AT43DK370 USB Host/Function Development Kit

User Guide for Revision 1.3



Table of Contents

Section 1

Introduction 1-1

1.1 Development Kit Features 1-1

Section 2

Getting Started 2-1

2.1 Electrostatic Warning 2-1
2.2 Unpacking the System 2-1
2.3 System Requirements 2-1
2.4 Connecting the Hardware 2-2
2.5 Installing AT43DK370 and Starting Up the USB Clinic 2-2
2.6 Testing the Hardware 2-3

Section 3

Hardware Description 3-1

3.1 Jumper Settings 3-2

Section 4

USB Clinic 4-1

4.1 Main Menu 4-1
4.2 Output Window 4-2
4.2.1 Command/Status Window 4-2
4.2.2 User Program Output Window 4-2
4.3 Connection 4-2
4.3.1 Test Connectivity 4-3
4.3.2 Check Device Enumerated 4-3
4.4 Download Code 4-3
4.5 Memory 4-5
4.5.1 Fill Memory 4-5
4.5.2 Read Memory 4-5
4.6 Enumeration 4-5
4.6.1 Get Device Descriptor 4-6
4.6.2 Get Configuration Descriptor 4-6
4.6.3 Get String Descriptor 4-7
4.6.4 Set Configuration 4-7
4.6.5 Set Interface 4-7
4.7 Data Transfer 4-8

4.7.1 Get ISO Data 4-8
4.7.2 Send ISO Data 4-8
4.7.3 Get Data 4-8
4.7.4 Send Data 4-8
4.7.5 Control Transfer 4-9
4.7.6 Custom Transfer 4-9
4.8 Port Features 4-10
4.8.1 Set Port Feature 4-10
4.8.2 Clear Port Feature 4-10
4.9 Device State Control 4-11
4.9.1 Reset Device 4-11
4.9.2 Suspend Device 4-11
4.9.3 Resume Device 4-11
4.10 Miscellaneous Notes 4-11

Section 5

Building Firmware for the AT43DK370 Development Kit 5-1
5.1 Sample Directory and File Structure 5-1
5.1.1 USBP ARM Project Guide 5-2
5.1.2 “Make” Project 5-4
5.2 ADS Settings 5-5
5.3 Modifying a Sample Application 5-8

Section 6

Converting Between FLASH and ICE Mode 6-1
6.1 Introduction 6-1
6.2 Converting to Flash Mode from ICE Mode 6-1
6.3 Converting to Flash Mode from Download Mode 6-3
6.4 Converting to Download Mode from ICE Mode 6-3
6.5 Converting to Download Mode from Flash Mode 6-4
6.6 Converting to ICE Mode from Flash Mode 6-4
6.7 Converting to ICE Mode from Download Mode 6-4
6.8 Summary 6-4

Section 7

Generating Hex Files for Flash Mode in the AT43USB370 Development Board with ADS 7-1
7.1 Introduction 7-1
7.2 Procedure 7-1

Section 8

Technical Support 8-1

Section 9

Appendices 9-1

9.1 AT43USB370 Bill of Materials (BOM) 9-1

9.2 AT43DK370 Schematics 9-3

Section 1

Introduction

Congratulations on your purchase of the AT43DK370 Development Kit. The AT43DK370 is a complete starter kit and development system for the AT43USB370 Host/Function Processor. It is designed to allow real-time firmware development and evaluation of the AT43USB370 USB Host/Function Processor.

1.1 Development Kit Features

The AT43DK370 development kit consists of the following features:

- AT43USB370 USB Host/Function Processor Reference Design Board featuring
  - ARM System Processor
  - 1 MBytes Flash ROM
  - 2 MBytes Static RAM
  - USB Type A/B Ports
  - RS-232 Serial Port
  - MII Expansion Connector
  - Generic 8-bit Expansion Connector
  - 20-pin JTAG interface connector
  - Reset Button
  - Power Indicator LEDs
  - In-system Firmware Programming Capability
- USB Firmware Library including USB Host Stack, User Class Device Drivers and High Level APIs in C
- USB Clinic In-circuit Emulation Tool

The latest version of the USB Clinic can be found in the USB section of the Atmel web site at http://www.atmel.com/ad/plugplayhost. Please refer to the same section for up-to-date information on new USB software, documentation releases and tool upgrades.

Section 2

Getting Started

2.1 Electrostatic Warning

The AT43DK370 Development Board is shipped in protective anti-static packaging. The board must not be subjected to high electrostatic potentials. A grounding strap or similar protective device should be worn when handling the board. Avoid touching the component pins or any other metallic elements.

2.2 Unpacking the System

The development kit is supplied with the following:

- AT43DK370 Reference Design Board
- Female-Female DB9 Null-modem Cable
- 2m Fully Rated USB Cable
- 5V Regulated Power Supply
- Atmel USB CD-ROM with Software and Documentation
- Atmel Products CD-ROM

Please contact your local Atmel distribution or E-mail usb@atmel.com if any of the aforementioned items is missing from the package.

2.3 System Requirements

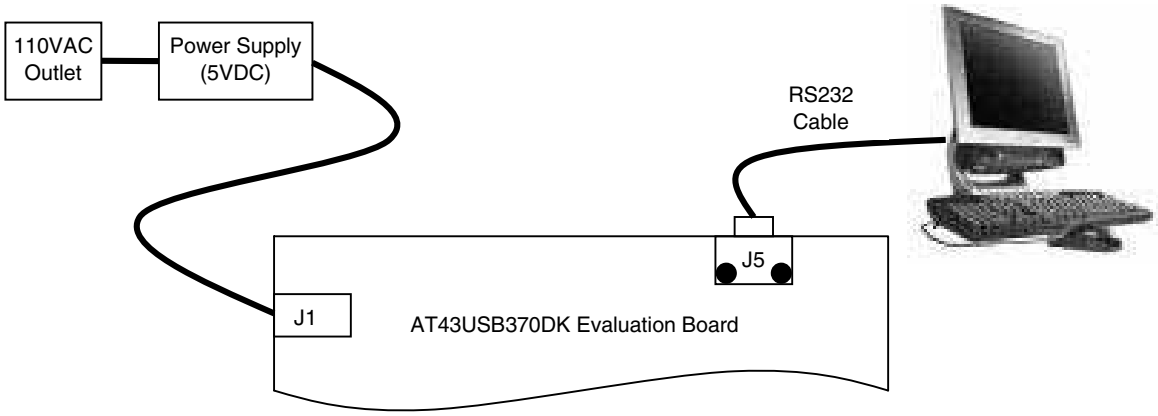
The minimum hardware and software requirements are:

- 486 Processor (Pentium® is recommended)
- 128 MB RAM
- 10 MB Free Hard Disk Space
- Windows® 98/2000/ME/XP
- RS-232 Port (COM port)

2.4 Connecting the Hardware

Atmel has taken great care in creating a reliable demonstration kit for its customers. In order to ensure proper operation, the supplied components in the kit must be used in the setup as shown in Figure 2-1. Atmel does NOT recommend substitution of these components.

Figure 2-1. Connection to the AT43DK370



Connect the AT43DK370 evaluation board as follows:

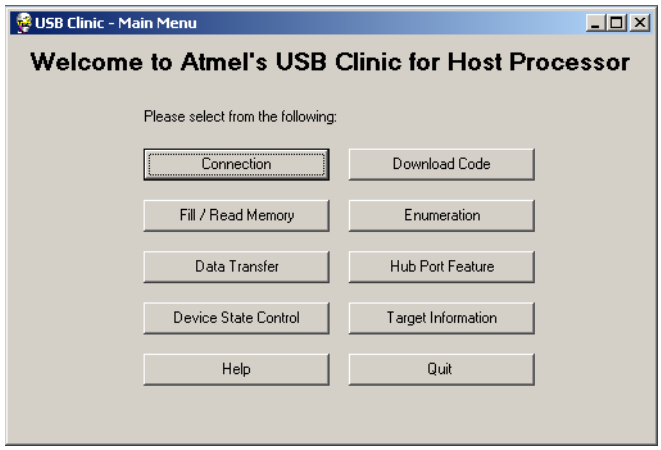
1. Connect the serial cable from J5 on the evaluation board to a COM port on the PC.
2. Connect the AC input connector of the power supply to an AC wall outlet (110 VAC).
3. Connect the DC output connector of the power supply to J1 on the evaluation board.

2.5 Installing AT43DK370 and Starting Up the USB Clinic

To install the AT43DK370:

1. Insert the “Atmel AT43DK370” CD into the CD-ROM drive of the PC or notebook and follow the instructions.
2. After installation, the AT43USB370 documents, firmware, and software should be installed on the **C:\Program Files\ATMEL USB\AT43DK370** , if the default installation directory is used. One of the software programs installed is USB Clinic. It is an integrated diagnostic and debugging tool that provides communication between the AT43DK370 and the PC. Please refer to Section 4 of this User's Guide for detailed description of this tool.
3. To invoke the USB Clinic program from desktop go to **Start** > **Programs** > **Atmel USB Clinic** > **Atmel USB Clinic**. The screen in Figure 2-2-will appear.

Figure 2-2. USB Clinic - Main Menu



2.6 Testing the Hardware

From the USB Clinic **Main Menu** click **Connection (**Figure 2-3 and Figure 2-4 will appear**)**. The **Connection** button allows testing of the RS-232 serial port and the AT43DK370 development board connection. Prior to going to the **Connection** menu, please make sure that the AT43DK370 is physically connected to the PC with the supplied serial cable or any standard serial cable.

Figure 2-3. USB Clinic - Connection

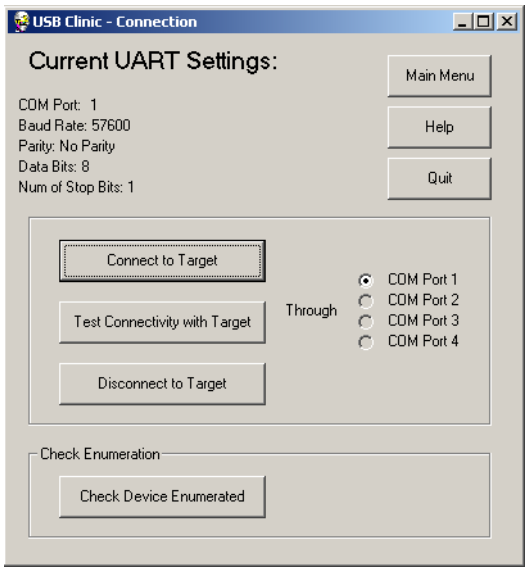
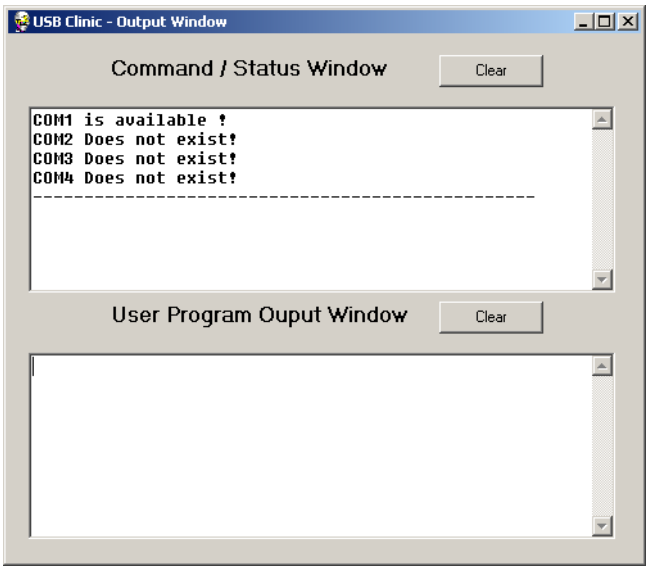
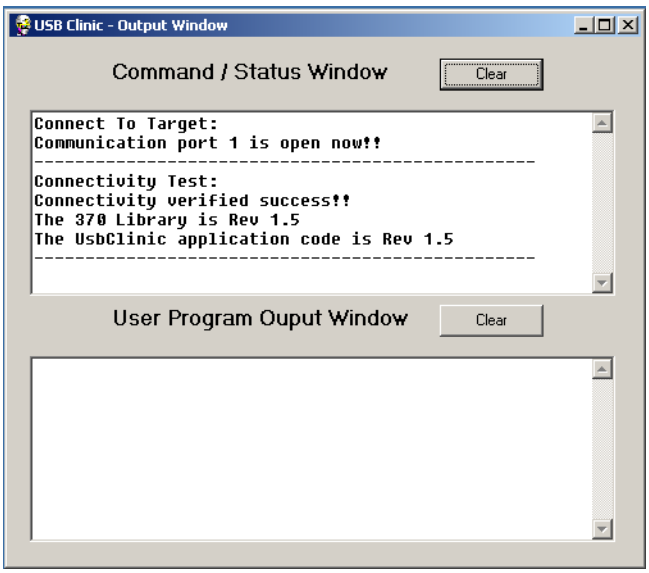


Figure 2-4. COM Port Availability



The USB Clinic automatically detects and displays the availability of COM ports from COM1 to COM4 in the **Connection** window. Users can use any available COM port by simply selecting the desired COM port and then clicking on the **Connect to Target** button to open the selected COM port. To verify connectivity between the selected COM port and the AT43DK370 development board, click on **Test Connectivity with Target** and look for *Connection verified* in the **Output Window** (see Figure 2-5). The **Test Connectivity with Target** also checks the AT43DK370 firmware revision to see if it supports the current USB Clinic version.

Figure 2-5. Connection Verified Output Window



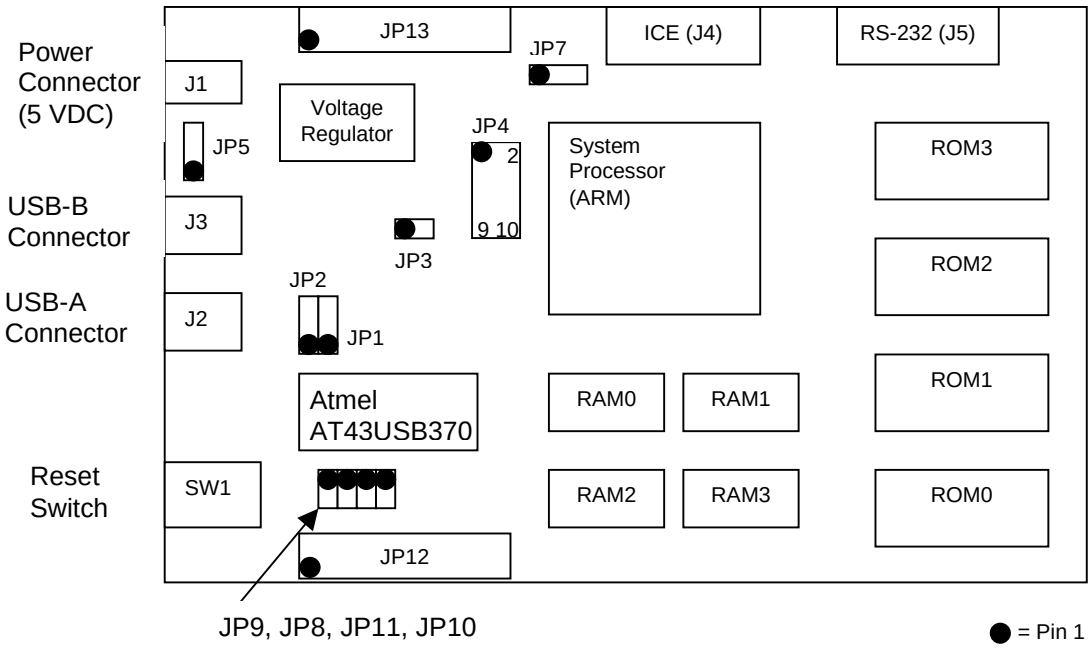
Once the connection is established, USB Clinic is ready for use. Please refer to Section 4 for more details.

Section 3

Hardware Description

The AT43DK370 development board comes pre-configured as a USB Host in StandAlone (Flash) mode. Alternative settings are available through jumper configurations. Complete descriptions of the jumper settings are explained in Section 3.1.

Figure 3-1. AT43USB370 Development Board Components Layout



The AT43DK370 Development Board consists of an ARM® processor with 32-bit external bus, the AT43USB370 Host/Function Processor, 2 Mbytes of flash ROM, 2 Mbytes of SRAM, USB type A/B ports and RS-232 serial port. The AT43USB370 connects to the ARM processor through its 32-bit generic host interface. The RS-232 port provides access to the ARM processor and the AT43USB370, and the in-system programming to SRAM.

Section 8 of this User's Guide contains complete AT43DK370 BOM and schematics.

3.1 Jumper Settings

The AT43DK370 Development Board supports two modes of operation, the default Standalone (or Flash) Mode and the ICE (In-Circuit Emulator) mode. In Standalone Mode, the AT43DK370 Development Board executes code from its on board flash ROM. While in ICE Mode, an In-Circuit Emulator for the ARM processor can be connected through J4 to further facilitate code development. The corresponding jumper settings for the different modes are shown in Table 3-1.

Table 3-1. Development Board Environment

	Standalone Mode (Default)	**ICE Mode**
JP4 (nRCS<4> is 1)	Close: 3-5, 7-9, 2-4	Close: 1-3, 5-7, 2-4
JP7	Close: 1-2	Close: 2-3

The AT43USB370 is a dual role processor that can be configured either as a full/low speed host or a full speed device. Its exact personality is defined by the jumper settings shown in Table 3-2. Please be aware that the AT43SUB370 can operate either in the host mode or in the device mode, but not simultaneously.

Table 3-2. Development Board Configuration

	USB Host (Default)	**USB Function FS**
JP5	Close: 2-3	Close: 1-2
JP8	Open	Open
JP9	Close	Open
JP10	Open	Close
JP11	Close	Open

Table 3-3 contains the jumper settings for the clock source. In normal operations, a 6 MHz crystal clock source is used. For debugging purposes, an external 48 MHz oscillator can be used as the clock source to the AT43USB370 by bypassing its internal PLL.

Table 3-3. AT43USB370 Clock Source

	Internal PLL with 6 MHz Crystal (Default)	**External 48 MHz Oscillator**
JP1	Close: 1-2	Close: 2-3
JP2	Close: 1-2	Close: 2-3

JP3 was used in DK 1.0 for Revision A of the AT43USB370's minor erratum relating to DMA acknowledgement. This erratum is resolved with minor re-works of the AT43DK370 Development Board and leaving the JP3 jumper open. Both the re-works and the JP3 setting are done at the factory.

DK 1.1, or Revision B of the AT43USB370 requires no such workaround. However, to simplify user usage, JP3 can still be left open, the way it is from the factory setting.

JP12 and JP13 are currently unused and are reserved for future expansion.

Section 4

USB Clinic

USB Clinic is a Graphical User Interface (GUI) based diagnostic and debugging tool that provides basic control of the AT43USB370 USB Host/Function Processor via the RS-232 serial port. Its main feature set includes user firmware download/execution, direct read/write access of the AT43USB370 internal memory, manipulation of USB device enumeration, data transfer through various endpoints, hub feature selection, and USB device state control. All of the AT43USB370 Library APIs can be called through USB Clinic. This section explains the capability and the usage of USB Clinic in detail.

Please note since the function calls used in USB Clinic allows the flexibility of user inputs, it is the user's responsibility to ensure the inputs are bound to the USB Specification 2.0 in order to obtain the correct response from the USB Clinic and the AT43USB370 Library.

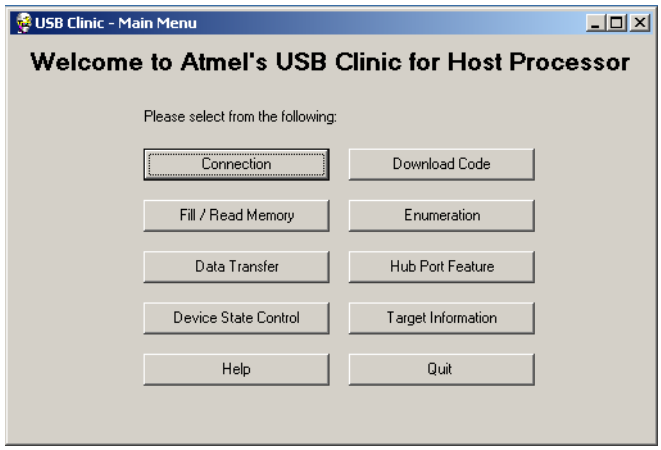
Atmel will continue to extend the capability of the USB Clinic. The following section is intended for USB Clinic Rev. 1.2. For software upgrades, please refer to the USB section of the Atmel web site at http://www.atmel.com/ad/plugplayhost.

4.1 Main Menu

Once properly installed, USB Clinic can be invoked by double-clicking on the **USB Clinic** icon, or if default installation options are used, from the Windows' **Startup** menu by selecting **Start** > **Programs** > **USB Clinic**.

The **Main Menu** will appear on screen (see Figure)

Table 4-1. USB Clinic - Main Menu



Each button in the **Main Menu** represents a category of functions. USB Clinic Rev. 1.2 currently supports seven categories of functions: Connection, Download Code, Fill/Read Memory, Enumeration, Data Transfer, Hub Port Feature, and Device State Control. The the last four categories, functions are tied with the AT43USB370 Library APIs directly. For detailed usage regarding those function calls, please refer to the “AT43USB370 Software Development Guide for Host Mode”.

4.2 Output Window

Executing any function button will bring up the **Output Window**. There are two sections in the **Output Window**: the **Command/Status Window** and the **User Program Output Window**.

4.2.1 Command/Status Window

The **Command/Status Window** displays all messages and outputs resulting from the execution of USB Clinic commands. Click the **Clear** button to clear the content of the window see Figure 4-2).

4.2.2 User Program Output Window

The **User Program Output Window** displays text messages and outputs from the user developed firmware. In the user firmware source code, users will need to insert ***\n*** at the end of every message printing statement to ensure that messages are properly displayed in the **User Program Output Window**. Clicking the **Clear** button clears the content of the window (see Figure 4-2).

4.3 Connection

From the **Main Menu**, the **Connection** button allows testing of the RS-232 serial port and the AT43DK370 Development Board connection. Prior to go to the **Connection** menu, please make sure that the AT43DK370 is physically connected to the PC with the supplied serial cable.

Click on the **Connection** button and the windows in Figure 4-1 and Figure 4-2 appear.

Figure 4-1. USB Clinic - Connection Window

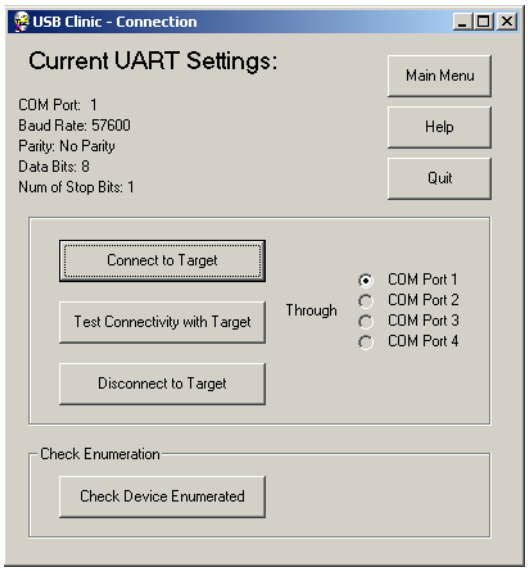
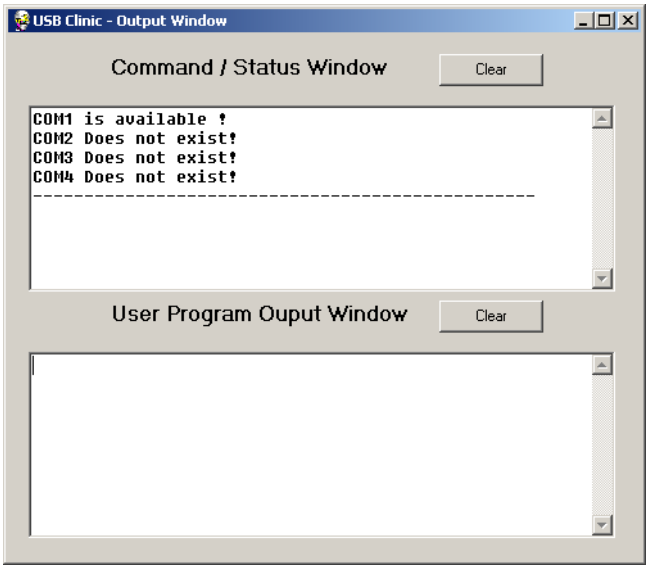


Figure 4-2. USB Clinic - Output Window



4.3.1 Test Connectivity

The USB Clinic automatically detects and displays the availability of COM ports from COM1 to COM4 in the **Connection** window. Users can use any available COM port by simply selecting the desire COM port and then clicking on the **Connect to Target** button to open the selected COM port. To verify connectivity between the selected COM port and the AT43DK370 Development Board, click on **Test Connectivity with Target** and look for **Connection verified** in the **Output Window**. The **Test Connectivity with Target** also checks the AT43DK370 firmware revision and its USB Clinic application code revision.

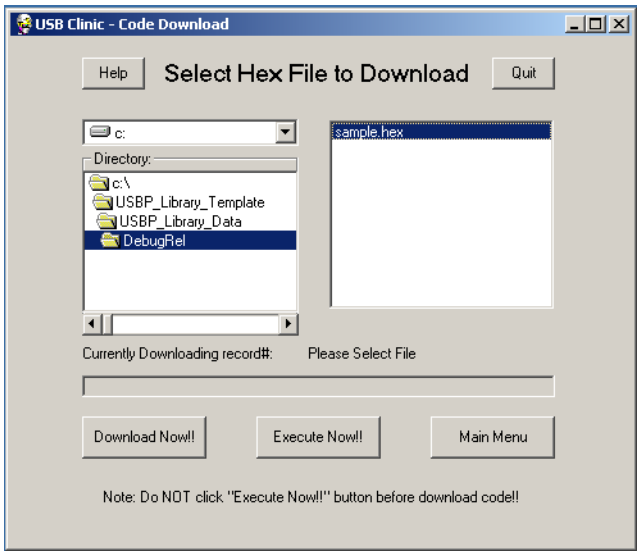
4.3.2 Check Device Enumerated

The **Check Device Enumerated** command allows the user to see if the connected target USB devices are enumerated or not. It is done by checking if the device addresses have been assigned to the target USB devices. AT43USB370 can host up to 7 USB devices, so the device address ranges from 1 to 7, and is assigned by the enumeration timing order in which a device is connected to the AT43USB370 Host.

4.4 Download Code

The **Download Code** button allows downloading user developed firmware onto the system processor's program memory space for debug and testing. Please be aware that this button is NOT for downloading AT43DK370 firmware. The AT43DK370 boots off from the system processor and its firmware is stored in an external flash attached to the system processor.

Figure 4-3. USB Clinic - Download/Execute Code

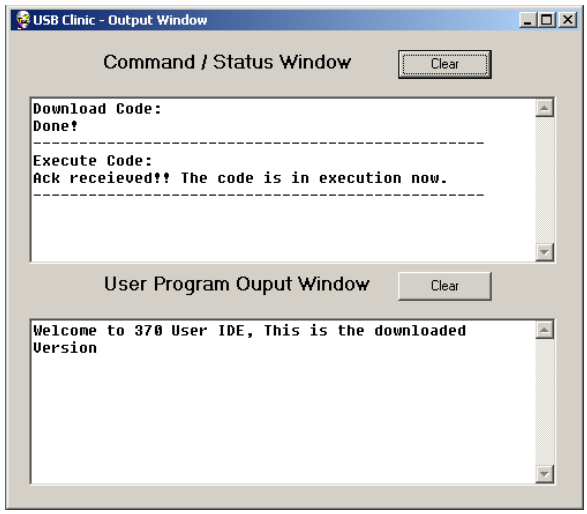


The ARM code requires 32-bit Intel Hex File Format with the entry point at address 0x108000. Select the appropriate ARM firmware in the file selection window first and then double-click on the file or click on the **Download Now!!** to start the downloading process. The progress bar and the associated messages indicate the code download status. A sample download Hex file is included in the CD at the **C:\Program Files\Atmel USB\AT43DK370\Firmware** directory.

Once code download is complete, the user has the option of downloading the firmware again without executing the code or executing the downloaded firmware by clicking on **Execute Now**. Resetting the AT43DK370 Development Board will not erase the firmware from the ARM's memory space.

The **Download Now!!** command executes the downloaded firmware. The user should only click on **Download Now!!** after code download is complete. During execution, the only way to return control back to USB Clinic is to press the reset button on the AT43USB370 Development Board or to power-cycle the board. The **User Program Output Window** displays the test output of the user developed firmware during execution as shown in Figure 4-4.

Figure 4-4. USB Clinic - User Program Output Window



4.5 Memory

There are two types of memory commands are available, **Fill Memory** and **Read Memory**.

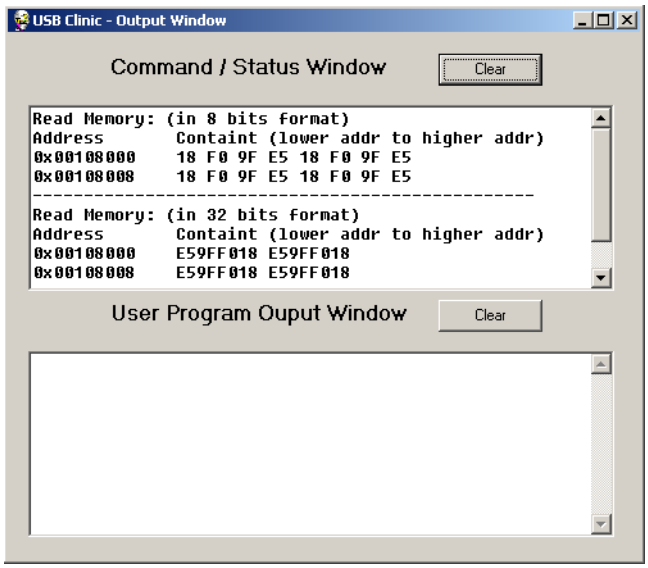
4.5.1 Fill Memory

The **Fill Memory** command allows the user to write to the memory location of the target device with the desired data size in Little Endian format. The user will need to select the desired data size (8 bits, 16 bits, or 32 bits), then enter the number of data size to fill (4 digits decimal value), starting address (Hex value), and the pattern (Hex value) to be written to the target address.

4.5.2 Read Memory

The **Read Memory** command allows the user to read to the desired memory location of the target device in the Little Endian format. The user will need to select the data size to be read (8 bits, 16 bits, or 32 bits), and then enter the starting address (Hex value) to read from the target. The output displays a maximum of 8 address values (for 8 bytes or 4 words or 2 long words) of Hex value per line, and it displays from lower address to higher address (left to right, and top to bottom). Please note that the available memory address in the AT43DK370 Development Board ranges up to 0x3FFFFFFF. The user will not be able to read or write to memory addresses beyond this limit.

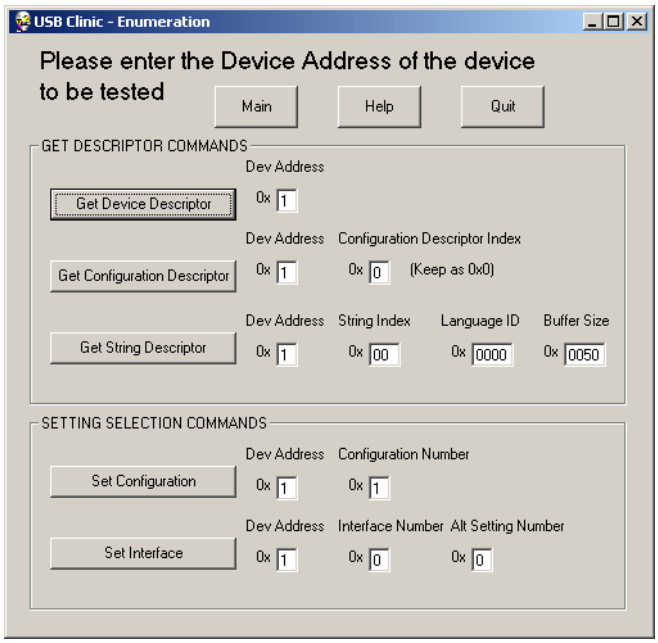
Figure 4-5. USB Clinic - Read Memory



4.6 Enumeration

The **Enumeration** functions allow for retrieving and potentially altering the USB device interface setting upon successful device connection and enumeration. There are five **Enumeration** functions, as shown in Figure 4-6, they are the **Get Device Descriptor**, **Get Configuration Descriptor**, **Get String Descriptor**, **Set Configuration**, and **Set Interface**.

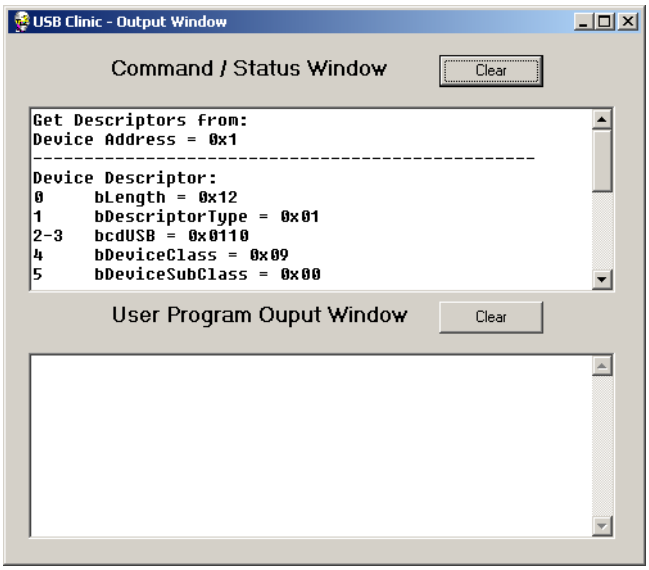
Figure 4-6. USB Clinic Enumeration



4.6.1 Get Device Descriptor

The **Get Device Descriptor** command allows the user to get the device descriptor of a target USB device. A target USB device is defined as one of the USB devices connected to the AT43DK370 development board. To retrieve the target device's descriptor, the user needs to assign its device address as input. The device address ranges from 1 to 7, and is assigned by the enumeration timing order in which a device is connected to the AT43DK370 Host.

Figure 4-7. USB Clinic - Get Device Descriptor



4.6.2 Get Configuration Descriptor

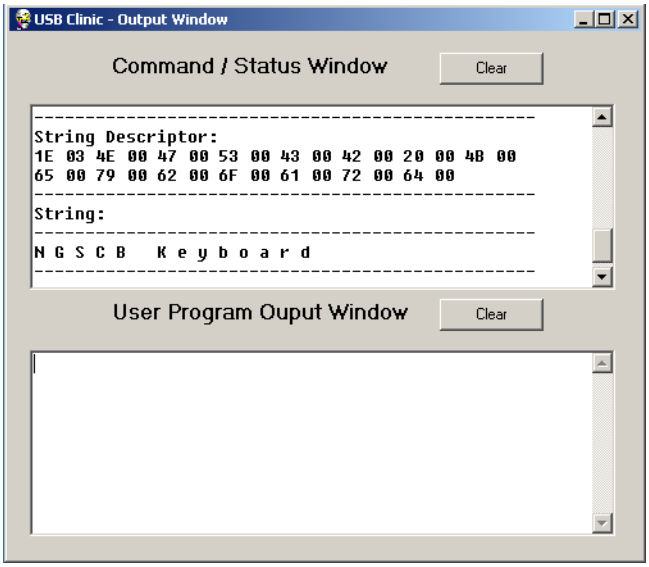
The **Get Configuration Descriptor** command allows the user to get the configuration descriptor of the first configuration from the target device. The current FW does not support multiple configurations. To get the configuration descriptor, the user will need to assign the right device address and the configuration index as input. Configuration Descriptor Index (CDIndex) zero always retrieves the first configuration descriptor,

regardless its configuration number, so the CDIndex should be kept at zero. The device address ranges from 1 to 7, and it is assigned by the enumeration timing order in which a device is connected to the AT43DK370 Host.

4.6.3 Get String Descriptor

The **Get String Descriptor** command allows the user to obtain the string descriptors. To obtain a string descriptor, the user first needs to determine the string index. The string index can be obtained from device, configuration, or interface descriptors. If string index 0x00 is entered, regardless of the Language ID, the **Output Window** will display the list of Language ID that the device supports. The USB Clinic Rev. 1.2 decodes only the United States English (Language ID 0x0409) string. All other Language IDs will display only the string descriptor without the properly decoded string.

Figure 4-8. USB Clinic - Get String Descriptor



4.6.4 Set Configuration

The **Set Configuration** command allows the user to set the device to a particular configuration. To set the configuration, the user will need to assign the device address and the configuration number associated with the target device as inputs. The configuration number can be obtained from the *bConfigurationValue* field of the configuration descriptors. Since the AT43USB370 Library does not support multiple configuration settings, the **Set Configuration** command is used mainly in the enumeration stage only to set the device from Address stage to Configured state. The enumeration is done by the AT43DK370 firmware upon device connection, therefore the user will not need to execute this command through the USB Clinic manually. Please refer to “USB Processor Library Software Development Guide for Host Mode’ and “USB Specification Revision 2.0” for details.

4.6.5 Set Interface

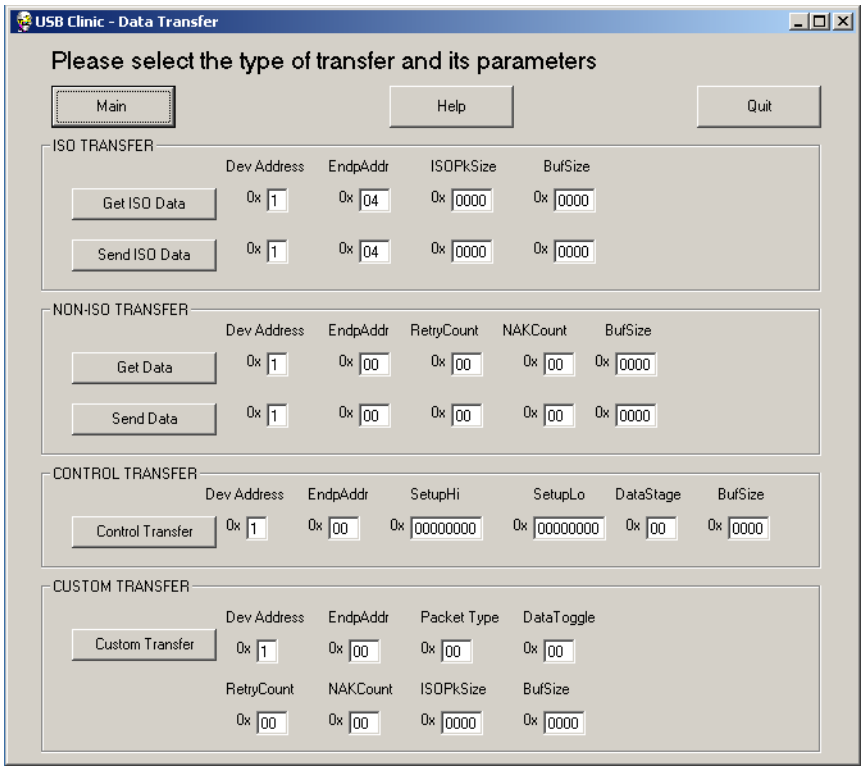
For configurations with multiple interfaces/alternate settings, the **Set Interface** command allows the user to set the particular interface/alternate interface setting for the configuration on the target device. To set the right interface, the user will need to assign the device address, the interface number, and the alternate number associated with the target device as inputs.

The interface number or the alternate setting number can be obtained from the *bInterfaceNumber* field and the *bAlternateSetting* field of the interface descriptor (that is returned followed by the configuration descriptor) respectively.

4.7 Data Transfer

The **Data Transfer** functions contain all types of data exchange methods between the devices and the Host. The functions includes Get/Send ISO Data (for Isochronous Transfer), Get/Send Data (for Interrupt, Bulk, and Control Transfer), Control Transfer (for Control Transfer), and Custom Transfer (for All Transfer Types).

Figure 4-9. USB Clinic - Data Transfer



4.7.1 Get ISO Data

The **Get ISO Data** command gets data from a USB device from its Isochronous (ISO) endpoints. The user has to enter the Device Address of the device with the proper ISO Endpoint, Packet Size, and the Buffer Size allocated for this operation.

4.7.2 Send ISO Data

The **Send ISO Data** command sends data to a USB device to its Isochronous (ISO) endpoints. The user has to enter the Device Address of the device with the proper ISO Endpoint, Packet Size, and the Buffer Size allocated for this operation.

4.7.3 Get Data

The **Get Data** command gets data from a USB device from the Non-Isochronous endpoints, such as Interrupt, Bulk and Control. The user has to enter the Device Address of the device, the Endpoint intended, the number of Retries allowed, the number of NAKs allowed, and the Buffer Size allocated for the operation.

4.7.4 Send Data

The **Send Data** command sends data to a USB device to the Non-Isochronous endpoints, such as Interrupt, Bulk and Control. The user has to enter the Device Address of the device, the Endpoint intended, the number of Retries allowed, the number of NAKs allowed, and the Buffer Size allocated for the operation.

4.7.5 Control Transfer

The **Control** command performs transfers to the control endpoint of the device. The user has to enter the Device Address of the device, the Control Endpoint, the least significant 4 bytes of the 8-byte setup data in the Setup Hi field, the most significant 4 bytes of the 8-byte setup data in Setup Lo field, the Data Stage field, and the Buffer Size allocated for this operation. For more information using this command, please refer to the section “ControlTransfer() API” in the “AT43USB370 Software Development Guide for Host Mode”.

Table 4-2. Data Stage

Data Stage	Value	Description
DATA_STAGE_NULL	0x00	Setup stage will be followed by the status stage.
DATA_STAGE_IN	0x01	Data stage following the setup stage will be in the IN direction. The data will be received from the device.
DATA_STAGE_OUT	0x02	Data stage following the setup stage will be in the OUT direction. The data will be sent to the device.

4.7.6 Custom Transfer

The **Custom Transfer** command allows users to customize their own transfer type. It supports all transfer types endpoints (Isochronous, Interrupt, Bulk and Control). The user has to enter the Device Address of the device, the Endpoint intended, the Packet Type (see Table 4-3), Data Toggle (see Table 4-4), the number of Retries allowed, the number of NAKs allowed, the ISO Packet Size if ISO endpoint is used, and the Buffer Size allocated for this operation. For more details on how to use the **Custom Transfer** command, please refer to the section “USBP_H_CustomTransfer() API” of the “AT43USB370 Software Development Guide for Host Mode”.

Table 4-3. Packet Types

Packet Type	Value
PACKET_OUT	0x00
PACKET_IN	0x01
PACKET_SETUP	0x02

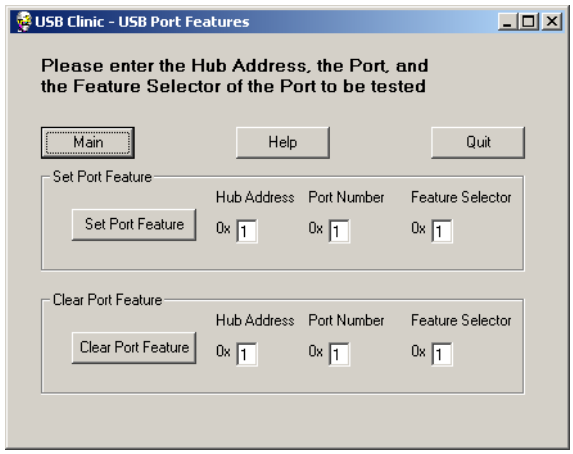
Table 4-4. Data Toggle

Data Toggle	Value
DATA_TOGGLE_0	0x00
DATA_TOGGLE_1	0x01

4.8 Port Features

The **Port Features** functions enable or disable a particular feature on the selected hub port. The functions are **Set Port Feature**, and **Clear Port Feature**.

Figure 4-10. USB Clinic - Port Features



4.8.1 Set Port Feature

The **Set Port Feature** command is used to enable a particular feature on the selected hub port. The user has to input the hub Device Address, the Port, and the Feature (see Table 4-4) to be enabled. For more details, please refer to Chapter 11 of the “USB Specification Rev 2.0”, and to the “AT43USB370 Software Development Guide for Host Mode”.

4.8.2 Clear Port Feature

The **Clear Port Feature** command is used to disable a particular feature on the selected hub port. The user has to input the hub Device Address, the Port, and the featUre (see Table 4-5) to be enabled. For more details, please refer to Chapter 11 of the “USB Specification Rev 2.0”, and to the “AT43USB370 Software Development Guide for Host Mode”.

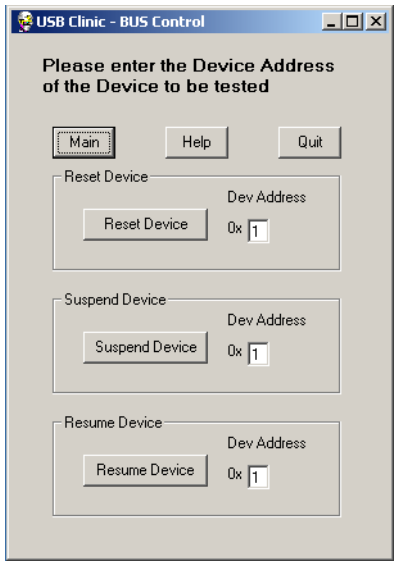
Table 4-5. Feature Selector Values

Feature Selector	Value
PORT_CONNECTION	0
PORT_ENABLE	1
PORT_SUSPEND	2
PORT_OVER_CURRENT	3
PORT_RESET	4
PORT_POWER	8
PORT_LOW_SPEED	9
C_PORT_CONNECTION	16
C_PORT_ENABLE	17
C_PORT_SUSPEND	18
C_PORT_OVER_CURRENT	19
C_PORT_RESET	20
PORT_TEST	21
PORT_INDICATOR	22

4.9 Device State Control

The **Device State Control** allows the Host to control the states of the downstream devices. The functions available are **Reset Device**, **Suspend Device**, and **Resume Device**.

Figure 4-11. USB Clinic - Device State Control



4.9.1 Reset Device

The **Reset Device** function resets the state of the selected device to the default state. The device is selected by the Device Address entered.

4.9.2 Suspend Device

The **Suspend Device** command allows the Host to force the selected device to go to suspend. If the device has Remote Wakeup capability (bit 5 of *bmAttributes* field of the **Configuration Descriptor** is set) and **DEVICE_REMOTE_WAKEUP** (0x1) feature is enabled by the Host, it can Wakeup itself from suspended state. Otherwise, the device can only be woken up using the **Resume Device** command described below. The device is selected by the Device Address entered. The device address ranges from 1 to 7, and is assigned by the enumeration timing order in which a device is connected to the AT43DK370 Host. For details about how to set the **DEVICE_REMOTE_WAKEUP** feature, please refer to the “USB Specification Rev 2.0”, and to the “AT43USB370 Software Development Guide for Host Mode”.

4.9.3 Resume Device

The **Resume Device** command allows the Host to wake up the selected device under suspend. The device is selected by the Device Address entered. The device address ranges from 1 to 7, and is assigned by the enumeration timing order in which a device is connected to the AT43DK370 Host.

4.10 Miscellaneous Notes

- The **Target Information** lists information regarding the USB Clinic version compatibility and the DK hardware. It does not contain any function calls.
- Wherever there is the text **0x** before any input prompt, that input prompt is expecting a Hex value (0 to 9, A/a to F/f) as an input.
- After a hardware power-cycle or reset (sometimes, depending on the reset condition), the user needs to click on **Connect to Target** to establish the serial

connection between the AT43DK370 Development Board and the PC. This serial connection is NOT active until the **Connect to Target** button is clicked.

– There is a 1000-character limitation on a single transmission. The user will need to break the text to fit the limitation.

– There is a **Help** button on every window that describes how to use the functions in every window.

Section 5

Building Firmware for the AT43DK370 Development Kit

Developing firmware for the AT43DK370 requires an ARM development tool, of the user's choice, that can build ARM code in Intel 32-bit Hex File Format. This tutorial illustrates how to build the firmware using a template application and the ARM® Developer Suite™ version 1.2 (ADS) from ARM Limited.

There are three modes of firmware: Flash mode, ICE mode, and Download mode. The Flash mode firmware generates an Intel 32-bit Hex file that is split into 4 to be programmed onto the AT43DK370 board's flash sets. For information about generating the Flash mode firmware, please refer to Section 7. The ICE mode firmware is used by the In-Circuit Emulator that runs code on the AT43DK370 board's SRAM. The Download mode firmware also generates an Intel 32-bit Hex file that is used by the USB Clinic that downloads the firmware to run on the AT43DK370 board's SRAM. The sample template included in the DK is the Download mode version. For details on how to convert between the firmware modes, please refer to Section 6.

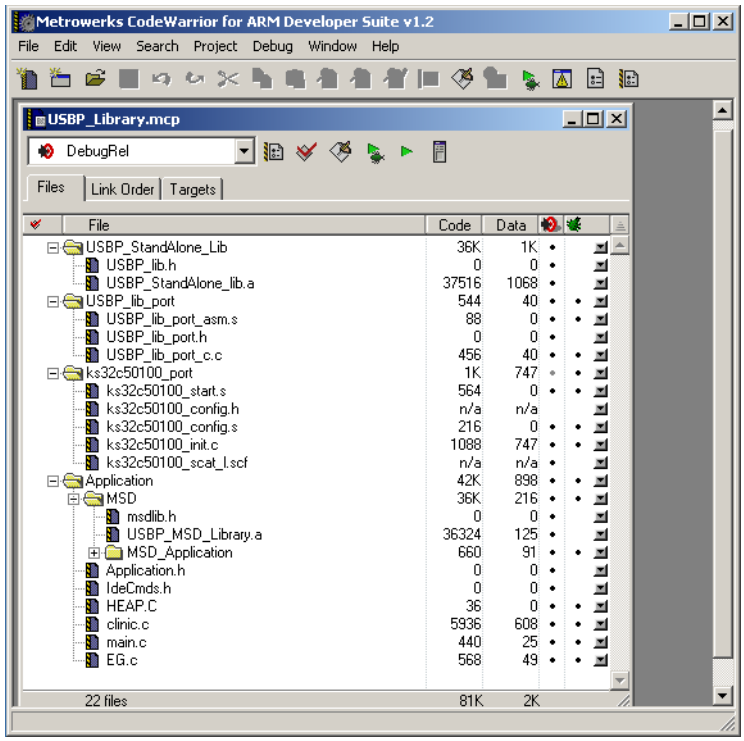
5.1 Sample Directory and File Structure

The default sample code directory for the **USBP_Library_Rev_x.x_Template** resides in **C:\Program Files\Atmel USB\AT43DK370\Firmware**.

Note: This User Guide uses the **USBP_Library_Rev_x.x_Template** as an example. The actual library revision number might be updated.

From the ARM Developer Suite (ADS), go to **File** > **Open** > **C:\Program Files\Atmel USB\AT43DK370\Firmware** > **USBP_Library_Rev_x.x_Template**. Select and open the sample project file named **USBP_Library.mcp** as shown in Figure 5-1.

Figure 5-1. USBP_Library.mcp with Metrowerks® CodeWarrior™ for ADS v1.2



The project file contains the build information for the sample source code and the required ARM library.

5.1.1 USBP ARM Project Guide

5.1.1.1 Overview

The USB Processor's ARM Project is presented in the **USBP_Library_Rev_x.x_Template** folder. This directory contains the project file: **USBP_Library.mcp**.

USBP_Library.mcp is the main project file of ARM Code. It includes the **USBP_StandAlone_lib.a** library file, processor-specific code, library porting code, application library and the application code.

5.1.1.2 Directory Structure

This directory structure of the project is described below:

```
USBP_Library
|
|-- USBP_StandAlone_lib
    |
    |-- USBP_lib.h
    |-- USBP_StandAlone_lib.a
|
|-- USBP_lib_port
    |--  USBP_lib_port_h.h
```

```
    |--  USBP_lib_port_asm.s
    |-- USBP_lib_port_c.c
|
|-- ks32c50100_port
    |--  ks32c50100_start.s
    |--  ks32c50100_config.h
    |--  ks32c50100_config.s
    |--  ks32c50100_init.c
    |--  ks32c50100_scat_l.scf
|
|-- Application
    |
    |-- MSD
          |-- msdlib.h
          |-- USBP_MSD_Library.a
          |-- MSD_Application
    |-- Application.h
    |-- IdeCmds.h
    |-- HEAP.c
    |-- Clinic.c
    |-- main.c
    |-- Eg.c
```

A brief description of the folders and files is given below.

1. USBP_Library

This is the main directory of the project. It contains the main project file (USBP_Library.mcp) and all the project folders. The project is built with ADS (ARM Development Suite) Version 1.2

2. USBP_StandAlone_Lib

The **USBP_StandAlone_lib.a** is the binary library that contains the AT43USB370's firmware and all the high/low level APIs. The **USBP_lib.h** is the header file used for the stand alone library.

3. **USBP_lib_port**

This folder contains the processor-specific port required to be implemented by the User and integrated into the USBP library. It contains the following files.

a. **USBP_lib_port.h**: This is the header file for the USBP Library port. It contains various definitions required for the USB Processor's configuration.

b. **USBP_lib_port_asm.s**: This file contains the processor-specific assembly C port required by the USBP Library.

c. **USBP_lib_port_c.c**: This file contains the processor-specific C port required by the USBP Library.

4. ks32c50100_port

This folder contains the port for the sample target processor. i.e. ks32c50100. It contains the following files.

a. **ks32c50100_start.s**: This file contains the startup code for the ks32c50100 processor.

b. **ks32c50100_config.h**: This file contains ks32c50100 processor definitions.

c. **ks32c50100_config.s**: This file contains the system manager initialization routine for the ks32c50100 processor.

d. **ks32c50100_init.c**: This file contains various peripheral initialization functions for the ks32c50100 processor.

e. **ks32c50100_scat_l.scf**: This is the scatter loading file. It defines the various mapped regions of the DK board. It specifies the address ranges for Read Only (Flash), Read/Write (SRAM), and the Internal SRAM area.

5. **Application**

   This folder contains the Application (System Processor Software) code for the project. It contains the following files:

   a. **MSD:** This directory contains the **USB_MSD_Library.a**, **msdlib.h,** and the **MSD_Application** code. The **USB_MSD_Library.a** is the binary library for the MSD driver. It contains the MSD APIs for applications. The **msdlib.h** is the header files used for the MSD library. The MSD_Application directory is for MSD application code. Currently this directory contains all the required routines for MSD applications, but it does not contain a sample application code.

   b. **Application.h**: This is application's header file.

   c. **HEAP.c**: This file contains the heap management code for the application.

   d. **Clinic.c**: This file contains all the source routines applicable to the USB Clinic.

   e. **IdeCmds.h**: This file contains the protocol used between the USB Clinic application code and the PC USB Clinic.

   f. **main.c**: This file contains the entry point of the application code. The **main()** routine is defined in this file.

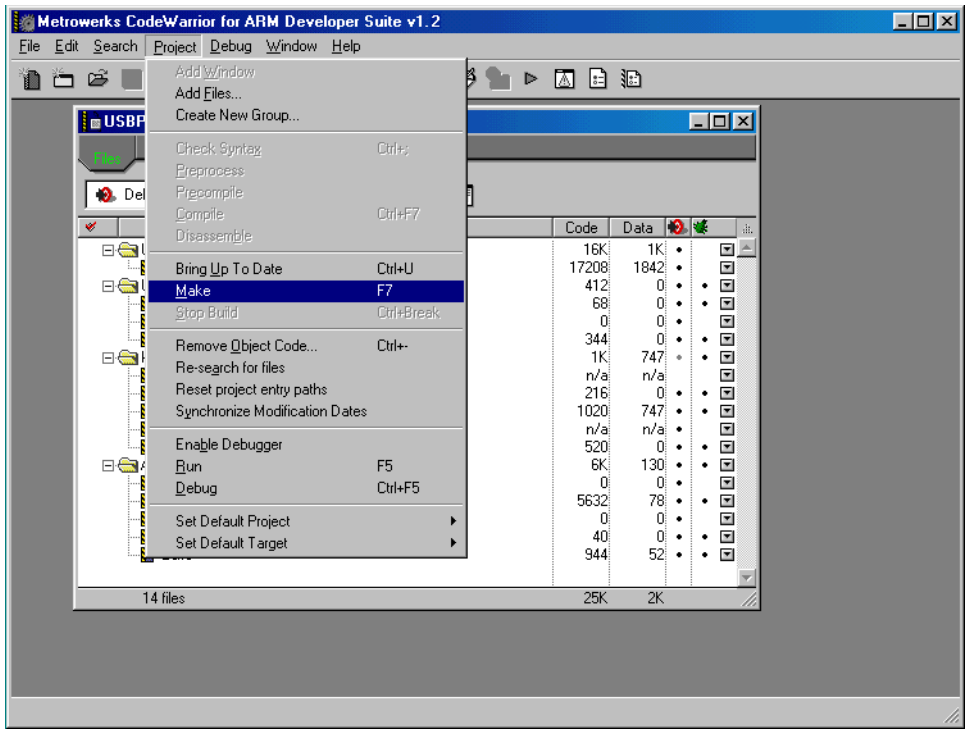
   g. **EG.c**: This file contains the application code.

Other than the directories shown in the **USBP_Library.mcp**, there is another directory called **USBP_Library_Data** that is created by the project. This directory contains the project image, objects and output files. Typically those files are located in the **DebugRel** sub-directory.

5.1.2 "Make" Project

To build a project, go to **Project** > **Make** as shown in Figure 5-2. To ensure a clean build, remove the existing object files first by going to **Project** > **Remove Object Code** and then rebuild the project by selecting **Project** > **Make**.

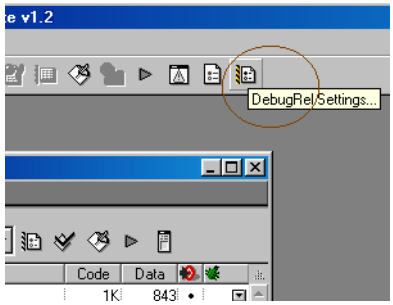
Figure 5-2. Project Make



5.2 ADS Settings

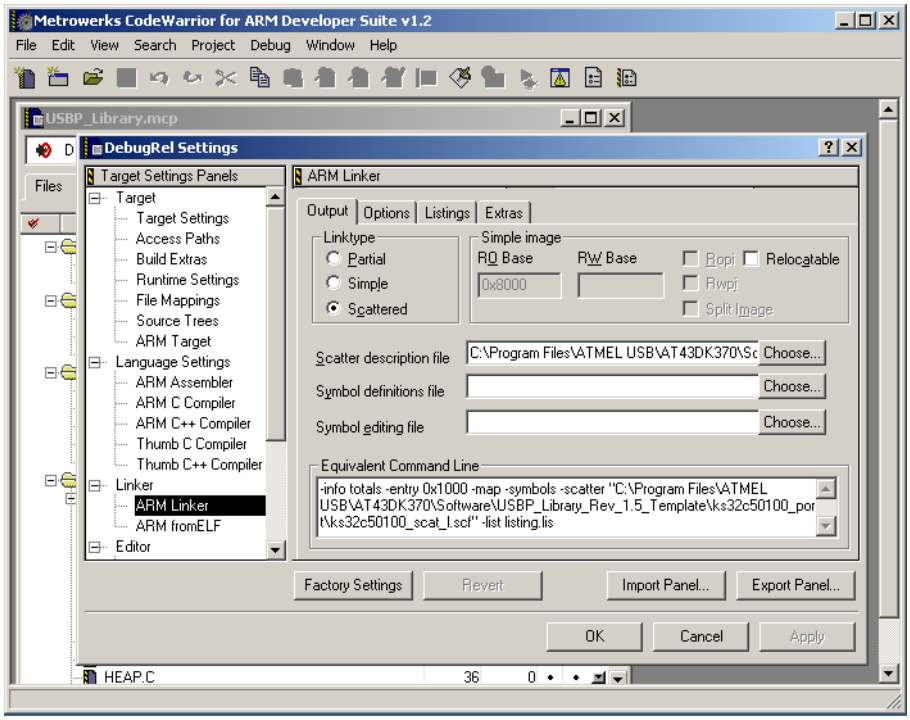
To Make a project, the ADS requires to know the format to compile code and how to link its object. Those information are stored in a scatter file. To assign a scatter file, click on the **DebugRel Setting** icon on the top right corner of the ADS IDE as shown in Figure 5-3.

Figure 5-3. DebugRel Setting Icon



The following window appears on the screen.

Figure 5-4. DebugRel Settings Window



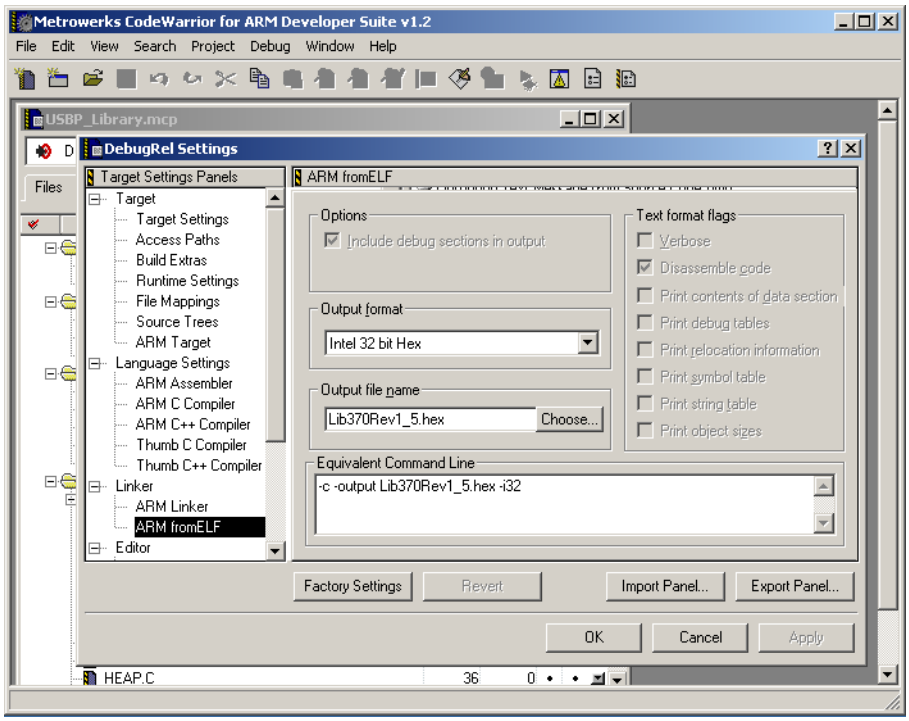
The left-hand side of the **DebugRel Settings** window contains available options including Target, Language Settings and Linker. Go to **Linker** > **ARM Linker** > **Output**. There is a **Scatter description file** prompt in the **ARM Linker** window. Click on **Choose**, and go to **USBP_Library_Template** > **ks32c50100_port** directory and select the **ks32c50100_scat_l file** to set the proper Scatter file path.

The **DebugRel Settings** window is also important for other settings such as, first, the setting for generating **Intel 32 bit Hex** format files for USB Clinic download executables. To generate the correct file format, the ADS configuration must be properly set.

In the **DebugRel Settings** window, check for **ARMfromELF** under **Linker**. If **ARMfromELF** does not exist, go to **Target** > **Target Settings**. Under the **Target Settings** window, go to **Post Linker** and select **ARMfromELF**. The **ARMfromELF** should then appear under **Linker**. Go to **Linker** > **ARMfromELF**.

In the **ARMfromELF** window, go to the **Output format** prompt and set the Hex output format to **Intel 32 bit Hex**. In the same window, the user can also specify the output name by typing it in the **Output file name** prompt as shown in Figure 5-5.

Figure 5-5. Output Format/Name Selection



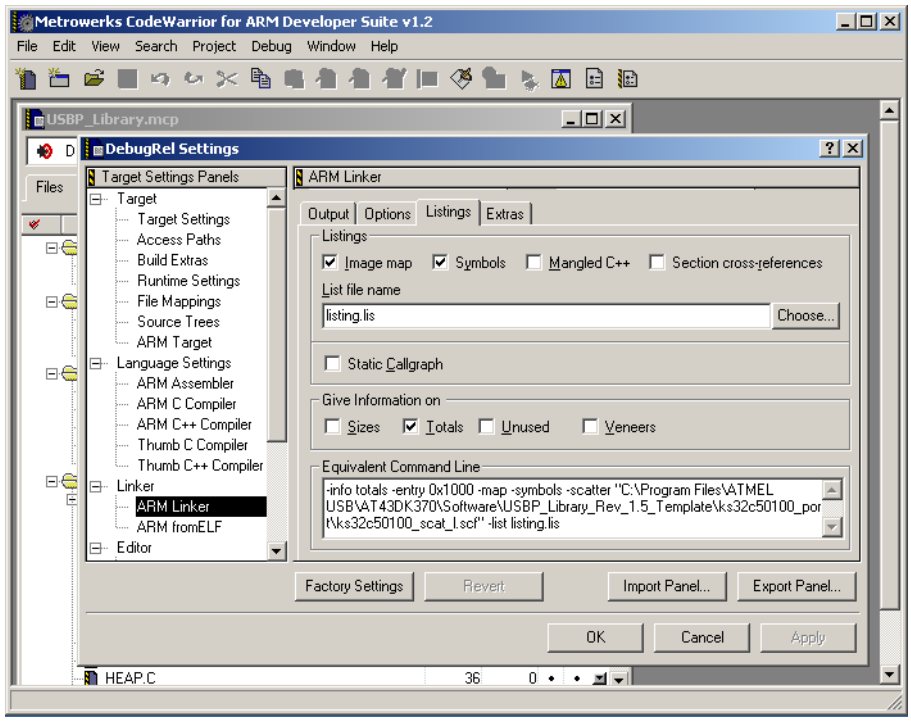
Secondly, for the ICE mode, the firmware entry point needs to be 0x1000. Go to **Linker** > **ARM Linker** > **Options** and look for the **Image entry point** prompt. At the prompt enter ***0x1000*** to set the proper entry point.

Thirdly, in addition to the Hex files, ADS can also create supporting files such as the **Image Map** file and **Symbol** file during the **Make** process. The **Image Map** file maps each section of the object file to the actual location in memory, and the **Symbol** file maps each variable to the actual memory location. These two files provide the necessary memory locations that the user can use, along with the **Fill/Read Memory** function in the USB Clinic, to check the validity of software.

To configure the ADS to create the **Image Map** and **Symbol** files, go to **Linker** > **ARM Linker** > **Listings**. On the **Listings** window, there is an **Image Map** and **Symbols** check box. Check the desired file(s) and enter the file name at the **List file name** prompt, as shown in Figure 5-6.

Please note that creating the **Symbol** file increases the **Make** time significantly.

Figure 5-6. Image Map file and Symbol file selection.

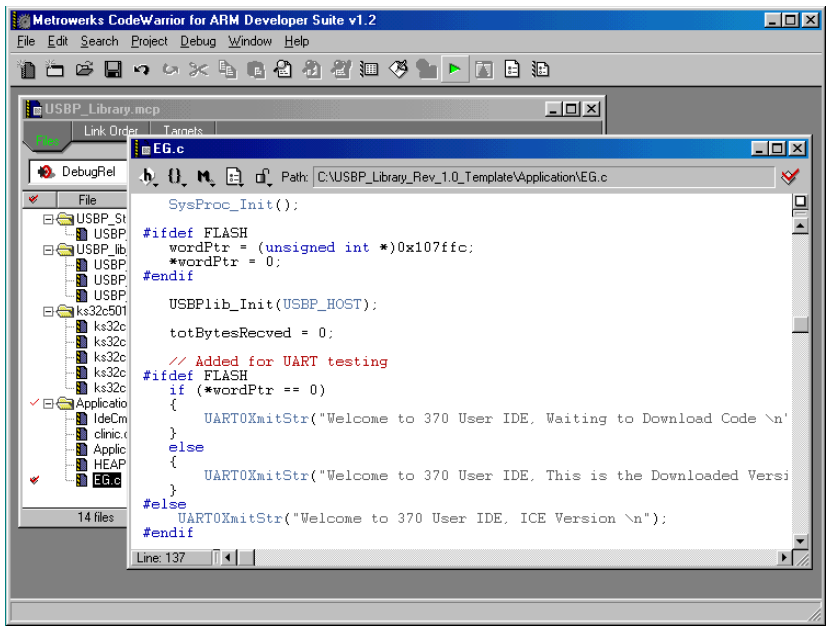


Note: The entry point 0x1000 is for ICE (In-Circuit Emulation) mode.

5.3 Modifying a Sample Application

Go back to the project files (see Figure 5-1 on page 2). In **USBP_Library.mcp**, there is an **EG.c** file. This file contains the main routine. Double-click on the **EG.c** to bring the source code (shown in Figure 5-7)

Figure 5-7. EG.c Source Code



The ARM firmware can print messages to the **Output Window** of the USB Clinic during execution. This is accomplished by calling the **UART0XmitStr()** function with a **\n** at the end of the text string.

For example, add `UART0XmitStr("Welcome to 370 User IDE, This is the Downloaded Version!! \n")` to the ARM Download mode firmware (Figure 5-8). During execution, the USB Clinic **Output Window** displays the text as shown in Figure 5-9.

Figure 5-8. Outputting Text Message From Source Code

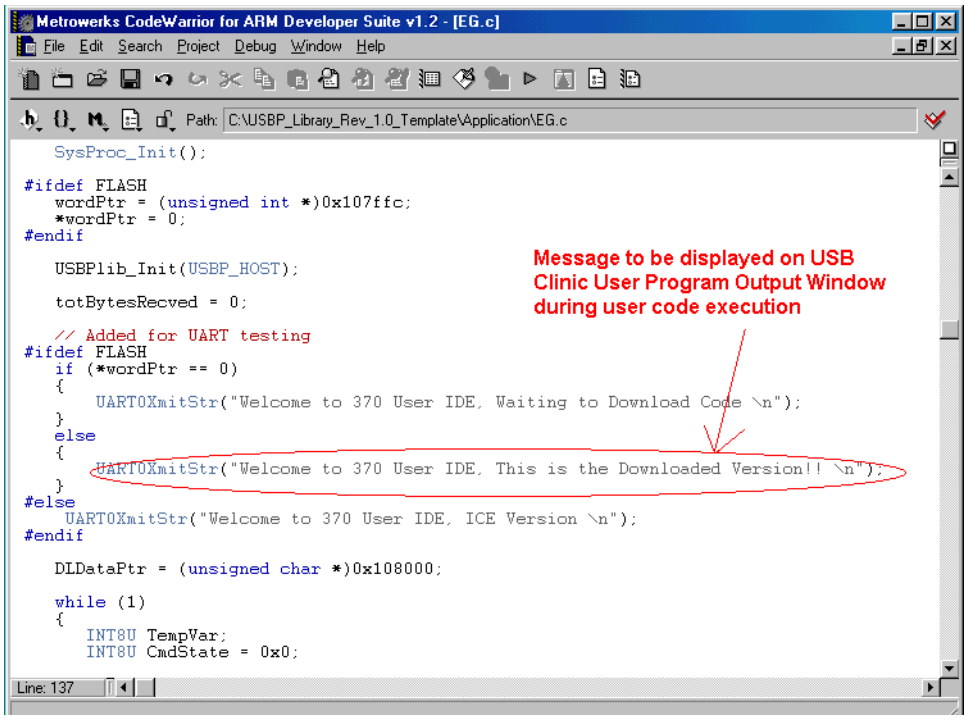
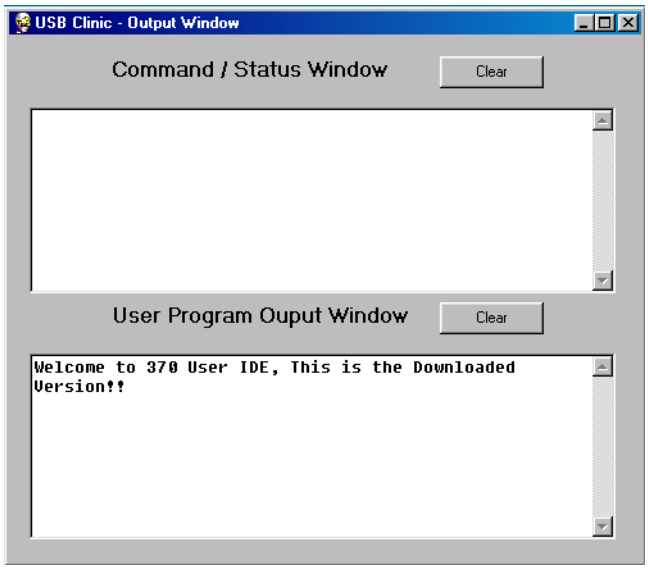


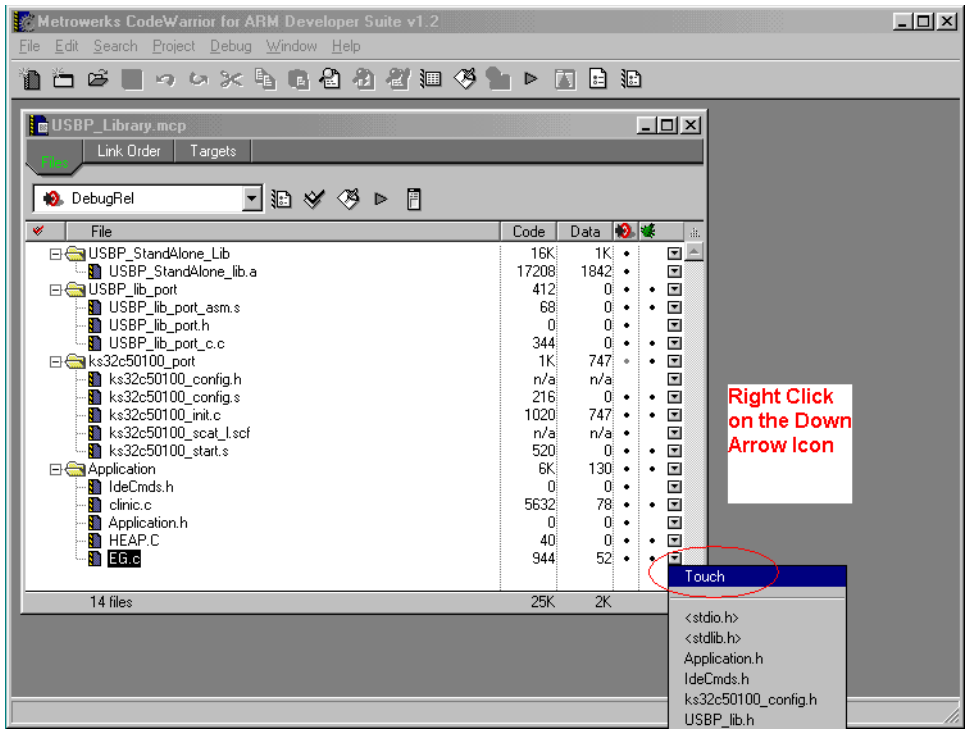
Figure 5-9. USB Clinic - User Program Output Window



There is a 1000-character limitation in the length of the text string. If string length exceeds 1000, simple break up the string into two or more short string and repeat the **UART0XmitStr().**

After editing the source files, if the user choose to re-compile only a single or a few selected files, right-click on the down-arrow icon at the right-most end of each of the files, and then select **Touch** (Figure 5-10).

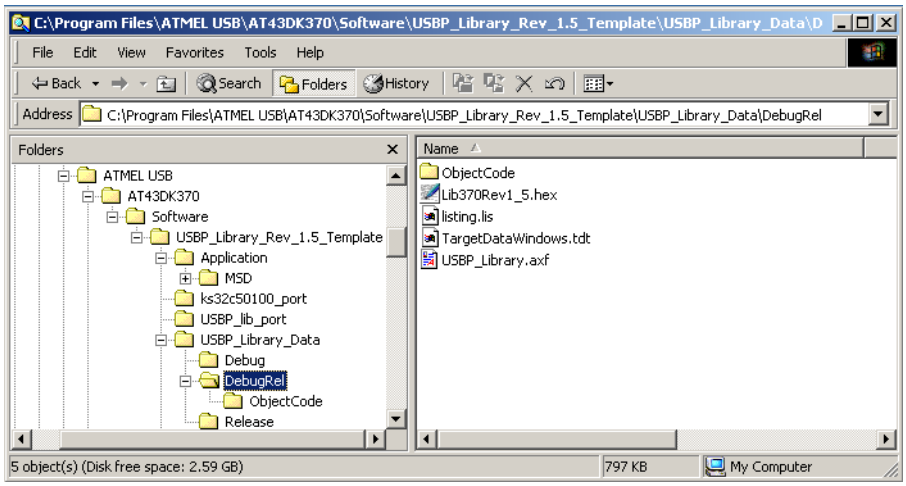
Figure 5-10. File Touch Selection



The user can then go on the **Project** > **Make** and that will compile the *touched* files only.

Once the output file is created (ICE mode axf file, Flash/Download mode hex files), it is placed into the directory **C:\Program Files\Atmel USB\AT43DK370\Software\USBP_Library_Rev_x.x_Template\USBP_Library_Data\DebugRel**, under the previous assigned file name (see Figure 5-5 on page 7 and Figure 5-11 on page 11). Use the USB Clinic to download and execute this firmware. Every time an output is generated, a **Work_Directory_Name_Data** directory is created that stores the output files. In this case, **Sample_DLVer_Data** is the output directory.

Figure 5-11. User Program Output Directory



Section 6

Converting Between FLASH and ICE Mode

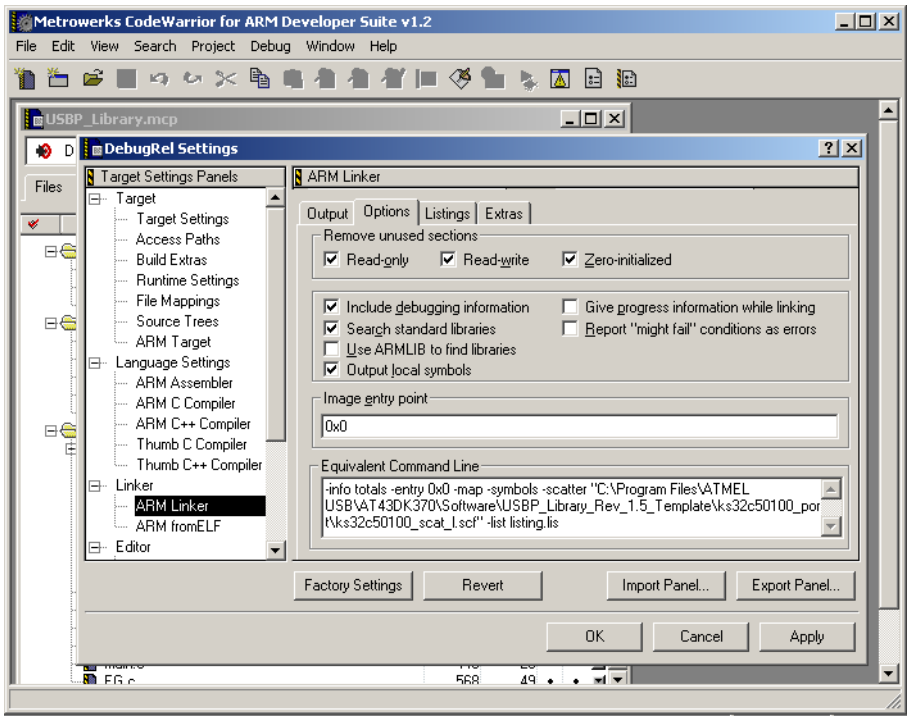
6.1 Introduction

By default the sample template is configured in ICE mode to run with In-Circuit Emulator. It can also be converted to Flash mode to generate the Hex file to be programmed onto the DK flash set, and the Download mode to generate the Hex file to be downloaded onto the DK SRAM using USB Clinic. The steps to convert between ICE mode, Flash mode and Download mode are as follows.

6.2 Converting to Flash Mode from ICE Mode

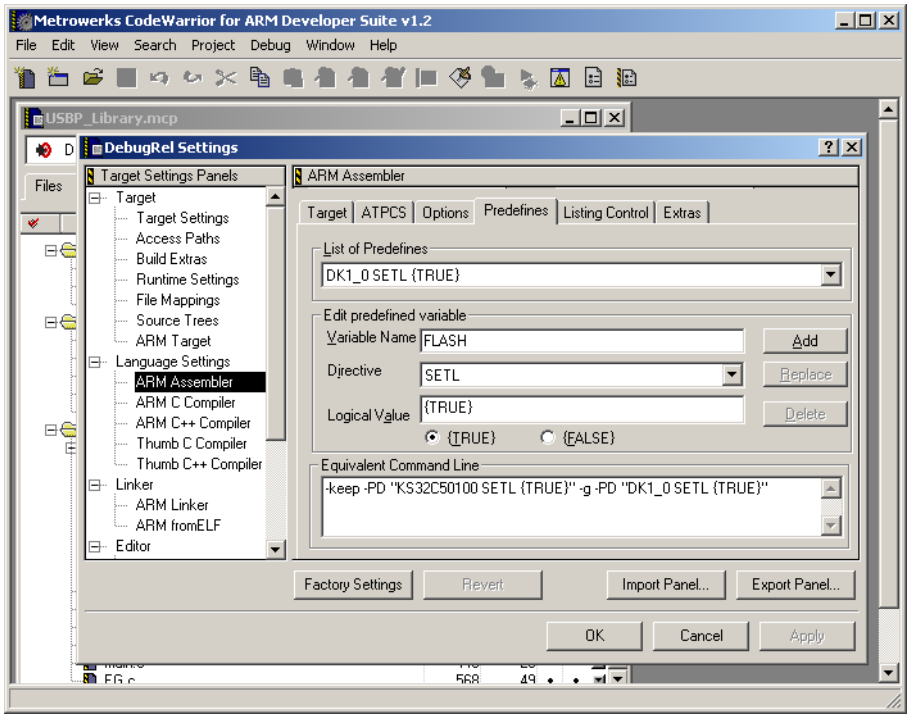
1. In ADS go to **DebugRel** > **Linker** > **ARM Linker** > **Options**
2. Under **Image entry point** specify address **0x0** (see Figure 6-1)

Figure 6-1. DebugRel Settings Window



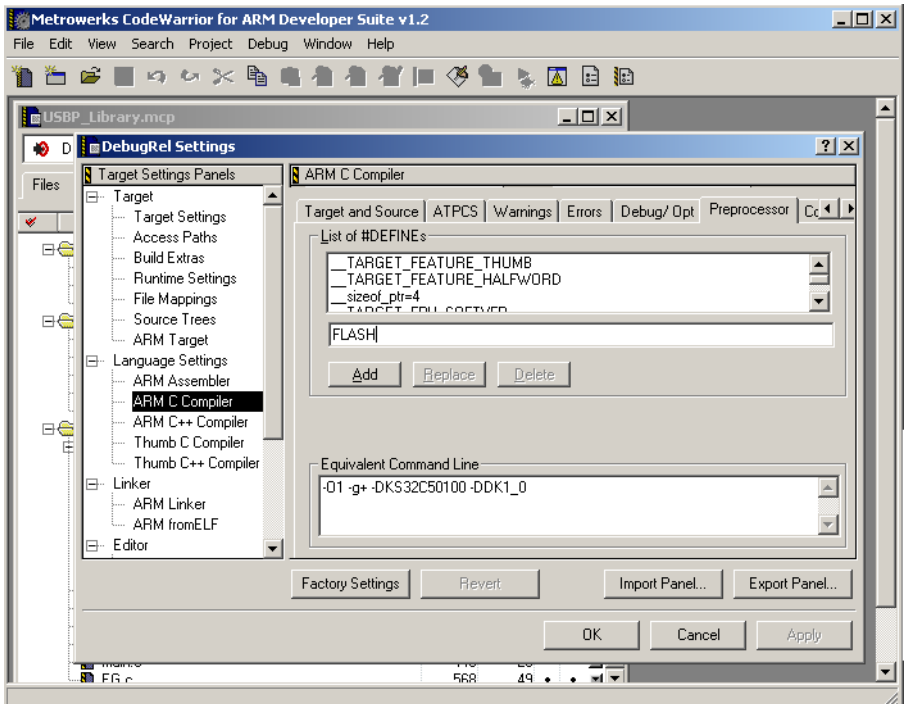
3. In project view, modify the **ks32c50100_scat_l** file to use the Flash Mode portion only.
4. In ADS go to **DebugRel** > **Language Settings** > **ARM Assembler**.
5. Click on the **Predefines** tab.
6. Add assembly constant: ***FLASH*** and click **Apply** (see Figure 6-2)

Figure 6-2. DebugRel Settings Window - Predefines Tab



7. Go to Language Settings > **ARM C Compiler** > **Preprocessor.**
8. Add C constant: ***FLASH*** and click **OK** (see Figure 6-3)

Figure 6-3. DebugRel Settings Window



For more information on generating the hex files for Flash mode, please refer to Section 7.

6.3 Converting to Flash Mode from Download Mode

1. In ADS go to **DebugRel** > **Linker** > **ARM Linker** > **Options**.
2. Under **Image entry point** specify address **0x0.**
3. In project view, modify the **ks32c50100_scat_l** file to use the Flash Mode portion only.
4. In ADS go to **DebugRel** > **Language Settings** > **ARM Assembler.**
5. Click on the **Predefines** tab.
6. Delete assembly constant: **DOWNLOAD** and click **Apply**.
7. Go to **Language Settings** > **ARM C Compiler** > **Preprocessor.**
8. Delete C constant: **DOWNLOAD** and click **OK.**

6.4 Converting to Download Mode from ICE Mode

1. In ADS go to **DebugRel** > **Linker** > **ARM Linker** > **Options**.
2. Under **Image entry point** specify address **0x108000.**
3. In project view, modify the **ks32c50100_scat_l** file to use the Download Mode portion only.
4. In ADS go to **DebugRel** > **Language Settings** > **ARM Assembler.**
5. Click on the **Predefines** tab.
6. Add assembly constants: **FLASH**, **DOWNLOAD** and click **Apply**.
7. Go to **Language Settings** > **ARM C Compiler** > **Preprocessor.**
8. Add C constants: **FLASH**, **DOWNLOAD** and click **OK.**

6.5 Converting to Download Mode from Flash Mode

1. In ADS go to **DebugRel** > **Linker** > **ARM Linker** > **Options**.
2. Under **Image entry point** specify address **0x108000.**
3. In project view, modify the **ks32c50100_scat_l** file to use the Download Mode portion only.
4. In ADS go to **DebugRel** > **Language Settings** > **ARM Assembler.**
5. Click on the **Predefines** tab.
6. Add assembly constants: **DOWNLOAD** and click **Apply**.
7. Go to **Language Settings** > **ARM C Compiler** > **Preprocessor.**
8. Add C constant: **DOWNLOAD** and click **OK.**

6.6 Converting to ICE Mode from Flash Mode

1. In ADS go to **DebugRel** > **Linker** > **ARM Linker** > **Options**.
2. Under **Image entry point** specify address **0x1000.**
3. In project view, modify the **ks32c50100_scat_l** file to use the ICE Mode portion only.
4. In ADS go to **DebugRel** > **Language Settings** > **ARM Assembler.**
5. Click on the **Predefines** tab.
6. Delete assembly constant: **FLASH**.
7. Go to **Language Settings** > **ARM C Compiler** > **Preprocessor.**
8. Delete C constant: **FLASH.**

6.7 Converting to ICE Mode from Download Mode

1. In ADS go to **DebugRel** > **Linker** > **ARM Linker** > **Options**.
2. Under **Image entry point** specify address **0x1000.**
3. In project view, modify the **ks32c50100_scat_l** file to use the ICE Mode portion only.
4. In ADS go to **DebugRel** > **Language Settings** > **ARM Assembler.**
5. Click on the **Predefines** tab.
6. Delete assembly constant: **FLASH** and **DOWNLOAD**.
7. Go to **Language Settings** > **ARM C Compiler** > **Preprocessor.**
8. Delete C constants: **FLASH** and **DOWNLOAD.**

6.8 Summary

ICE Mode: Entry point = 0x1000, ks32c50100_scat_l file use ICE mode portion, No “FLASH” or “DOWNLOAD” as predefine constants.

Flash Mode: Entry point = 0x0, ks32c50100_scat_l file use Flash mode portion, has “FLASH” as predefine constant.

Download Mode: Entry point = 0x108000, ks32c50100_scat_l file use Download mode portion, has both “FLASH” and “DOWNLOAD” as predefine constants.

Section 7

Generating Hex Files for Flash Mode in the AT43USB370 Development Board with ADS

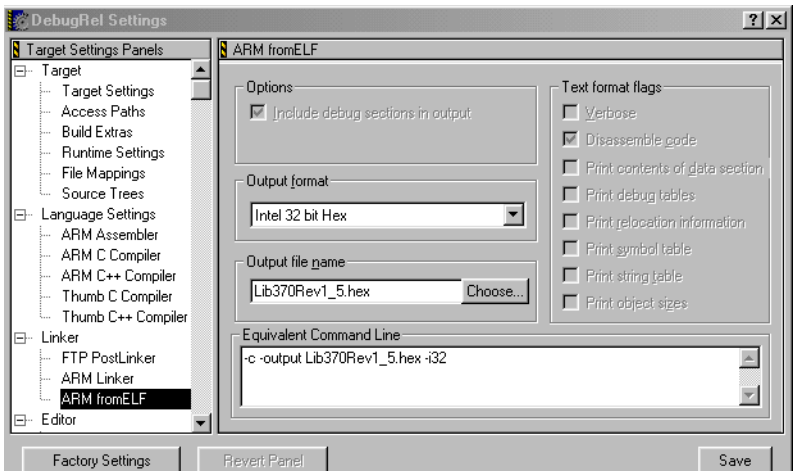
7.1 Introduction

This section gives directions on generating hex file for Flash mode in the AT43USB370 Development Board with ADS.

7.2 Procedure

1. To put the board in Flash mode jumper the following pins on the headers:
   a. In JP4, jumper together FLASH_CS and nRCS<0>
   b. In JP4, jumper together SRAM_CS and nRCS<1>
   c. In JP7, jumper together pin 1-2
2. The project entry point must be changed from to 0x0. To change the entry point, go to **DebugRel** > **ARM Linker** > **Options** > **Image Entry Point** column and set it to 0x0.
3. Add the **Predefine** constant ***FLASH*** in the ARM Assembler and Compiler (see Section 6)
4. To create a Hex file do the following:

Figure 7-1. Post-linker Setting: ARM fromELF



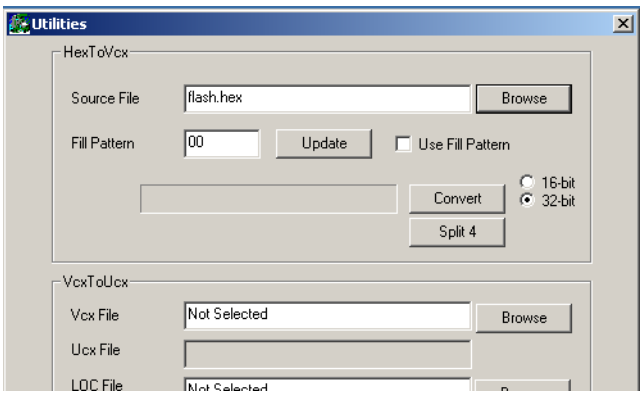
Once the hex file is created, open the hex file in a text editor. In the hex file there is a 16-bit address that can address a 16 KB space. Since we're splitting this file into 4 separate hex files, we would not need any extended hex records. Therefore, delete the lines beginning with: ":02000…" except for the first record (leave that one in).

Note: For the DOWNLOAD mode hex file, taking out the ":02000…" lines is not required.

5. Use **Hex2Vcx.exe** to split this file into 4 hex files for each flash. The **Hex2Vcx.exe** can be found at the **C:\Program Files\Atmel USB\AT43DK370\Software\USBP_Library_Rev_x.x_Template\USBP_Library_Data\DebugRel** directory.

6. Choose the following from the GUI that appears (see Figure 7-2):
   a. Specify the hex file that is to be split in the source file text box
   b. Srt **Fill Pattern** to **00**
   c. Select **32-bit**
   d. Click on the **[Split 4]** button

Figure 7-2. Hex2Vcx GUI



7. Once the hex file has been split, program the flash sets with the hex files according to their labels.

Section 8

Technical Support

For technical support, please fill out the Customer Problem Report html form in the **C:\Program Files\ATMEL USB\AT43DK370** directory.

Alternatively, fill out an online support form available in the Product Section of the Atmel web site at **http://www.atmel.com**. Please make sure the following information is included:

- Revision number of the AT43DK370 Development Board
- Version number of the USB Clinic
- A detailed description of the problem

Section 9

Appendices

9.1 AT43USB370 Bill of Materials (BOM)

Item	Qty	Reference	Part Description	Manufacturer	Manufacturer Part No.	Distributor	Distributor Part No.
Capacitor							
1	2	C1, C24	2.2 nF Ceramic 0805	Panasonic	ECJ-2VB1H222K	Digikey	PCC222BNCT-ND
2	1	C2	22 nF Ceramic 0805	Panasonic	ECJ-2VB1H223K	Digikey	PCC223BGCT-ND
3	46	C3, C4, C5, C6, C7, C8, C9, C10, C11, C12, C13, C14, C15, C26, C27, C28, C29, C30, C31, C32, C33, C34, C35, C36, C37, C39, C40, C41, C42, C43, C49, C50, C51, C52, C53, C54, C55, C56, C57, C58, C59, C60, C61, C62, C63, C66	0.1 µF Ceramic 0805	Panasonic	ECJ-2VB1C104K	Digikey	PCC1812CT-ND
4	2	C17, C22	1000 pF Ceramic 0805	Panasonic	ECJ-2VC1H102J	Digikey	PCC102CGCT-ND
5	3	C18, C23, C25	33 pF Ceramic 0805	Panasonic	ECJ-2VC1H330J	Digikey	PCC330CGCT-ND
6	4	C44, C45, C46, C47	100 pF Ceramic 0805	Panasonic	ECJ-2VC1H101J	Digikey	PCC101CGCT-ND
7	1	C38	820 pF Ceramic 0805	Panasonic	ECJ-2VC1H821J	Digikey	PCC821CGCT-ND
8	3	C16, C19, C21	10 µF Tantalum C	Kemet	T491C106K016AS	Digikey	399-1595-1-ND
9	1	C48	100 µF Electrolytic D	Panasonic	ECE-V1CA101WP	Digikey	PCE3182CT-ND
10	1	C20	4.7 µF Electrolytic A	Panasonic	ECE-V1ES4R7SR	Digikey	PCE3065CT-ND
11	2	C64, C65	33 pF Ceramic 0805	Panasonic	ECJ-2VC1H330J	Digikey	PCC330CGCT-ND
Resistor							
12	4	R1, R14, R15, R16	470 5% 0805	Panasonic	ERJ-6GEYJ471V	Digikey	P470ACT-ND
13	6	R2, R27, R28, R29, R30, R31	10K 5% 0805	Panasonic	ERJ-6GEYJ103V	Digikey	P10KACT-ND

Item	Qty	Reference	Part Description	Manufacturer	Manufacturer Part No.	Distributor	Distributor Part No.
14	1	R3	162K 1% 0805	Panasonic	ERJ-6ENF1623V	Digikey	P162KCCT-ND
15	1	R4	53.6K 1% 0805	Panasonic	ERJ-6ENF5362V	Digikey	P53.6KCCT-ND
16	1	R5	86.6K 1% 0805	Panasonic	ERJ-6ENF8662V	Digikey	P86.6KCCT-ND
17	6	R6, R10, R11, R26, R38, R39	100K 5% 0805	Panasonic	ERJ-6GEYJ104V	Digikey	P100KACT-ND
18	3	R7, R12, R13	33K 5% 0805	Panasonic	ERJ-6GEYJ333V	Digikey	P33KACT-ND
19	1	R8	22K 5% 0805	Panasonic	ERJ-6GEYJ223V	Digikey	P22KACT-ND
20	1	R9	14K 1% 0805	Panasonic	ERJ-6ENF1402V	Digikey	P14.0KCCT-ND
21	7	R17, R18, R19, R20, R21, R22, R23	33 Resistor Array EXB-2HV SMD	Panasonic	EXB-2HV330JV	Digikey	Y1330CT-ND
22	4	R24, R25, R34, R36	1.5K 5% 0805	Panasonic	ERJ-6GEYJ152V	Digikey	P1.5KACT-ND
23	2	R32, R33	22 5% 0805	Panasonic	ERJ-6GEYJ220V	Digikey	P22ACT-ND
24	2	R37, R35	15K 5% 0805	Panasonic	ERJ-6GEYJ153V	Digikey	P15KACT-ND
LED, Inductors							
25	3	LED1, LED2, LED3	LED Green 0805 SMD	Lumex	SML-LXT0805GW-TR	Digikey	67-1553-1-ND
26	3	L1, L2, L3	4.7 µH D73C SMD	TOKO	636CY-4R7M	Digikey	TKS2505CT-ND
27	2	L4, L5	1.2 µH 1008 SMD	Panasonic	ELJ-FC1R2JF	Digikey	PCD1229CT-ND
Semiconductor							
28	1	U1	USB Host Processor	Atmel	AT43USB370		
29	1	U8	Triple Voltage Regulator	Maxim	MAX1702BEGX		
30	1	U9	ARM µProcessor	Samsung	S3C4510		
31	1	U10	RS-232 Line Transceiver	Maxim	MAX3222	Digikey	296-13082-1-ND
32	1	U12	MIC2505-1BM	Micrel	MIC2505-1BM	Future	MIC2505-1BM
33	4	U13, U14, U15, U16	512K x 8 Static RAM	Alliance Semicondutor	AS7C34096-12TC	Future	S628512CV-12TF
Oscillator, Crystal							
34	1	OSC1	10 MHz Oscillator SG-636 SMD	EPSON	SG-8002JC-PCC	Digikey	SG-8002JC-PCC-ND 10MHz
35	1	XTAL1	6 MHz Crystal ATS-SM SMD	CTS	ATS060SM-T	Digikey	CTX505-ND
Socket, Header, Connector, Switch							
36	4	U4, U5, U6, U7	32-Pin Dip Socket	AMP	2-382189-1	Digikey	A24812-ND
37	1	U17	20-Pin PLCC Socket	Mill-Max	940-99-020-24-000000	Digikey	ED80021-ND
38	4	JP1, JP2, JP5, JP7	HEADER 3x1 Straight Male			Jameco	109575
39	1	JP4	HEADER 5x2 Straight Male			Jameco	67820
40	1	J4	HEADER 10X2 Right Angle Male	AMP	103311-5	Digikey	A26292-ND
41	5	JP3, JP8, JP9, JP10, JP11	HEADER 2x1 Straight Male			Jameco	108337
42	2	JP12, JP13	HEADER 17x2 Straight Male			Jameco	53516
43	1	J1	Power Jack 3 Terminals			Digikey	CP-202A-ND
44	1	J5	Serial Port DB9 Male	AMP	747840-5	Digikey	A23278-ND

Item	Qty	Reference	Part Description	Manufacturer	Manufacturer Part No.	Distributor	Distributor Part No.
45	1	J2	USB - A	AMP	AMP787616-1	Digikey	787616-1
46	1	J3	USB - B	AMP	AMP787780-1	Digikey	787780-1
47	1	SW1	Momentary Push Button Switch	E-Switch	520-02-RED	Digikey	EG1415-ND
Second Level Components Stuffing							
48	4	U4, U5, U6, U7	ROM 256K x 8, 90 ns	Atmel	AT49LV002-90PC		
49	1	U17	PAL 16v8 PLCC 10 ns	Atmel	ATF16LV8C-10JC		
Components Not Stuffed on Board							
50	1	OSC2	48 MHz Oscillator SG-636 SMD	EPSON	SG-8002JC-PCC	Digikey	SG-8002JC-PCC-ND 48MHz
Miscellaneous							
51	1		Adhesive Rubber Feet 100/Pk	3M	SJ5018BLKC	Jameco	142682

9.2 AT43DK370 Schematics

Figure 9-1 on page 4 through Figure 9-6 on page 9 cover the schematic diagrams for this system.

Figure 9-1. ARM Schematic

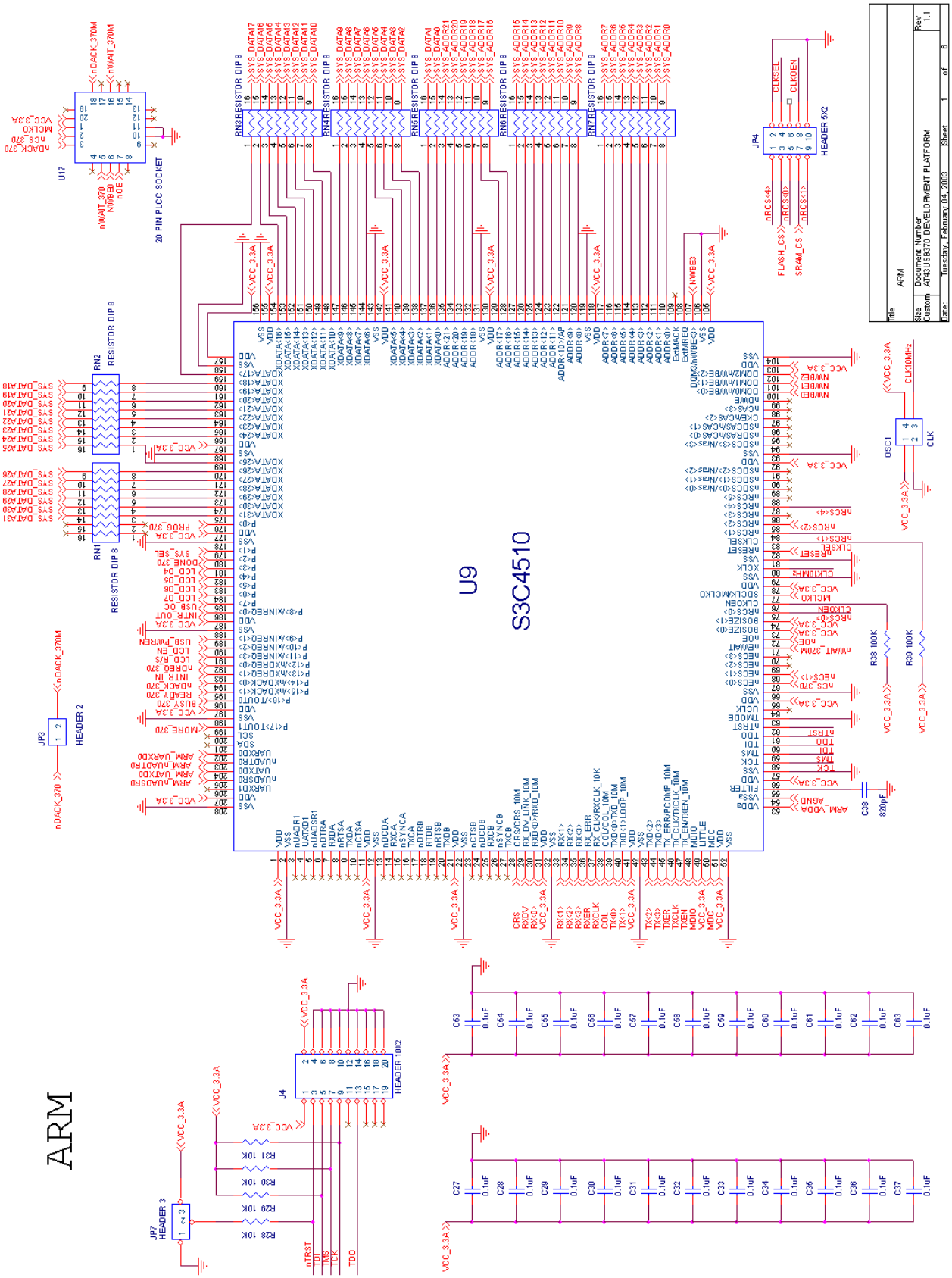


Figure 9-2. AT43USB370 Schematic

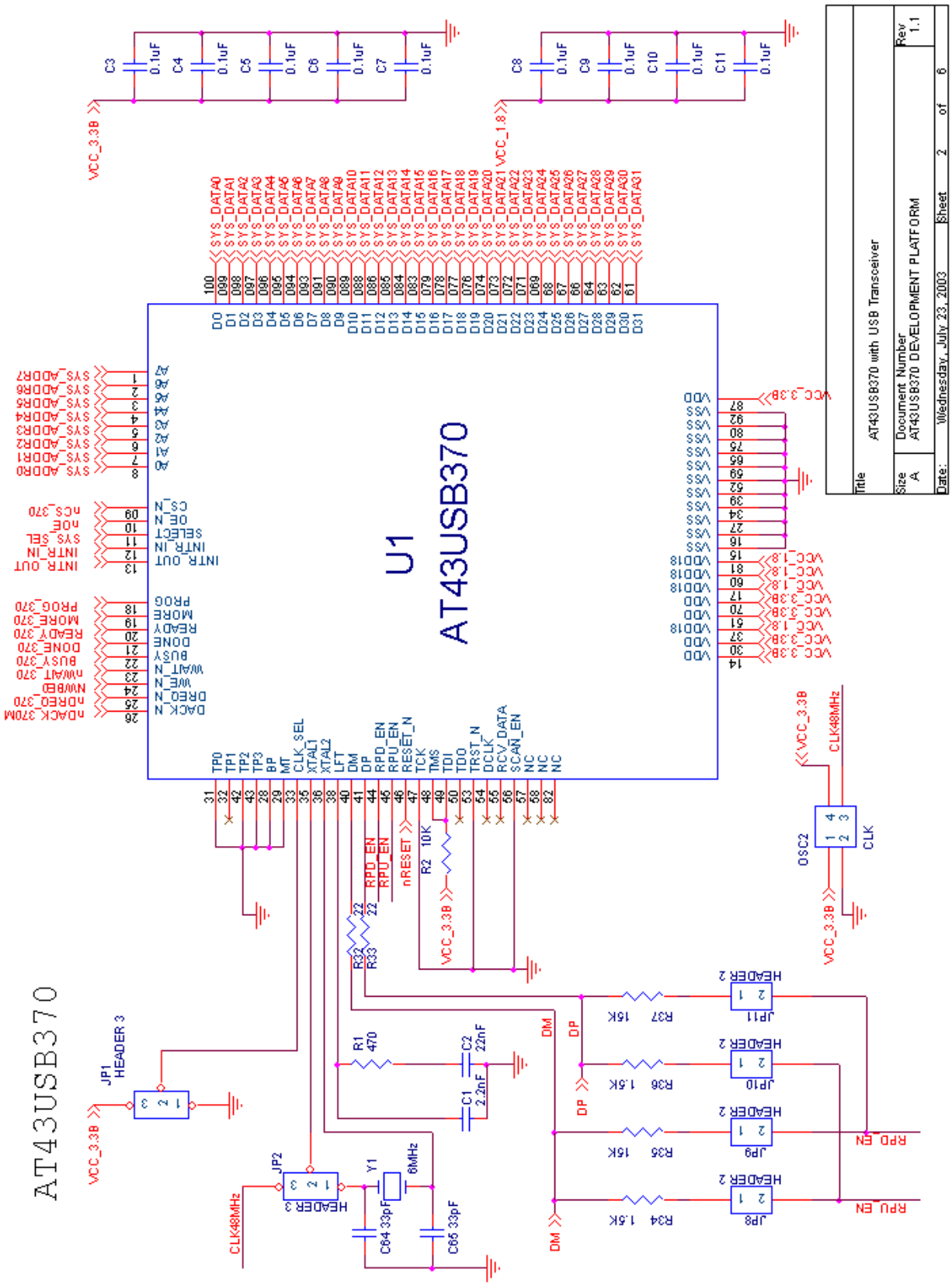


Figure 9-3. SRAM Schematic

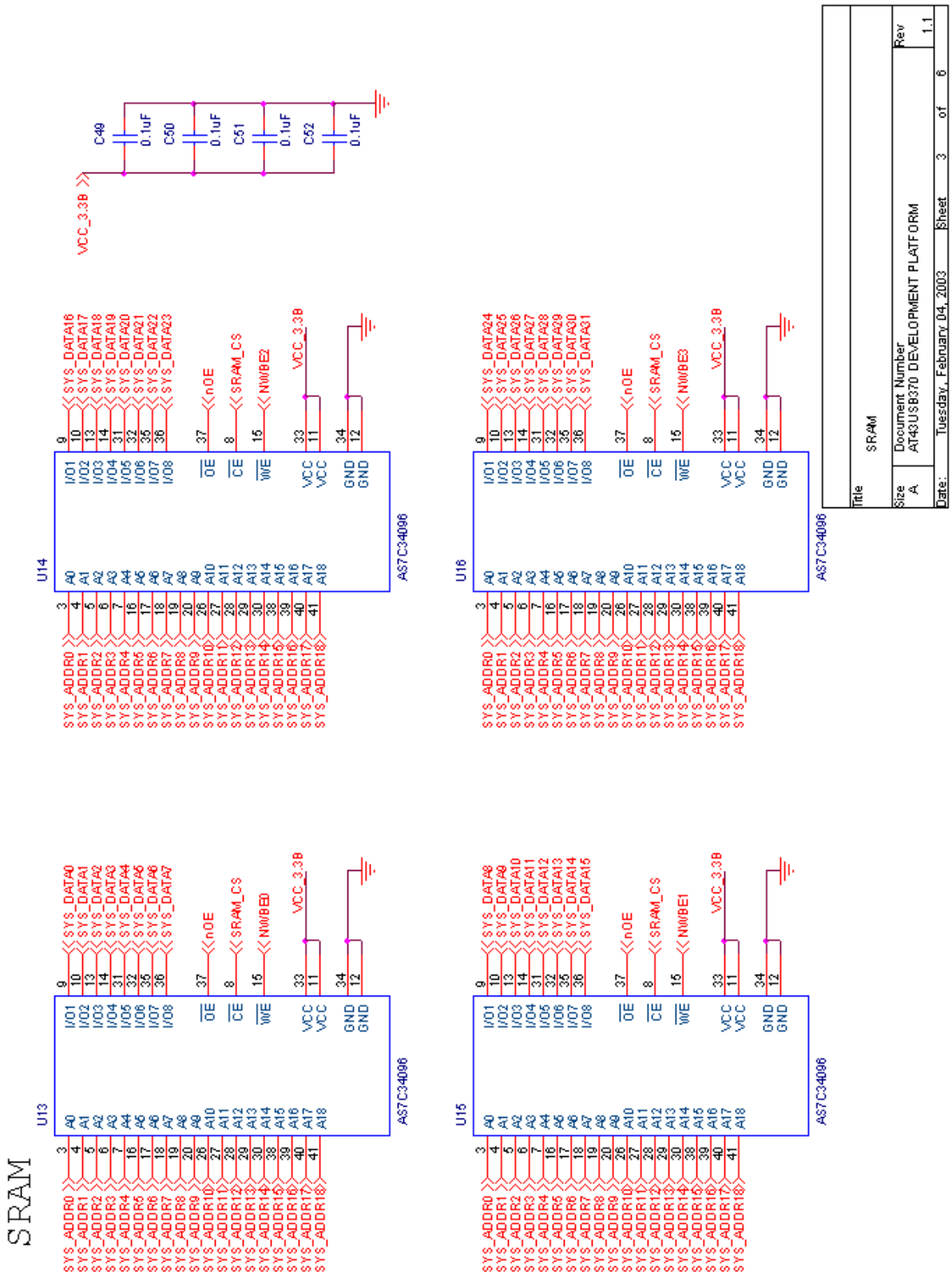


Figure 9-4. FLASH Schematic

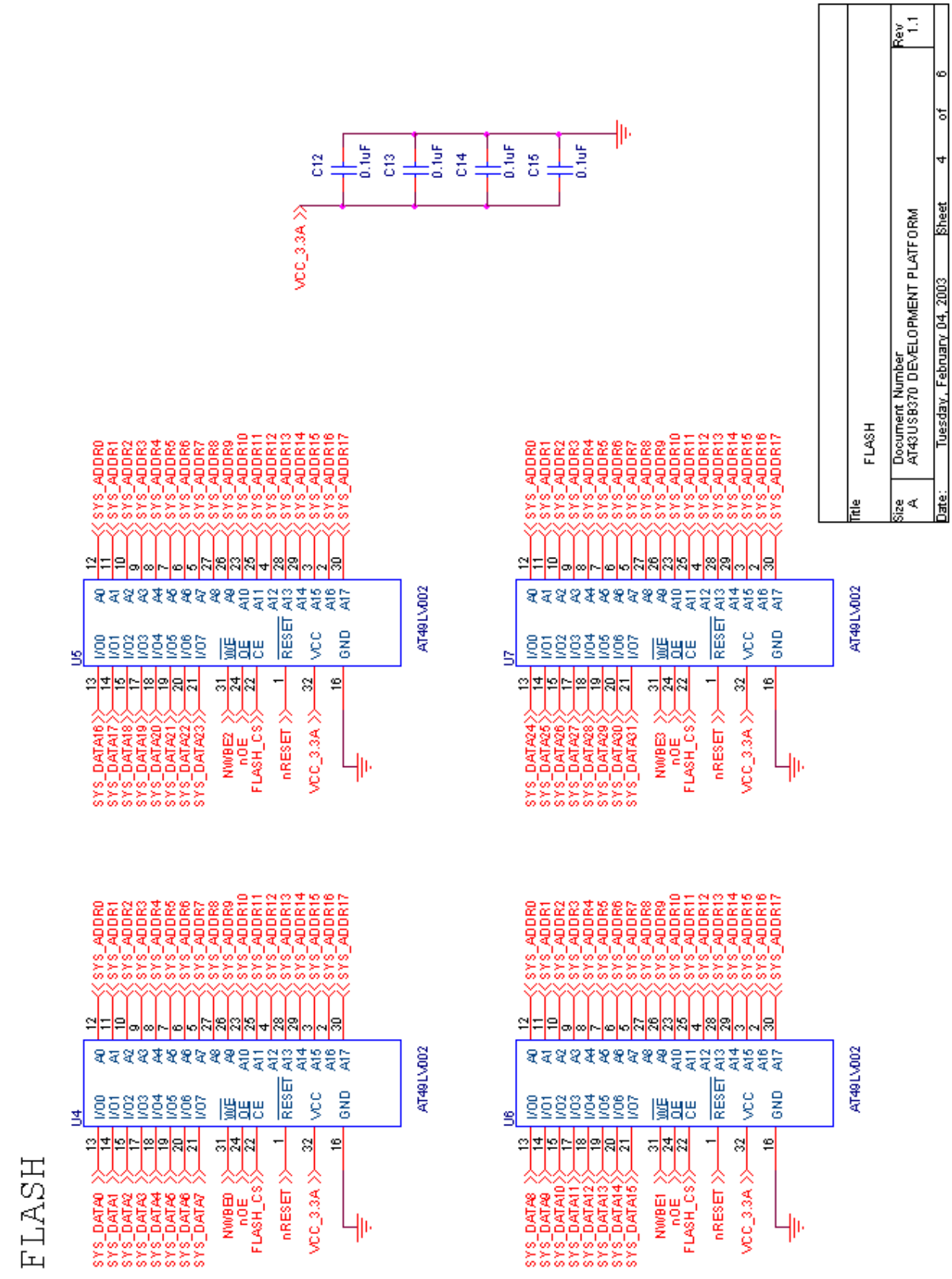


Figure 9-5. Serial Port, Connectors and USB Power Schematic

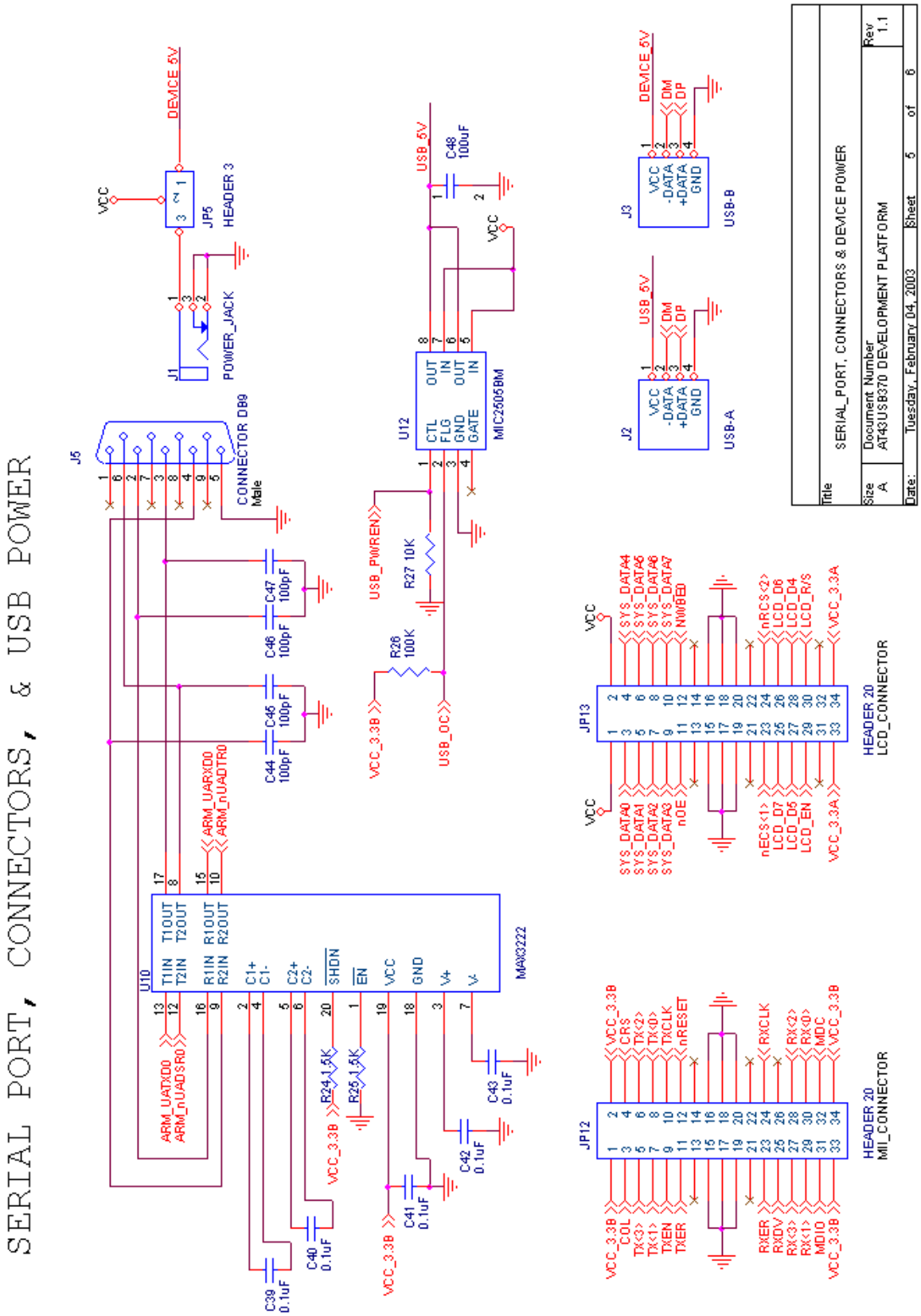
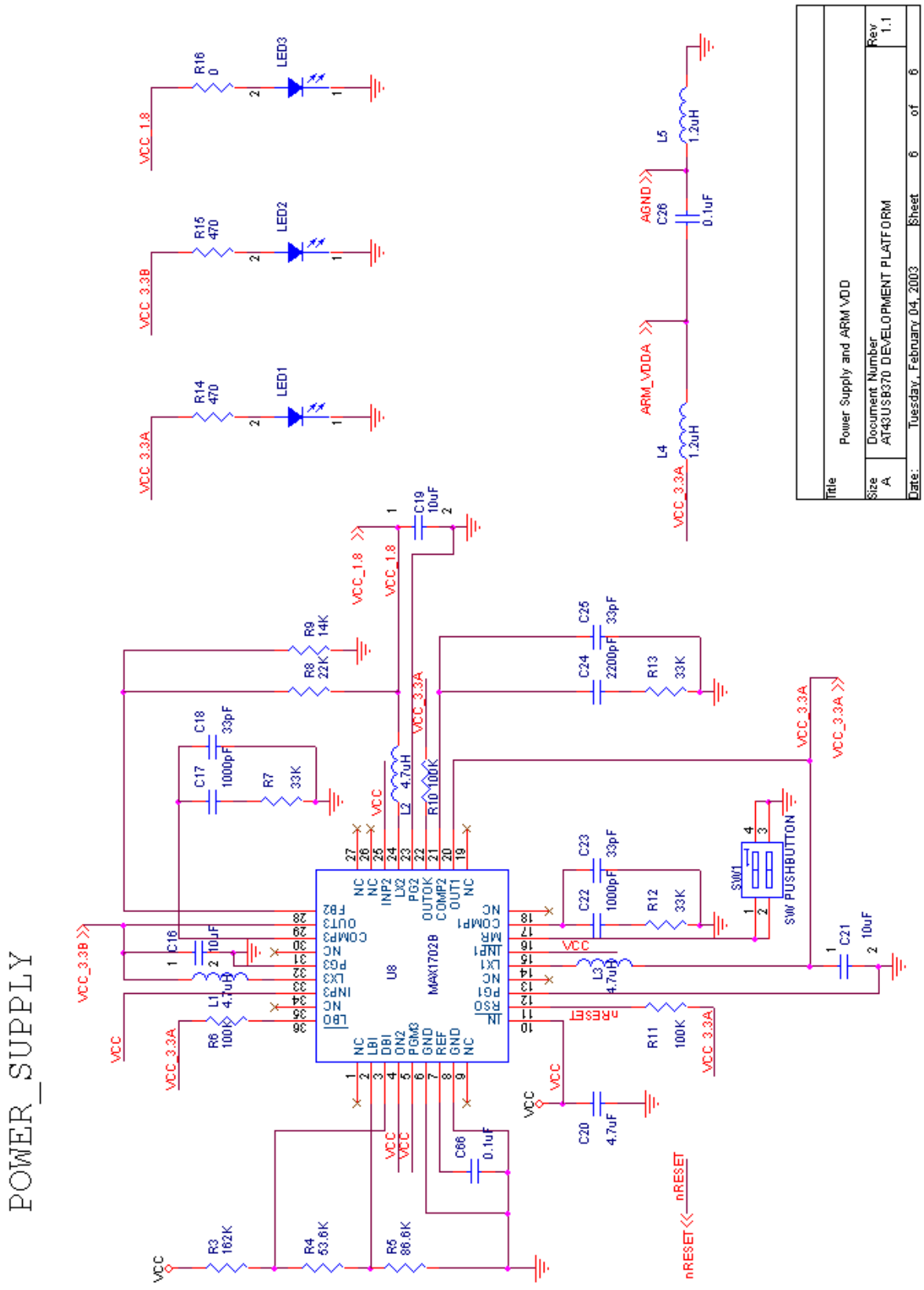


Figure 9-6. Power Supply Schematic



Atmel Corporation

2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

Regional Headquarters

Europe

Atmel Sarl
Route des Arsenaux 41
Case Postale 80
CH-1705 Fribourg
Switzerland
Tel: (41) 26-426-5555
Fax: (41) 26-426-5500

Asia

Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimshatsui
East Kowloon
Hong Kong
Tel: (852) 2721-9778
Fax: (852) 2722-1369

Japan

9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

Atmel Operations

Memory

2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

Microcontrollers

2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

La Chantrerie
BP 70602
44306 Nantes Cedex 3, France
Tel: (33) 2-40-18-18-18
Fax: (33) 2-40-18-19-60

ASIC/ASSP/Smart Cards

Zone Industrielle
13106 Rousset Cedex, France
Tel: (33) 4-42-53-60-00
Fax: (33) 4-42-53-60-01

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Scottish Enterprise Technology Park
Maxwell Building
East Kilbride G75 0QR, Scotland
Tel: (44) 1355-803-000
Fax: (44) 1355-242-743

RF/Automotive

Theresienstrasse 2
Postfach 3535
74025 Heilbronn, Germany
Tel: (49) 71-31-67-0
Fax: (49) 71-31-67-2340

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Biometrics/Imaging/Hi-Rel MPU/
High Speed Converters/RF Datacom

Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex, France
Tel: (33) 4-76-58-30-00
Fax: (33) 4-76-58-34-80

Literature Requests
www.atmel.com/literature

Disclaimer: Atmel Corporation makes no warranty for the use of its products, other than those expressly contained in the Company's standard warranty which is detailed in Atmel's Terms and Conditions located on the Company's web site. The Company assumes no responsibility for any errors which may appear in this document, reserves the right to change devices or specifications detailed herein at any time without notice, and does not make any commitment to update the information contained herein. No licenses to patents or other intellectual property of Atmel are granted by the Company in connection with the sale of Atmel products, expressly or by implication. Atmel's products are not authorized for use as critical components in life support devices or systems.

© Atmel Corporation 2004. **All rights reserved.** Atmel® and combinations thereof, are the registered trademarks of Atmel Corporation or its subsidiaries. ARM® is the registered trademark, and Developer Suite™ is the trademark of ARM Limited. Metrowerks® is the registered trademark, and CodeWarrior™ is the trademark of Metrowerks Corp. Intel® and Pentium® are the registered trademarks of Intel Corporation. Windows® 98/2000/ME/XP is the registered trademarks of Microsoft Corp. Other terms and product names may be the trademarks of others.

Printed on recycled paper.

3423C–USB–1/04 /xM