the installed development components.

Go to **Project** > **Components, Environment and Books** and select the **Folders/Extensions** tab.


Uncheck
Set path for FX2LP and Keil *.inc files as shown or as installed on your system
Set path for Keil library as shown
Components, Environment and Books
Project Components
Folders/Extensions
Books
OK
Cancel
Defaults

Development Tool Folders
- [ ] Use Settings from TOOLS.INI:
- Tool Base Folder: C:\Keil\C51\
- BIN : C:\Keil\C51\BIN\
- INC: C:\Cypress\USB\CY3684_EZ-USB_FX2LP_DVK\1.1\Tar
- LIB: C:\Keil\C51\LIB\
- Regfile:

Default File Extensions:

| | |
|---|---|
| C Source: | *.c |
| C++ Source: | *.cpp |
| Asm Source: | *.s*; *.src; *.a* |
| Object: | *.obj |
| Library: | *.lib |
| Document: | *.txt; *.h; *.inc |

Select Folder for 3rd Party Development Tools
- [ ] Use GNU Tools
- Cygnus Folder:
- Keil Root Folder:
- [ ] Use ARM Tools
- RealView Folder:
- Keil Root Folder:

**Figure 8 Setup File Extensions Dialog Box**

Enter the INC field as follows:

*C:\CYPRESS\USB\CY3684_EZ-USB_FX2LP_DVK\1.1\TARGET\INC\;C:\Keil\C51\INC\*

Click **OK** in the Options to save the settings.

## 3.2 Debugging Session

To start debugging, select the **Debug** icon in the Keil µVision2 application (**Figure 9**).

View the Output Window, usually located in the lower left of the IDE, to verify that you are connected to the monitor and the program has loaded (that is, it says something such as, “Connected to Monitor-51 V3.0”). An error message such as, “Monitor error 22: no code memory at address 0045h”, does not prevent you from debugging.

Refer to the following Knowledge Base articles for additional information:

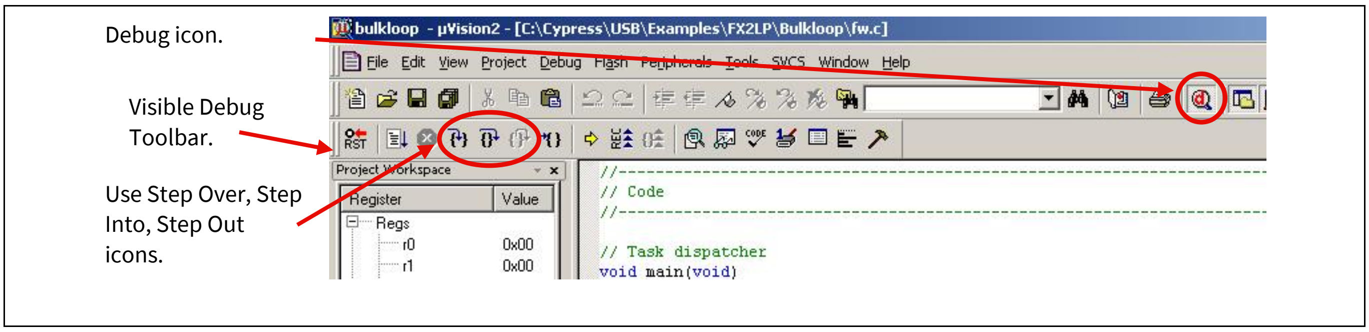
- **How to use Keil µVision IDE**
- **Settings for ‘Connection to Target’ dialog box in Keil debugger**
- **Usage of Serial to USB convertors with Keil Debugger**

Also, you should see the IDE switch to Debug mode, with a yellow arrow indicating the Program Counter location.

Make the Debug Toolbar visible, if it is not already.

Go to **View** > **Debug Toolbar**.

Now you can use the Step Over, Step Into, and Step Out icons to step through the code.

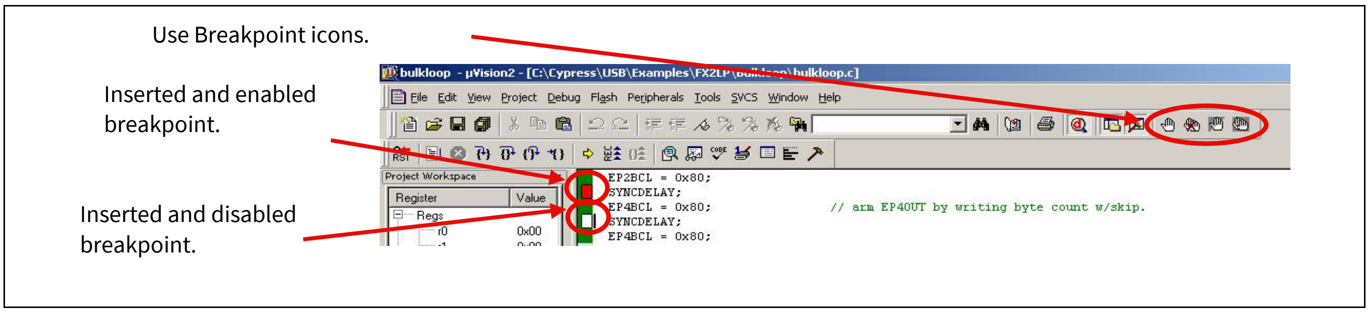

Debug icon.
Visible Debug Toolbar.
Use Step Over, Step Into, Step Out icons.
bulkloop - µVision2 - [C:\Cypress\USB\Examples\FX2LP\Bulkloop\fw.c]
File Edit View Project Debug Flash Peripherals Tools SVCS Window Help
Project Workspace
Register Value
Regs
r0 0x00
r1 0x00

```
//------------------------------------------------------------------
// Code
//------------------------------------------------------------------

// Task dispatcher
void main(void)
```

Figure 9 Keil IDE

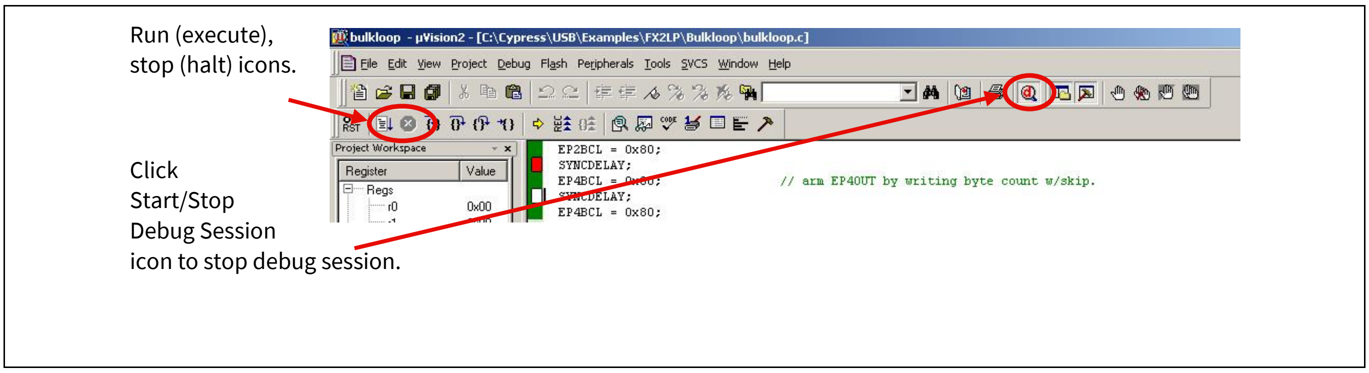
You can easily insert, remove, enable, or disable corresponding breakpoints using the **Insert/Remove Breakpoint**, **Enable/Disable Breakpoint**, **Disable All Breakpoints**, and **Kill All Breakpoints** icons. Place your cursor at the required line and use the icons mentioned to insert, remove, enable, or disable a breakpoint.


Use Breakpoint icons.
Inserted and enabled breakpoint.
Inserted and disabled breakpoint.
bulkloop - µVision2 - [C:\Cypress\USB\Examples\FX2LP\Bulkloop\bulkloop.c]
File Edit View Project Debug Flash Peripherals Tools SVCS Window Help
Project Workspace
Register Value
Regs
r0 0x00

```
EP2BCL = 0x80;
SYNCDELAY;
EP4BCL = 0x80;                // arm EP4OUT by writing byte count w/skip.
SYNCDELAY;
EP4BCL = 0x80;
```

Figure 10 Keil IDE Debugging

Using the **Run** icon, you can run (execute) the firmware until the active breakpoint occurs. Also, you can Stop (halt) program execution using the Stop icon (so you can halt the 8051). Be sure to check (enable) Serial Interrupt in **Figure 7** as indicated above.


Run (execute), stop (halt) icons.
Click Start/Stop Debug Session icon to stop debug session.
bulkloop - µVision2 - [C:\Cypress\USB\Examples\FX2LP\Bulkloop\bulkloop.c]
File Edit View Project Debug Flash Peripherals Tools SVCS Window Help
Project Workspace
Register Value
Regs
r0 0x00

```
EP2BCL = 0x80;
SYNCDELAY;
EP4BCL = 0x80;                // arm EP4OUT by writing byte count w/skip.
SYNCDELAY;
EP4BCL = 0x80;
```

Figure 11 Keil IDE Debugging

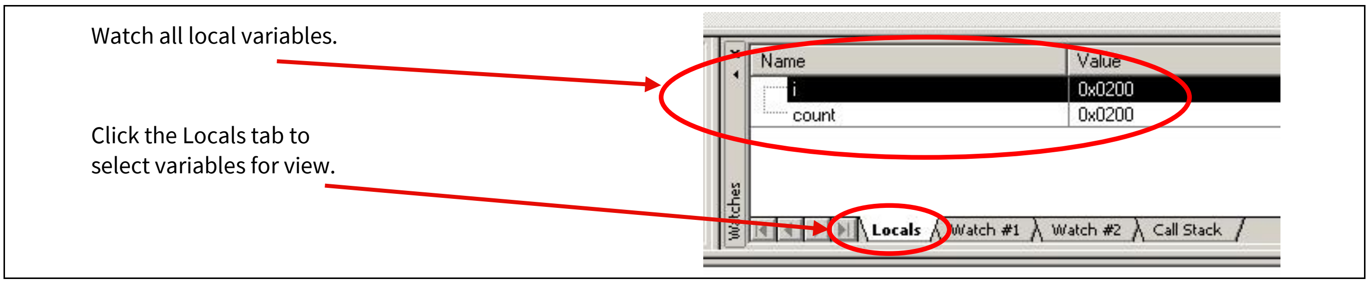
To stop a debug session, use the **Start/Stop Debug Session** icon.

During the debug session, you can watch the values change as the code executes. You can view variable values while the debug process halts or is in single-step mode debug. Hovering your mouse over a variable while debug halts or is in single-step mode shows you its current value.

There are also other features available while debugging your design. They include use of the Watch Window, CPU Registers, and Memory Window.

Make the Watch and Call Stack Window visible, if it is not already.

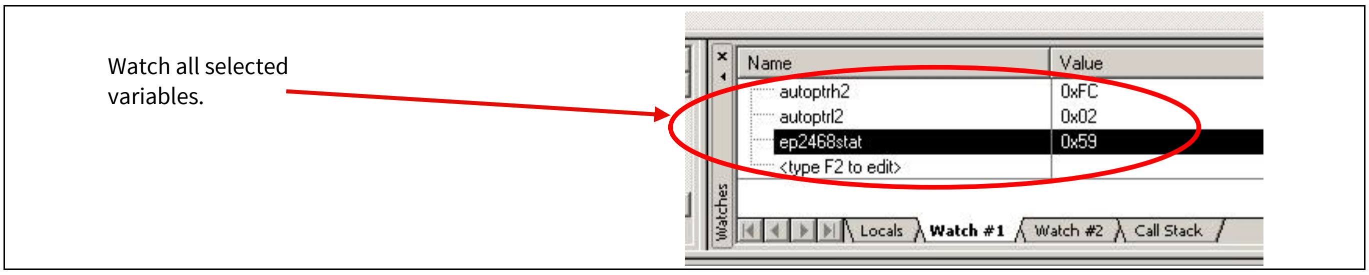
Go to **View** > **Watch and Call Stack Window**. The following window displays.


Watch all local variables.
Click the Locals tab to select variables for view.
Watches
Locals
Watch #1
Watch #2
Call Stack

| Name | Value |
|---|---|
| i | 0x0200 |
| count | 0x0200 |

**Figure 12** **Keil IDE Local Variables Tab**

The **Locals** tab shows all local function variables of the current function. The Watch tabs display user-specified program variables. You can specify all variables that you want to watch while debug halts.

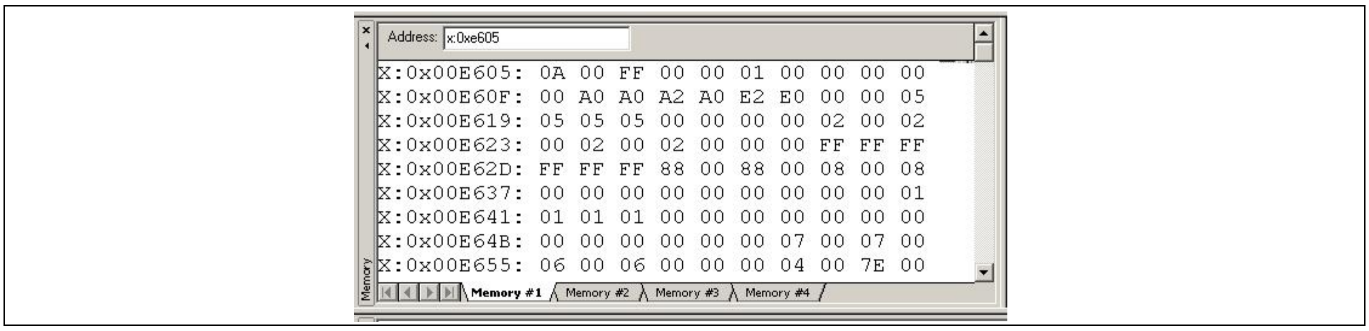


| Name | Value |
|---|---|
| autoptrh2 | 0xFC |
| autoptrl2 | 0x02 |
| ep2468stat | 0x59 |
| <type F2 to edit> | |

**Figure 13** **Keil IDE Watch Variables Tab**

Make the Memory window visible, if it is not already.

Go to **View** > **Memory Window**. The following window displays.

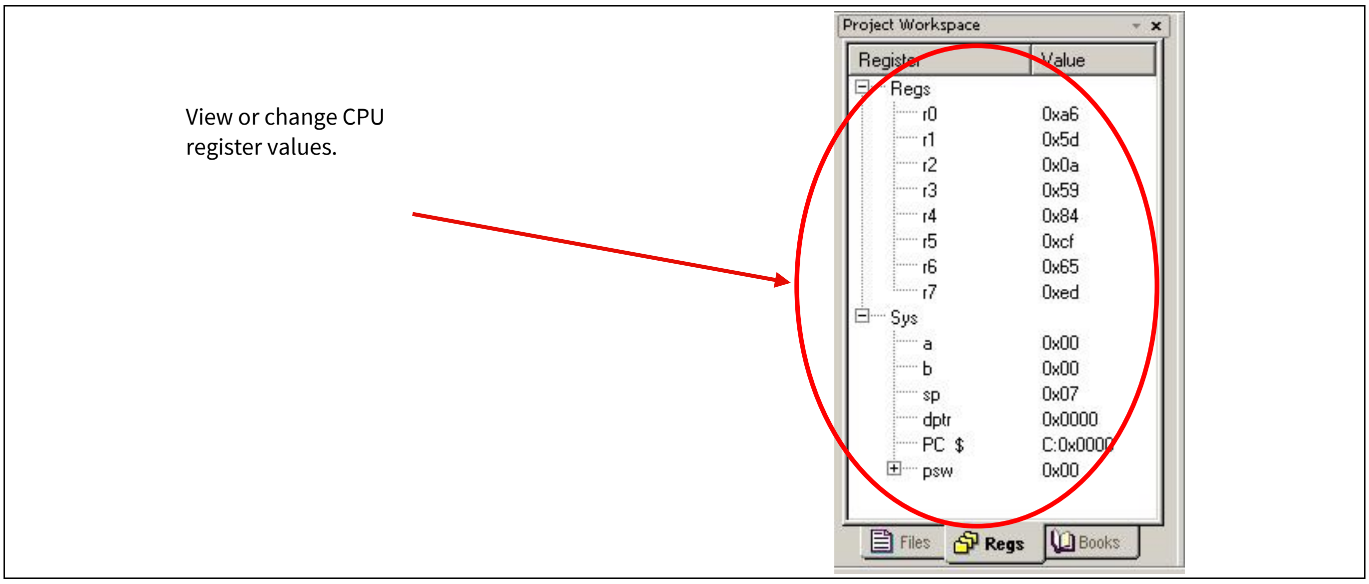


```
X:0x00E605: 0A 00 FF 00 00 01 00 00 00 00
X:0x00E60F: 00 A0 A0 A2 A0 E2 E0 00 00 05
X:0x00E619: 05 05 05 00 00 00 00 02 00 02
X:0x00E623: 00 02 00 02 00 00 00 FF FF FF
X:0x00E62D: FF FF FF 88 00 88 00 08 00 08
X:0x00E637: 00 00 00 00 00 00 00 00 00 01
X:0x00E641: 01 01 01 00 00 00 00 00 00 00
X:0x00E64B: 00 00 00 00 00 00 07 00 07 00
X:0x00E655: 06 00 06 00 00 00 04 00 7E 00
```

**Figure 14** **Keil IDE Memory Tab**

The Memory window displays the contents of the various memory areas. Up to four different areas can be reviewed in the different tabs.

You can also view CPU registers. They are displayed in the **Register** tab of the Project Workspace, and can be modified in the same way as variables in the Watch Window. When you run through some portion of your firmware, the registers with change in value are highlighted in the Register tab. This helps to easily identify the changes during debugging.

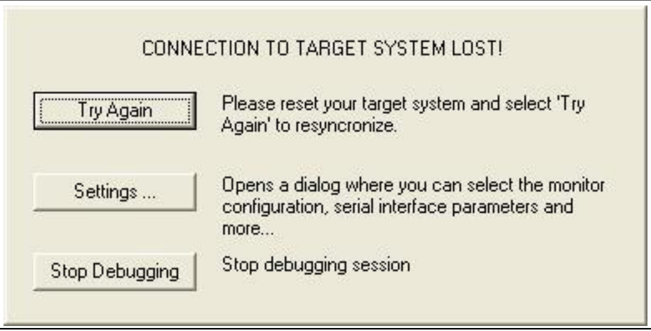


```
Register      Value
Regs
  r0          0xa6
  r1          0x5d
  r2          0x0a
  r3          0x59
  r4          0x84
  r5          0xcf
  r6          0x65
  r7          0xed
Sys
  a           0x00
  b           0x00
  sp          0x07
  dptr        0x0000
  PC  $       C:0x0000
  psw         0x00
```

**Figure 15 Keil IDE Register Tab**

# 4 Troubleshooting the Keil Debugger

**Q1**: When I am trying to run a debug session, the following message appears: “CONNECTION TO TARGET SYSTEM LOST!”



CONNECTION TO TARGET SYSTEM LOST!

Try Again — Please reset your target system and select 'Try Again' to resyncronize.

Settings ... — Opens a dialog where you can select the monitor configuration, serial interface parameters and more...

Stop Debugging — Stop debugging session

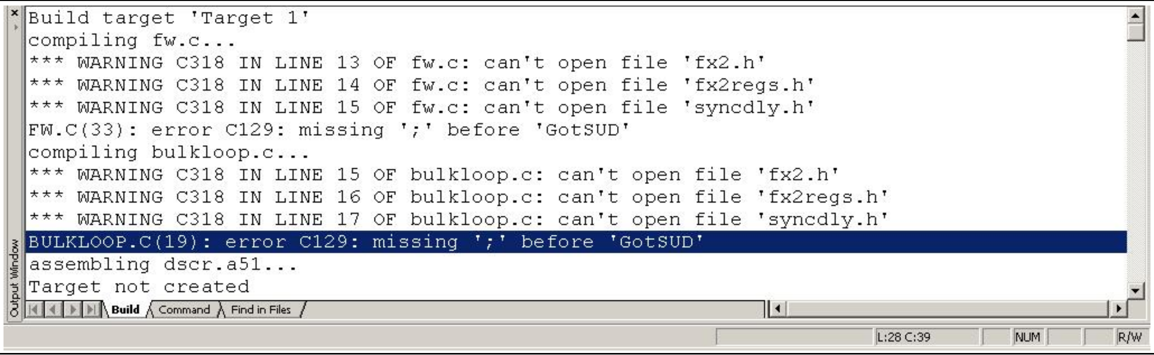
**Figure 16 Connection Error Message**

**A1**: You must check the following:

- Serial cable – make sure the serial cable is connected to the correct port on the development board and that the correct COM port being used on the host PC is selected in the debug settings.
- Verify all Debug settings – Baud rate set to 38400, serial interrupt enabled, correct COM port selected.
- Make sure your code and the monitor code locations do not overlap.

It is recommended to choose a monitor code that resides in external memory (mon-ext…) and your code resides in the internal memory so they do not step over each other.

**Q2**: When I try to compile my project, I receive an error message in the output window as follows:

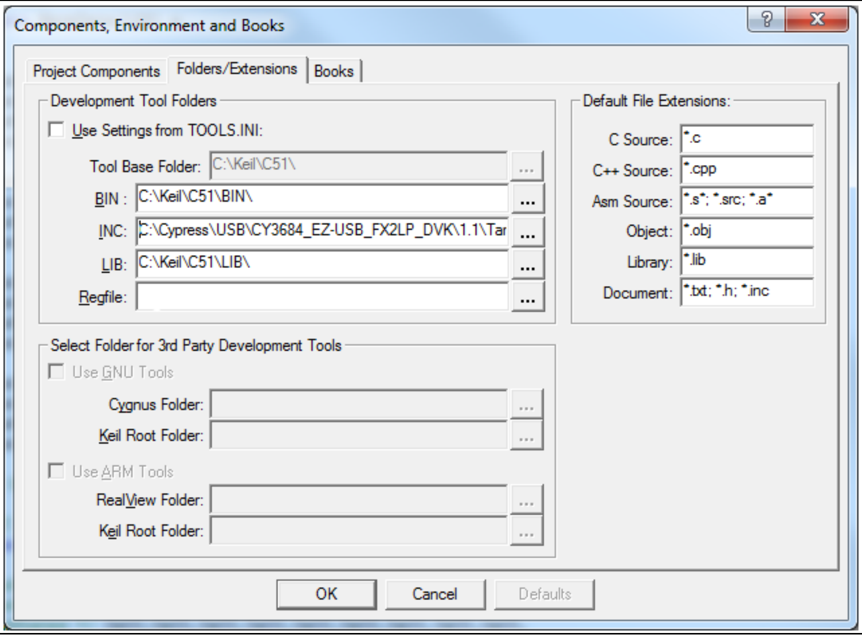


```
Build target 'Target 1'
compiling fw.c...
*** WARNING C318 IN LINE 13 OF fw.c: can't open file 'fx2.h'
*** WARNING C318 IN LINE 14 OF fw.c: can't open file 'fx2regs.h'
*** WARNING C318 IN LINE 15 OF fw.c: can't open file 'syncdly.h'
FW.C(33): error C129: missing ';' before 'GotSUD'
compiling bulkloop.c...
*** WARNING C318 IN LINE 15 OF bulkloop.c: can't open file 'fx2.h'
*** WARNING C318 IN LINE 16 OF bulkloop.c: can't open file 'fx2regs.h'
*** WARNING C318 IN LINE 17 OF bulkloop.c: can't open file 'syncdly.h'
BULKLOOP.C(19): error C129: missing ';' before 'GotSUD'
assembling dscr.a51...
Target not created
```

**Figure 17 Compile Error Message**

**A2**: Check the development tool environment folder. Verify the correct path is given for the FX2LP *.inc* files.

Go to **Project** > **Components, Environment and Books…** Click on the **Folders/Extensions** tab. Verify all folder settings.


Components, Environment and Books

Project Components | Folders/Extensions | Books

Development Tool Folders

☐ Use Settings from TOOLS.INI:

Tool Base Folder: C:\Keil\C51\
BIN : C:\Keil\C51\BIN\
INC: C:\Cypress\USB\CY3684_EZ-USB_FX2LP_DVK\1.1\Tar
LIB: C:\Keil\C51\LIB\
Regfile:

Default File Extensions:

C Source: *.c
C++ Source: *.cpp
Asm Source: *.s*; *.src; *.a*
Object: *.obj
Library: *.lib
Document: *.txt; *.h; *.inc

Select Folder for 3rd Party Development Tools

☐ Use GNU Tools
Cygnus Folder:
Keil Root Folder:

☐ Use ARM Tools
RealView Folder:
Keil Root Folder:

**Figure 18** **Folder Settings**

Enter the fields as below:

- BIN Folder: *C:\Keil\C51\BIN\*
- INC Folder: *C:\Cypress\USB\CY3684_EZ-USB_FX2LP_DVK\1.1\Target\Inc\;C:\Keil\C51\INC\*
- LIB Folder: *C:\Keil\C51\LIB\*

**Q3**: I have experience using IAR IDEs for embedded applications. Can I use an IAR Embedded Workbench instead of Keil µVision?!

**A3**: Yes, you can use another IDE besides Keil but Cypress cannot offer support for or guarantee properly working firmware compiled by this IDE or other development tools. Therefore, we recommend using the Keil IDE for our products.

**Q4**: Where can I get more information about the Keil IDE, especially about the C51 compiler?

**A4**: You can find additional answers at this location*: C:\Keil\C51\HLP*. There are several documents that can help your work with the Keil IDE. There is also a document that describes the C51, error codes and their definitions, debug commands, and other useful information.

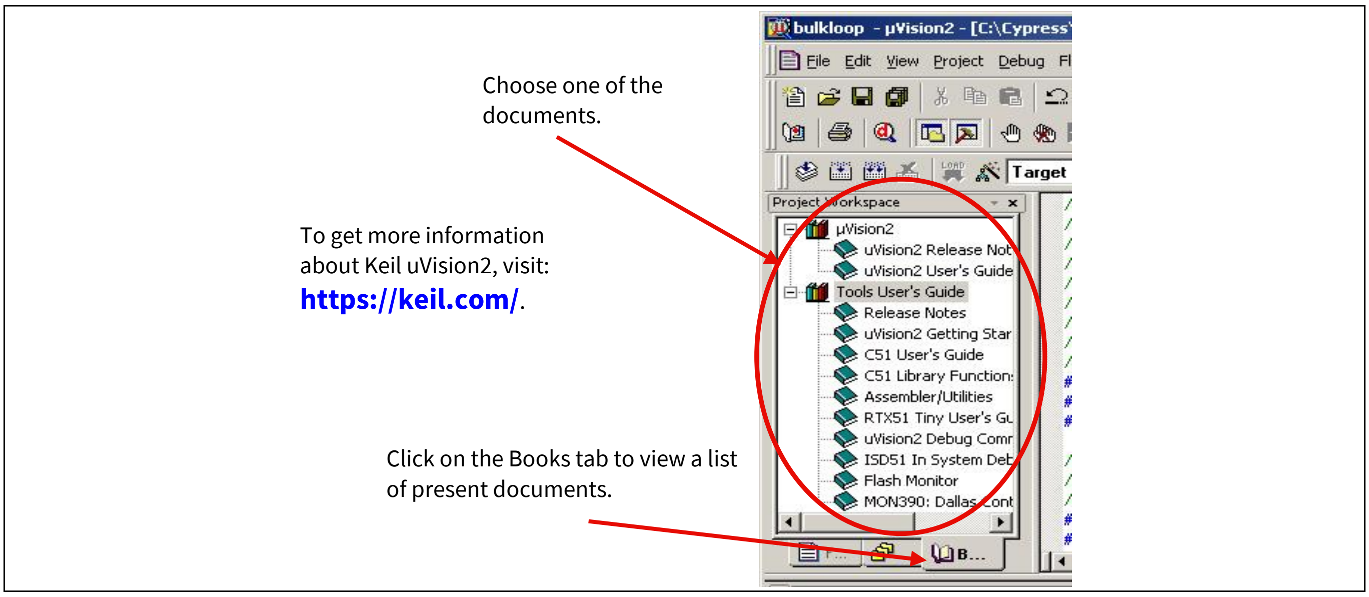


**Figure 19** **Resource Access**

**Q5**: Can the Keil monitor be placed in Read Only Memory?

**A5**: No, Cypress does not have a configuration of the monitor that can be placed into Read Only Memory.

**Q6**: The Keil debugger will not start. How do you make it work?

**A6**: The setup of the Keil debugger requires an RS-232 cable. First, determine what port number you are going to use on the computer (or later do it by trial and error if you cannot determine the port number you are using). Second, ensure the monitor has been downloaded to the development board. If it is in the default configuration, it loads when the board is plugged into the bus. To determine if it has been loaded, unplug the board and plug it in. Did the green LED on the development board light up? If it did, the monitor loaded. If it did not, open the USB Interface dialog box and select the "Load Mon" icon. The green LED should then light.

Once the monitor is loaded, open the Keil tool and select the debug option and select **Start**. If you have the evaluation copy, it will be reported at this time and you should select **OK**.

The system should load the program and start the debugger at this time. If it does not, then the communication across the RS-232 did not occur. Select the setting options ensure all boxes are checked. Set the baud rate to 38400. Click **OK** and retry the connection.

If it still fails, most likely the COM port is not set correctly. Go back to the setting options and select a different port and retry the connection. Do this until you try each of the ports or find the right port.

**Q7**: Can I use a USB-to-Serial adapter to use the debug function of the CY3684 development board?

**A7**: Most USB-to-Serial adapters work perfect but there are some low cost adapters that do not work with Keil monitors. You can read about this at the Keil web site **https://www.keil.com/** Technical Support Knowledgebase by typing "General: USB INTERFACE FOR KEIL MONITOR" in search.

**Q8**: What resources does the Keil monitor use? I need to know so I can avoid using 8051 resources used by the Keil monitor, to avoid having my program crash the monitor, or the monitor crash my program.

**A8**: There are several monitors available that use external or internal memory and use different part of memory.

Regarding the selected Keil monitor file, Keil monitor uses:

- Specific memory and specific part of memory
- Specific serial port

**Q9**: I have purchased the **CY3684 EZ-USB® FX2LP Development Kit**. I have followed all initial steps of installing the board. The problem is that the FX2LP development board is not being detected when I start USB Control Center (see **Figure 20**).

**A9**: This problem occurs because you have bound the device to the wrong driver or you did not bind to any driver. The USB Control Center works only with CyUsb.sys. Update the drivers to your board and specify CyUsb.sys driver.

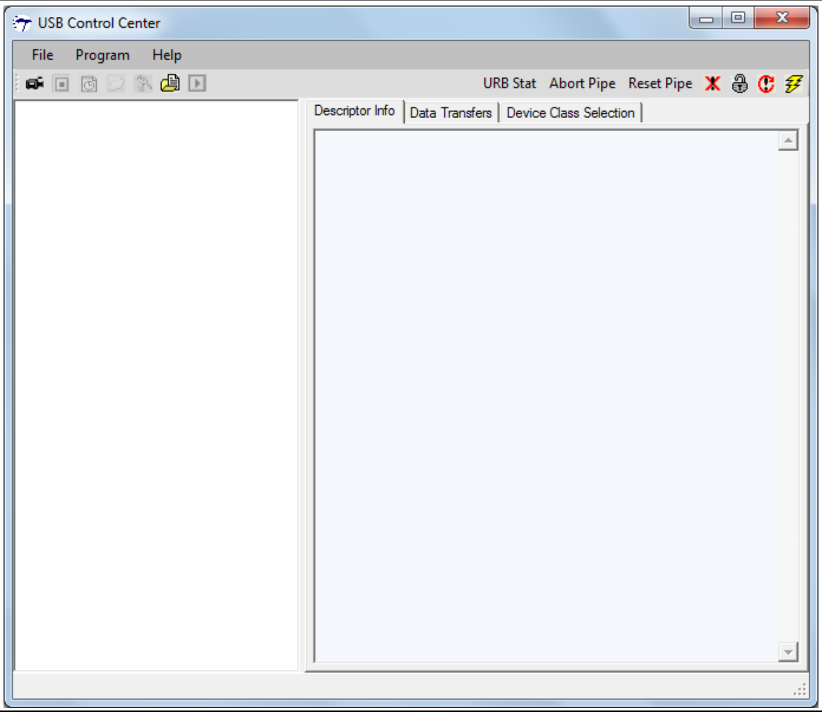


**Figure 20** No Device Detected

# 5 Debug FX1/FX2 Using Internal Monitor Code

In this section, we will give more resources to let you use Keil C easily when you are designing and debugging.

The following is the example of how to debug FX1/FX2™ using internal monitor code. Download the *AN42499_Example.zip* and unzip it to run the example. Follow these steps to run it.

## 5.1 Choosing the Monitor File

Cypress provides four monitor files for debugging:

- *mon-ext-sio0-c0.hex* – connects to SIO-0, loads to external memory
- *mon-ext-sio1-c0.hex* – connects to SIO-1, loads to external memory
- *mon-int-sio0.hex* - connects to SIO-0, loads to internal memory
- *mon-int-sio1.hex* - connects to SIO-1, loads to internal memory

For debugging using internal monitor code, '*mon-int-sio0.hex*' and '*mon-int-sio1.hex*' are used for Serial port 0 and Serial port 1 respectively. On installing the CY3684 kit, these are available in the folder 'Monitor'. The *.M51* files and the *Readme* file related to the Monitor code are available in the folder 'Monitor' attached along with this application note. After installing the CY3684, go to:

*C:\Cypress\USB\CY3684_EZ-USB_FX2LP_DVK\1.1\Target\Monitor*

## 5.2 Specifying the Code and XData Range

All the steps given in this application note hold good while using the internal monitor too. However, the only exception is in specifying the code range and the Xdata range. The main intention of specifying the right code range and the Xdata range is to ensure that the code you wish to debug and the monitor code do not overlap.

The monitor code resides in the memory range '0x0000 to 0x1075', as given in the *Readme* file. The Xdata of the monitor code resides in memory '0x1100 to 0x11CE' (this can be verified from the .M51 files of the monitor code). Note that this setting is the same for both 'mon-int-sio0' and 'mon-int-sio1'.

Now, you need to find the code and Xdata requirements of the firmware that you wish to debug. After the project is built, this information is displayed in the output window, as for example:

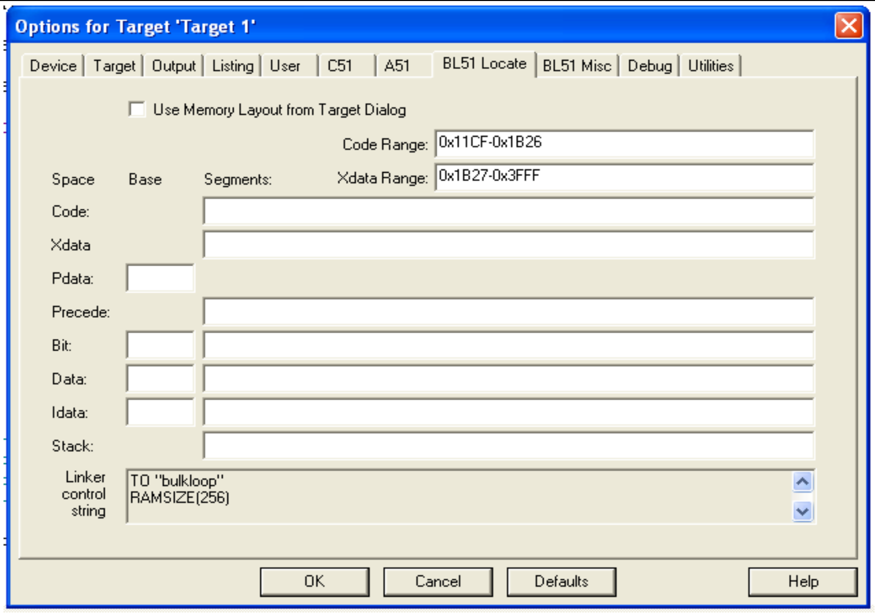
Program Size: data = 45 Xdata = 4400 code = 2300

The same can also be found in the respective *.M51* file of the project.

For the example project used here, you have the memory requirement as:

Program Size: data = 45.5 Xdata = 4473 code = 2391

Since the code and the Xdata of the monitor occupies till 0x11CE, the code to be debugged can be placed from 0x11CF. Since the code requires 2391 bytes, the code range should be minimum: 0x11CF–0x1B26. Further, the Xdata can be placed after this memory location; that is, starting from 0x1B27–0x3FFF. See **Figure 21** to know how to setup.


Options for Target 'Target 1'
Device | Target | Output | Listing | User | C51 | A51 | BL51 Locate | BL51 Misc | Debug | Utilities
OK
Cancel
Defaults
Help

Use Memory Layout from Target Dialog: unchecked

Code Range: 0x11CF-0x1B26

Xdata Range: 0x1B27-0x3FFF

Space, Base, Segments: Code, Xdata, Pdata, Precede, Bit, Data, Idata, Stack (empty)

Linker control string:

```
TO "bulkloop"
RAMSIZE(256)
```

Figure 21 Code and Xdata Range Setup

These are the major changes to be made from the settings given in the application note; all other steps can be followed as given in the application note.

## 5.3 Testing the Project

The example project to be debugged can be found in the folder ‘Debug_Internal_Monitor’.

This project is based on the ‘Bulkloop’ example available as part of the CY36l84 development kit. It uses EP2 and EP4 as Bulk OUT endpoints and EP6 and EP8 as Bulk In Endpoints and does a loop back between the pair of endpoints EP2, EP6 and EP4, EP8.

When you start debugging, a yellow arrow is seen indicating the program counter location. Later, depending on the requirement, you can set breakpoints at specific locations or monitor the various registers, among others.

# 6 Hardware Connections

The project is tested on the CY3684 development kit. The schematic of this can be found in the folder 'Hardware'.

*Note:* *Though the interrupt vector table resides at 0x00 to 0x80, the monitor code does not overlap it, as it only modifies the interrupt vector table at 0x0023 or 0x003B, depending on whether serial port 0 or 1 is used and retains everything else as it is in this code space.*

*Note:* *The monitor code uses two resources: One of the Serial ports and Timer 1. Timer 1 is used for generating the serial clock. Make certain not to use these resources in the project to avoid conflict.*

*Note:* *As long as the device is in the Debug mode, it is bound to the Keil Monitor-51 driver, which is specified in the Debug settings (refer to **Figure 6**).*

If your code size exceeds the 64 KB, you may choose code banking method to avoid the restriction. You can refer another application note - **AN58170 - Code/Memory Banking Using EZ-USB®** to learn more about it.

## 7 Summary

This application note mainly provides a brief introduction for using the KEIL Debugger Environment while you are using an FX2LP product design in your application. After you go through this application note, you will be able to understand how to download monitor firmware into FX2LP to enable the debugger function and also setup the UART Baud Rate correctly to link up with FX2LP silicon to start using the debugger feature. This application note gives basic information to guide new users of FX2LP to start debugging their application. Advanced users may need to consult with **KEIL** for detailed information or you can use the Cypress MyCase (login in at **www.cypress.com** as a partner or a customer and click “Create a MyCase” in the **Customer Service Support** tab).

# 8 Application Notes and Reference Designs

## 8.1 Application Notes

- **AN65209- Getting started with FX2LP™**

  AN65209 introduces you to the EZ-USB FX2LP USB 2.0 device controller. This application note helps you build a project for FX2LP and explore its various development tools, and then guides you to the appropriate documentation to accelerate in-depth learning about FX2LP.

- **AN58764 - Implementing a Virtual COM Port in FX2LP™**

  This application note explains how to implement a virtual COM port device using the standard Windows driver in FX2LP. This information helps to migrate from UART to USB.

- **AN58009- Serial (UART) Port Debugging of EZ-USB® FX1/FX2LP™ Firmware**

  AN58009 describes the code to be added to EZ-USB FX2LP firmware for serial port debugging. This code enables you to print debug messages using the UART in the HyperTerminal program on a Windows computer or to capture them in a file.

- **AN15456 - Guide to Successful EZ-USB® FX2LP™ Hardware Design**

  This application note identifies possible USB hardware design issues, especially when operating at high-speed. It also facilitates the process of catching potential problems before building a board and assists in the debugging when getting a board up and running.

- **AN50963 - EZ-USB® FX1™/FX2LP™ Boot Options**

  This application note discusses the various methods to download firmware in to FX1/FX2LP.

- **AN66806 - Getting Started with EZ-USB® FX2LP™ GPIF**

  This application note describes the steps necessary to develop GPIF waveforms using the GPIF Designer.

- **AN61345 - Designing with EZ-USB® FX2LP™ Slave FIFO Interface**

  This application note provides a sample project to interface an FX2LP with an FPGA. The interface implements Hi-Speed USB connectivity for FPGA-based applications such as data acquisition, industrial control and monitoring, and image processing. FX2LP acts in Slave-FIFO mode and the FPGA acts as the master. This application note also gives a sample FX2LP firmware for Slave-FIFO implementation and a sample VHDL and Verilog project for FPGA implementation.

- **AN57322 - Interfacing SRAM with FX2LP™ over GPIF**

  This application note discusses how to connect the Cypress CY7C1399B SRAM to FX2LP using the General Programmable Interface (GPIF). It describes how to create read and write waveforms using GPIF Designer. This application note is also useful as a reference to connect FX2LP to other SRAMs.

- **AN4053 - Streaming Data through Isochronous/Bulk Endpoints on EZ-USBR FX2 and EZUSB FX2LP™**

  This application note provides background information for a streaming application using the EZ-USB FX2 or the EZ-USB FX2LP part. It provides information on streaming data through BULK endpoints, ISOCHRONOUS endpoints, and high bandwidth ISOCHRONOUS endpoints along with design issues to consider when using the FX2/FX2LP in high-bandwidth applications.

- **AN58170 - Code/Memory Banking Using EZ-USB**

  The EZ-USBFX2 family of chips contains an 8051 core. The 8051 core has 16-bit address lines and is able to access 64KB of memory. However, some applications require more than 64KB. This application note describes methods of overcoming this 64KB boundary.

- **AN1193 - Using Timer Interrupt in Cypress EZ-USB FX2LP™ Based Applications**

  This application note helps EZ-USBR FX2LP firmware developers to use timer interrupts in their applications.

- **AN63787 - EZ-USB® FX2LP™ GPIF and Slave FIFO Configuration Examples using 8-bit Asynchronous Interface**

  This application note discusses how to configure the General Programmable Interface (GPIF) and slave FIFOs in EZ-USB FX2LP in both manual mode and auto mode to implement an 8-bit asynchronous parallel interface. This application note is tested with two FX2LP development kits connected back-to-back; the first one operating in master mode and the second operating in slave mode.

- **AN61244 - Firmware Optimization in EZ-USB**

  This application note describes firmware optimization methods in EZ-USB. Some of these methods are common to any processor and some are specific to the 8051 core of EZ-USB FX2LP.

- **AN45471 - Create Your Own USB Vendor Commands Using FX2LP™**

  This application note demonstrates how to code USB vendor commands to perform specific product. In addition, the note explains how to use the CyConsole utility to issue vendor commands.

## 8.2 Reference Designs

Several reference designs of FX2LP for popular applications are available. The reference designs include demonstration source code, reference schematics, and a BOM, where appropriate, for the design.

The reference designs available on the Cypress website are:

- **CY4661 - External USB Hard Disk Drives (HDD) with Fingerprint Authentication Security**

  The CY4661 reference design kit from Cypress and UPEK provides customers with a turnkey solution for an external USB hard disk drive (HDD), with fingerprint authentication, and security to protect and authenticate data. The reference design uses UPEK's Touch Strip Fingerprint Authentication Solution (TCS3 swipe fingerprint sensor and TCD42 security ASIC).

- **FX2LP™ DMB-T/H TV Dongle reference design**

  This reference design kit is based on FX2LP and Legend Silicon's chipset. A captured and demodulated RF signal converted to an MPEG2 TS stream by the Legend Silicon chipset is sent to the PC through an FX2LP.The PC plays these streams using a media player. This is a complete design, including all required files.

## Revision history

| Document version | Date of release | Description of changes |
|---|---|---|
| ** | 2007-11-14 | New application note. |
| *A | 2011-01-25 | Added quick reference to each figure in all instances across the document.<br>Updated Introduction:<br>Updated hyperlink of CY3684.<br>Updated SuiteUSB version to 3.4.<br>Updated Download of the Debug Monitor File:<br>Updated Manual Download:<br>Changed the external memory to external SRAM memory of the DVK. Added the extra link to FX2LP TRM for more external memory usage introduction.<br>Updated Debug FX1/FX2™ Using External Monitor Code:<br>Updated Debugging Session:<br>Added additional information on debugging session with KB articles and another app note index link.<br>Added “Debug FX1/FX2™ Using Internal Monitor Code”.<br>Added “Hardware Connections”.<br>Completing Sunset Review. |
| *B | 2011-04-28 | Minor change:<br>Corrected document number in footer. |
| *C | 2014-02-10 | Updated to new template.<br>Completing Sunset Review. |
| *D | 2016-03-29 | Added Related Resources reference in page 1.<br>Updated Abstract:<br>Updated description.<br>Updated Introduction:<br>Updated description.<br>Updated hyperlinks.<br>Updated Download of the Debug Monitor File:<br>Updated Manual Download:<br>Updated Figure 1.<br>Updated Figure 2.<br>Updated Figure 3.<br>Updated Troubleshooting the Keil Debugger:<br>Updated Figure 20.<br>Added “Application Notes and Reference Designs”. |
| *E | 2017-02-21 | Updated to new template.<br>Completing Sunset Review. |
| *F | 2017-04-20 | Updated Cypress Logo and Copyright. |

| Document version | Date of release | Description of changes |
|---|---|---|
| *G | 2018-03-13 | Updated Document Title to read as “AN42499 - Debugging EZ-USB FX1/FX2LP Firmware Using the Keil Debugger Environment”.<br>Updated Debug FX1/FX2™ Using External Monitor Code:<br>Updated Keil Debug Monitor, Debug Settings:<br>Updated Figure 8.<br>Updated Troubleshooting the Keil Debugger:<br>Updated Figure 18.<br>Updated Debug FX1/FX2™ Using Internal Monitor Code:<br>Updated Choosing the Monitor File:<br>Updated description.<br>Updated Application Notes and Reference Designs:<br>Updated Application Notes:<br>Updated titles of AN45471, AN61345, AN66806, and AN15456.<br>Updated hyperlink of AN58009.<br>Completing Sunset Review. |
| *H | 2019-03-28 | Updated Application Notes and Reference Designs:<br>Updated Application Notes:<br>Removed AN74505 (as application note is obsolete).<br>Completing Sunset Review. |
| *I | 2020-04-13 | Updated Application Notes and Reference Designs:<br>Updated Application Notes:<br>Removed AN58069 (as application note is obsolete).<br>Completing Sunset Review. |
| *J | 2021-04-08 | Updated Debug FX1/FX2™ Using External Monitor Code:<br>Updated Debugging session:<br>Updated hyperlinks.<br>Updated Hardware Connections:<br>Updated hyperlinks.<br>Updated Summary:<br>Updated hyperlinks.<br>Updated Application Notes and Reference Designs:<br>Updated Application Notes:<br>Updated hyperlinks.<br>Updated Reference Designs:<br>Updated hyperlinks.<br>Updated to Infineon template.<br>Completing Sunset Review. |

**Edition 2021-04-08**

**Published by**

**Infineon Technologies AG**

**81726 Munich, Germany**

**Do you have a question about this document?**

**Go to [www.cypress.com/support](www.cypress.com/support)**

**Document reference**

**001-42499 Rev. *J**