AT43DK380 USB Host/Function Development Kit

User Guide for Revision 1.1

Table of Contents

Section 1

Introduction 1-1

1.1 Development Kit Packages and 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 AT43DK380 and Starting Up the USB Clinic 2-2

2.6 Testing the Hardware 2-3

Section 3

Hardware Description 3-1

3.1 Description of the Main Development Platform 3-1

3.1.1 AT43DK380 Processor Interface Connector 3-2

3.1.2 Jumper Settings 3-4

3.2 AT43DC380-PDC1 Processor Daughter Card 3-6

3.2.1 PDC1 Jumper Settings 3-6

3.3 AT43DC380-PDC2 Processor Daughter Card 3-7

3.3.1 PDC2 Jumper Settings 3-7

3.4 AT43DK380 Main Development Platform to PDC Connection 3-8

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 4-4

4.5 Memory 4-5

4.5.1 Fill Memory 4-5

4.5.2 Read Memory 4-5

4.6 Enumeration....4-6
4.6.1 Get Device Descriptor....4-7
4.6.2 Get Configuration Descriptor....4-7
4.6.3 Get String Descriptor....4-7
4.6.4 Set Configuration....4-8
4.6.5 Set Interface....4-8
4.7 Data Transfer....4-8
4.7.1 Get ISO Data....4-9
4.7.2 Send ISO Data....4-9
4.7.3 Get Data....4-9
4.7.4 Send Data....4-9
4.7.5 Control Transfer....4-9
4.7.6 Custom Transfer....4-10
4.8 Port Features....4-11
4.8.1 Set Port Feature....4-11
4.8.2 Clear Port Feature....4-11
4.9 Device State Control....4-12
4.9.1 Reset Device....4-12
4.9.2 Suspend Device....4-12
4.9.3 Resume Device....4-12
4.10 Miscellaneous Notes....4-12

Section 5

Building Firmware for the AT43DK380 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....5-5
5.2 ADS Settings....5-6
5.3 Modifying a Sample Application....5-8

Section 6

Converting Between FLASH and ICE Mode and Download 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....6-1
6.4 Converting to Download Mode from ICE Mode....6-2
6.5 Converting to Download Mode from Flash Mode....6-2
6.6 Converting to ICE Mode from Flash Mode....6-2
6.7 Converting to ICE Mode from Download Mode....6-2
6.8 Summary....6-2
6.8.1 ICE Mode....6-2

6.8.2 Flash Mode 6-3
6.8.3 Download Mode 6-3

Section 7

Switching Between PDC1 and PDC2 Project Template 7-1
7.1 Switching from PDC1 to PDC2 Project Template 7-1
7.2 Switching from PDC2 to PDC1 Project Template 7-1
7.3 Summary 7-1
7.3.1 For PDC1 7-1
7.3.2 For PDC2 7-1

Section 8

Generating Hex Files for Flash Mode in the AT43USB380 Development Platform with ADS 8-1
8.1 Introduction 8-1
8.2 Procedure 8-1

Section 9

Technical Support 9-1

Section 10

Appendices 10-1
10.1 AT43DK380 Schematics 10-1
10.1.1 AT43DK380 Main Development Platform Schematics 10-1
10.1.2 AT43DK380-PDC1 Schematic 10-6
10.1.3 AT43DK380-PDC2 Schematic 10-7
10.2 AT43DK380 Bill of Materials (BOM) 10-8
10.2.1 AT43DK380 Main Development Platform BOM 10-8
10.2.2 AT43DK380-PDC1 BOM 10-11
10.2.3 AT43DK380-PDC2 BOM 10-12
10.3 AT43DK380 Main Development Platform CPLD VHDL Code Ref. 10-13

Section 1

Introduction

Congratulations on your purchase of the AT43DK380 Development Kit. The AT43DK380 Main Development Platform is the AT43USB380 Host/Function/OTG Processor Development Platform to be used with a user target processor board or the AT43DK380 Processor Daughter Cards (PDCs). It is designed to allow real-time firmware development and evaluation of the AT43USB380 USB Host/Function/OTG Processor.

1.1 Development Kit Packages and Features

The AT43DK380 Development Kit comes in the following packages:

- **AT43DK380-BD1** AT43USB380 Development Kit bundled set 1:
  - AT43DK380 Main Development Platform with pre-programmed flash set for the PDC1 32-bit ARM7TDMI® Processor
  - AT43DK380-PDC1 32-bit ARM7TDMI Processor Daughter Card
  - Pre-compiled ANSI-C compliant USB Firmware Libraries including USB Host Stack and Device Class Drivers for the PDC1 32-bit ARM7TDMI Processor
  - USB Clinic In-Circuit Emulation tool
- **AT43DK380-BD2** AT43USB380 Development Kit bundled set 2:
  - AT43DK380 Main Development Platform with pre-programmed flash set for the PDC2 AT91R40008 16-bit ARM7TDMI Processor
  - AT43DK380-PDC2 AT91R40008 16-bit ARM7TDMI Processor Daughter Card
  - Pre-compiled ANSI-C compliant USB Firmware Libraries including USB Host Stack and Device Class Drivers for the PDC2 AT91R40008 16-bit ARM7TDMI Processor
  - USB Clinic In-Circuit Emulation tool
- **AT43DK380-PDC1** 32-bit ARM7TDMI Processor Daughter Card set:
  - AT43DK380-PDC1 32-bit ARM7TDMI Processor Daughter Card
  - Pre-programmed flash set for the PDC1 32-bit ARM7TDMI Processor
  - Pre-compiled ANSI-C compliant USB Firmware Libraries including USB Host Stack and Device Class Drivers for the PDC1 32-bit ARM7TDMI Processor
  - USB Clinic In-Circuit Emulation tool

- **AT43DK380-PDC2** Atmel AT91R40008 16-bit ARM7TDMI Processor Daughter Card set:
  - AT43DK380-PDC2 AT91R40008 16-bit ARM7TDMI Processor Daughter Card
  - Pre-programmed flash set for the PDC2 AT91R40008 16-bit ARM7TDMI Processor
  - Pre-compiled ANSI-C compliant USB Firmware Libraries including USB Host Stack and Device Class Drivers for the PDC2 AT91R40008 16-bit ARM7TDMI Processor
  - USB Clinic In-Circuit Emulation tool

A complete development kit contains a Main Development Platform and a Daughter Card. The pre-programmed flash set and the software tools are functional only for the complete development kit. We recommend purchasing a bundled set for initial development.

The AT43DK380 Main Development Platform consists of the following features:

- AT43USB380 USB Host/Function/OTG Processor
- 1M Bytes Flash ROM
- 2M Bytes Static RAM
- USB MINI - A/B ports
- 4-Ports USB Hub
- OTG Interface Charge Pump
- RS-232 Serial Port
- 8/16/32-bit System Processor Interface
- 64 I/Os CPLD
- EEPROM Connector
- Reset buttOn
- Power Indicator LEDs
- In-System Firmware Programming Capability

Please register at **http://www.atmel.com/products/usb/forms/softwareg.asp** to access and be informed of the latest up-to-date information on new USB software, documentation releases and tool upgrades.

Section 2

Getting Started

2.1 Electrostatic Warning

The AT43DK380 Development Platform 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 bundled development kit is supplied with the following:

- AT43DK380 Main Development Platform
- AT43DK380 Processor Daughter Card
- Female-female DB9 Null-modem Cable
- 2m Fully Rated USB Cable
- 5V regulated power supply
- Atmel USB CD-ROM with Software and Documentation
- CD-ROM of Atmel Products

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:

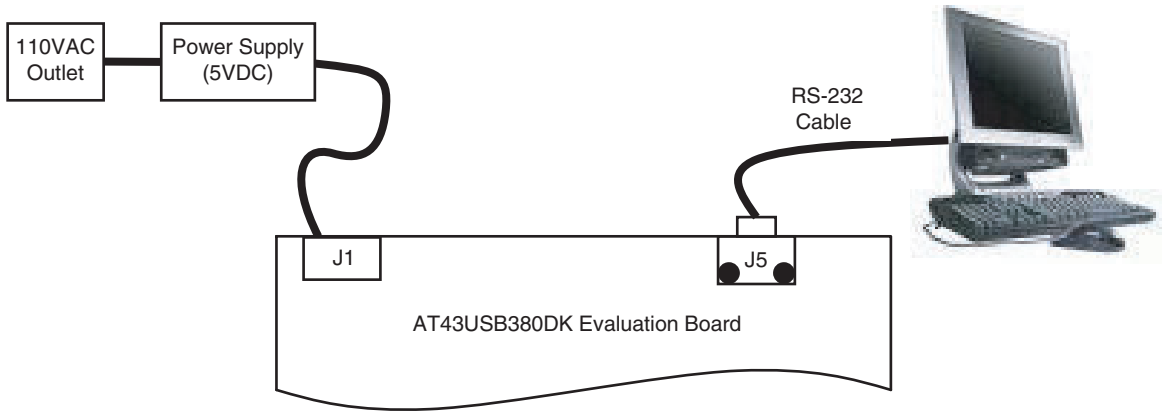
- 486 Processor (Pentium® is recommended)
- 128 MB RAM
- 10 MB Free Hard Disk Space
- Windows® 98/2000/ME/XP
- RS-232 Port (COM port)

In order to use the AT43DK380 Development Platform with the tools provided, one of the Processor Daughter Cards (PDC) needs to be properly configured and plugged in. The hardware test will only work if the PDC is plugged in.

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 AT43DK380



Connect the AT43DK380 Evaluation Board as follows:

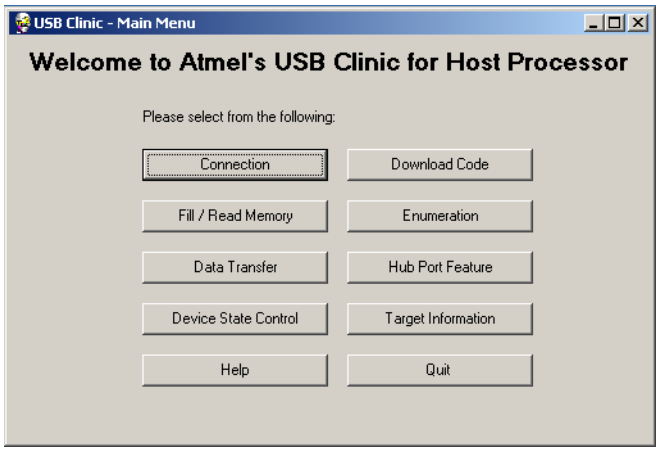
1. Connect the serial cable from J5 on the evaluation board to a COM port on the PC.
2. Connect the AC input connector of the power supply to an AC wall outlet (110 VAC).
3. Connect the DC output connector of the power supply to J1 on the evaluation board.

2.5 Installing AT43DK380 and Starting Up the USB Clinic

To install the AT43DK380:

1. Insert the “Atmel AT43DK380” CD into the CD-ROM drive of the PC or notebook, double-click on the **MSI** file and follow the instructions.
2. After installation, the AT43USB380 documents, firmware, and software should be installed on the **C:\Program Files\ATMEL USB\AT43DK380_Date**, if the default installation directory is used. One of the software programs installed is the USB Clinic. It is an integrated diagnostic and debugging tool that provides communication between the AT43DK380 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 AT43DK380 Development Platform connection. Prior to going to the **Connection** menu, please make sure that the AT43DK380 is physically connected to the PC with the supplied serial cable or any standard serial cable.

Figure 2-3. USB Clinic - Connection

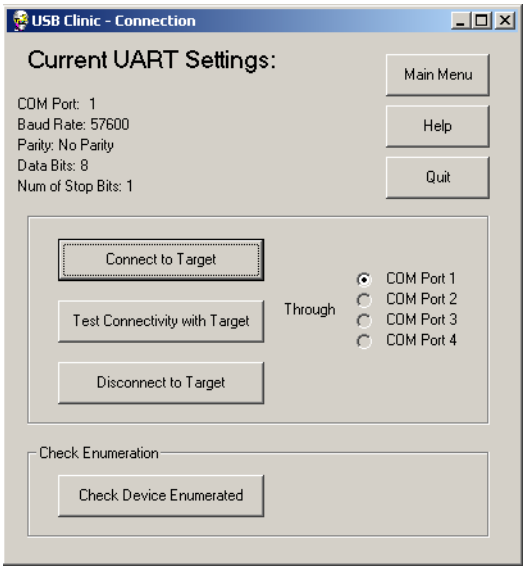
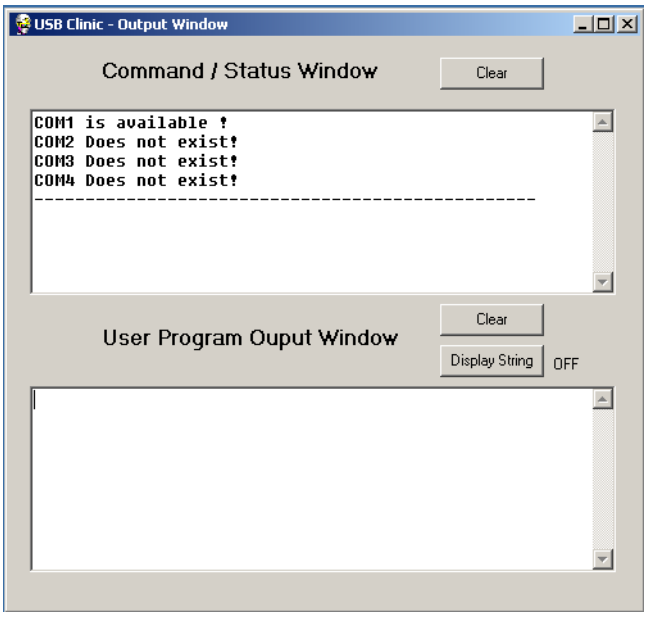
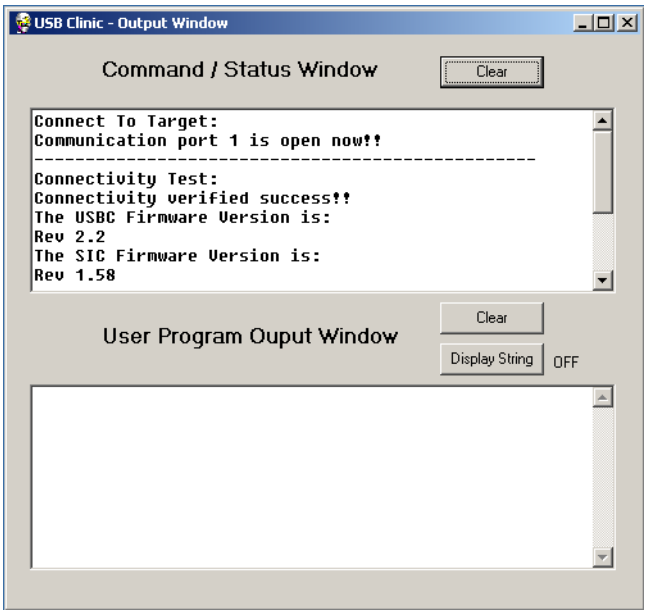


Figure 2-4. COM Port Availability



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 the **Connect to Target** button to open the selected COM port. To verify connectivity between the selected COM port and the AT43DK380 Development Platform, 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 AT43DK380 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

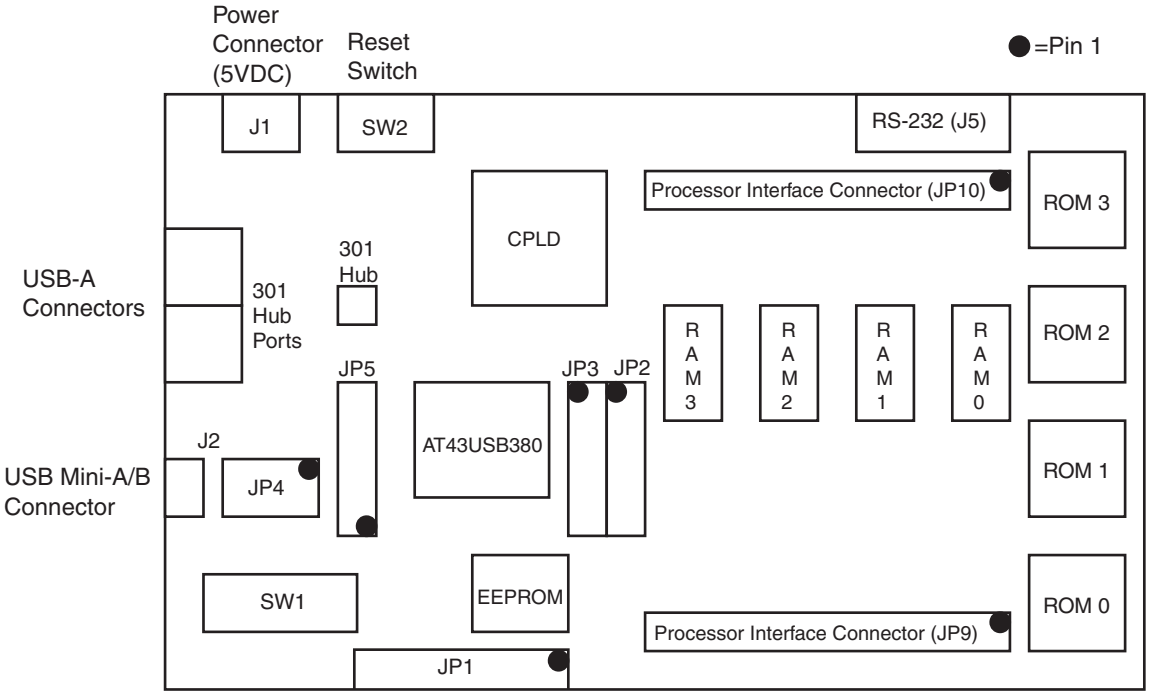
The AT43DK380 Development Platform comes pre-configured as a USB Host in Flash (Standalone) mode. Alternative settings are available through jumper configurations. This section gives the basic description about the hardware components and the jumper settings of the AT43DK380 Development Platforms.

Section 10 contains the complete AT43DK380 schematics and bill of materials.

3.1 Description of the Main Development Platform

The AT43DK380 Main Development Platform consists of the AT43USB380 Host/Function/OTG processor, 1 Mbytes of Flash ROM, 2 Mbytes of SRAM, 4 port USB Hub, USB type MINI-A/B port, RS-232 serial port, and pre-programmed CPLD and glue logic for configurable bus width interface. The AT43USB380 connects to the system processor through the configurable bus width interface connectors. The RS-232 port provides access to the system processor and the AT43USB380, and the In-System Programming to SRAM.

Figure 3-1. AT43USB380 Main Development Platform Components Layout



3.1.1 AT43DK380 Processor Interface Connector

The AT43DK380 Main Development Platform provides a generic processor interface to allow a wide variety of processors to communicate with the on-board RAM, ROM and the AT43USB380 USB Host. The pin assignments and descriptions for the connectors are as follows:

Table 3-1. J9: Processor Interface Connector 1

Pin #	Pin Name	Signal Source	Pin Description
1, 2, 49, 50	VCC_3.3		Power: 3.3 VDC
3	D0	Bi-Direction	Data Bit 0
4	D1	Bi-Direction	Data Bit 1
5	D2	Bi-Direction	Data Bit 2
6	D3	Bi-Direction	Data Bit 3
7	D4	Bi-Direction	Data Bit 4
8	D5	Bi-Direction	Data Bit 5
9	D6	Bi-Direction	Data Bit 6
10	D7	Bi-Direction	Data Bit 7
13	D8	Bi-Direction	Data Bit 8
14	D9	Bi-Direction	Data Bit 9
15	D10	Bi-Direction	Data Bit10
16	D11	Bi-Direction	Data Bit 11
17	D12	Bi-Direction	Data Bit 12
18	D13	Bi-Direction	Data Bit 13
19	D14	Bi-Direction	Data Bit 14
20	D15	Bi-Direction	Data Bit 15
23	D16	Bi-Direction	Data Bit 16
24	D17	Bi-Direction	Data Bit 17
25	D18	Bi-Direction	Data Bit 18
26	D19	Bi-Direction	Data Bit 19
27	D20	Bi-Direction	Data Bit 20
28	D21	Bi-Direction	Data Bit 21
29	D22	Bi-Direction	Data Bit 22
30	D23	Bi-Direction	Data Bit 23
33	D24	Bi-Direction	Data Bit 24
34	D25	Bi-Direction	Data Bit 25
35	D26	Bi-Direction	Data Bit 26
36	D27	Bi-Direction	Data Bit 27
37	D28	Bi-Direction	Data Bit 28
38	D29	Bi-Direction	Data Bit 29
39	D30	Bi-Direction	Data Bit 30
40	D31	Bi-Direction	Data Bit 31

Table 3-1. J9: Processor Interface Connector 1

Pin #	Pin Name	Signal Source	Pin Description
43	NDACK	System Processor	DMA Acknowledge
44	NDREQ	AT43USB380	DMA Request
45	NWAIT	AT43USB380	
46	INTREQ	AT43USB380	Interrupt Request
47	NRESET	AT43DK380 (Main Development Platform and PDCs)	System Reset, Active Low
48	NWAKEUP	System Processor	AT43USB380 Wake-Up request from Suspend Mode
11, 12, 21, 22, 31, 32, 41, 42	GND		Power: Ground

Table 3-2. J10: Processor Interface Connector 2

Pin #	Pin Name	Signal Source	Pin Description
1, 2, 49, 50	VCC_3.3		Power: 3.3 VDC
3	A0	System Processor	Address Bit 0
4	A1	System Processor	Address Bit 1
5	A2	System Processor	Address Bit 2
6	A3	System Processor	Address Bit 3
7	A4	System Processor	Address Bit 4
8	A5	System Processor	Address Bit 5
9	A6	System Processor	Address Bit 6
10	A7	System Processor	Address Bit 7
11	A8	System Processor	Address Bit 8
12	A9	System Processor	Address Bit 9
15	A10	System Processor	Address Bit 10
16	A11	System Processor	Address Bit 11
17	A12	System Processor	Address Bit 12
18	A13	System Processor	Address Bit 13
19	A14	System Processor	Address Bit 14
20	A15	System Processor	Address Bit 15
21	A16	System Processor	Address Bit 16
22	A17	System Processor	Address Bit 17
23	A18	System Processor	Address Bit 18
24	A19	System Processor	Address Bit 19
25	A20	System Processor	Address Bit 20
26	EXTRA1	CPLD	Additional signal to/from CPLD
29	NRD	System Processor	Read Enable. Active Low

Table 3-2. J10: Processor Interface Connector 2

Pin #	Pin Name	Signal Source	Pin Description
30	EXTRA2	CPLD	Unused signal to/from CPLD
31	nWBE0	System Processor	Write Byte 0. Active Low
32	nWBE1	System Processor	Write Byte 1. Active Low
33	nWBE2	System Processor	Write Byte 2. Active Low
34	nWBE3	System Processor	Write Byte 3. Active Low
35	nCS0	System Processor	Chip Select 0. Active Low
36	nCS1	System Processor	Chip Select 1. Active Low
37	nCS2	System Processor	Chip Select 2. Active Low
38	nCS3	System Processor	Chip Select 3. Active Low
41	PCLK	System Processor	Processor Clock
42	EXTRA3	CPLD	Unused signal to/from CPLD
43	NC		No Connect
44	NC		No Connect
45	UTXD0		
46	URXD0		
47	UDTR0		
48	UDSR0		
13, 14, 28, 28, 39, 40	GND		Power: Ground

Section 10 of the User's guide contains the bus interface CPLD VHDL code.

3.1.2 Jumper Settings

The AT43DK380 Main Development Platform supports two modes of operation, Host Mode or On-The-Go (OTG) Mode. Host Mode is the default setting. In Host Mode, the AT43USB380 USB root port is directly connected to a 4-Port USB Hub (AT43301). Thus, allowing the AT43USB380 to communicate to multiple USB devices. Downstream USB devices are connected through J3 and J4. In OTG Mode, the AT43USB380 supports all requirements specified in the USB On-The-Go Supplement. Depending on the state of the ID signal through cable connection, the AT43USB380 is either configured as an USB Host or Device. Table 3-3 shows the configuration settings for mode selection.

Table 3-3. AT43DK380 Mode Configuration

	Host Mode with 4-Port USB Hub	OTG Mode (Single Port)
JP4	Close: 1-3, 2-4	Close: 3-5, 4-6

Configuration options to the AT43USB380 are selected through SW1. SW1 determines the bus sizing of the system processor, and the source of the AT43USB380 binary code.

Other switch settings are for debugging purposes only and it is NOT intended for the end user. Switch settings for SW1 is illustrated in Table 3-4.

Table 3-4. AT43USB380 SW1 Configuration

	Default Setting	Description
SW1.1	ON	Test Point 0 (Test Mode Only)
SW1.2	ON	Test Point 2 (Test Mode Only)
SW1.3	ON	AT43USB380 binary code source OFF: Upload from EEPROM ON : Download through System Processor
SW1.4	ON	Test Point 4 (Test Mode Only)
SW1.5 SW1.6	Depending on the bundled Processor Daughter Card bus width interface	AT43USB380 Bus Width Selection SW1.5 SW1.6 OFF don't care : 32 Bit interface ON OFF : 16 Bit interface ON ON : 8 Bit interface
SW1.7	ON	Clock Select (Test Mode Only)
SW1.8	OFF	TRESET (Test Mode Only)

The AT43DK380 Main Development Platform supports various addressing schemes to the AT43USB380. Depending on the system processor used, JP5 can be configured to match the address requirements of the system processor and that of the AT43USB380. AT43USB380's addressing scheme is on the byte boundary. For most processors with byte boundary addressing, the default setting is used. For some processors with internal hardware address shifting to word or long word boundary, either use software to shift the address back to byte boundary, or use JP5 for hardware shifting scheme. Do not use both software and hardware shifting. Table 3-5 shows the jumper settings for byte, word, and long word addressing schemes.

Table 3-5. AT43USB380 Address Configuration

	Default: Byte Addressing	Word Addressing	Long Word Addressing
JP5	Close: 2-4, 3-5, 6-8, 7-9,10-12, 11-13, 14-16, 15-17	Close: 3-4, 5-6, 7-8, 9-10,11-12, 13-14, 15-16, 17-18	Close: 4-6, 5-7, 8-10, 9-11, 12-14, 13-15, 16-18, 17-19

For an 8/16-bit mode interface, the AT43USB380's upper 16 data bits can be configured through software to be high, low, or floating. For 32-bit mode, the upper 16 data bits are connected to the system. The AT43DK380 Main Development Platform's JP2 and JP3 can be used to connect the upper 16 bits to the system or left floating to match the bus width requirements of the system processor. Table x-x shows the jumper settings for different bus width interfaces.

Table 3-6. AT43USB380 D[31:16] Configuration

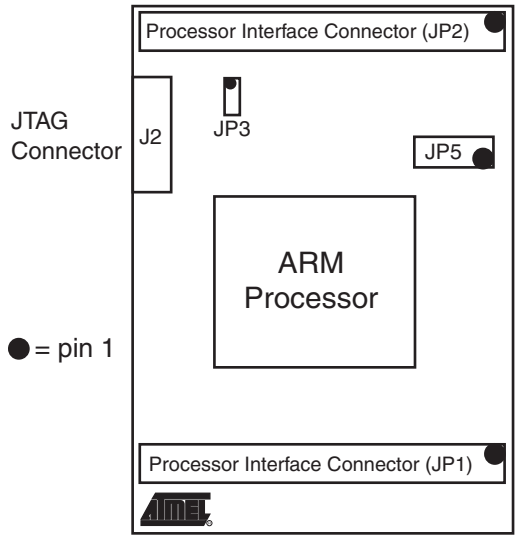
	Default: 8/16 Bit Interface	32 Bit Interface
JP2	Open: all	Close: 1-2, 3-4, 5-6, 7-8, 9-10, 11-12, 13-14, 15-16, 17-18, 19-20
JP3	Open: all	Close: 1-2, 3-4, 5-6, 7-8, 9-10, 11-12, 13-14, 15-16, 17-18, 19-20

The AT43DK380 Main Development Platform supports two modes of operation, the default Flash (or Standalone) Mode and the ICE (In-circuit Emulator) mode. In Flash Mode, the AT43DK380 Main Development Platform executes code from its on-board flash ROM. While in ICE Mode, a In-Circuit Emulator for the system processor can be connected through the JTAG connector of the Processor Daughter Card (PDC) to further facilitate code development. For the jumper settings for the different modes, please refer to the corresponding PDC section.

3.2 AT43DC380-PDC1 Processor Daughter Card

The AT43DC380-PDC1 Processor Daughter Card has a 32-bit bus interface connection. It comes pre-configured in Flash (Standalone) Mode. Alternative settings are available through jumper configurations. Complete descriptions of the jumper settings are explained in Section 3.2.1 on page 6.

Figure 3-2. AT43DC380-PDC1 Processor Daughter Card Layout



The AT43DK380-PDC1 Processor Daughter Card consists of a 32-bit ARM® processor with JTAG accessibility (J2). Eight GPIO pins from the ARM processor are tied to a connection header for interfacing with AT43DK380 Main Development Platform, and eight other GPIOs are tied to JP6 for monitoring the ARM processor. In addition, the AT43DC380-PDC1 Processor Daughter Card provides the footprint for a MII connector (J2) if a network connection is required. Communication to the AT43DK380 Main Development Platform is established through a proprietary interface. Section 10 of this User Guide contains the complete AT43DK380-PDC1 schematics and bill of material.

3.2.1 PDC1 Jumper Settings

The AT43DK380-PDC1 Processor Daughter Card supports two modes of operations, the default Flash (or Standalone) Mode and the ICE (In-Circuit Emulator) Mode. In Flash Mode, the AT43DK380-PDC1 Processor Daughter Card executes code from the Flash ROM of the AT43DK380 Main Development Platform. While in ICE Mode, an In-Circuit Emulator for the ARM processor can be connected through J2 to facilitate code develop-

ment. The corresponding jumper settings for the different modes are shown in Table 3-7.

Table 3-7. AT43DK380-PDC1 Development Mode Selection

	Flash Mode (Default)	ICE Mode
JP3	Close: 1-2	Close: 2-3
JP5	Close: 3-5, 7-9	Close: 1-3, 5-7

The ARM processor can operate at either 10 MHz or 50 MHz. Table x-x contains the jumper settings for the clock selection for the ARM processor.

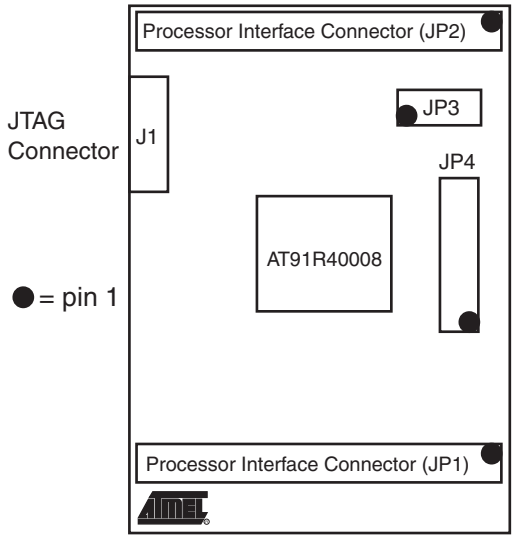
Table 3-8. ARM Processor Clock Select

	10 MHz	50 MHz
JP6	Open: 2-4	Close: 2-4

3.3 AT43DC380-PDC2 Processor Daughter Card

The AT43DC380-PDC2 Processor Daughter Card has a 16-bit bus interface connection. It comes pre-configured in Flash (Standalone) Mode. Alternative settings are available through jumper configurations. Complete descriptions of the jumper settings are explained in Section 3.3.1 on page 7.

Figure 3-3. AT43DC380-PDC2 Processor Daughter Card Layout



The AT43DK380-PDC2 Processor Daughter Card consists of the AT91R40008, a 16-bit ARM processor, with JTAG accessibility (J1). Six GPIO pins from the AT91R40008 processor are tied to a connection header for interfacing with the AT43DK380 Main Development Platform, and sixteen GPIO pins (with some overlap to the six GPIO pins mentioned) are tied to JP4 for monitoring the AT91R40008 processor. Communication to the AT43DK380 Main Development Platform is established through a proprietary interface. Section 10 of this User Guide contains the complete AT43DK380-PDC2 schematics and bill of materials.

3.3.1 PDC2 Jumper Settings

The AT43DK380-PDC2 Processor Daughter Card supports two modes of operations, the default Flash (or Standalone) Mode and the ICE (In-Circuit Emulator) Mode. In Flash Mode, the AT43DK380-PDC2 Processor Daughter Card executes code from the Flash

ROM of the AT43DK380 Main Development Platform. While in ICE Mode, an In-Circuit Emulator for the ARM processor can be connected through J1 to facilitate code development. The corresponding jumper settings for the different modes are shown in Table 3-9.

Table 3-9. AT43DK380-PDC2 Development Mode Selection

	Flash Mode (Default)	ICE Mode
JP3	Close: 3-5, 7-9	Close: 5-7

The External Interrupt signal can come from P9/IRQ0 or P12/FIRQ:

Table 3-10. AT43DK380-PDC2 External Interrupt Selection

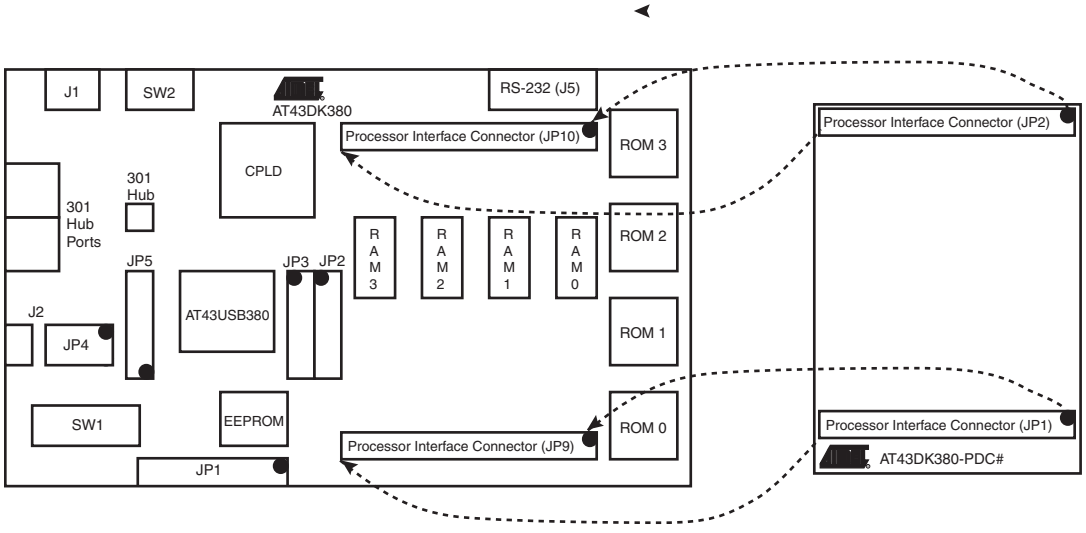
	P9/IRQ0 (Default)	P12/FIRQ
JP3	Close: 4-6	Close: 2-4

3.4 AT43DK380 Main Development Platform to PDC Connection

The AT43DK380-PDCx-xx provides two connectors that directly connect to the AT43DK380 Main Development Platform.

Making the connection:

1. Take all safety and electrostatic precautions.
2. Mate together the connectors with labels JP1 and JP2 on the AT43DK380-PDCx-xx to JP9 and JP10 on the AT43DK380 Main Development Platform respectively. If proper connection is established, the Atmel logos on both the AT43DK380-PDCx-xx and the AT43DK380 Main Development Platform have the same orientation.



Section 4

USB Clinic

USB Clinic is a Graphical User Interface (GUI) based diagnostic and debugging tool that provides basic control of the AT43USB380 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 AT43USB380 internal memory, manipulation of USB device enumeration, data transfer through various endpoints, hub feature selection, and USB device state control. All of the AT43USB380 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 AT43USB380 Library.

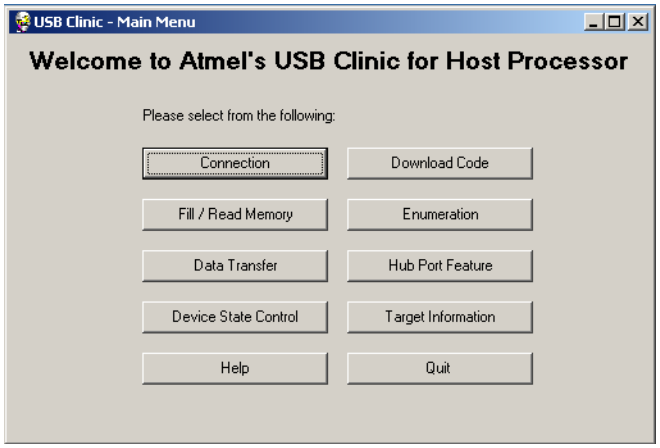
Atmel will continue to extend the capability of the USB Clinic. The following section is intended for USB Clinic Rev. 1.2. For software upgrades, please refer to the USB section of the Atmel web site at http://www.atmel.com/ad/plugplayhost.

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)

Figure 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 AT43USB380 Library APIs directly. For detailed usage regarding those function calls, please refer to the “AT43USB380 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 4-3).

4.2.2 User Program Output Window

The **User Program Output Window** displays text messages and outputs from the user developed code, when the **Display String** button is **ON**. Press the **Display String** button once to enable the message display to be **ON**, and press again to disable it to **OFF**. In the user source code, users will need to insert **\n** at the end of every message printing statement to allow the messages to be displayed in the **User Program Output Window**, and to be displayed properly. Pressing the **Clear** button clears the content of the window (see Figure 4-3).

While using the USB Clinic features with printing user message statement, the user would need to pay attention not to insert the message string in the USB Clinic source code or return data path, as they share the same communication channel, and the message statement might corrupt the return data from normal USB Clinic operation.

4.3 Connection

From the **Main Menu**, the **Connection** button allows testing of the RS-232 serial port and the AT43DK380 Development Platform connection. Prior to go to the **Connection** menu, please make sure that the AT43DK380 is physically connected to the PC with the supplied serial cable.

Click on the **Connection** button and the windows in Figure 4-2 and Figure 4-3 appear.

Figure 4-2. USB Clinic - Connection Window

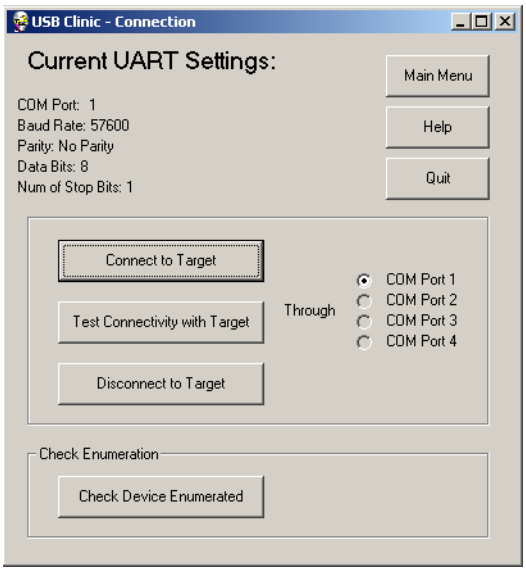
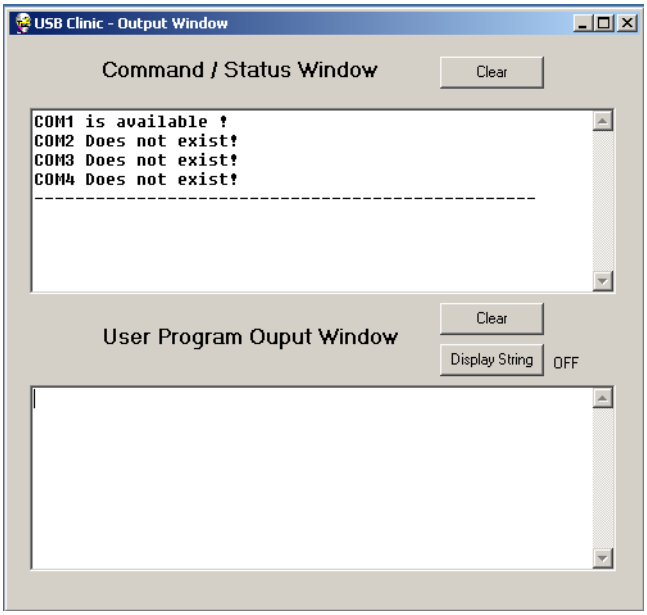


Figure 4-3. USB Clinic - Output Window



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 AT43DK380 Development Platform, click on **Test Connectivity with Target** and look for **Connection verified** in the **Output Window**. The **Test Connectivity with Target** also displays the stacked AT43DK380 firmware and library revisions.

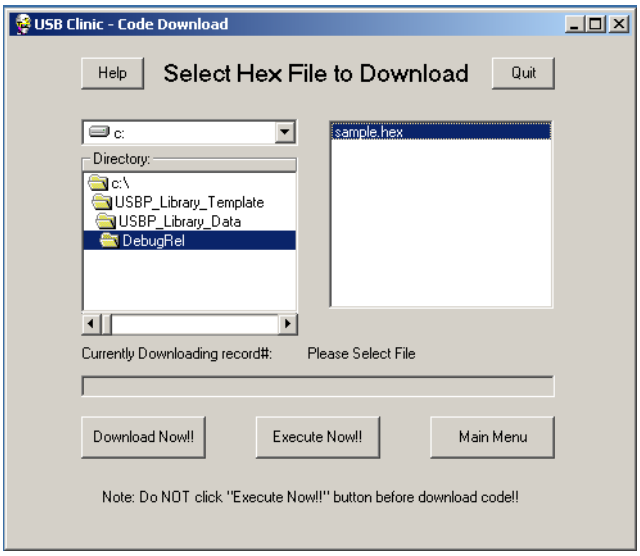
4.3.2 Check Device Enumerated

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. AT43USB380 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 AT43USB380 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 AT43DK380 firmware. The AT43DK380 boots off from the system processor and its firmware is stored in an external flash attached to the system processor.

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



The ARM® code requires 32-bit Intel® Hex File Format. 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\AT43DK380_Data\380 DK Board Reference** directory.

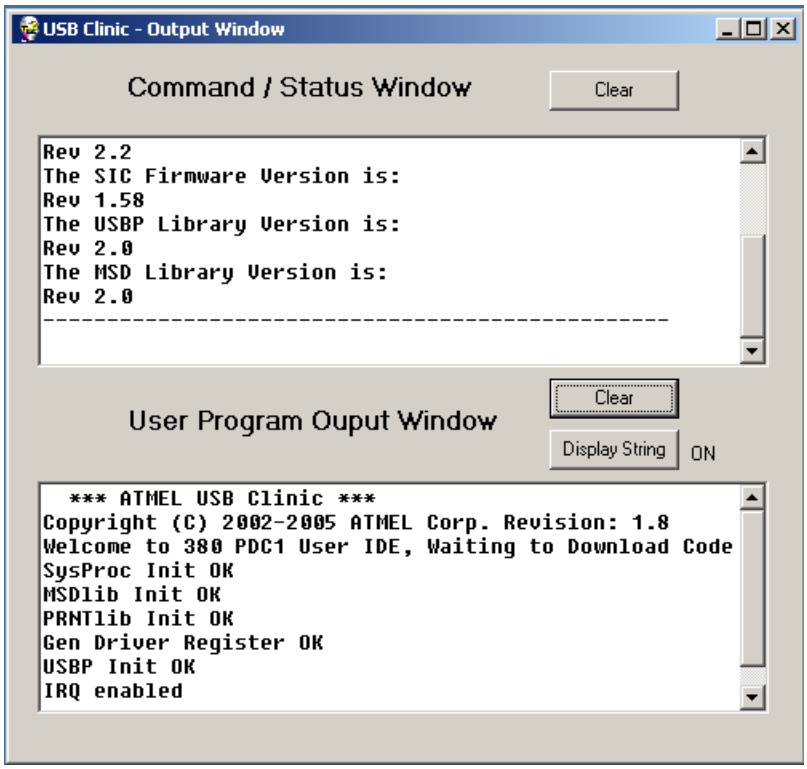
Prior to downloading, the user should set the **Display String** to **OFF** in the **User Program Output Window**, as it might slow down the downloading process or corrupt the acknowledge message.

Once code download is complete, the user has the option of downloading again without executing the code or executing the downloaded code by clicking on the **Execute Now!!** button. The user should only click on **Execute Now!!** after code download is complete. Prior to downloading, the user should set the **Display String** to **ON** in the **User Program Output Window**, if the user had inserted the print message statement in the source code and would like to monitor the program flow through the printed message.

If the downloaded code contains the USB Clinic application code, as in the case of the sample download hex file, the user can still use the USB Clinic functions. Otherwise only the **Display String** feature is enabled, if it had been set to be **ON** prior to clicking **Execute Now!!**.

To terminate the execution, either press the reset button on the AT43USB380 Development Board or power-cycle the board. The downloaded user code will be erased from the SRAM after reset or power-cycle.

Figure 4-5. USB Clinic - User Program Output Window



4.5 Memory

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

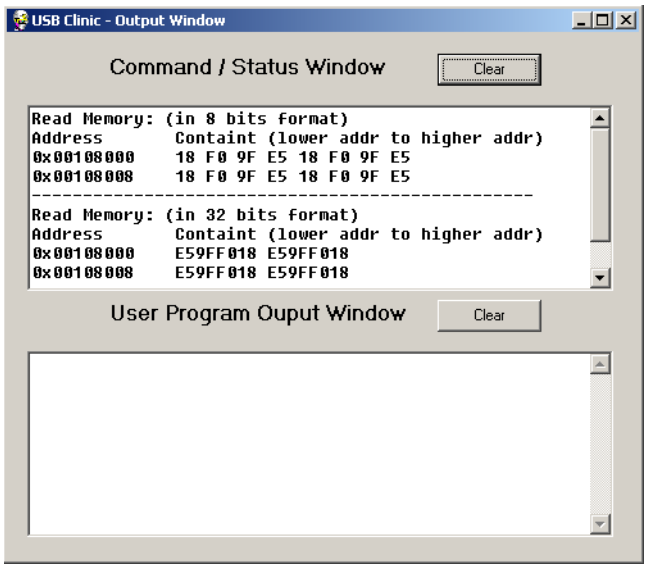
4.5.1 Fill Memory

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 AT43DK380 Development Platform ranges up to 0x3FFFFFFF. The user will not be able to read or write to memory addresses beyond this limit.

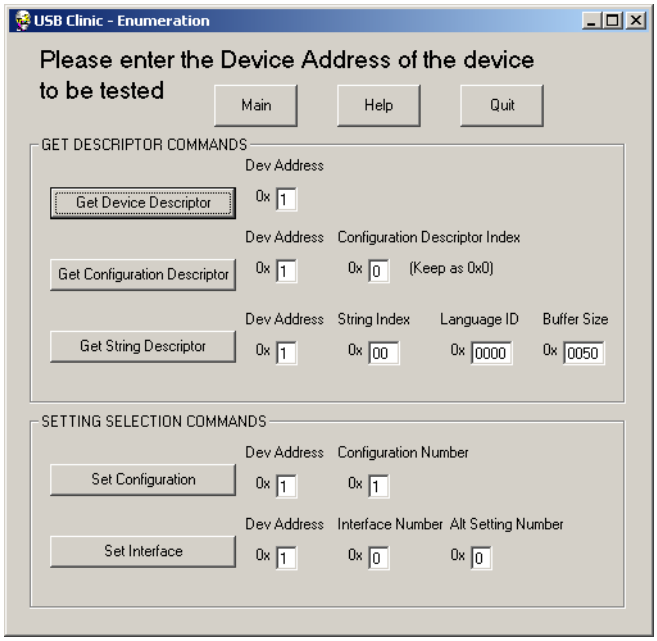
Figure 4-6. USB Clinic - Read Memory



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 4-7, they are the **Get Device Descriptor**, **Get Configuration Descriptor**, **Get String Descriptor**, **Set Configuration**, and **Set Interface**.

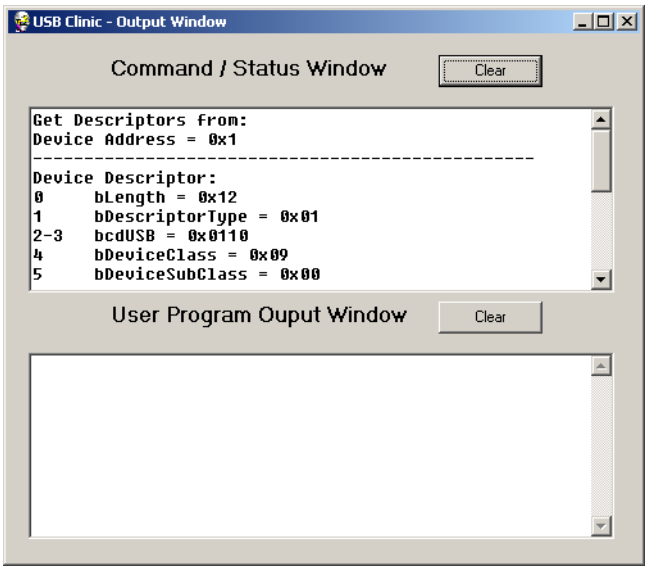
Figure 4-7. USB Clinic Enumeration



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 AT43DK380 Development Platform. 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 AT43DK380 Host.

Figure 4-8. USB Clinic - Get Device Descriptor



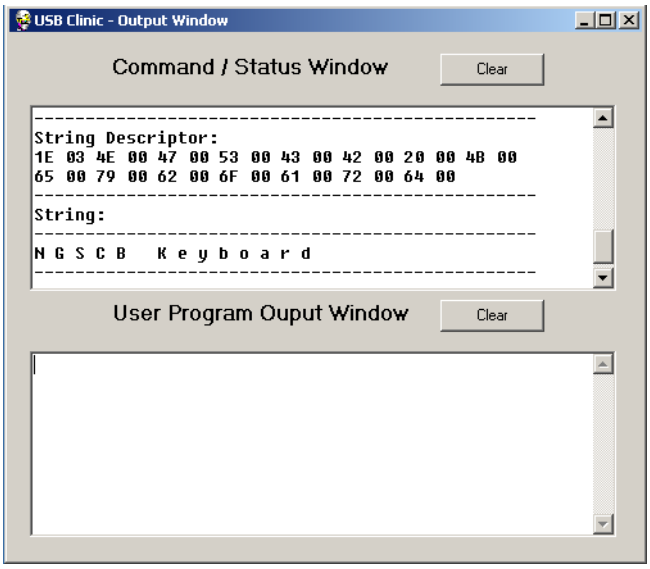
4.6.2 Get Configuration Descriptor

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 AT43DK380 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 4-9. 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 AT43USB380 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 AT43DK380 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

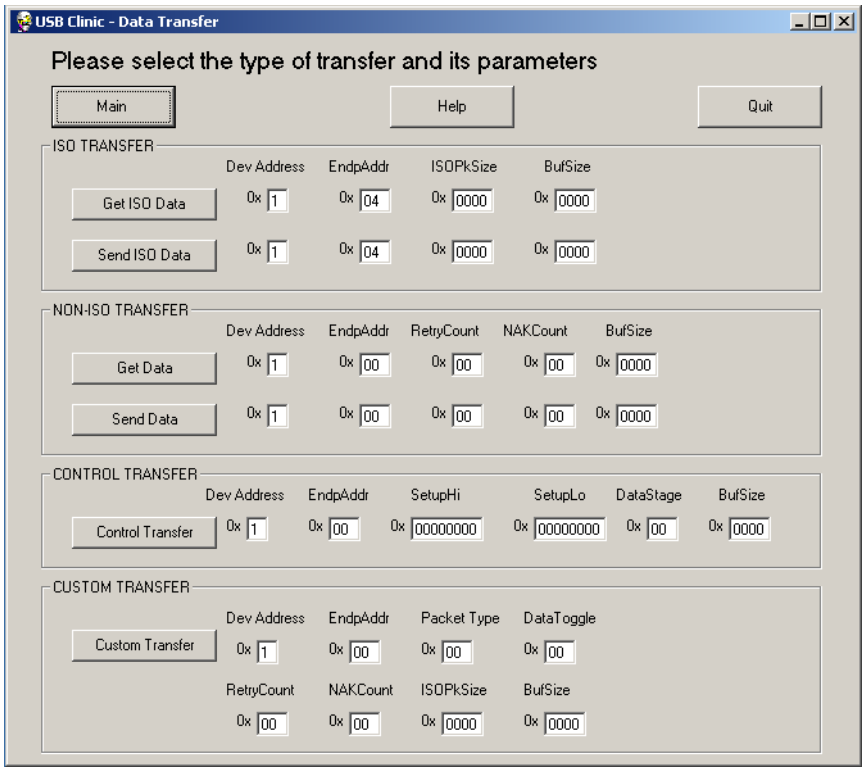
For configurations with multiple interfaces/alternate settings, the **Set Interface** command allows the user to set the particular interface/alternate interface setting for the configuration on the target device. To set the right interface, the user will need to assign the device address, the interface 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 4-10. 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 “AT43USB380 Software Development Guide for Host Mode”.

Table 4-1. 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 4-2), Data Toggle (see Table 4-3), 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 “AT43USB380 Software Development Guide for Host Mode”.

Table 4-2. Packet Types

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

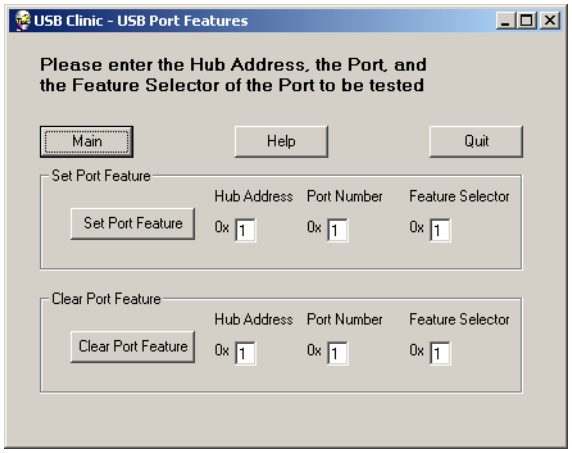
Table 4-3. Data Toggle

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 4-11. 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 4-3) to be enabled. For more details, please refer to Chapter 11 of the “USB Specification Rev 2.0”, and to the “AT43USB380 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 4-4) to be enabled. For more details, please refer to Chapter 11 of the “USB Specification Rev 2.0”, and to the “AT43USB380 Software Development Guide for Host Mode”.

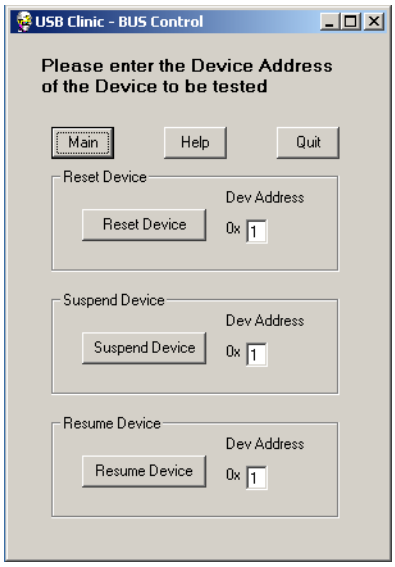
Table 4-4. Feature Selector Values

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 4-12. 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 AT43DK380 Host. For details about how to set the **DEVICE_REMOTE_WAKEUP** feature, please refer to the “USB Specification Rev 2.0”, and to the “AT43USB380 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 AT43DK380 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 AT43DK380 Development Platform 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 AT43DK380 Development Kit

Developing firmware for the AT43DK380 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 AT43DK380 board's flash sets. For information about generating the Flash mode firmware, please refer to Section 9. The ICE mode firmware is used by the In-Circuit Emulator that runs code on the AT43DK380 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 AT43DK380 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 8.

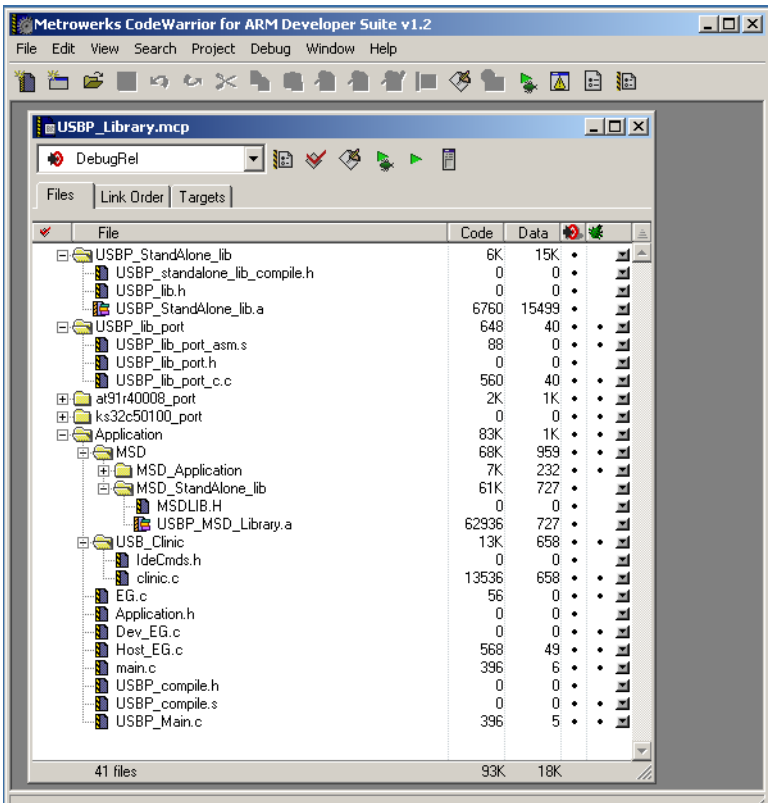
5.1 Sample Directory and File Structure

The default sample code directory for the **USBP_Library_Rev_x.x_Template** resides in **C:\Program Files\Atmel USB\AT43DK380_Date\380 Software**.

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

From the ARM Developer Suite (ADS), go to **File** > **Open** > **C:\Program Files\Atmel USB\AT43DK380_Date\380 Software** > **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_compile.h
      |-- USBP_StandAlone_lib.a
|
|-- USBP_lib_port
```

```
      |--  USBP_lib_port_h.h
      |--  USBP_lib_port_asm.s
      |-- USBP_lib_port_c.c
|
|-- at91r40008_port
      |-- at91r40008_irq
      |-- at91r40008_include
      |-- at91r40008_start.s
      |-- at91r40008_config.h
      |-- eb40A.h
      |-- at91r40008_init.c
      |-- at91r40008_heap.c
      |-- at91r40008_scat_l.scf
|
|-- ks32c50100_port
      |-- ks32c50100_start.s
      |-- ks32c50100_config.h
      |-- ks32c50100_config.s
      |-- ks32c50100_init.c
      |-- ks32c50100_heap.c
      |-- ks32c50100_scat_l.scf
|
|-- Application
      |
      |-- MSD
           |-- msdlib.h
           |-- USBP_MSD_Library.a
           |-- MSD_Application
      |-- USB_Clinic
           |-- IdeCmds.h
           |-- clinic.c
      |-- Application.h
      |-- USBP_compile.h
      |-- USBP_compile.s
      |-- USBP_Main.c
      |-- main.c
      |-- EG.c
      |-- Host_EG.c
      |-- Dev_EG.c
```

A brief description of the folders and files follow.

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 Developer Suite) version 1.2.

2. **USBP_StandAlone_Lib**

   The USBP_StandAlone_lib.a is the binary library that contains the AT43USB380's firmware and all the high/low level APIs. The USBP_lib.h and USBP_standalone_lib_coimpile.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 USBP Library port. It contains various definitions required for USB processor's configuration.

   b. **USBP_lib_port_asm.s:** This file contains the processor-specific assembly 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. **at91r40008_port**

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

   a. **at91r40008_start.s:** This file contains the startup code for at91r40008 processor.

   b. **at91r40008_config.h:** This file contains at91r40008 processor definitions.

   c. **eb40A.h:** This file contains the system manager initialization routine for at91r40008 processor.

   d. **at91r40008_init.c:** This file contains various peripheral initialization functions for at91r40008 processor.

   e. **at91r40008_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 Internal SRAM areas.

   f. **at91r40008_irq:** This file contains interrupt and UART initializations for the at91r40008 processor.

   g. **at91r40008_include:** This folder includes the inline functions and register setting and definition for the at91r40008 processor.

   h. **at91r40008_heap.c:** This file contains the heap space specification for the DK.

5. **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 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 Internal SRAM areas.

   f. **ks32c50100_heap.c:** This file contains the heap space specification for the DK.

6. **Application**

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

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 MSD driver. It contains the MSD APIs for application. 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 application, but it does not contain a sample application code yet.

b. **USB_Clinic:** This directory contains the source and header files for the USB Clinic.

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

d. **USBP_compile.h:** This file specifies the platform and mode for the project file for the C files.

e. **USBP_compile.s:** This file specifies the platform and mode for the project file for the assembly files.

f. **USBP_Main.c:** This file contains the reference USBP project initialization and application main routine.

g. **main.c:** This file is for the user's main routine.

h. **EG.c:** This file contains the reference required AT43USB380 processor's initialization setting.

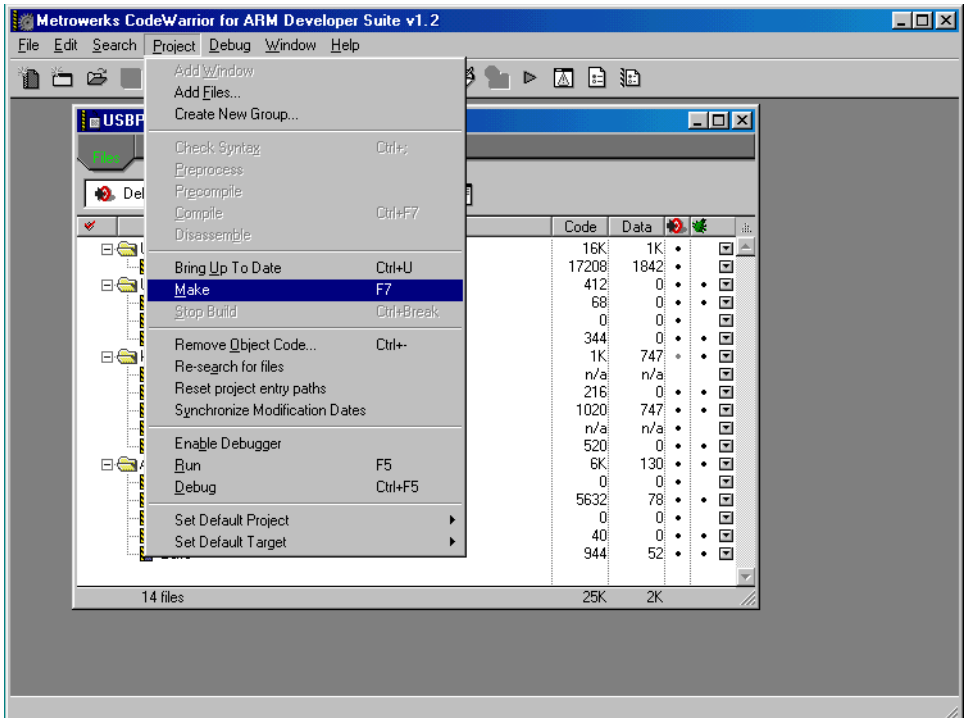
i. **Host_EG.c:** This file contains the reference required AT43USB380 processor's initialization setting for Host mode.

j. **Dev_EG.c:** This file contains the reference required AT43USB380 processor's initialization setting for Device mode.

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**.

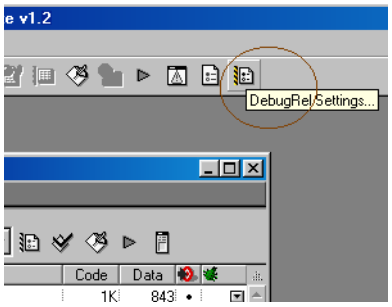
Figure 5-2. Project Make



5.2 ADS Settings

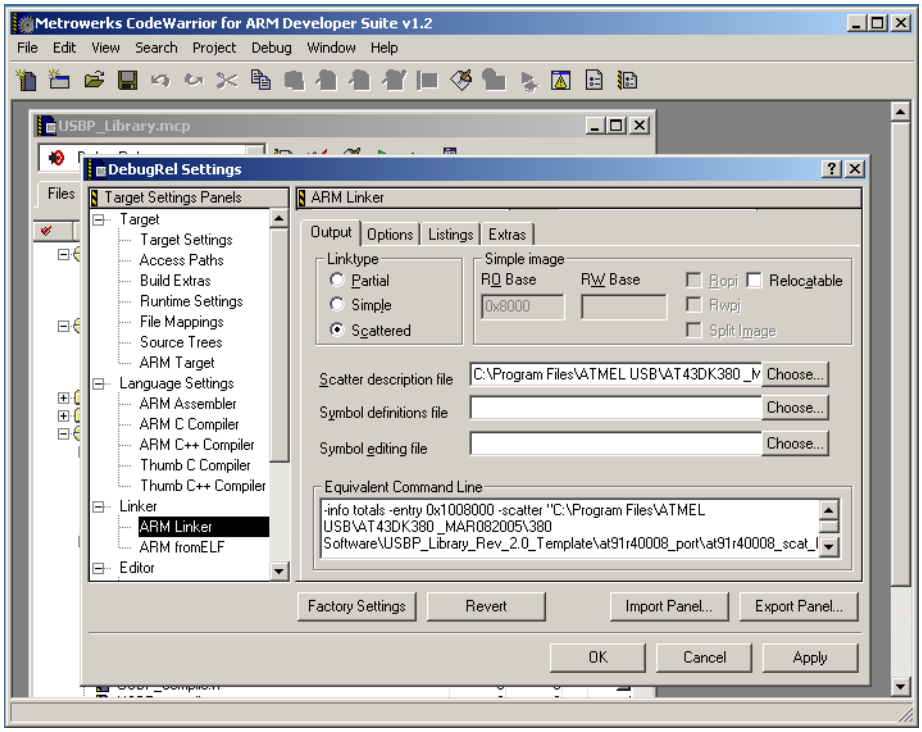
To Make a project, the ADS requires to know the format to compile code and how to link its object. Those information are stored in a scatter file. To assign a scatter file, click on the **DebugRel Setting** icon on the top right corner of the ADS IDE as shown in Figure 5-3.

Figure 5-3. DebugRel Setting Icon



The following window appears on the screen.

Figure 5-4. DebugRel Settings Window



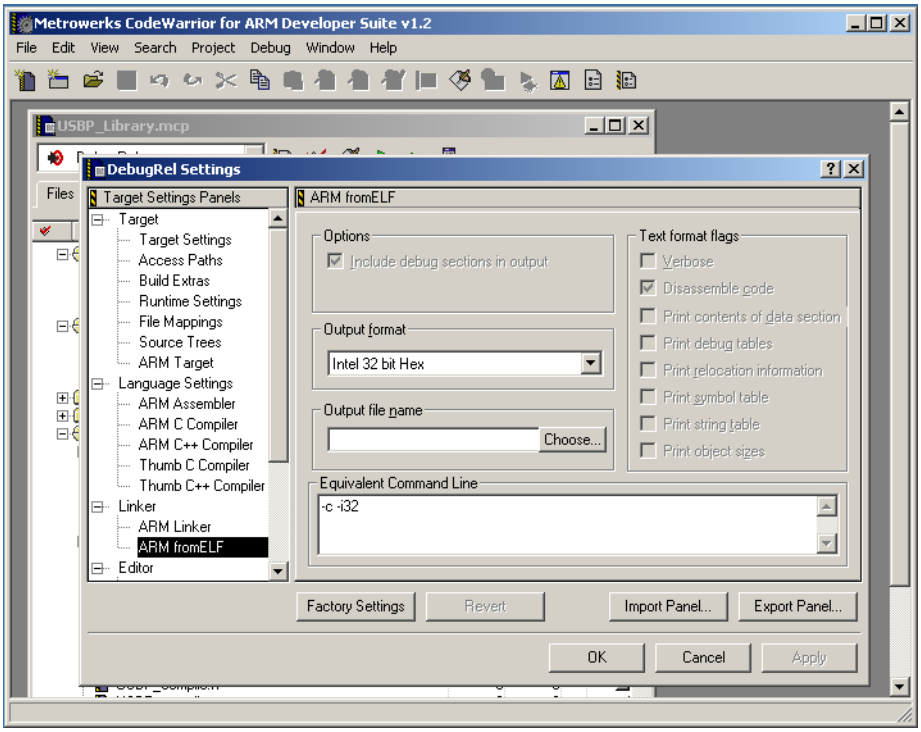
The left-hand side of the **DebugRel Settings** window contains available options including Target, Language Settings and Linker. Go to **Linker** > **ARM Linker** > **Output**. There is a **Scatter description file** prompt in the **ARM Linker** window. Click on **Choose**, and go to **USBP_Library_Template** > **ks32c50100_port** or "**at91r40008_port**" directory and select the **ks32c50100_scat_l** or "**at91r40008_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 for PCD1, and 0x1000000 for PCD2. Go to **Linker** > **ARM Linker** > **Options** and look for the **Image entry point** prompt 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.

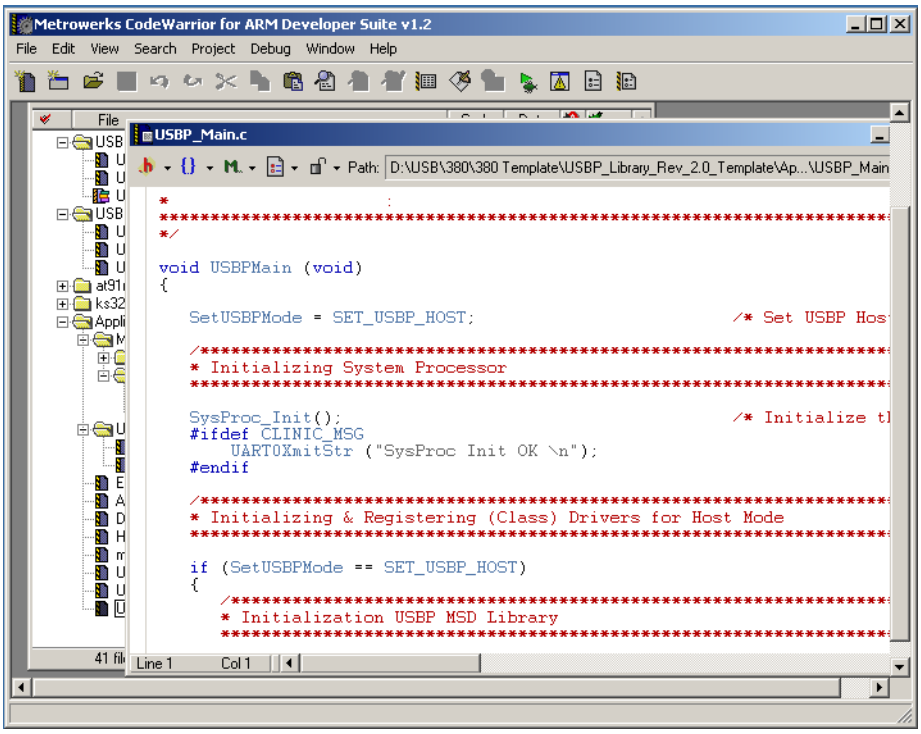
To configure the ADS to create the **Image Map** and **Symbol** files, go to **Linker** > **ARM Linker** > **Listings**. On the **Listings** window, there is an **Image Map** and **Symbols** check box. Check the desired file(s) and enter the file name at the **List file name** prompt.

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

5.3 Modifying a Sample Application

Go back to the project files (see Figure 5-1). In **USBP_Library.mcp**, there is an **USBP_main.c** file. This file contains the main routine. Double-click on the **USBP_main.c** to bring the source code (shown in Figure 5-6)

Figure 5-6. USBP_main.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("SysProc Init OK \n")` to the ARM Download mode firmware (Figure 5-7). During execution, the USB Clinic **Output Window** displays the text as shown in Figure 5-8.

Figure 5-7. Outputting Text Message From Source Code

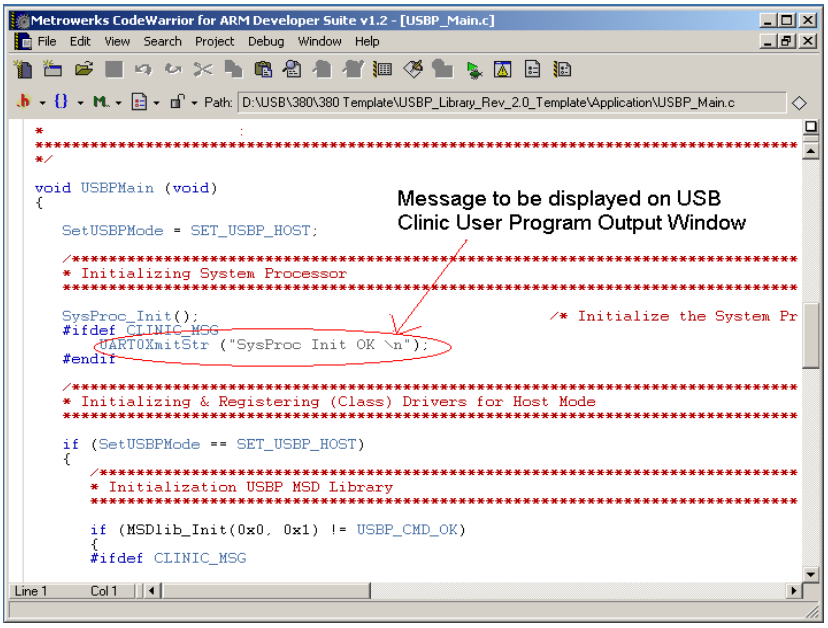
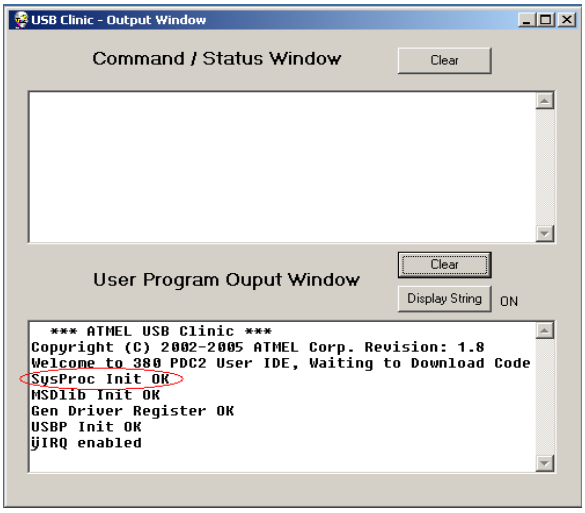


Figure 5-8. USB Clinic - User Program Output Window



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**.

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\AT43DK380_Date380 \Software\USBP_Library_Rev_x.x_Template\USBP_Library_Data\DebugRel**, under the previously assigned file name (see Figure 5-5).

Section 6

Converting Between FLASH and ICE Mode and Download 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 the entry point address. For PDC1 the entry point is **0x0**, for PDC2 the entry point is **0x2000000**.
3. In project view, modify the corresponding PDC scatter file (PDC1: ks32c50100_scat_l.scf; PDC2: at91r40008_scat_l.scf) to comment the ICE mode portion and uncomment the Flash Mode portion.
4. In **USBP_compile.h**, define ***FLASH*** (leave ***DOWNLOAD*** undefined).
5. In **USBP_compile.s**, uncomment ***FLASH*** (leave ***DOWNLOAD*** commented).

For more information on generating hex files for Flash mode, please refer to Section 8.

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 the entry point address. For PDC1 the entry point is **0x0**, for PDC2 the entry point is **0x2000000**.
3. In project view, modify the corresponding PDC scatter file (PDC1: ks32c50100_scat_l.scf; PDC2: at91r40008_scat_l.scf) to comment the Download Mode portion and uncomment the Flash Mode portion.
4. In **USBP_compile.h**, undefine ***DOWNLOAD*** (leave ***FLASH*** defined).
5. In **USBP_compile.s**, comment ***DOWNLOAD*** (leave ***FLASH*** uncommented).

For more information on generating hex files for Flash mode, please refer to Section 8.

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 the entry point address**.** For PDC1 the entry point is **0x108000**, for PDC2 the entry point is **0x1008000**.
3. In project view, modify the corresponding PDC scatter file (PDC1: ks32c50100_scat_l.scf; PDC2: at91r40008_scat_l.scf) to comment the ICE mode portion and uncomment the Download Mode portion.
4. In **USBP_compile.h**, define both ***FLASH*** and ***DOWNLOAD*.**
5. In **USBP_compile.s**, uncomment both ***FLASH*** and ***DOWNLOAD***.

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 the entry point address**.** For PDC1 the entry point is **0x108000**, for PDC2 the entry point is **0x1008000**.
3. In project view, modify the corresponding PDC scatter file (PDC1: ks32c50100_scat_l.scf; PDC2: at91r40008_scat_l.scf) to uncomment the Download Mode portion and comment the Flash Mode portion.
4. In **USBP_compile.h**, define ***DOWNLOAD*** (leave ***FLASH*** defined).
5. In **USBP_compile.s**, uncomment ***DOWNLOAD*** (leave ***FLASH*** uncommented).

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 the entry point address**.** For PDC1 the entry point is **0x1000**, for PDC2 the entry point is **0x1000000**.
3. In project view, modify the corresponding PDC scatter file (PDC1: ks32c50100_scat_l.scf; PDC2: at91r40008_scat_l.scf) to uncomment the ICE mode portion and comment the Flash Mode portion.
4. In **USBP_compile.h**, undefine ***FLASH*** (leave ***DOWNLOAD*** undefined).
5. In **USBP_compile.s**, comment ***FLASH*** (leave ***DOWNLOAD*** commented).

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 the entry point address**.** For PDC1 the entry point is **0x1000**, for PDC2 the entry point is **0x1000000**.
3. In project view, modify the corresponding PDC scatter file (PDC1: ks32c50100_scat_l.scf; PDC2: at91r40008_scat_l.scf) to uncomment the ICE mode portion and comment the Download Mode portion.
4. In **USBP_compile.h**, undefine both ***FLASH*** and ***DOWNLOAD***.
5. In **USBP_compile.s**, comment both ***FLASH*** and ***DOWNLOAD***.

6.8 Summary

6.8.1 ICE Mode

- **Entry Point:** For PDC1 the entry point = 0x1000; for PDC2 the entry point = 0x1000000.

- **Scatter File:** For all PDCs use ICE mode portion.
- **Predefine Constants:** No ***FLASH*** or ***DOWNLOAD*** as predefine constants in both USBP_compile.h and USBP_compile.s.

6.8.2 Flash Mode

- **Entry Point:** For PDC1 the entry point = 0x0; for PDC2 the entry point = 0x2000000.
- **Scatter File:** For all PDCs use Flash mode portion.
- **Predefine Constants:** Define and uncomment ***FLASH*** in USBP_compile.h and USBP_compile.s.

6.8.3 Download Mode

- **Entry Point:** For PDC1 the entry point = 0x108000; for PDC2 the entry point = 0x1008000.
- **Scatter file:** For all PDCs use Download mode portion.
- **Predefine Constants:** Define and uncomment both ***FLASH*** and ***DOWNLOAD*** in both USBP_compile.h and USBP_compile.s.

Section 7

Switching Between PDC1 and PDC2 Project Template

7.1 Switching from PDC1 to PDC2 Project Template

1. In ADS go to **DebugRel** > **Linker** > **ARM Linker** > **Options**
2. In the **Scatter** description file, click **Choose** to choose the **at91r40008_scat_l** from the **at91r40008_port** folder.
3. In **USBP_compile.h**, undefine **KS32C50100** and define **AT91R40008**.
4. In **USBP_compile.s**, comment **KS32C50100** and uncomment **AT91R40008**.
5. In **USBP_lib_port.h**, set the **SET_BUS_WIDTH** to **SET_BUS_WIDTH_16**, and the **SET_XFR_MODE** to **SET_XFR_MODE_DFIFO**.

7.2 Switching from PDC2 to PDC1 Project Template

1. In ADS go to **DebugRel** > **Linker** > **ARM Linker** > **Options**
2. In the **Scatter** description file, click **Choose** to choose the **ks32c50100_scat_l** from the **ks32c50100_port** folder.
3. In **USBP_compile.h**, undefine **AT91R40008** and define **KS32C50100**.
4. In **USBP_compile.s**, comment **AT91R40008** and uncomment **KS32C50100**.
5. In **USBP_lib_port.h**, set the **SET_BUS_WIDTH** to **SET_BUS_WIDTH_32**, and select the desired setting for **SET_XFR_MODE**.

7.3 Summary

7.3.1 For PDC1

- **Scatter File:** Use **ks32c50100_scat_l.scf**.
- **Predefine Constants:** define and uncomment **KS32C50100**, and undefine and comment **AT91R40008** in USBP_compile.h and USBP_compile.s.
- **USBP_lib_port.h:** Set bus width to 32 bits and select the desired setting for and data transfer mode.

7.3.2 For PDC2

- **Scatter File:** Use **at91r40008_scat_l.scf**.
- **Predefine Constants:** define and uncomment **AT91R40008**, and undefine and comment **KS32C50100** in USBP_compile.h and USBP_compile.s.

- **USBP_lib_port.h:** Set bus width to 16 bits and set the data transfer mode to direct FIFO mode.

Section 8

Generating Hex Files for Flash Mode in the AT43USB380 Development Platform with ADS

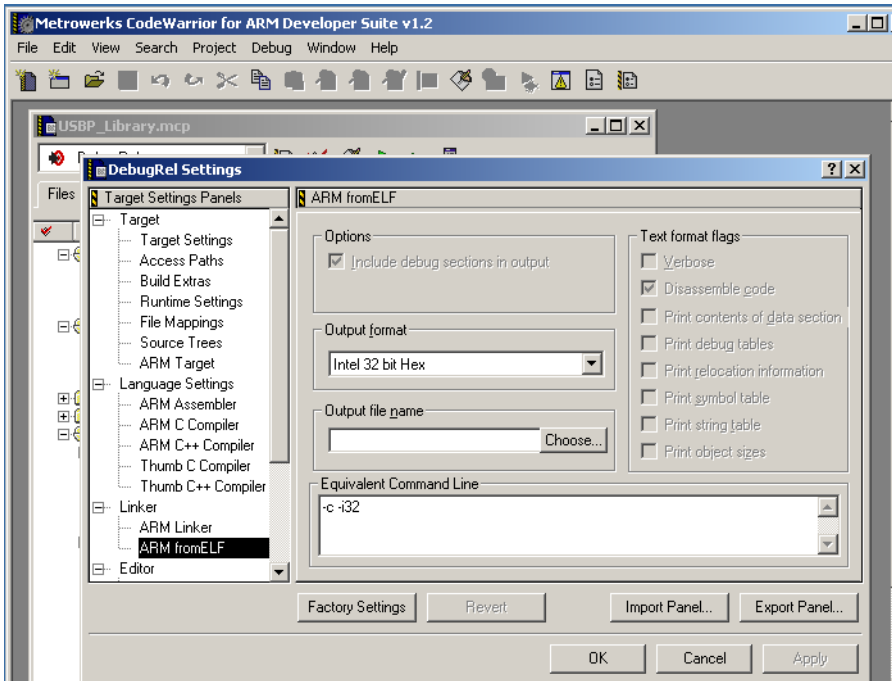
8.1 Introduction

This section gives directions on generating hex file for Flash mode in the AT43USB380 Development Platform with ADS.

8.2 Procedure

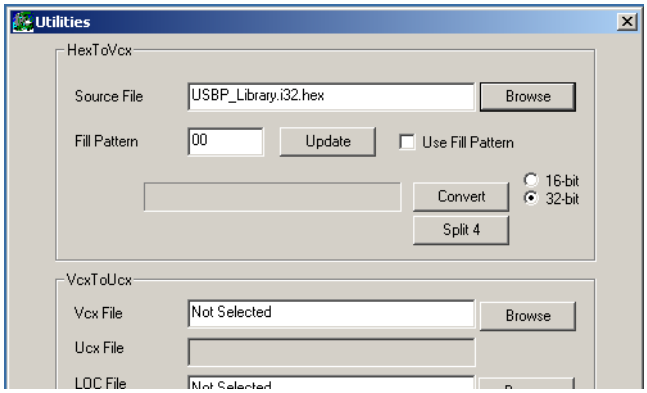
1. Put the PDC jumpers to the Flash Mode setting.
2. The project entry point must be changed to 0x0. To change the entry point, go to **DebugRel** > **ARM Linker** > **Options** > **Image Entry Point** column and set it to 0x0 for PDC1 or 0x2000000 for PDC2.
3. In the **DebugRel Target** setting, select **Target Settings.** In the **Post-linker** field, select **ARM fromELF**.

Figure 8-1. Post-linker Setting: ARM fromELF



4. In the **DebugRel Linker** setting, select **ARM fromELF** setting. In the **Output format** field, select **Intel 32 bit Hex**. In the **Out file name** field, leave it blank (see Figure 8-1). A default name of USBP_Library.i32 will be created. Once the file is created, add a ".hex" to the file name. The name **USBP_Library.i32.hex** will be used and is required to use this default name.
5. 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 Max address 16-KB space. Since we're splitting this file into 4 separate hex files, we would not need any extended hex records. So delete the lines beginning with: "02:0000…" **except** for the first record (leave that one in).
6. Use the **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\AT43DK380_Date\380 Software \USBP_Library_Rev_x.x_Template\USBP_Library_Data\DebugRel** directory.
7. Choose the following from the GUI:
   a. Specify the hex file that is to be split in the source file text box
   b. Set **Fill Pattern** to **00**
   c. Select **32-bit**
   d. Click on the **[Split 4]** button

Figure 8-2. Hex2Vcx GUI



8. There is a bug on the AT43DK380 Main Development such that the Flash data bits are reversed. As a workaround, the user would need to reverse the hex file bits before programing them onto Flash. To do so, double-click on the **Reverse.bat** script. The final hex files created and ready for programing are named **USBP_Library.i32.rev.0**, **1**, **2**, and **3**.
9. Once the hex file has been split, program the Flash sets with the hex files according to their labels.

Section 9

Technical Support

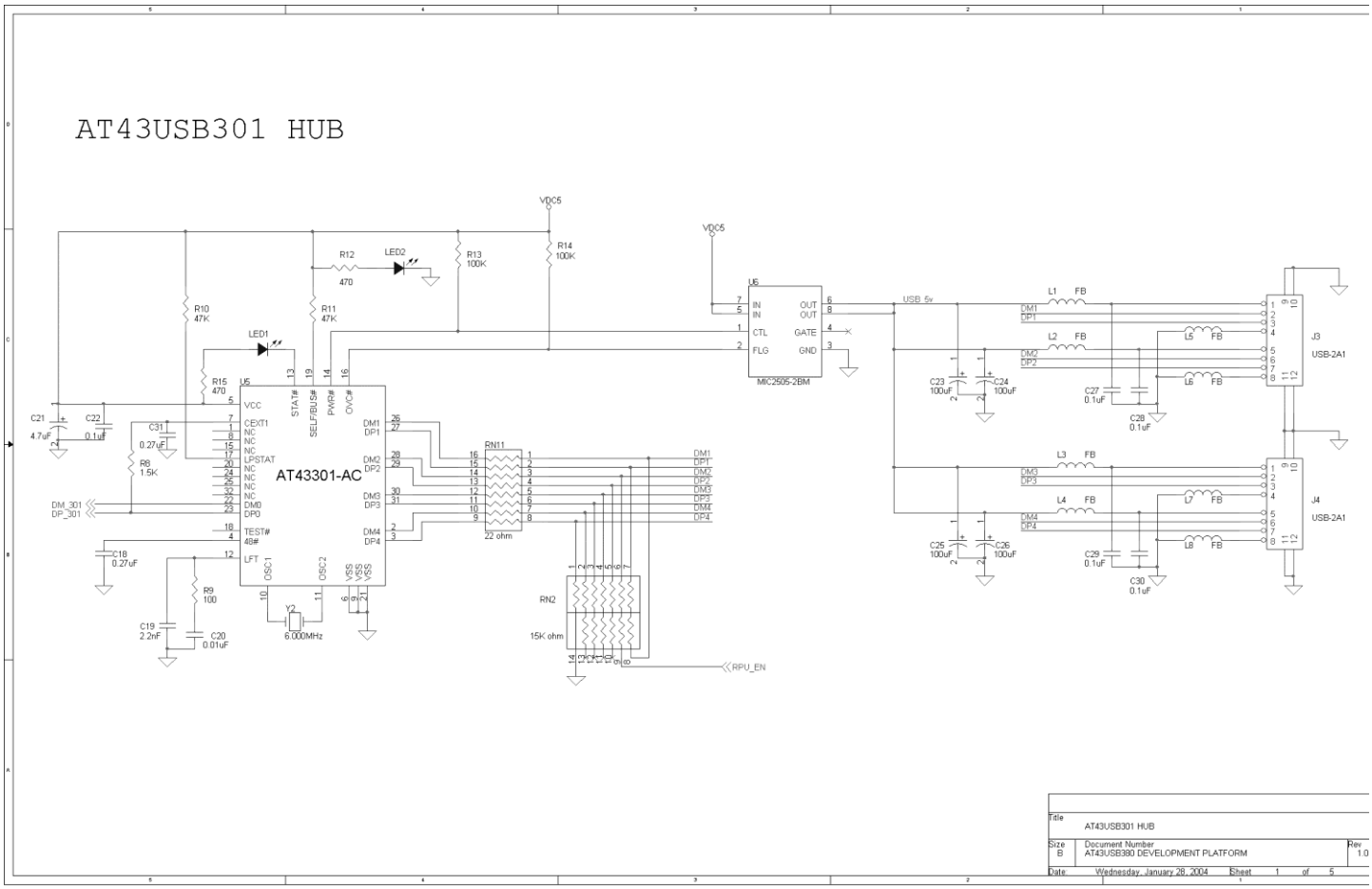
For technical support, please fill out the Customer Support Form at **http://www.atmel.com/products/USB/Forms/USB-support.asp**.

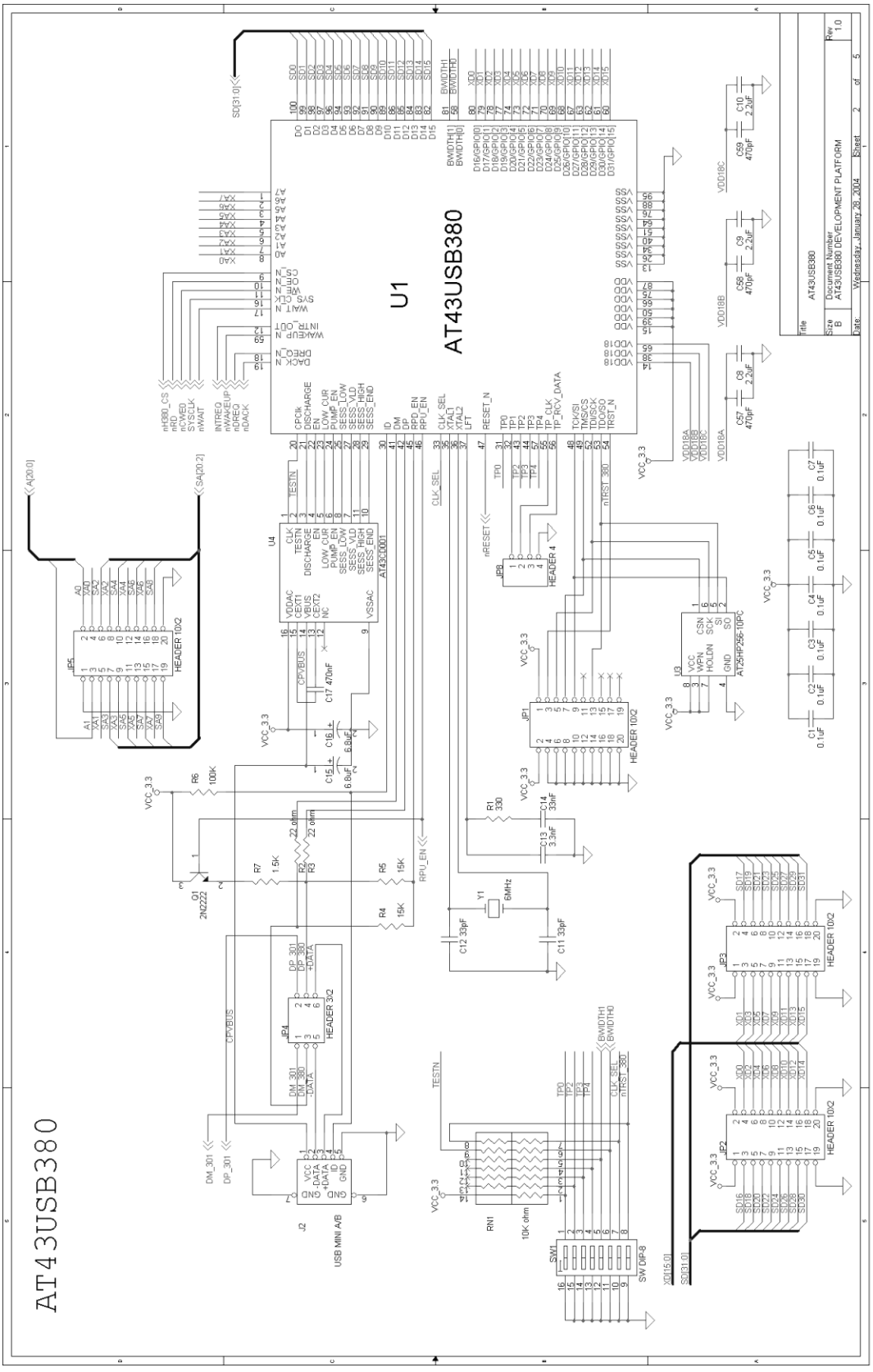
Section 10

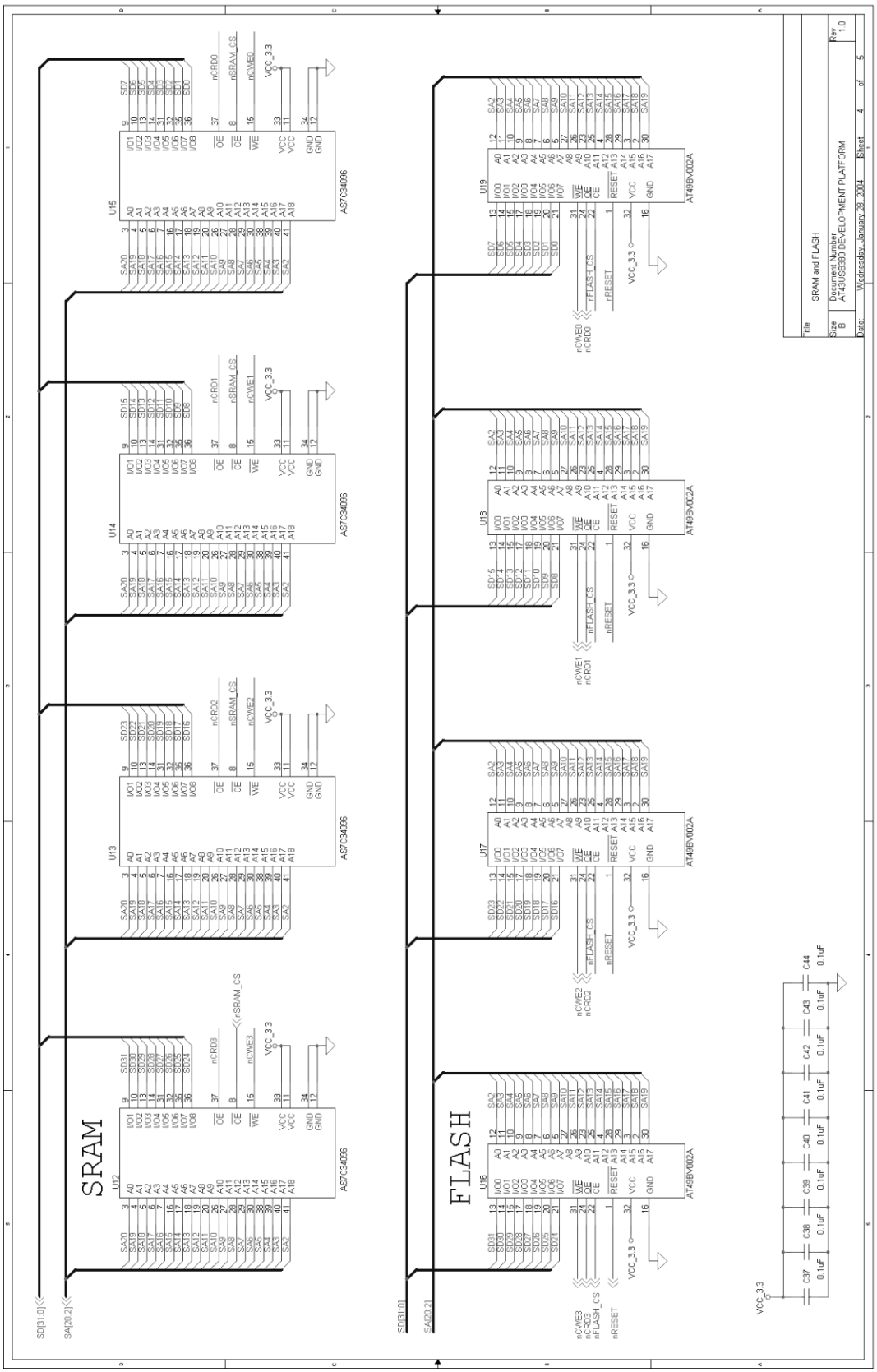
Appendices

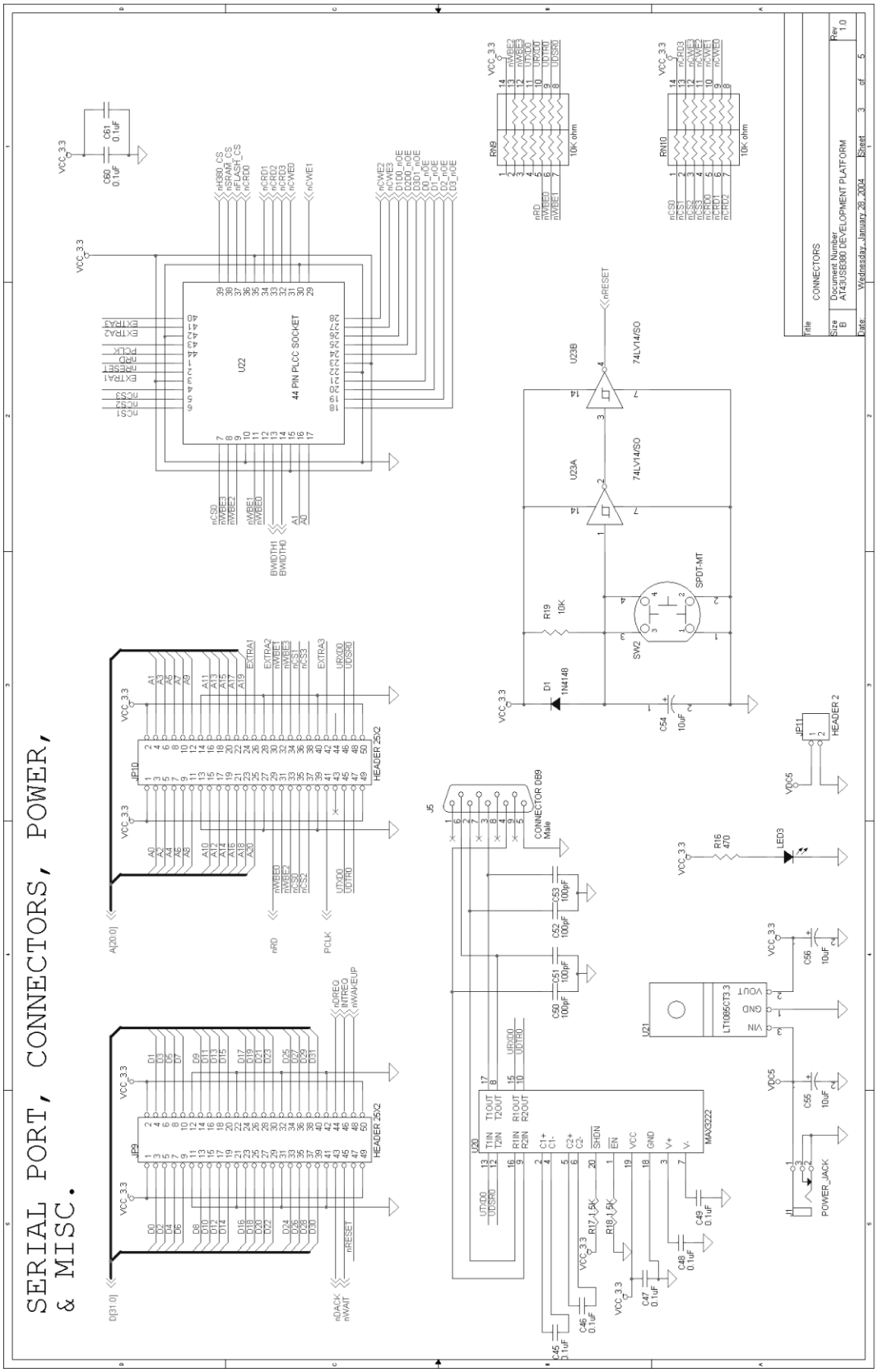
10.1 AT43DK380 Schematics

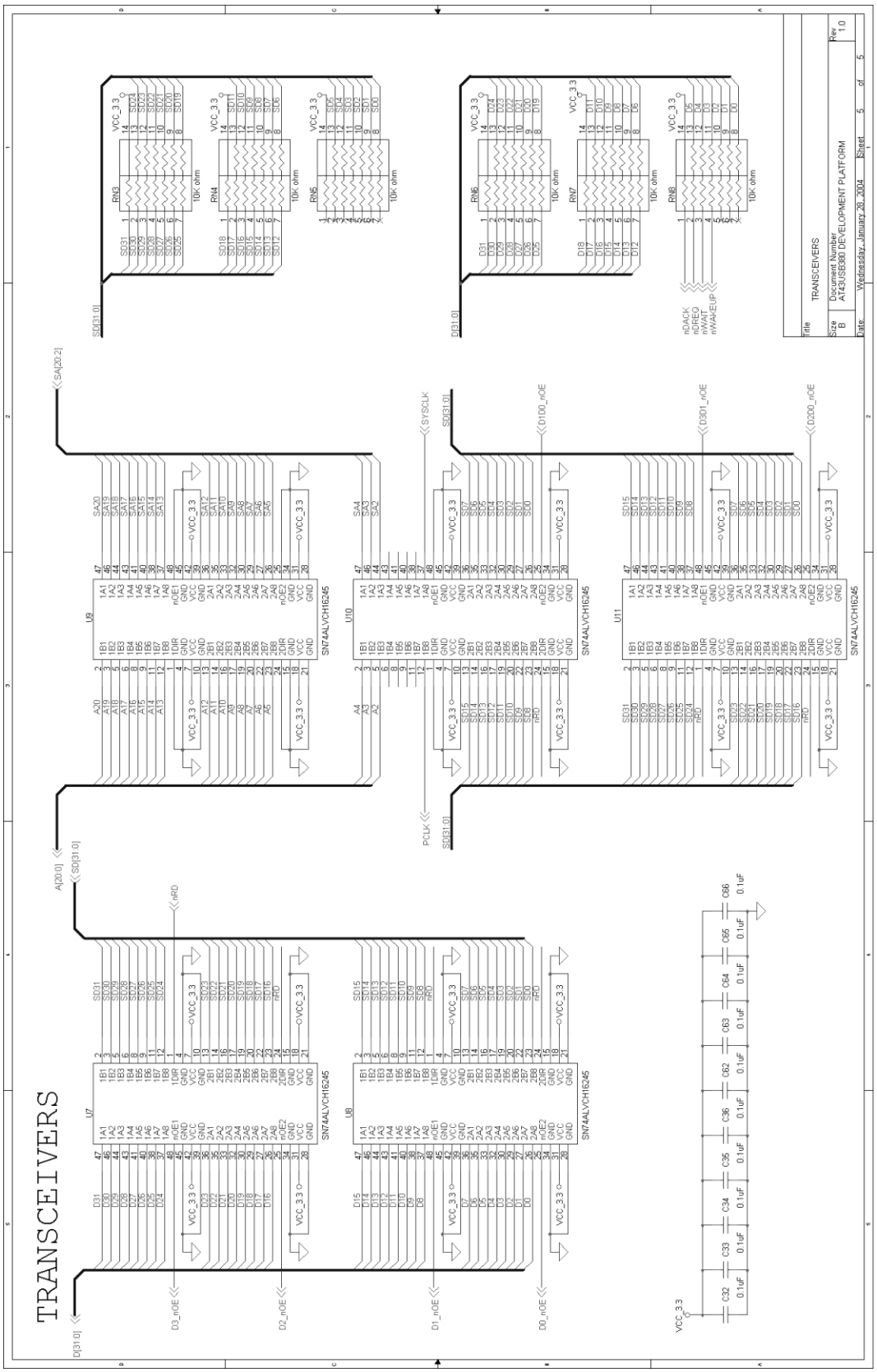
10.1.1 AT43DK380 Main Development Platform Schematics



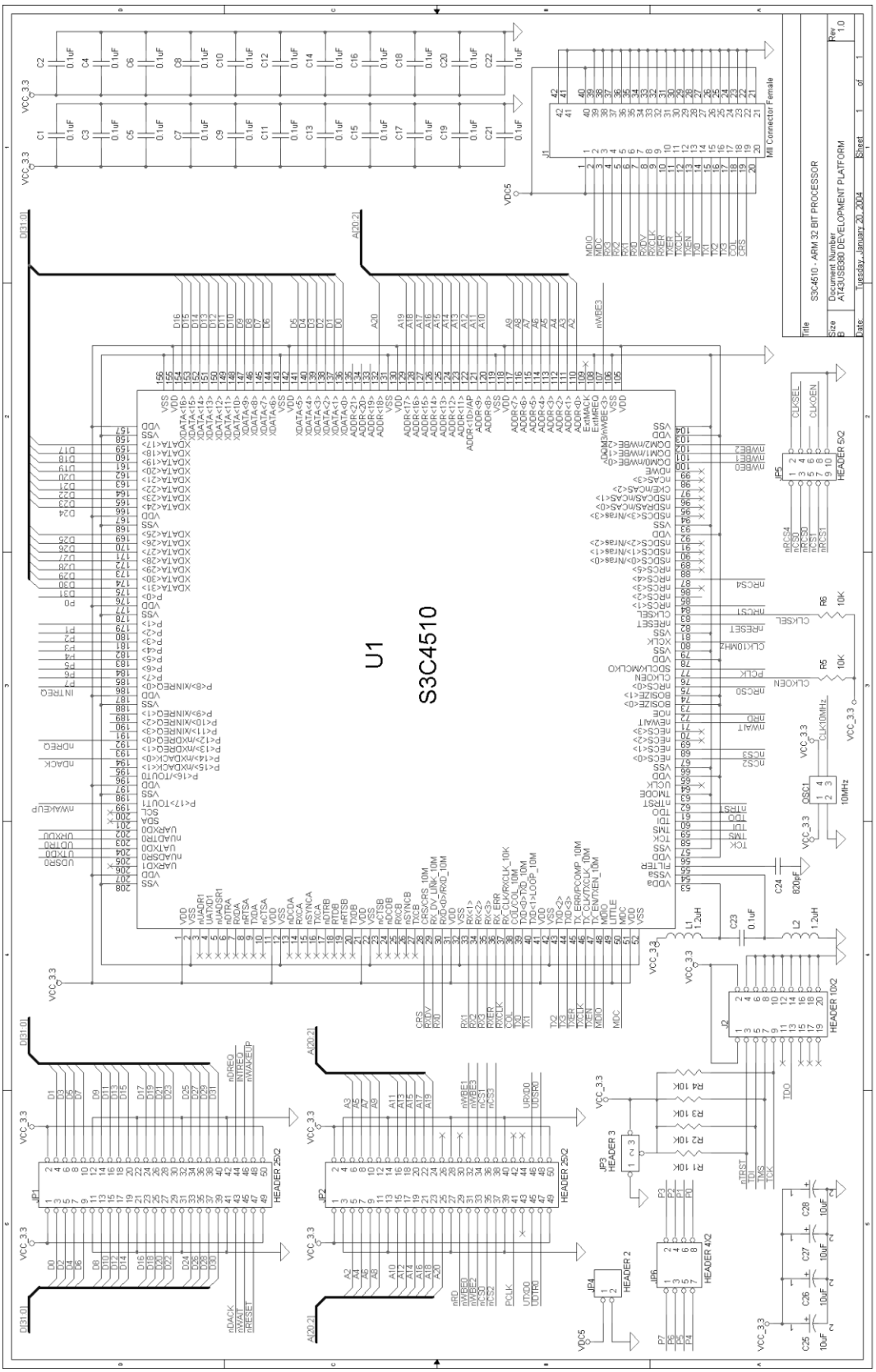

AT43USB380
U1
AT43USB380
Title
AT43USB380
Size
B
Document Number
AT43USB380 DEVELOPMENT PLATFORM
Rev
1.0
Date:
Wednesday, January 28, 2004
Sheet
2
of
5
HEADER 10X2
HEADER 3X2
HEADER 4
USB MINI A/B
SW DIP-8
AT25HP256-10PC
2N2222
6MHz
10K ohm


SRAM
FLASH
SRAM and FLASH
AT43USB380 DEVELOPMENT PLATFORM
Wednesday, January 28, 2004
AS7C34096
AT49BV002A
nSRAM_CS
nFLASH_CS
nRESET
VCC_3.3
SD[31:0]
SA[20:2]

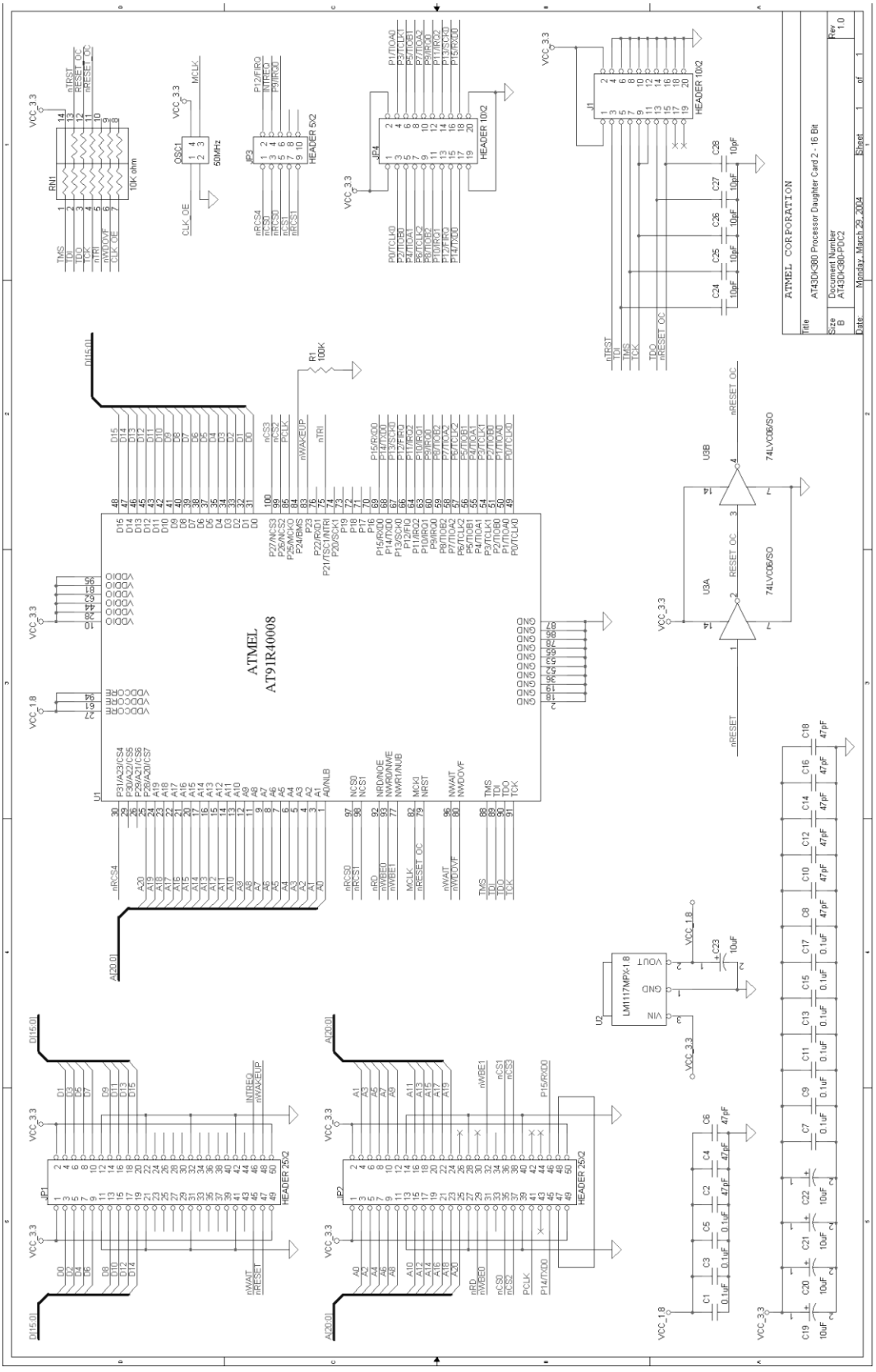

SERIAL PORT, CONNECTORS, POWER,
& MISC.
CONNECTORS
AT43USB380 DEVELOPMENT PLATFORM
Wednesday, January 28, 2004
44 PIN PLCC SOCKET
HEADER 25X2
CONNECTOR DB9 Male
MAX3222
LT1086CT-3.3
74LV14/SO
SPDT-MT
POWER_JACK


TRANSCEIVERS
SN74ALVCH16245
VCC_3.3
10k ohm
SD[31:0]
SA[20:2]
A[20:0]
D[31:0]
PCLK
SYSCLK
nRD
D3_nOE
D2_nOE
D1_nOE
D0_nOE
D100_nOE
D301_nOE
D200_nOE
nDACK
nDREQ
nWAIT
nWAKEUP
Title TRANSCEIVERS
Size B
Document Number AT43USB380 DEVELOPMENT PLATFORM
Rev 1.0
Date: Wednesday, January 28, 2004
Sheet 5 of 5

10.1.2 AT43DK380-PDC1 Schematic



10.1.3 AT43DK380-PDC2 Schematic



10.2 AT43DK380 Bill of Materials (BOM)

10.2.1 AT43DK380 Main Development Platform BOM

Atmel AT43USB380 Main Board							
AT43USB380 Development Platform Revision: 1.0							
Bill Of Materials							
Item	**Qty**	**Reference**	**Part Description**	**Manufacturer**	**Manufacturer Part No.**	**Distrib.**	**Distributor Part No.**
Capacitor							
1	37	C1, C2, C3, C4, C5, C6, C7, C22, C27, C28, C29, C30, C32, C33, C34, C35, C36, C37, C38, C39, C40, C41, C42, C43, C44, C45, C46, C47, C48, C49, C60, C61, C62, C63, C64, C65, C66	0.1uF Ceramic 0805	Panasonic	ECJ-2VB1C104K	Digikey	PCC1812CT-ND
2	3	C8, C9, C10	2.2uF Ceramic 1206	Panasonic	ECJ-3YB1C225K	Digikey	PCC1931CT-ND
3	2	C11, C12	33pF Ceramic 0805	Panasonic	ECJ-2VC1H330J	Digikey	PCC330CGCT-ND
4	1	C13	3.3nF Ceramic 0805	Panasonic	ECJ-2VB1H332K	Digikey	PCC332BNCT-ND
5	1	C14	33nF Ceramic 0805	Panasonic	ECJ-2VB1H333K	Digikey	PCC1834CT-ND
6	2	C15, C16	6.8uF Tantalum 3528	Kemet	T491B685K010AS	Digikey	399-1567-1-ND
7	1	C17	0.47uF Ceramic 1206	Panasonic	ECJ-3VB1C474K	Digikey	PCC1878CT-ND
8	2	C18, C31	0.27uF Ceramic 0805	Panasonic	ECJ-2YB1C274K	Digikey	PCC1916CT-ND
9	1	C19	2.2nF Ceramic 0805	Panasonic	ECJ-2VB1H222K	Digikey	PCC222BNCT-ND
10	1	C20	0.01uF Ceramic 0805	Panasonic	ECJ-2VB1H103K	Digikey	PCC103BNCT-ND
11	1	C21	4.7uF Tantalum 3216	Kemet	T491A475K010AS	Digikey	399-1561-1-ND
12	4	C23, C24, C25, C26	100uF Tantalum 7343	Kemet	T491D107K010AS	Digikey	399-1579-1-ND
13	4	C50, C51, C52, C53	100pF Ceramic 0805	Panasonic	ECJ-2VC1H101J	Digikey	PCC101CGCT-ND
14	3	C54, C55, C56	10uF Tantalum	Kemet	T491A106K010AS	Digikey	399-1563-1-ND
15	3	C57, C58, C59	470pF Ceramic 0805	Panasonic	ECJ-2VB2A471K	Digikey	PCC1983CT-ND
Resistor							
16	9	RN1, RN3, RN4, RN5, RN6, RN7, RN8, RN9, RN10	10K Resistive Network, Bussed	CTS		Digikey	767-141-R10K-ND
17	1	RN2	15K Resistor Network, Bussed	CTS		Digikey	767-141-R15K-ND
18	3	RN11	22 Resistor Network, Isolated	CTS		Digikey	767-163-R22-ND
19	1	R1	330 5% 0805	Panasonic	ERJ-6GEYJ331V	Digikey	P330ACT-ND

Atmel AT43USB380 Main Board							
AT43USB380 Development Platform Revision: 1.0							
Bill Of Materials							
Item	**Qty**	**Reference**	**Part Description**	**Manufacturer**	**Manufacturer Part No.**	**Distrib.**	**Distributor Part No.**
20	3	R2, R3	22 5% 0805	Panasonic	ERJ-6GEYJ220V	Digikey	P22ACT-ND
21	2	R4, R5	15K 5% 0805	Panasonic	ERJ-6GEYJ153V	Digikey	P15KACT-ND
22	3	R6, R13, R14	100K 5% 0805	Panasonic	ERJ-6GEYJ104V	Digikey	P100KACT-ND
23	4	R7, R8, R17, R18	1.5K 5% 0805	Panasonic	ERJ-6GEYJ152V	Digikey	P1.5KACT-ND
24	1	R9	100 5% 0805	Panasonic	ERJ-6GEYJ101V	Digikey	P100ACT-ND
25	2	R10, R11	47K 5% 0805	Panasonic	ERJ-6GEY473V	Digikey	P47KACT-ND
26	3	R12, R15, R16	470 5% 0805	Panasonic	ERJ-6GEY471V	Digikey	P470ACT-ND
27	1	R19	10K 5% 0805	Panasonic	ERJ-6GEY103V	Digikey	P10KACT-ND
Inductor							
28	8	L1, L2, L3, L4, L5, L6, L7, L8	Ferrite Bead 1206 SMD	Stewart	HI1206N800R-00	Digikey	240-1010-1-ND
Diode							
29	1	D1	1N4148 Diode SMD	MCC	DL4148	Digikey	DL4148MSCT-ND
30	3	LED1, LED2, LED3	LED Green 0805 SMD	Lumex	SML-LXT0805GW-TR	Digikey	67-1553-1-ND
31	2	Y1, Y2	6MHz Crystal ATS-SM SMD	CTS	ATS060SM-T	Digikey	CTX505-ND
Semiconductor							
32	1	U1	AT43USB380	Atmel	AT43USB380-AC		
33	1	U4	AT43CD001	Atmel			
34	1	U5	AT43301	Atmel	AT43USB301-AC		
35	1	U6	MIC2505-2BM	Micrel	MIC2505-2BM	Future	MIC2505-2BM
36	5	U7, U8, U9, U10, U11	SN74ALVCH16245	TI	SN74ALVCH16245DGGR	Digikey	296-5190-1-ND
37	4	U12, U13, U14, U15	512Kx8 Static RAM	Alliance Semicondutor	AS7C34096-12TC	Future	AS7C34096-12TC
38	1	U20	RS232 Line Transceiver	TI	MAX3222CDBR	Digikey	296-13082-1-ND
39	1	U21	3.3 Volt Regulator TO-220	Linear Technology	LT1085CT3.3	Digikey	LT1085CT-3.3-ND
40	1	U23	74LV14 14pins TSSOP	TI	SN74ALVC14PWR	Digikey	296-5121-1-ND
41	1	Q1	MMBT2222 SOT-23	Fairchild	MMBT2222A	Digikey	MMBT2222AFSCT-ND
Socket, Header, Connector, Switch							
42	1	U3	8 PIN DIP SOCKET			Jameco	51570CF

Atmel AT43USB380 Main Board							
AT43USB380 Development Platform Revision: 1.0							
Bill Of Materials							
Item	**Qty**	**Reference**	**Part Description**	**Manufacturer**	**Manufacturer Part No.**	**Distrib.**	**Distributor Part No.**
43	4	U16, U17, U18, U19	32 PIN PLCC SOCKET	Mill-Max	940-99-032-24-000000	Digikey	ED80023-ND
44	1	U22	44 PIN PLCC SOCKET	Mill-Max	940-99-044-24-000000	Digikey	ED80024-ND
45	1	JP1	HEADER 10X2, Double Row, 20 Pin, Right Angle, Male	AMP	103311-5	Digikey	A26292-ND
46	3	JP2, JP3, JP5	HEADER 10X2, Double Row, 20 Pin, Straight, Male			Jameco	53479CF
47	1	JP4	HEADER 3X2, Double Row, 6 Pin, Straight, Male			Jameco	115035CF
48	1	JP8	HEADER 4x1, Single Row, 4 Pin, Straight, Male			Jameco	117559CF
49	2	JP9, JP10	HEADER 25X2, Double Row, 50 Pin, Straight, Male			Jameco	53559CF
50	1	JP11	HEADER 2x1, Single Row, 2 Pin, Straight, Male			Jameco	108337CF
51	1	J1	Power Jack 3 Terminals			Digikey	CP-202A-ND
52	1	J2	USB MINI A/B (THM)	Molex	56579-0511		
53	2	J4, J3	USB-A, Dual Port, Right Angle	Mill-Max	896-30-008-90-000000	Digikey	ED90002-ND
54	1	J5	Serial Port DB9 Male	AMP	747840-5	Digikey	A23278-ND
55	1	SW1	SW DIP-8	CTS	206-8	Digikey	CT2068-ND
56	1	SW2	Momentary Push Button Switch	E-Switch	520-02-RED	Digikey	EG1415-ND
Miscellaneous							
57	4		Adhesive Rubber Feet 100/Pk	3M	SJ5018BLKC	Jameco	142682CF
Non-stuffed Components							
58	1	U3	AT25HP256 10ns DIP-8	Atmel	AT25HP256-10PI-2.7		
59	4	U16, U17, U18, U19	AT49BV002 90ns PLCC-32	Atmel	AT49BV002-90JC		
60	1	U22	ATF1504ASV 15ns PLCC-44	Atmel	ATF1504ASV-15 JC44		

10.2.2 AT43DK380-PDC1 BOM

Processor Daughter Card 1: ARM® 32Bit (Samsung)							
AT43USB380 Development Platform Revision: 1.0							
Bill Of Materials							
Item	**Qty**	**Reference**	**Part Description**	**Manufacturer**	**Manufacturer Part No.**	**Distrib.**	**Distributor Part No.**
Capacitor							
1	23	C1, C2, C3, C4, C5, C6, C7, C8, C9, C10, C11, C12, C13, C14, C15, C16, C17, C18, C19, C20, C21, C22, C23	0.1uF Ceramic 0805	Panasonic	ECJ-2VB1C104K	Digikey	PCC1812CT-ND
2	1	C24	820pF Ceramic 0805	Panasonic	ECJ-2VC1H821J	Digikey	PCC821CGCT-ND
3	4	C25, C26, C27, C28	10uF Tantalum	Kemet	T491A106K010AS	Digikey	399-1563-1-ND
Resistor							
4	6	R1, R2, R3, R4, R5, R6	10K 5% 0805	Panasonic	ERJ-6GEYJ103V	Digikey	P10KACT-ND
Inductors							
5	2	L1, L2	1.2uH 1008 SMD	Panasonic	ELJ-FC1R2JF	Digikey	PCD1229CT-ND
Semiconductor							
6	1	U1	ARM uProcessor	Samsung	S3C4510		
Oscillator, Crystal							
7	1	OSC1	10MHz Oscillator SG-636 SMD	EPSON	SG-8002JC-PCC	Digikey	SG-8002JC-PCC-ND 10MHz
Header, Connector							
8	2	JP1, JP2	HEADER 25x2, 0.1", Double Row, 50 Pin, Vertical Female	Sullins	PPPC252LFBN	Digikey	S4325-ND
9	1	JP3	HEADER 3x1, 0.1", Single Row, 3 Pin, Vertical Male			Jameco	109575CF
10	1	JP4	HEADER 2x1, 0.1", Single Row, 2 Pin, Vertical Male			Jameco	108337CF
11	1	JP5	HEADER 5x2, 0.1", Double Row, 10 Pin, Vertical Male			Jameco	67820CF
12	1	JP6	HEADER 4x2, 0.1", Double Row, 8 Pin, Vertical Male			Jameco	109516CF
13	1	J2	HEADER 10X2, Double Row, 20 Pin, Right Angle Male	AMP	103311-5	Digikey	A26292-ND
Non-stuffed Components							
14	1	J1	Connector, MII, 40 Pin Female	AMP	787171-4	Digikey	A23734-ND

10.2.3 AT43DK380-PDC2 BOM

Atmel Processor Daughter Card 2: ARM 16Bit (Atmel)							
AT43USB380 Development Platform Revision: 1.0							
Bill Of Materials							
Item	**Qty**	**Reference**	**Part Description**	**Manufacturer**	**Manufacturer Part No.**	**Distrib.**	**Distributor Part No.**
Capacitor							
1	9	C1, C3, C5, C7, C9, C11, C13, C15, C17	0.1uF Ceramic 0805	Panasonic	ECJ-2VB1C104K	Digikey	PCC1812CT-ND
2	9	C2, C4, C6, C8, C10, C12, C14, C16, C18	47pF Ceramic 0805	Panasonic	ECJ-2VC1H470J	Digikey	PCC470CGCT-ND
3	5	C19, C20, C21, C22, C23	10uF Tantalum	Kemet	T491A106K010AS	Digikey	399-1563-1-ND
4	5	C24, C25, C26, C27, C28	10pF Ceramic 0805	Panasonic	ECJ-2VC1H100D	Digikey	PCC100CNCT-ND
Resistor							
5	1	RN1	10K 5% 0805	CTS		Digikey	767-141-R10K-ND
6	1	R1	100K 5% 0805	Panasonic	ERJ-6GEYJ104V	Digikey	P100KACT-ND
Semiconductor							
7	1	U1	AT91R40008	Atmel	AT91R40008-66AI		
8	1	U2	LM1117MPX-1.8	National Semiconductor	LM1117MPX-1.8	Digikey	LM1117MP-1.8CT-ND
9	1	U3	74LVC06 14 Pin TSSOP	TI	SN74LVC06APWR	Digikey	296-8437-1-ND
Oscillator							
10	1	OSC1	50MHz Oscillator SMT	Epson	SG-8002JC-PCC	Digikey	SG-8002JC-PCC-ND 50MHz
Header, Connector							
11	1	J1	HEADER 10X2, Double Row, 20 Pin, Right Angle Male	AMP	103311-5	Digikey	A26292-ND
12	2	JP2, JP1	HEADER 25X2, Double Row, 50 Pin, Vertical Female	Sullins	PPPC252LFBN	Digikey	S4325-ND
13	1	JP3	HEADER 5X2, Double Row, 10 Pin, Vertical Male			Jameco	67820CA
14	1	JP4	HEADER 10X2, Double Row, 20 Pin, Vertical Male			Jameco	53479CA

10.3 AT43DK380 Main Development Platform CPLD VHDL Code Ref.

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
--USE IEEE.STD_LOGIC_UNSIGNED.ALL;

ENTITY AT43USB380 is
PORT(-- PCLK:  IN STD_LOGIC;     --not used.
           nRD: IN STD_LOGIC;
          -- nRESET: IN STD_LOGIC;    --not used.
          -- nCS3: IN STD_LOGIC;     --not used.
           nCS2: IN STD_LOGIC;
           nCS1: IN STD_LOGIC;
           nCS0: IN STD_LOGIC;
           nWBE3: IN STD_LOGIC;
           nWBE2: IN STD_LOGIC;
           nWBE1: IN STD_LOGIC;
           nWBE0: IN STD_LOGIC;
           A1: IN STD_LOGIC;
           A0: IN STD_LOGIC;
           BWIDTH1: IN STD_LOGIC;
           BWIDTH0: IN STD_LOGIC;
           D3_nOE: OUT STD_LOGIC;
           D2_nOE: OUT STD_LOGIC;
           D1_nOE: OUT STD_LOGIC;
           D0_nOE: OUT STD_LOGIC;
           D3D1_nOE: OUT STD_LOGIC;
           D2D0_nOE: OUT STD_LOGIC;
           D1D0_nOE: OUT STD_LOGIC;
           nCWE3: OUT STD_LOGIC;
           nCWE2: OUT STD_LOGIC;
           nCWE1: OUT STD_LOGIC;
           nCWE0: OUT STD_LOGIC;
           nCRD3: OUT STD_LOGIC;
           nCRD2: OUT STD_LOGIC;
           nCRD1: OUT STD_LOGIC;
           nCRD0: OUT STD_LOGIC;
           nFLASH_CS: OUT STD_LOGIC;
           nSRAM_CS: OUT STD_LOGIC;
           nH380_CS: OUT STD_LOGIC
          );
END ENTITY;
```

```
ARCHITECTURE BEHAVIOR OF AT43USB380 IS

BEGIN

--D3_nOE/D2_nOE are only used for 32bit bus memory byte (nWBE3/2) and 380 (nWBE0) word access.
--D1_nOE is used for 32/16bit bus for memory byte (nWBE1), and 380 (nWBE0) word/half-word access.
--D0_nOE is used for all access.
D3_nOE <= '0' WHEN (nRD = '0' AND BWIDTH1 = '1') ELSE
        '0' WHEN ((nWBE3 = '0' OR (nWBE0 = '0' AND nCS2 = '0')) AND BWIDTH1 = '1') ELSE '1';
D2_nOE <= '0' WHEN (nRD = '0' AND BWIDTH1 = '1') ELSE
        '0' WHEN ((nWBE2 = '0' OR (nWBE0 = '0' AND nCS2 = '0')) AND BWIDTH1 = '1') ELSE '1';
D1_nOE <= '0' WHEN ((nRD = '0' AND BWIDTH1 = '1') OR (nRD = '0' AND BWIDTH0 = '1')) ELSE
        '0' WHEN (((nWBE1 = '0' OR (nWBE0 = '0' AND nCS2 = '0')) AND BWIDTH1 = '1') OR
                ((nWBE1 = '0' OR (nWBE0 = '0' AND nCS2 = '0')) AND BWIDTH0 = '1')) ELSE '1';
D0_nOE <= '0' WHEN nRD = '0' ELSE
        '0' WHEN nWBE0 = '0' ELSE '1';

--D3D1_nOE/D2D0_nOE are used for 16/8bit bus upper half-word/4th byte memory access.  They are not
--used for 380 access.
--D1D0_nOE is used for 8bit bus 2nd/4th memory byte access; it is not used for 380 access.
D3D1_nOE <= '0' WHEN ((nCS0 = '0' OR nCS1 = '0') AND nRD = '0' AND BWIDTH1 = '0' AND
                BWIDTH0 = '1' AND A1 = '1') ELSE -- 16 Bit Read
        '0' WHEN ((nCS0 = '0' OR nCS1 = '0') AND nWBE1 = '0' AND BWIDTH1 = '0' AND
                BWIDTH0 = '1' AND A1 = '1') ELSE -- 16 Bit Write
        '0' WHEN ((nCS0 = '0' OR nCS1 = '0') AND nRD = '0' AND BWIDTH1 = '0' AND
                BWIDTH0 = '0' AND A1 = '1' AND A0 = '1') ELSE -- 8 Bit Read
        '0' WHEN ((nCS0 = '0' OR nCS1 = '0') AND nWBE0 = '0' AND BWIDTH1 = '0' AND
                BWIDTH0 = '0' AND A1 = '1' AND A0 = '1') ELSE '1'; -- 8 Bit Write
D2D0_nOE <= '0' WHEN ((nCS0 = '0' OR nCS1 = '0') AND nRD = '0' AND BWIDTH1 = '0' AND
                BWIDTH0 = '1'  AND A1 = '1') ELSE -- 16 Bit Read
        '0' WHEN ((nCS0 = '0' OR nCS1 = '0') AND nWBE0 = '0' AND BWIDTH1 = '0' AND
                BWIDTH0 = '1' AND A1 = '1') ELSE -- 16 Bit Write
        '0' WHEN ((nCS0 = '0' OR nCS1 = '0') AND nRD = '0' AND BWIDTH1 = '0' AND
                BWIDTH0 = '0' AND A1 = '1' AND A0 = '0') ELSE -- 8 Bit Read
        '0' WHEN ((nCS0 = '0' OR nCS1 = '0') AND nWBE0 = '0' AND BWIDTH1 = '0' AND
                BWIDTH0 = '0' AND A1 = '1' AND A0 = '0') ELSE '1'; -- 8 Bit Write
D1D0_nOE <= '0' WHEN ((nCS0 = '0' OR nCS1 = '0') AND nRD = '0' AND BWIDTH1 = '0' AND
                BWIDTH0 = '0' AND A0 = '1') ELSE-- 8 Bit Read
 '0' WHEN ((nCS0 = '0' OR nCS1 = '0') AND nWBE0 = '0' AND BWIDTH1 = '0' AND
                BWIDTH0 = '0' AND A0 = '1') ELSE '1'; -- 8 Bit Write

--nCWE3,2,1 are used for 32/16/8bit bus memory byte access nWE; they are not use for 380.
--nCWE0 is used for 32/16/8 bus memory byte access nWE, as well as 380 bus nWE.
nCWE3 <= nWBE0 WHEN (BWIDTH1 = '0' AND BWIDTH0 = '0'
                AND A1 = '1' AND A0 = '1') ELSE  -- 8 Bit
        nWBE1 WHEN (BWIDTH1 = '0' AND BWIDTH0 = '1'
                AND A1 = '1') ELSE  -- 16 Bit
        nWBE3 WHEN BWIDTH1 = '1' ELSE '1';  -- 32 Bit
nCWE2 <= nWBE0 WHEN (BWIDTH1 = '0' AND BWIDTH0 = '0'
                AND A1 = '1' AND A0 = '0') ELSE  -- 8 Bit
```

```
        nWBE0 WHEN (BWIDTH1 = '0' AND BWIDTH0 = '1'
                AND A1 = '1') ELSE  -- 16 Bit
        nWBE2 WHEN BWIDTH1 = '1' ELSE '1';  -- 32 Bit
nCWE1 <= nWBE0 WHEN (BWIDTH1 = '0' AND BWIDTH0 = '0'
                AND A1 = '0' AND A0 = '1') ELSE  -- 8 Bit
        nWBE1 WHEN (BWIDTH1 = '0' AND BWIDTH0 = '1'
                AND A1 = '0') ELSE  -- 16 Bit
        nWBE1 WHEN BWIDTH1 = '1' ELSE '1';  -- 32 Bit
nCWE0 <= nWBE0 WHEN (BWIDTH1 = '0' AND BWIDTH0 = '0'
                AND A1 = '0' AND A0 = '0') ELSE  -- 8 Bit
        nWBE0 WHEN (BWIDTH1 = '0' AND BWIDTH0 = '1'
                AND A1 = '0') ELSE  -- 16 Bit
        nWBE0 WHEN BWIDTH1 = '1' ELSE  -- 32 Bit
        nWBE0 WHEN (nCS2 = '0') ELSE '1';

--nCRD3,2,1 are used for 32/16/8bit bus memory byte access nOE; they are not use for 380.
--nCRD0 is used for 32/16/8 bus memory byte access nOE, as well as 380 bus nOE.
nCRD3 <= nRD WHEN (BWIDTH1 = '0' AND BWIDTH0 = '0'
                AND A1 = '1' AND A0 = '1') ELSE  -- 8 Bit
        nRD WHEN (BWIDTH1 = '0' AND BWIDTH0 = '1'
                AND A1 = '1') ELSE  -- 16 Bit
        nRD WHEN BWIDTH1 = '1' ELSE '1';  -- 32 Bit
nCRD2 <= nRD WHEN (BWIDTH1 = '0' AND BWIDTH0 = '0'
                AND A1 = '1' AND A0 = '0') ELSE  -- 8 Bit
        nRD WHEN (BWIDTH1 = '0' AND BWIDTH0 = '1'
                AND A1 = '1') ELSE  -- 16 Bit
        nRD WHEN BWIDTH1 = '1' ELSE '1';  -- 32 Bit
nCRD1 <= nRD WHEN (BWIDTH1 = '0' AND BWIDTH0 = '0'
                AND A1 = '0' AND A0 = '1') ELSE  -- 8 Bit
        nRD WHEN (BWIDTH1 = '0' AND BWIDTH0 = '1'
                AND A1 = '0') ELSE  -- 16 Bit
        nRD WHEN BWIDTH1 = '1' ELSE '1';  -- 32 Bit
nCRD0 <= nRD WHEN (BWIDTH1 = '0' AND BWIDTH0 = '0'
                AND A1 = '0' AND A0 = '0') ELSE  -- 8 Bit
        nRD WHEN (BWIDTH1 = '0' AND BWIDTH0 = '1'
                AND A1 = '0') ELSE  -- 16 Bit
        nRD WHEN BWIDTH1 = '1' ELSE  -- 32 Bit
        nRD WHEN nCS2 = '0' ELSE '1';

nH380_CS <= nCS2;
nSRAM_CS <= nCS1;
nFLASH_CS <= nCS0;

END  BEHAVIOR;
```

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

3485B–USB–3/9/05 /xM