APPLICATION NOTE

AT07694: SAM D11 USB Mass Storage Device

ATSAMD11D14AM

Introduction

The SAM D11 Universal Serial Bus interface (USB) module supports device mode operation, supporting full speed 12Mbits/s) and low speed (1.5Mbits/s) communication.

This application note describes the SAMD11 USB module and demonstrates implementing a USB application based on the Mass Storage (Bulk only) class to transfer data between a PC and SD card.

Features

- Compatible with the USB 2.1 specification
- Bulk-only Transport Protocol
- Complete Mass Storage solution based on SD card memory

Table of Contents

1 SAM D11 USB Peripheral Overview 3
1.1 Introduction 3
1.2 Features for USB Device Mode 4
1.2.1 Multi-packet Transfers 4
1.2.2 Ping-Pong Operation 4
1.2.3 Crystal-less Operation 5
1.2.4 Endpoint Management 5
1.2.5 Feedback Operation 5
1.2.6 Suspended and Remote Wake-up 5
1.2.7 Link Power Management 5
1.2.8 SOF Clock Output 6
2 Hardware Requirements 7
3 Software Requirements 7
4 Setup 7
4.1 Hardware Setup 7
4.2 Software Setup 8
5 USB Mass Storage Application Overview 11
6 Mass Storage Class Overview 12
6.1 Mass Storage Device Endpoints 12
6.1.1 Default Control Endpoint 12
6.1.2 BULK OUT Endpoint 12
6.1.3 BULK IN Endpoint 12
6.2 Data Transfers between Host and Device 12
6.2.2 Command Block Wrapper 13
6.2.3 Data Transport 14
6.2.4 Command Status Wrapper 14
6.3 SCSI Commands 15
7 Mass Storage Firmware Workflow 16
7.2 Module Configuration 18
7.2.1 Clock Initialization 18
7.2.2 SD Card Initialization 18
7.2.3 USB Initialization 19
7.3 USB Request 19
7.3.1 USB Standard Request 19
7.3.2 USB Mass Storage Request 19
8 Mass Storage Application 24
8.1 Programming SAM D11 Xplained Pro Kit 24
8.2 Mass Storage Enumeration 25
8.3 Load SD Card Memory 27
9 References 28
10 Revision History 29

1 SAM D11 USB Peripheral Overview

1.1 Introduction

The Universal Serial Bus interface (USB) module complies with the Universal Serial Bus (USB) 2.1 specification supporting device modes.

The USB device mode supports eight endpoint addresses. All endpoint addresses have one input and one output endpoint, for a total of 16 endpoints. Each endpoint is fully configurable in any of the four transfer types; control, interrupt, bulk, or isochronous. The maximum data payload size is selectable up to 1023 bytes.

Internal SRAM is used to keep the configuration and data buffer for each endpoint. The memory locations used for the endpoint configurations and data buffers are fully configurable. The amount of memory allocated is dynamic according to the number of endpoints in use, and the configuration of these. The USB module has a built-in Direct Memory Access (DMA) and will read/write data from/to the system RAM when a USB transaction takes place. No CPU or DMA Controller resources are required.

To maximize throughput, an endpoint can be configured for ping-pong operation. When this is done the input and output endpoint with the same address are used in the same direction. The CPU or DMA Controller can then read/write one data buffer while the USB module writes/reads from the other buffer. This gives double buffered communication.

Multi-packet transfer enables a data payload exceeding the maximum packet size of an endpoint to be transferred as multiple packets without any software intervention. This reduces the number of interrupts and software intervention needed for USB transfers.

For low power operation the USB module can put the microcontroller in any sleep mode when the USB bus is idle and a suspend condition is given. Upon bus resume the USB module can wake the microcontroller from any sleep mode.

Figure 1-1. Block Diagram

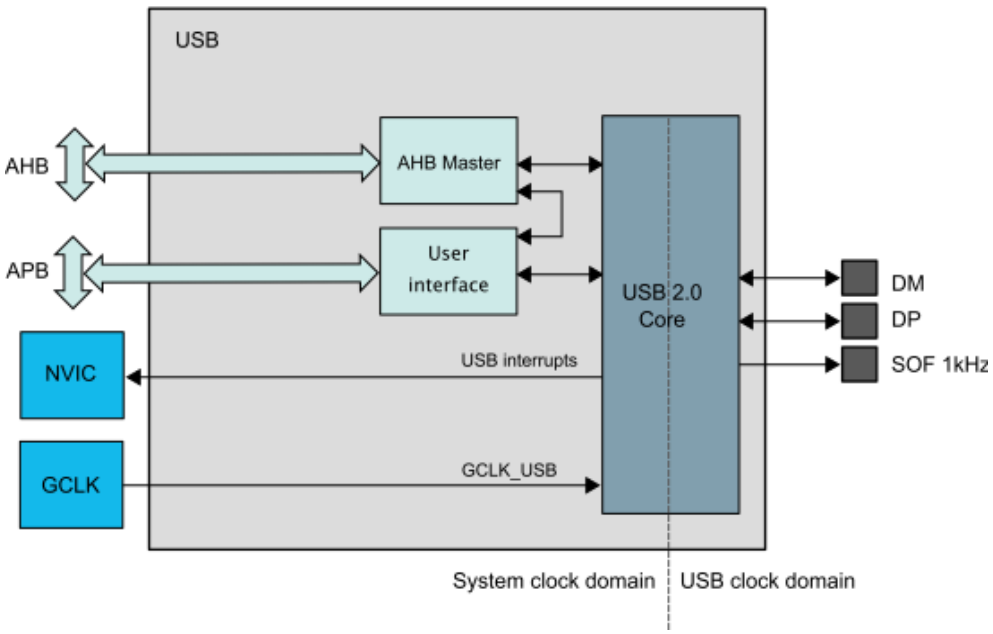


Figure 1-1 shows the USB module block diagram. The SAM D11 USB uses CLK_USB_APB for register access and CLK_USB_AHB for the DMA access to internal SRAM. A generic clock (GCLK_USB) is required to clock the USB. This generic clock is asynchronous to the bus clock (CLK_USB_AHB). Due to this asynchronicity, writes to certain registers will require synchronization between the clock domains.

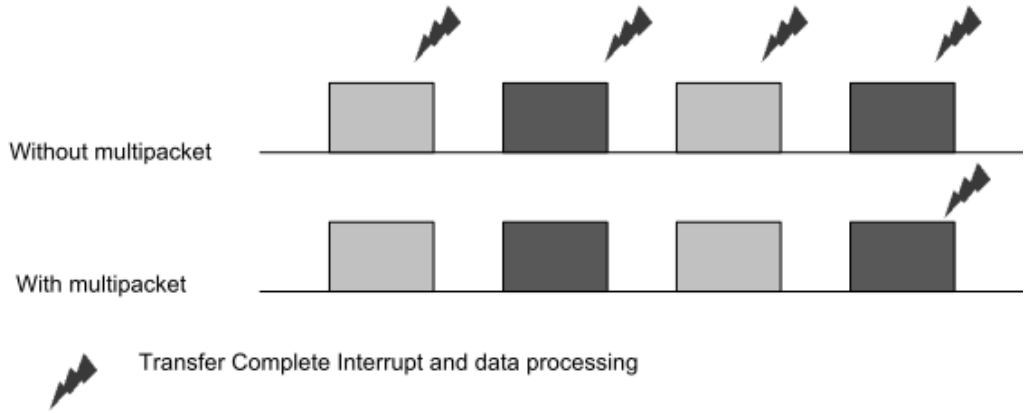
The USB module requires a GCLK_USB of 48MHz ±0.25% for low speed and full speed operation. To follow the USB data rate at 12Mbit/s in full-speed mode, the CLK_USB_AHB clock should be at minimum 8MHz.

1.2 Features for USB Device Mode

1.2.1 Multi-packet Transfers

Typically, an endpoint can transfer a maximum of endpoint packet size in a single USB transfer. For large amount of data, i.e., if the amount of data is exceeding the endpoint maximum packet size, the software has to manually split the data into packets and send them through the USB. When multi-packet transfer is used, the USB module itself splits the data into multiple packets and transfers them automatically on each USB data request. Thus, it enables transfer of data larger than the endpoint size without further interrupts or CPU intervention as shown in Figure 1-2 and provides higher data transfer rate.

Figure 1-2. USB Transfer with Multi-packet

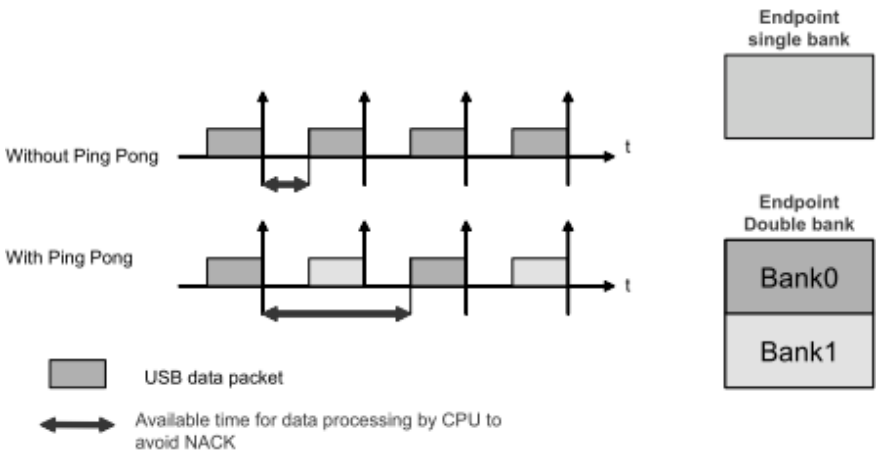


1.2.2 Ping-Pong Operation

When an endpoint is configured for ping-pong operation, it uses both the input and output data buffers (banks) for a given endpoint in a single direction.

When Ping-Pong mode/dual-bank is configured for an endpoint, it uses the data buffers for both the input and output endpoints of an endpoint address in the same direction. For example, if Endpoint 1 IN is configured to use ping-pong mode, it uses the data buffer of Endpoint 1 OUT for dual-banking. (Endpoint 1 OUT cannot be used when Endpoint 1 IN is used in ping-pong mode). An example for Ping-Pong operation is shown in Figure 1-3.

Figure 1-3. Ping-Pong Overview



1.2.3 Crystal-less Operation

SAM D11 provides an option to use the USB SOF (Start-Of-Frame) signal as a reference signal for DFLL to generate the required 48MHz for GCLK_USB. When the USB Clock Recovery Mode (USBCRM) is enabled in the DFLL, the SOF signal from the USB host will be used as the reference clock for DFLL. An auto jitter mechanism is enabled with USBCRM to manage the jitter within the USB specification limit. This feature eliminates the requirement of an external crystal in USB device applications.

1.2.4 Endpoint Management

The SAM D11 USB device mode supports a maximum of eight endpoint addresses. Each address can be configured to have one input and one output endpoint. All the endpoints can be configured in any of the four transfer types: control, interrupt, bulk, or isochronous.

SAM D11 uses an endpoint descriptor table to manage the endpoint configuration and status. The endpoint descriptor and the endpoint data buffer can be physically allocated anywhere in the internal RAM. The USB controller accesses the endpoint data directly through the AHB master with the help of the built-in DMA. Memory usage for each endpoint depends on the endpoint configuration (size, number of banks, etc) and can be allocated dynamically by the user based on the application requirements.

1.2.5 Feedback Operation

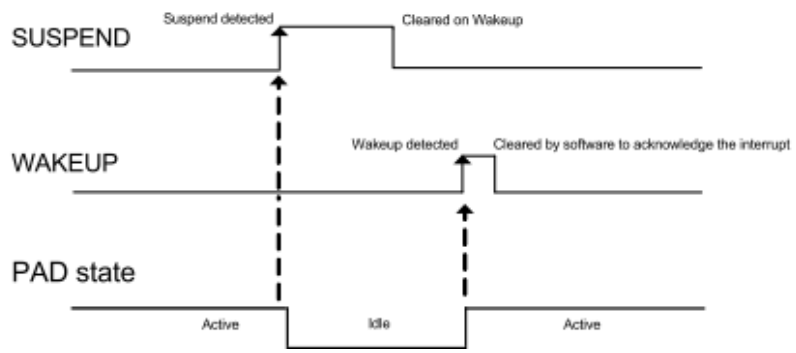
The feedback endpoint always has the opposite direction from the data endpoint(s). The feedback endpoint has the same endpoint number as the first (lower) data endpoint. A feedback endpoint can be created by configuring an endpoint with different endpoint size (PCKSIZE.SIZE) and different endpoint type (EPCFG.EPTYPE0/1) for the IN and OUT direction.

1.2.6 Suspended and Remote Wake-up

When a suspend condition is detected on the USB bus, the USB pads are put into idle state (low power consumption mode). On a resume signal from the host, the pads are put back into Active state. Figure 1-4 shows the USB pad behavior on suspend and resume events.

The USB operations will resume on receiving a downstream signal from the host. If remote wake-up is enabled in the device, it can send an Upstream Resume to the host to resume the USB operations. The rebroadcasted resume from the USB host is managed by the USB hardware and sets an interrupt flag when the resume event is completed.

Figure 1-4. Pad Events



1.2.7 Link Power Management

The SAM D11 USB device supports the Link Power Management Protocol (LPM-L1).

LPM-L1 allows a USB host to configure the USB device into inactive state much faster than the normal USB suspend mode (which requires 3ms of bus inactivity). It also provides much faster wake-up times in the order of

micro-seconds compared to the generic resume by host or upstream resume by device (which requires nearly 3 to 30ms). Further, it imposes no restrictions on the current drawn from the VBUS compared to suspend mode (which is limited to 500µA to 2mA).

An LPM transaction from host to device is necessary for the device to enter the L1 state. This L1-SLEEP state would occur after 9µs from receiving the LPM transaction. The remote wake-up feature from the device can be enabled/disabled through the LPM transaction. The L1-SLEEP exit time is specified by the Best Effort Service Latency (BESL) parameter provided through the USB descriptors. The device can conserve power in L1-SLEEP state by entering any of the sleep modes with the constraint that the wake-up latency should satisfy the BESL parameter.

If the LPM-L1 support is not enabled, the device will ignore the LPM transactions from the USB host and no handshake is returned.

1.2.8 SOF Clock Output

The USB SOF signal can be put on the SOF 1kHz pin. In this way, it can be fed to external components as a reference/synchronization signal.

2 Hardware Requirements

This application requires the following hardware:

- SAM D11 Xplained Pro board
- I/O1 Xplained Pro board with micro SD card
- USB Cable (Standard A to Micro-B USB)
- PC running on Windows® with USB 1.1 or 2.0 host

3 Software Requirements

The software needed for this application includes:

- Atmel® Studio 6.2

4 Setup

The example code given in this application note uses the SAM D11 Xplained Pro kit and as the hardware and Atmel Studio 6.2 as IDE for application development.

The overview of this chapter is:

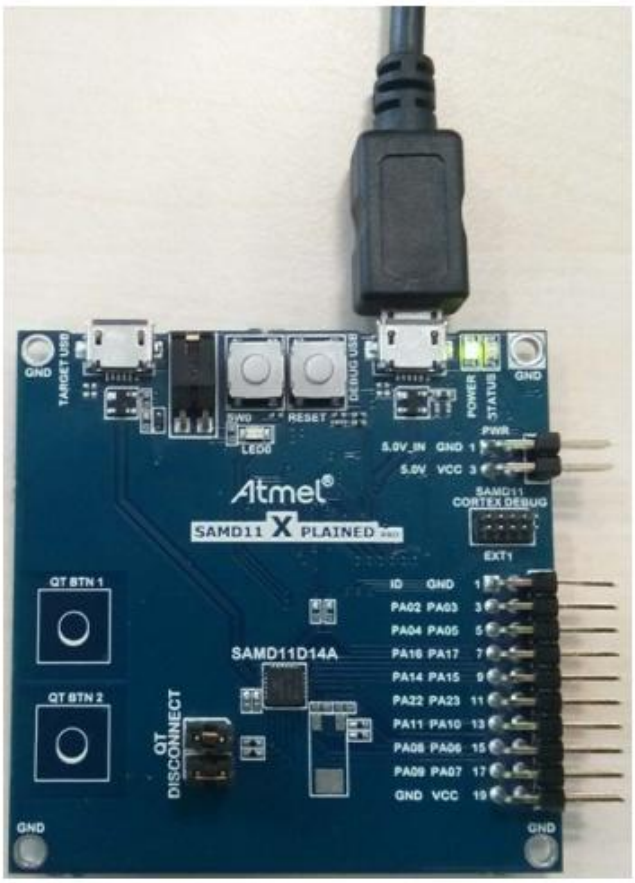
- Hardware Setup
- Software Setup

4.1 Hardware Setup

The SAM D11 Xplained Pro kit will be used to run the example application. This is an evaluation kit that allows connecting multiple external components via a wing connector. A wing is a self contained board that can be connected to the Xplained Pro using a wing connector. The SAM D11 Xplained Pro has one such wing connector marked as EXT1.

There are two USB ports on the SAM D11 Xplained Pro board; **DEBUG USB** and **TARGET USB**. For debugging using the Embedded Debugger EDBG, the **DEBUG USB** port has to be connected as shown in Figure 4-1.

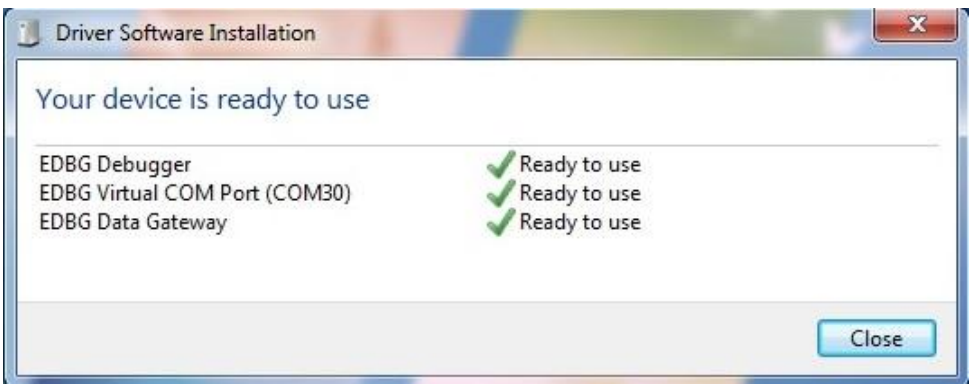
Figure 4-1. SAM D11 Xplained Pro with Debug USB Connected



4.2 Software Setup

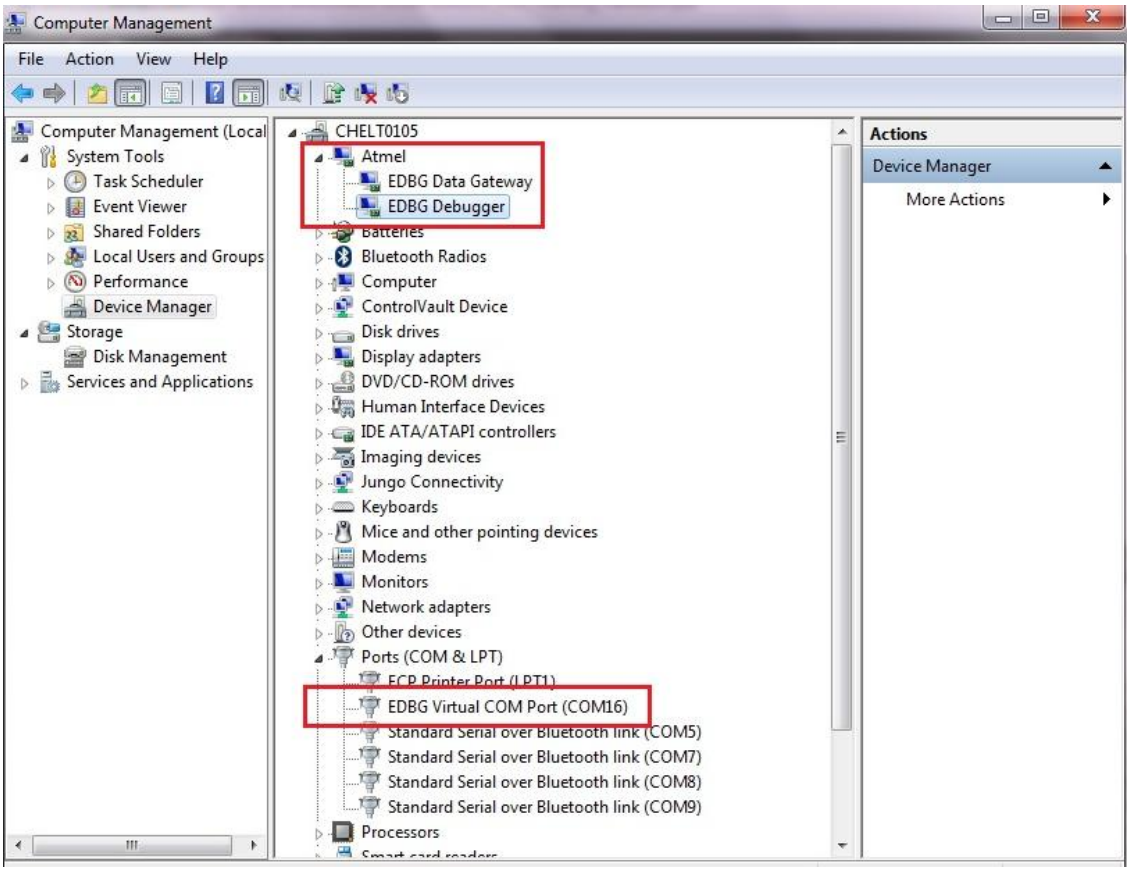
Once the SAM D11 Xplained Pro kit is connected to a PC, the required drivers for the EDBG will be automatically installed. Figure 4-2 shows the Driver Software Installation.

Figure 4-2. SAM D11 Xplained Pro Driver Installation



If the driver installation is proper, EDBG will be listed in the ‘Device Manager’ window which should show ‘EDBG Data Gateway’ and ‘EDBG Debugger’ under Atmel and ‘EDBG Virtual COM Port’ under ‘Ports (COM & LPT)’ as shown in Figure 4-3.

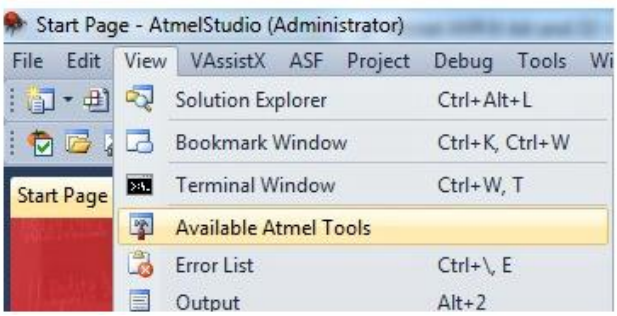
Figure 4-3. Successful EDBG Driver Installation



To ensure that the EDBG tool is getting detected in Atmel Studio, follow the steps below:

Open Atmel Studio 6.2 and go to 'View' → 'Available Atmel Tools' as shown in Figure 4-4.

Figure 4-4. Opening Available Atmel Tools in Atmel Studio



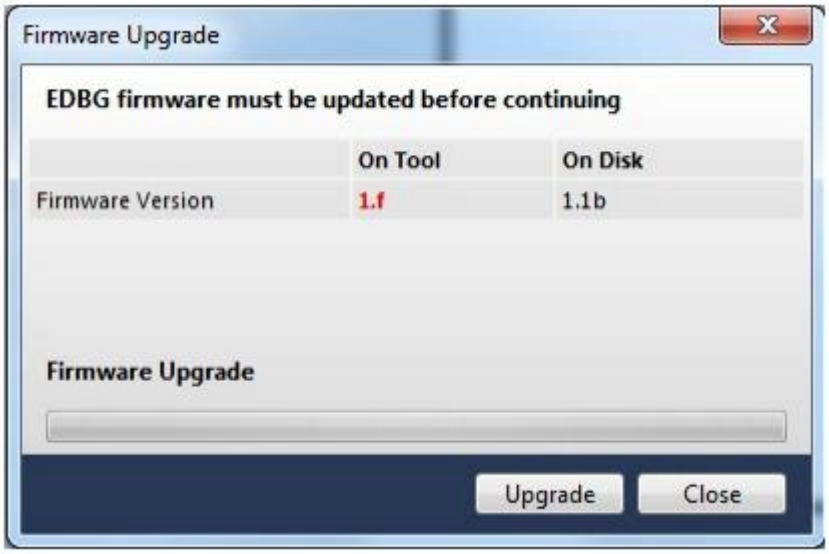
The EDBG should get listed in the tools as "EDBG" and the tool status should display as "Connected". This indicates that the tool is communicating properly with Atmel Studio. If the tool does not get displayed in 'Available Atmel Tools', disconnect the tool and reconnect again.

To verify that the firmware of the EDBG tool is the latest, right click on the EDBG in Available Atmel Tools and select the option 'Upgrade' as shown in Figure 4-5.

Figure 4-5. Upgrading EDBG from Available Atmel Tools Windows

If the firmware is not up-to-date, Atmel Studio will prompt for upgrade as shown in Figure 4-6. Click on the 'Upgrade' button to upgrade the firmware.

Figure 4-6. Firmware Upgrade Window



In case you get "Upgrade Failed" error, power cycle the tool and then try upgrading again.

5 USB Mass Storage Application Overview

The Mass Storage application is a simple file transfer application between the Host Computer and the SAMD11 Xplained Pro kit. The USB data exchange for this application is based on the SCSI (Small Computer System Interface) commands which use two bulk endpoints (one IN and one OUT) to perform the status and data transfer respectively.

The endpoint 0 (Control endpoint) is used only to perform the enumeration process, the errors management, to determine the LUN (Logical Unit Number) value and to perform a reset recovery in case of mass storage error. In computer storage, a logical unit number (LUN) is a number used to identify a logical unit (such as SD card, Data flash, etc.,), which is a device addressed by the SCSI protocol. The Mass Storage class allows one device to manage several storage units at the same time.

Mass storage application is a set of SCSI commands sent by the host to manage the file transfer. The SCSI commands are performed through both endpoints (IN or OUT). Each SCSI command is decoded and transmitted to the appropriate Storage Unit through a command set (Read, Write, etc.)

Figure 5-1. Application Overview

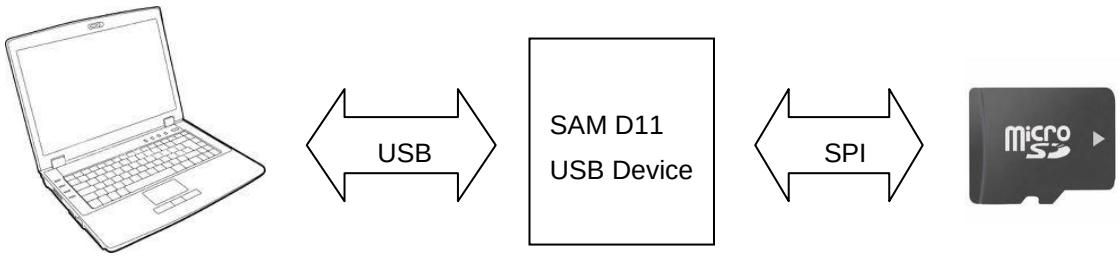


Figure 5-1 shows the application overview. This application uses a micro SD card as storage unit (logical unit), which is connected to the SAMD11 Xplained Pro kit via SPI lines.

The SAM D11 device acts as a mediator, which transfers data between Host Computer and SD card. The SAMD11 is used as a USB device and the data exchange between the Host Computer and SAMD11 is via USB (SCSI commands), whereas the data exchange between the SAMD11 and SD card is via SPI.

6 Mass Storage Class Overview

A Mass Storage device is composed of Default Control Endpoint 0, BULK IN Endpoint and BULK OUT Endpoint.

Figure 6-1 shows the Mass Storage application overview.

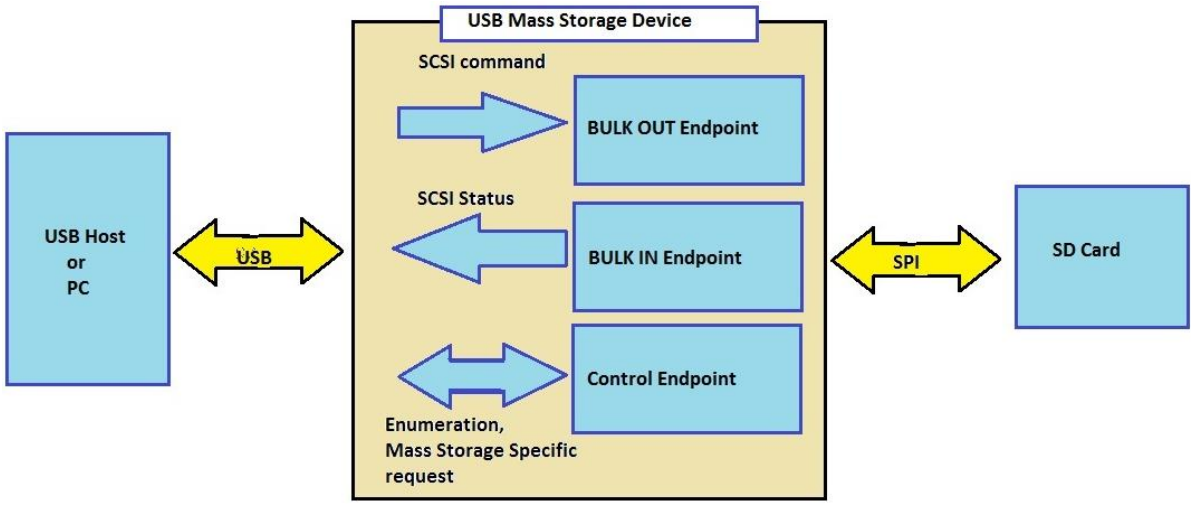
6.1 Mass Storage Device Endpoints

6.1.1 Default Control Endpoint

The standard enumeration process (USB chapter 9 support) is performed through the default control endpoint.

Control endpoint transfer data in both directions; therefore use both endpoint directions (IN and OUT) of a device address.

Figure 6-1. Mass Storage Application Overview



6.1.2 BULK OUT Endpoint

The Host sends Bulk OUT data in the endpoint 2 in the form of Command Block Wrapper. The device has to decode the Command Block Wrapper, handle the SCSI commands in software and read/write data from/to SD card according the received SCSI commands.

6.1.3 BULK IN Endpoint

A BULK IN Endpoint is used to send the status of the last received Command Block Wrapper (CBW from BULK OUT).

6.2 Data Transfers between Host and Device

As the USB bus is a single master bus (the USB Host), each data transfer is initiated by the USB Host, following a specific Command-Data-Status flow (see Figure 6-2).

A Bulk-only protocol transaction begins with the host sending a Command Block Wrapper (CBW) to the device and attempting to make the appropriate data transfer (IN, OUT). The device receives the CBW, checks and interprets it, attempts to satisfy the host's request, and returns status via a Command Status Wrapper (CSW).

Refer USB Mass Storage Bulk-Only Transport Protocol document. See Chapter 9 References.

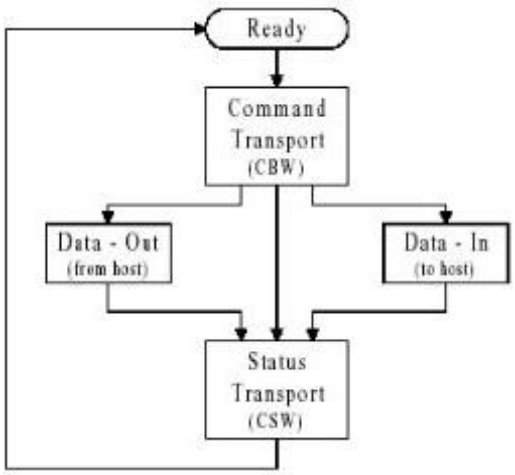
A Successful Bulk transaction (OUT + IN) has two/three stages.

- Command Transport (Command Block Wrapper)

- Data Transport (not necessary for some commands)
- Status Transport (Command Status Wrapper)

Refer the USB Mass Storage Class Bulk-Only Transport document.

Figure 6-2. Command/Data/Status Flow



6.2.2 Command Block Wrapper

The host shall send each CBW, which contains a command block, to the device via the Bulk-Out endpoint.

The CBW is 31 bytes long. The device shall indicate a successful transport of a CBW by acknowledging (ACKing) the CBW. The CBW contains some USB information such as the addressed LUN, the length of the SCSI command, and also contains the SCSI command for the memory.

Figure 6-3 example Command Block Wrapper send by the Host to device.

Figure 6-3. Command Block Wrapper Example

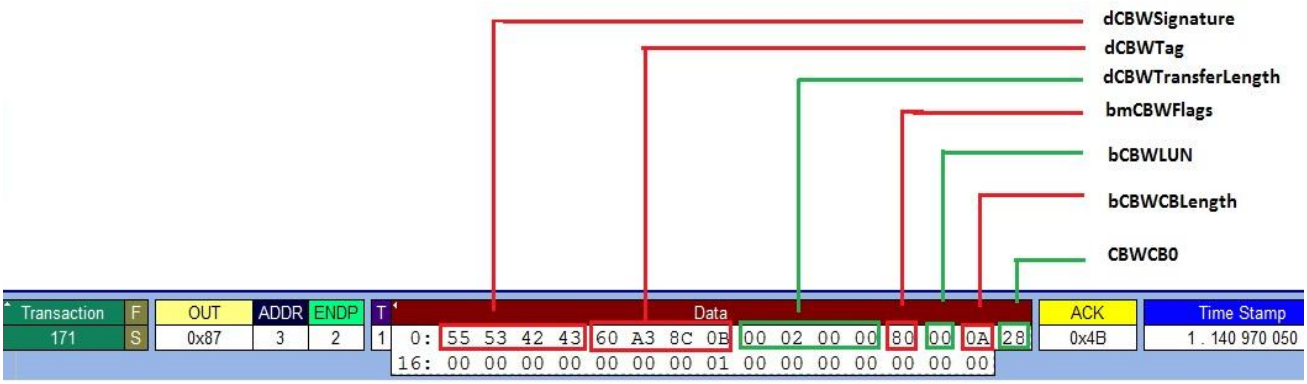


Table 6-1. Command Block Wrapper Description

Byte	Command	Byte count	Value	Description
0-3	dCBWSignature	4	0x55534243	55534243 indicate that the command is a Command Block wrapper.
4-7	dCBWTag	4	0x0B8CA360	A Command Block Tag sent by the host. The device shall echo the contents of this field back to the host in the dCSWTag field of the associated CSW.

Byte	Command	Byte count	Value	Description
8-11	dCBWDataTransferLength	4	0x00000200	0x00000200 – 512 bytes. 512 bytes of data that the host expects to transfer on the Bulk-In or Bulk-OUT endpoint (i.e., Host expects to transfer/receive 512 bytes of data).
12	bmCBWFlags	1	0x80	This flag indicates whether the 512 bytes are received/ transferred to Host. Since bit 7 is set to 1, Data-In from the device to the Host. So, Host expects 512 bytes of data from Device.
13	bCBWLUN	1	0x00	Logical unit is SD card (1 LUN). LUN value is set to 0 since single LUN is selected.
14	bCBCBLength	1	0x0A	This indicates the valid length of the upcoming field. (CBWCB). I.e., in the remaining 17 bytes of CBW only 10 (0x0A) bytes are valid.
15	CBWCB0	1	0x28	0x28 indicates the command is a Read_10 command. (Remaining CBWCB value is depending on the specific command).

6.2.3 Data Transport

The Host shall attempt to transfer the exact number of bytes to or from the device as specified in the Command Block Wrapper.

6.2.4 Command Status Wrapper

Command Status Wrapper contains the SCSI status for the last received CBW. CSW is 13 bytes long, which has CSW signature, Tag, status. If the Status is passed, the Host will send the next following command.

If the status is different from Passed (Failed, Phase error, etc.), the Host will ask for more information regarding the error by sending a REQUEST SENSE command.

Figure 6-4 shows an example Command Status Wrapper sent by the device to Host.

Figure 6-4. Command Status Wrapper example

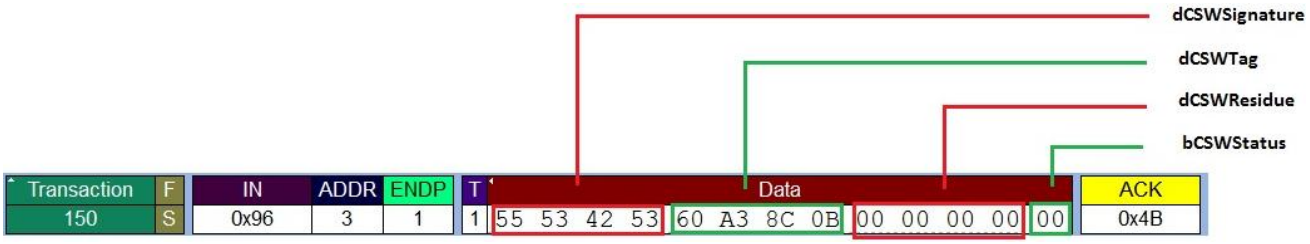


Table 6-2. Command Status Wrapper Description

Byte	Command	Byte count	Value	Description
0-3	dCSWSignature	4	0x55534253	55534253 indicates that the command is a Command Status wrapper.
4-7	dCBWTag	4	0x0B8CA360	The device shall set this field to value received in dCBWTag of CBW.

Byte	Command	Byte count	Value	Description
8-11	dCSWDataResidue	4	0x00000000	The difference between the amounts of data expected as stated in the dCBWDataTransferLength and the actual amount of relevant data sent by the device. 0 – indicates the device to send all 512 data bytes to Host.
12	bCSWStatus	1	0x00	00 indicates the command passed.

6.3 SCSI Commands

For a Mass storage class, a USB device should implement the following SCSI commands.

Sl. no	Command	Description	Op code
1	INQUIRY	Get Device Information	0x12
2	TEST UNIT READY	Request the device to report if it is ready	0x00
3	READ CAPACITY(10)	Report Current Media Capacity	0x25
4.	READ(10)	Transfer binary data from media(SD card (for example)) to Host	0x28
5.	WRITE(10)	Transfer binary data from the Host to media	0x2A
6.	REQUEST SENSE	Transfer Status sense data to Host whenever CSW failed occurs	0x03
7.	START/STOP	Request Removable media device to load or unload its media	0x1B

7 Mass Storage Firmware Workflow

Figure 7-1. USB MSC Application Flow Chart

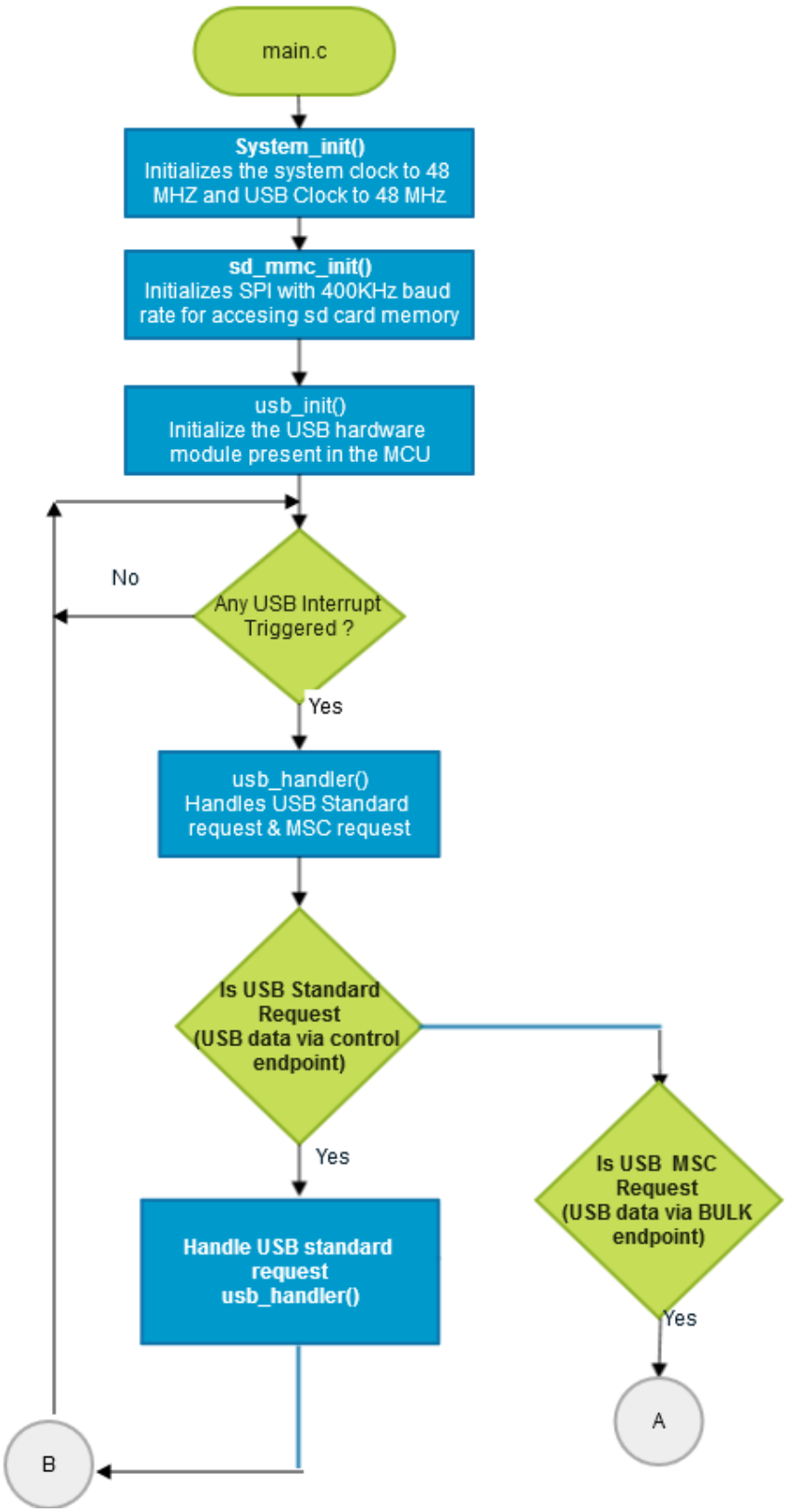
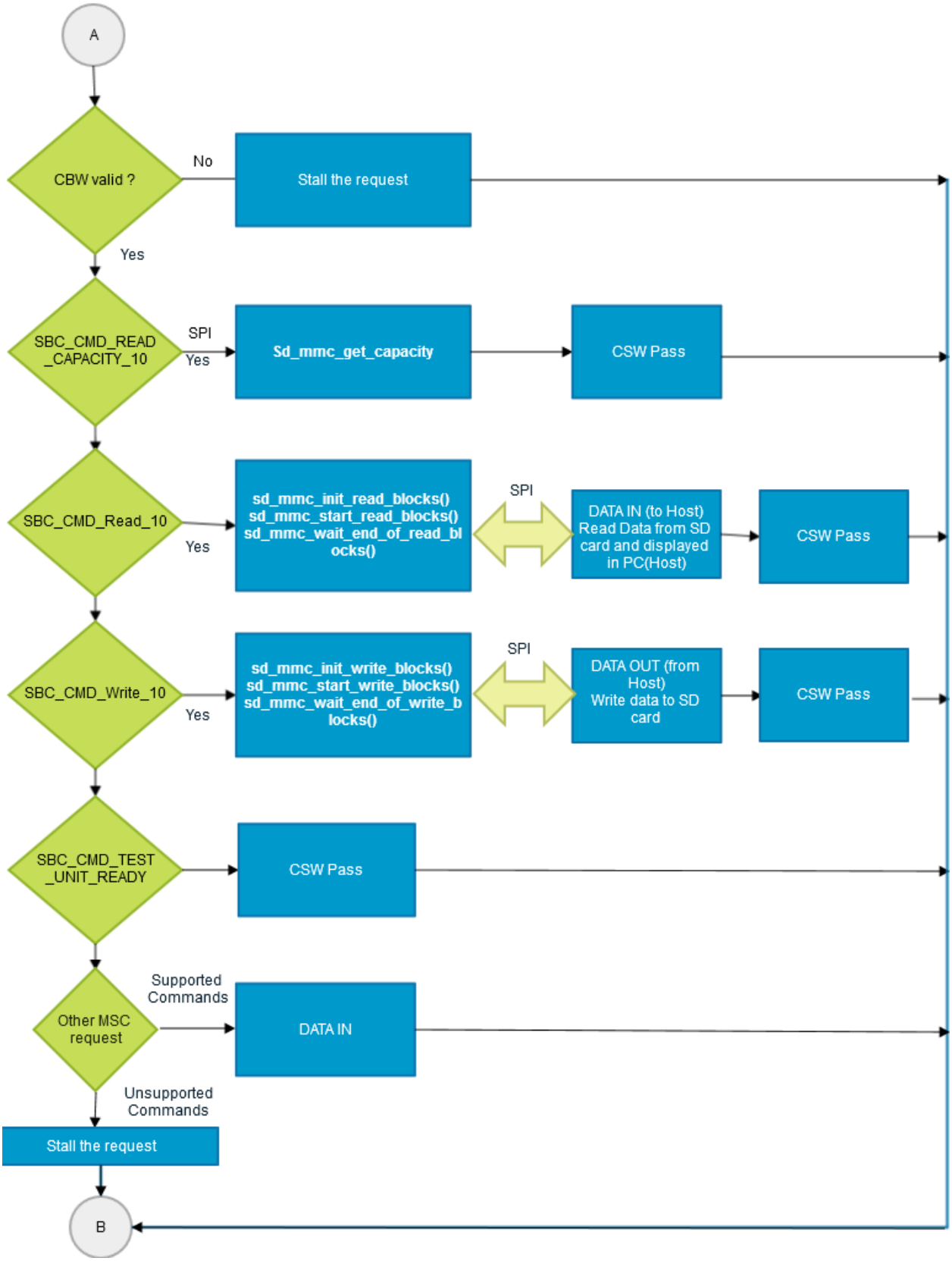


Figure 7-2. USB MSC Application Flow Chart (continued)



7.2 Module Configuration

The firmware uses ASF3.19 for USB, SPI clock configuration. ASF drivers are used for SPI communication between SAM D11 device and SD card.

7.2.1 Clock Initialization

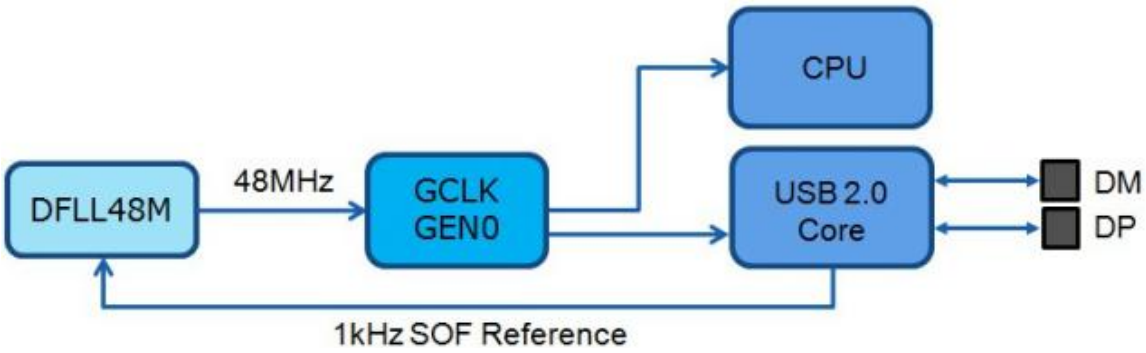
SAM D11 USB requires a 48MHz ±0.25% reference clock (GCLK_USB) for its operations. This GCLK_USB is required to clock the USB.

In addition to the GCLK_USB requirement, the CLK_USB_AHB should be a minimum of 8MHz to follow the USB data rate in full-speed mode.

The System_init() function initializes the DFLL48M in USB clock recovery mode with 1kHz USB SOF (Start Of Frame) signal as reference clock thereby initializes the system clock to 48MHz.

The 48MHz clock output is given as the source clock for GCLK Generator 0 which feeds the CPU and the USB module as shown in Figure 7-3.

Figure 7-3. USB Clock Configuration Block Diagram



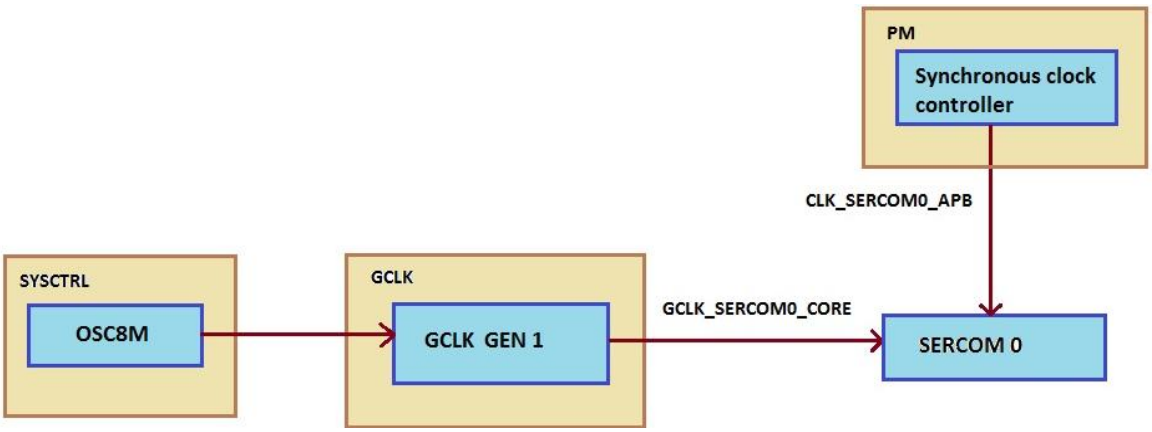
7.2.2 SD Card Initialization

Sd_mmc_init() function initializes the SERCOM0 SPI module with 400KHz SPI baud rate.

The 8MHz clock output is given as the source clock for GCLK Generator 1 which feeds the SERCOM 0 CORE.

Figure 7-4 shows the SERCOM0 clock configuration.

Figure 7-4. SERCOM0 Clock Configuration Block Diagram



This function also initialize SD Card detect pins (EXT1 pin 10 in I/O1 Xplained Pro kit) and set card detect pin as inputs which helps to identify whether SD Card is inserted in the slot or not. Table 7-1 shows SERCOM0 I/O pins for SPI module.

Table 7-1. SERCOM0 I/O Pins

Name	I/O pin	I/O peripheral MUX	Description
SERCOM[0] PAD [0]	PA06	C	DOPO (MOSI)
SERCOM[0] PAD [1]	PA07	C	SPI clock (SCK)
SERCOM[0] PAD [2]	PA08	D	SS
SERCOM[0] PAD [3]	PA09	D	DIPO (MISO)

7.2.3 USB Initialization

Usb_init() function initializes the USB module, set up USB I/O pins as shown in Table 7-2, set speed configuration to Full speed, initialize endpoint table RAM location to a known value 0.

Table 7-2. USB I/O Pins

Signal	I/O pin	I/O peripheral MUX	Description
USB/DM	PA24	G	USB Data Negative Line
USB/DP	PA25	G	USB Data Positive Line
USB/SOF 1KHz	PA23	G	USB SOF Output

7.3 USB Request

Once the configuration is done, the execution enters to infinite loop and listens for interrupt. If a USB interrupt occurs, the execution goes to usb_handler(). The firmware checks if the request is a standard request or mass storage request in usb_handler().

7.3.1 USB Standard Request

The standard enumeration process (USB chapter 9 support) is performed through the default control endpoint.

This process consists of a set of parameters sent by the device to the host to identify the device class and load the appropriate drivers. These parameters are called the descriptors.

7.3.2 USB Mass Storage Request

If the USB interrupt occurs from BULK OUT endpoint and if CBW received, the request is valid USB Mass Storage request. The firmware checks whether the received CBW is valid by checking the CBW signature.

SBC_CMD_INQUIRY

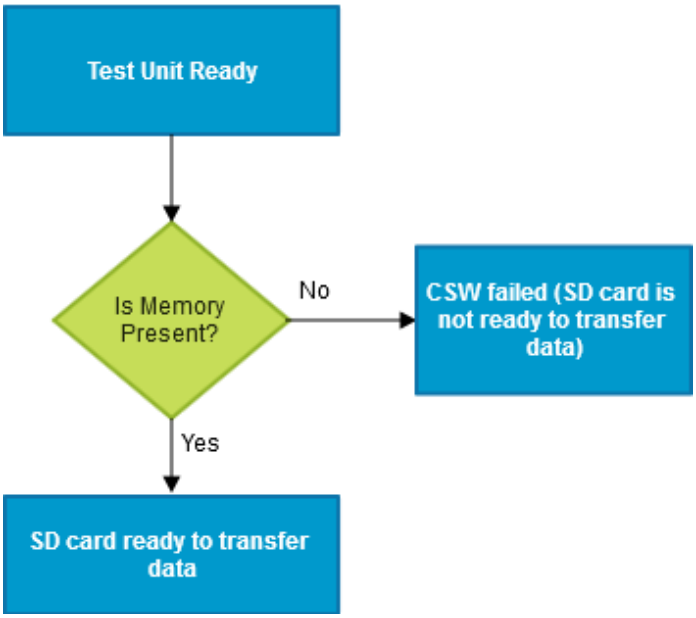
The Host sends INQUIRY command to request more details about the connected USB device such as whether the device is removable or not, VID, PID, Product revision level.

This returns the USB Device information such as VID, PID, Product Revision level, connected disk is Removable disk.

SBC_CMD_TEST_UNIT_READY

This case returns the connected memory state. Data transport is not required for this command. The device will return either command passed or failed. If the command passed, the disk is ready to transfer data. Figure 7-5 shows the flowchart representation for TEST UNIT READY command. The TEST UNIT READY command is useful in that it allows a Host Computer to poll a Device until it is ready.

Figure 7-5. Test Unit Ready



SBC_CMD_READ_CAPACITY_10

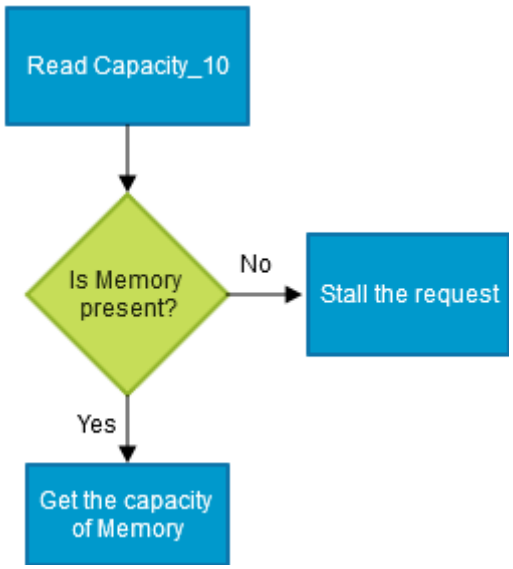
This is used to get the capacity of the connected SD card and display the capacity in PC. i.e., how much amount of data the device can store.

The Host requests the Last Logical Block address of the device (logical unit) and number of bytes in each sector. Each sector will have typically 512 bytes.

This case returns the address of the last valid sector. The sector size is typically 512 bytes.

For example, a memory of 16Kbytes returns ((16*1024)/512 -1) = 31. So, last valid sector is 31.

Figure 7-6. READ_CAPACITY_10



SBC_CMD_READ_10

This case is used to read data from SD card when host request to read a blocks of data from the memory.

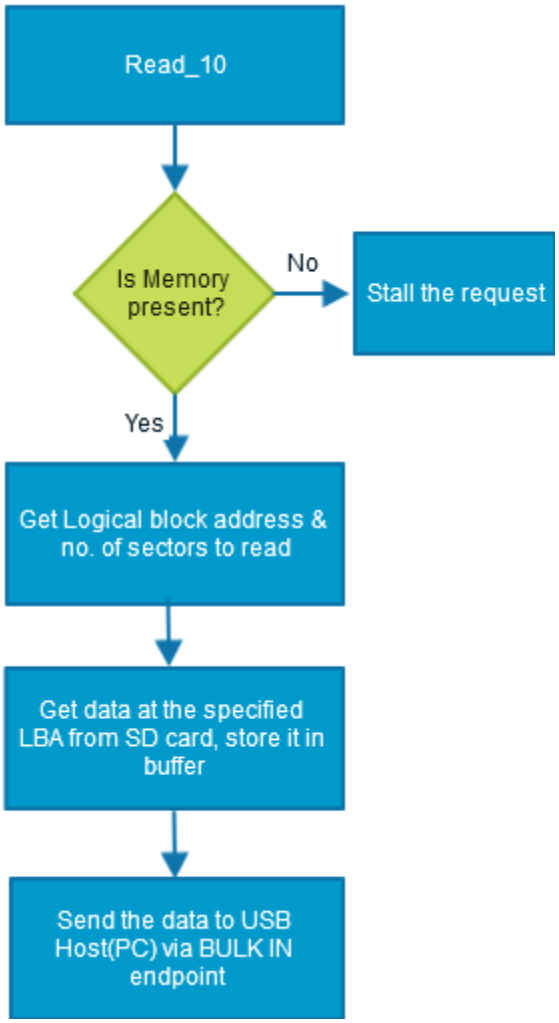
On receiving a READ (10) command, the device has to send the contents of the requested blocks to the host.

From the Host point of view, the USB memory is organized in logical blocks (sectors) of 512 bytes.

The Host always addresses the memory with:

- Logical sector number
- The number of contiguous sectors to read/write

Figure 7-7. Read(10) Command



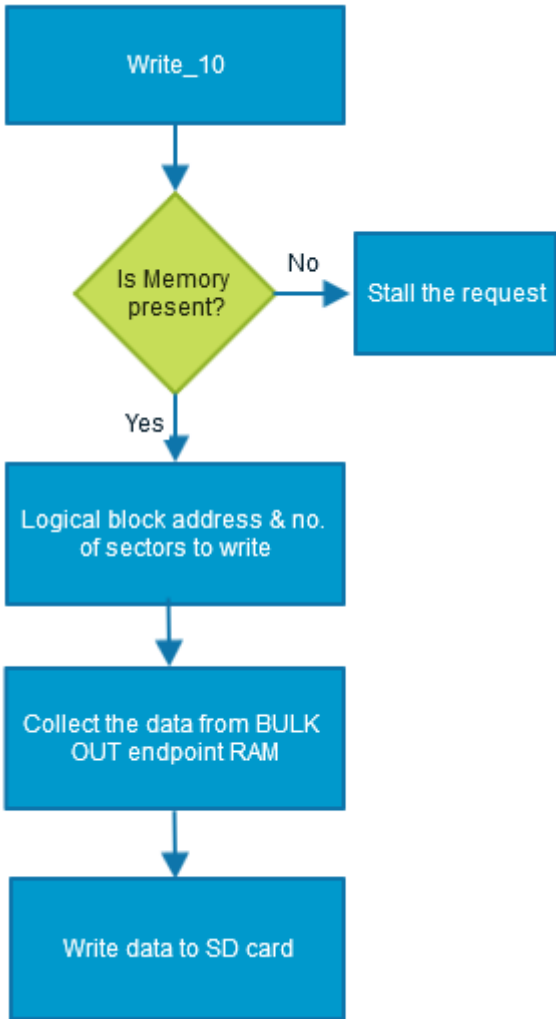
SBC_CMD_WRITE_10

This case is used to write data to SD card when host request to write certain blocks (512 bytes/block) of data.

After receiving a WRITE (10) command block, the device should receive the data in the bulk OUT endpoint.

Then the device should write the received data to the specified locations in the Memory.

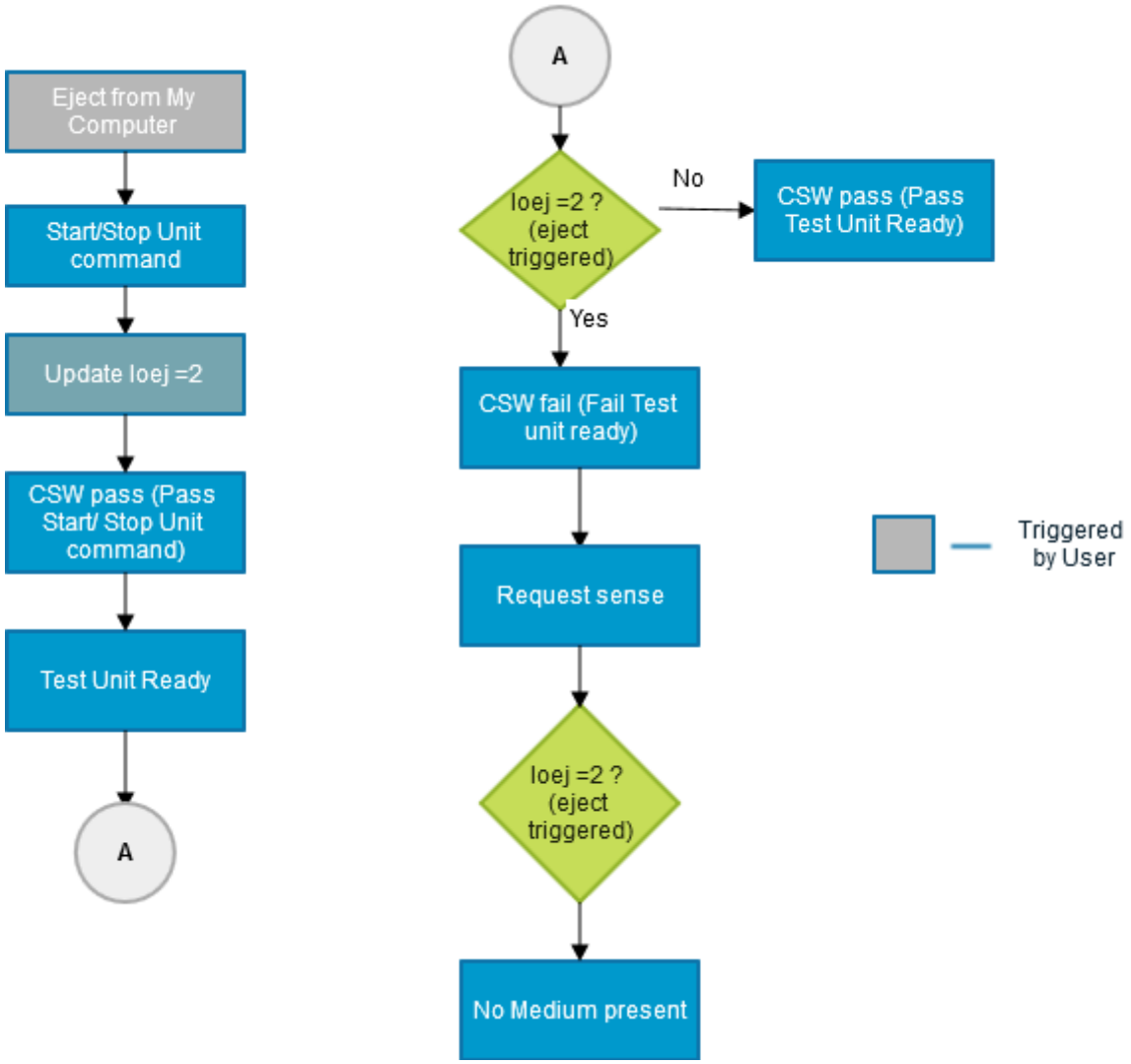
Figure 7-8. Write(10) Command



SBC_CMD_START_STOP_UNIT

This is used to eject the removable disk from PC. If you right click on the Removable disk and click Eject, Host issues Start/Stop unit command in the command block wrapper.

Figure 7-9. Eject Removable Disk



8 Mass Storage Application

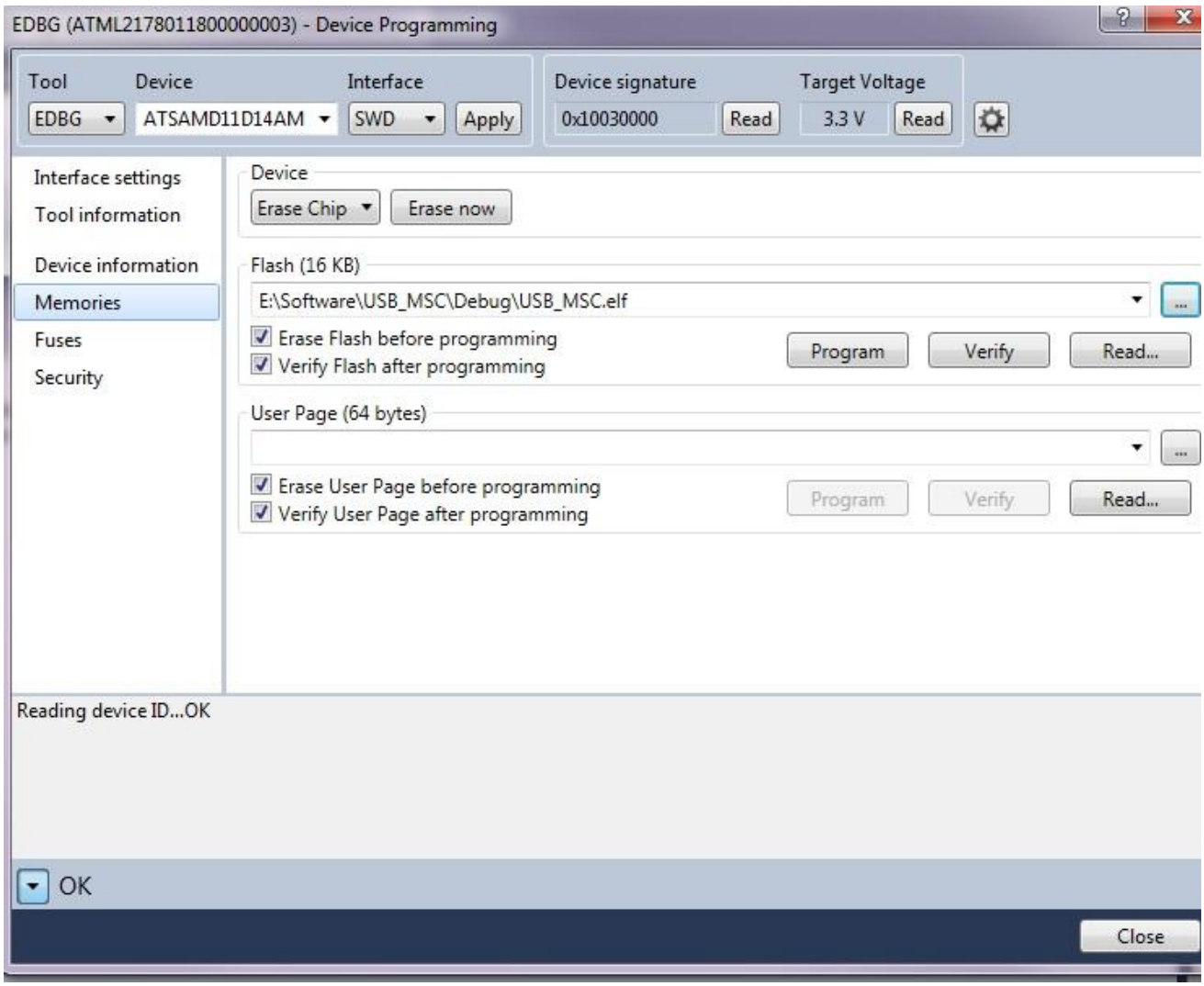
Ensure that the hardware connections are ready as shown in Figure 4-1.

8.1 Programming SAM D11 Xplained Pro Kit

1. The SAM D11 USB Mass Storage Device application code is available in the latest ASF with Atmel Studio. Follow the steps below to load the SAM D11 USB Mass Storage Device application code in the Atmel Studio:
2. To load the example project in Atmel Studio, go to ‘File’ → ‘New’ and click on ‘Example Project…’ The shortcut key for to do this is (CTRL +Shift + E):
3. Type “SAM D11 USB Mass Storage Device” in the search box from ‘New Example Project from ASF’. So that it will show the ‘SAM D11 USB Mass Storage Device’ project solution available in the ASF.
4. Compile the Project by selecting Build → Build solution.
5. The application occupies the following resources when compiling optimization for size.(-Os)

```
Program Memory Usage     :      12K   (approximately)
Data Memory Usage        :      1.8K  (approximately)
```

Figure 8-1. Device Programming

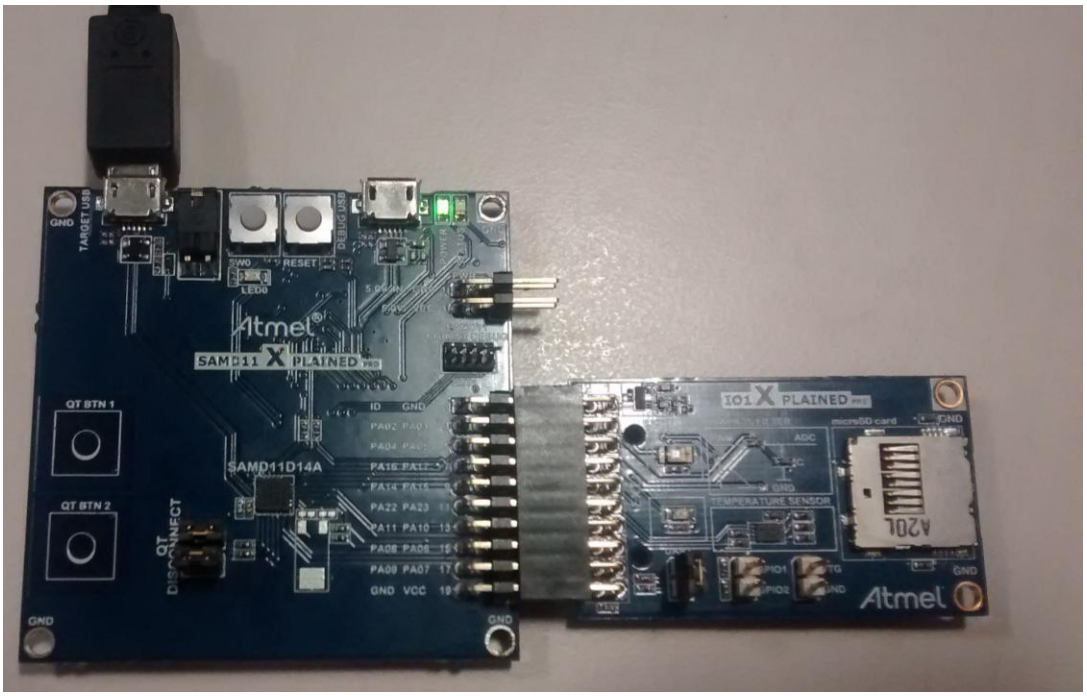


6. Open Device Programming window. Tools → Device Programming.
7. Select Tool as ‘EDBG’, Device as ‘ATSAMD11D14AM’, and interface as ‘SWD’ as shown in Figure 8-1.
8. Give the path for USB_MSC_DEVICE.elf in Memories tab under Flash.
9. Click ‘Program’.

8.2 Mass Storage Enumeration

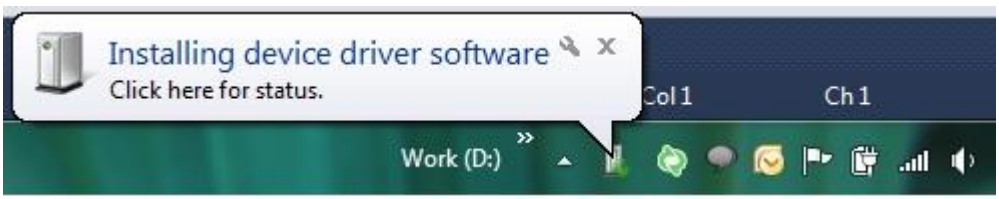
Plug the Micro-USB cable to SAM D11 Target USB, connect I/O1 Xplained kit in SAM D11 Xplained Pro EXT1 as shown in Figure 8-2.

Figure 8-2. SAM D11 Xplained Pro with Target USB Connected



If the SAMD 11 Xplained Pro kit is connected to a PC, a popup window will show in the Tray showing installing device driver software as shown in Figure 8-3. The required drivers for the Mass Storage application will be automatically installed.

Figure 8-3. Installing Device Driver Software



Click “Click here for status”, wait for the USB Mass Storage device, and the Atmel SD CARD USB Device is ready to use. Once the drivers are installed, your device is ready to use.

If the driver installation is proper, the USB Mass Storage will be listed in the ‘Device Manager’ window, which should show ‘USB Mass Storage Device’ under Universal Serial Bus Controller and ‘ATMEL SD CARD USB Device’ under Disk Drivers ’ as shown in Figure 8-5.

Figure 8-4. SAM D11 Xplained Pro MSC Driver Installation

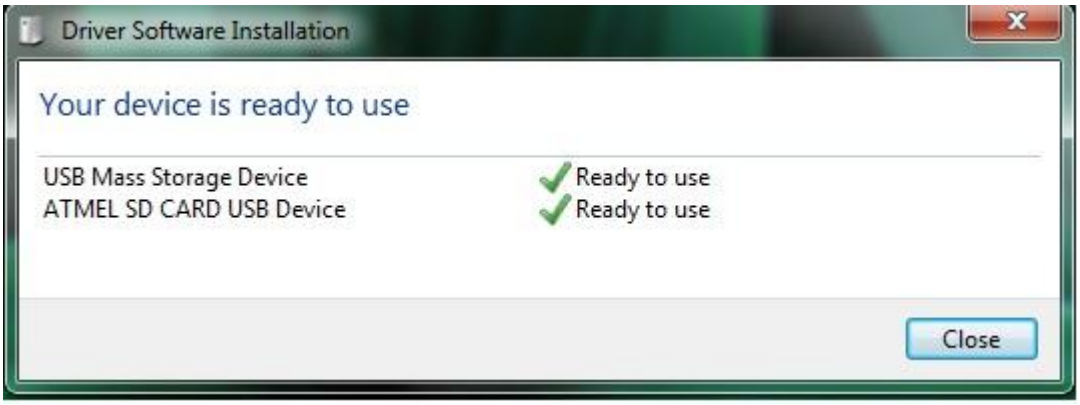
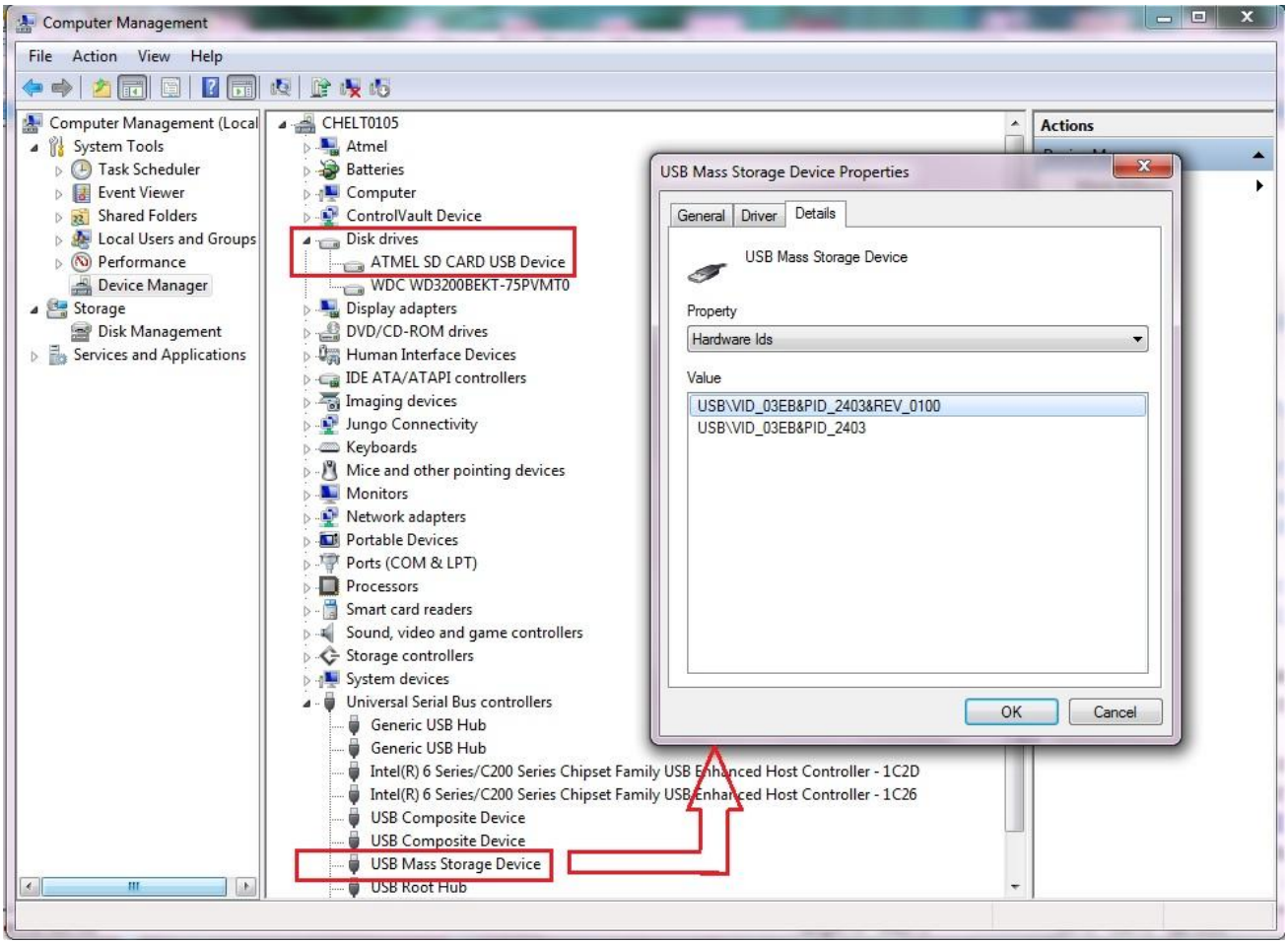


Figure 8-5. Mass Storage Enumeration

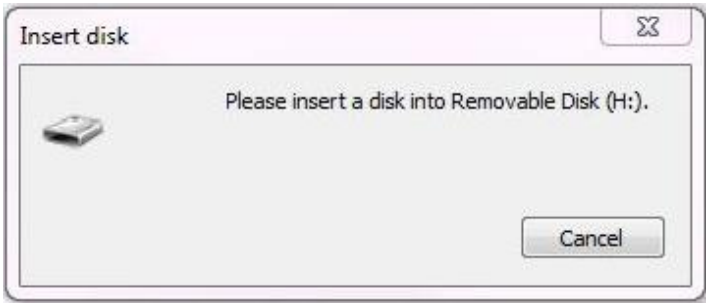


Open the My Computer window in the Host Computer, a Removable disk icon will show in My Computer. If the SD card is not inserted in the slot, a Removable disk icon without showing the capacity will show in My Computer as shown in Figure 8-6.

Figure 8-6. Removable Disk Icon

If you double-click the Removable disk, a popup window appears saying “Please insert a disk into Removable disk” as shown in Figure 8-7.This means that Memory, which is used to read/write data, is not connected.

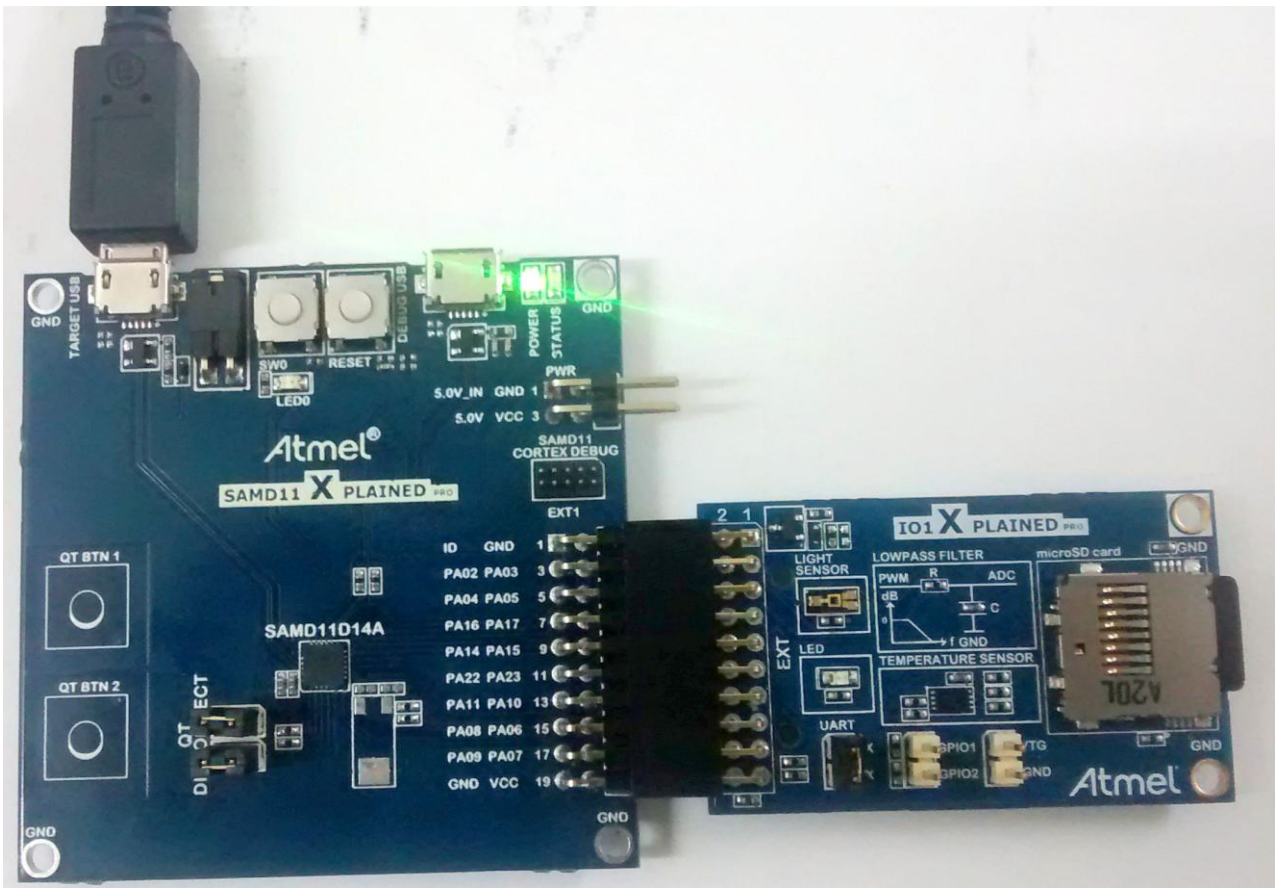
Figure 8-7. Insert a Disk



8.3 Load SD Card Memory

Insert the micro SD card into the slot.

Figure 8-8. SAM D11 Xplained Pro with Target USB Connected



If the I/O1 Xplained kit is properly connected to SAMD11 Xplained Pro kit, the capacity of the Removable disk, the remaining space in the disk, will show in My Computer as shown in Figure 8-9.

Figure 8-9. Removable Disk Showing Capacity

Now, you can start transferring files between the PC and SD card. Just drag the files to SD card and the files will be written to the disk.

Note: If Host Computer asks for formatting the disk, accept and format it.

If you remove the SD card at any time, the SD card memory will unloaded and the icon appears as shown in Figure 8-6.

9 References

USB Class Specific Documents

USB Mass Storage Bulk-Only Transport

10 Revision History

Doc Rev.	Date	Comments
42375A	11/2014	Initial document release.

Atmel | Enabling Unlimited Possibilities®

Atmel Corporation 1600 Technology Drive, San Jose, CA 95110 USA **T:** (+1)(408) 441.0311 **F:** (+1)(408) 436.4200 | **www.atmel.com**