AVR32760: AVR32 UC3 USB DFU Bootloader Protocol

32-bit AVR® Microcontrollers

Application Note

1. Introduction

This application note describes the USB DFU Protocol used in the AVR®32 UC3 USB bootloader.
In a first part of the document, it gives an overview the USB DFU class protocol and details the Atmel DFU protocol used by the bootloader in a second part of the document.

1.1 Intended Audience

This application note is intended for people who are interested in:

- modifying the UC3 USB DFU bootloader for instance to add specific commands.
- building their own UC3 DFU programmer tool.

1.2 Prerequisites

It is recommended to the audience to be familiar with the USB specification or at least with the USB transfer protocol.

1.3 Supported Devices

All the AVR32 UC3 parts implementing a USB device peripheral support this USB DFU Protocol.

1.4 Terms and Abbreviations

The meanings of some words have been stretched to suit the purposes of this document. These definitions are intended to clarify the discussions that follow.

- DFU — Device Firmware Upgrade
- Firmware — Executable software stored in a write-able, nonvolatile memory on a USB device.
- Upgrade — To overwrite the firmware of a device.
  (1) The act of overwriting the firmware of a device.
  (2) New firmware intended to replace a device's existing firmware.
- Download — To transmit information from host to device.
- Upload — To transmit information from device to host.
- IN — USB transfer packet from device to host
- OUT — USB transfer packet from host to device
- ZLP — USB Zero Length Packet

Rev 32131A–AVR32–09/09

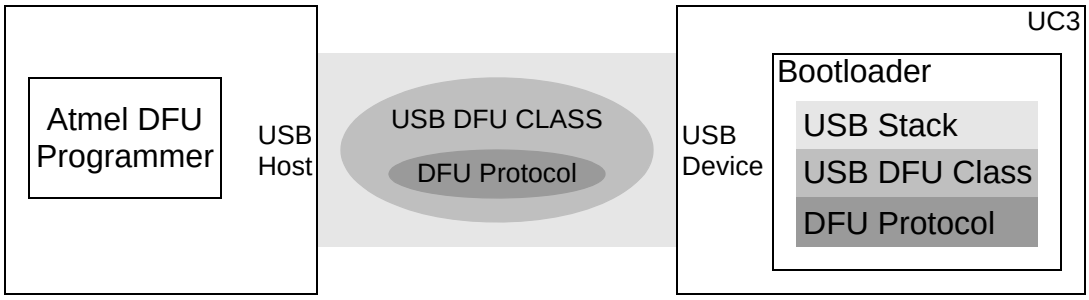
1.5 References

- AVR32 UC3 USB DFU Bootloader datasheet:
  http://www.atmel.com/dyn/resources/prod_documents/doc7745.pdf
- Universal Serial Bus Device Class Specification for Device Firmware Upgrade:
  www.usb.org/developers/devclass_docs/usbdfu10.pdf

1.6 System Overview

The Atmel DFU protocol is encapsulated in some USB DFU specific requests (see section "DFU Specific Requests" on page 3). Figure 1-1 shows the system environment.

Figure 1-1. System Environment



2. Device Firmware Upgrade Class

2.1 Introduction

The Device Firmware Upgrade (DFU) is the mechanism for accomplishing the task of upgrading the device firmware. Any class of USB device can exploit this capability by supporting the requirements specified in this document.

Because it is impractical for a device to concurrently perform both DFU operations and its normal run-time activities, those normal activities must cease for the duration of the DFU operations. Doing so means that the device must change its operating mode; i.e., a printer is **not** a printer while it is undergoing a firmware upgrade; it is a PROM programmer. However, a device that supports DFU is not capable of changing its mode of operation on its own. External (human or host operating system) intervention is required.

For more information about the USB DFU Class, refer to the Universal Serial Bus Device Class Specification for Device Firmware Upgrade specification (see section "References" on page 2).

2.2 DFU Specific Requests

In addition of the USB standard requests, 7 DFU class-specific requests are employed to accomplish the upgrade operations as detailed in Table 2-1.

Table 2-1. DFU Class-specific Requests

bmRequestType	bRequest	wValue	wIndex	wLength	Data
0010 0001b	DFU_DETACH	wTimeout	Interface	Zero	none
0010 0001b	DFU_DNLOAD	wBlock	Interface	Length	Atmel Specific(1)
1010 0001b	DFU_UPLOAD	wBlock	Interface	Length	Atmel Specific(1)
1010 0001b	DFU_GETSTATUS	Zero	Interface	6	Status
0010 0001b	DFU_CLRSTATUS	Zero	Interface	Zero	none
1010 0001b	DFU_GETSTATE	Zero	Interface	1	State
0010 0001b	DFU_ABORT	Zero	Interface	Zero	none

Note: 1. Refer to section "Atmel® DFU Protocol" on page 8 for detailed protocol.

2.3 DFU Descriptors Set

The device exports the DFU descriptor set, which contains:

- A DFU device descriptor
- A single configuration descriptor
- A single interface descriptor (including descriptors for alternate settings, if present)
- A single functional descriptor

2.3.1 DFU Device Descriptor

This descriptor is only present in the DFU mode descriptor set. The DFU class code is reported in the *bDeviceClass* field of this descriptor.

Table 2-2. DFU Mode Device Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	12h	Size of this descriptor, in bytes
1	bDescriptorType	1	01h	DFU FUNCTIONAL descriptor type
2	bcdUSB	2	0100h	USB specification release number in binary coded decimal
4	bDeviceClass	1	FEh	Application Specific Class Code
5	bDeviceSubClass	1	01h	Device Firmware Upgrade Code
6	bDeviceProtocol	1	00h	The device does not use a class specific protocol on this interface
7	bMaxPacketSize0	1	32	Maximum packet size for endpoint zero
8	idVendor	2	(1)	Vendor ID
10	idProduct	2	(1)	Product ID
12	bcdDevice	2	(1)	Device release number in binary coded decimal
14	iManufacturer	1	0	Index of string descriptor
15	iProduct	1	0	Index of string descriptor
16	iSerialNumber	1	0	Index of string descriptor
17	bNumConfigurations	1	01h	One configuration only for DFU

Note: 1. Refer to Table 2-3.

Table 2-3. Device Parameters

Field	Value	Description
idVendor	0x03EB	Vendor ID
idProduct	0x2FF8 (AT32UC3A0/1) 0x2FF6 (AT32UC3B0/1) 0x2FF1 (AT32UC3A3)	Product ID
bcdDevice	0x1000	Release Number

2.3.2 DFU Configuration Descriptor

This descriptor is identical to the standard configuration descriptor described in the USB DFU specification version 1.0, with the exception that the *bNumInterfaces* field must contain the value 01h.

2.3.3 DFU Interface Descriptor

This is the descriptor for the only interface available when operating in DFU mode. Therefore, the value of the *bInterfaceNumber* field is always zero.

Table 2-4. DFU Mode Interface Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	09h	Size of this descriptor, in bytes
1	bDescriptorType	1	04h	INTERFACE descriptor type
2	bInterfaceNumber	1	00h	Number of this interface

Offset	Field	Size	Value	Description
3	bAlternateSetting	1	00h	Alternate setting
4	bNumEndpoints	1	00h	Only the control pipe is used
5	bInterfaceClass	1	FEh	Application Specific Class Code
6	bInterfaceSubClass	1	01h	Device Firmware Upgrade Code
7	bInterfaceProtocol	1	00h	The device doesn't use a class specific protocol on this interface
8	iInterface	1	00h	Index of the String descriptor for this interface

2.3.4 DFU Functional Descriptor

Table 2-5. DFU Functional Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	07h	Size of this descriptor, in bytes
1	bDescriptorType	1	21h	DFU FUNCTIONAL descriptor type
2	bmAttributes	1	Bit mask	DFU Attributes: bit 7..3: reserved bit 2: device is able to communicate via USB after Manifestation phase 1 = yes, 0 = no, must see bus reset bit 1: *bitCanUpload*: upload capable 1 = yes, 0 = no bit 0: *bitCanDnload*: download capable 1 = yes, 0 = no
3	wDetachTimeOut	2	Number	Time in milliseconds that the device will wait after receipt of the DFU_DETACH request. If this time elapses without a USB reset, the device will terminate the re-configuration phase and revert back to normal operation. This represents the maximum time that the device can wait (depending on its timers, ...). The Host may specify a shorter time-out in the DFU_DETACH request.
5	wTransferSize	2	Number	Maximum number of bytes that the device can accept per control-write transaction

2.4 Device Status

2.4.1 Get Status

The Host employs the DFU_GETSTATUS request to facilitate synchronization with the device. This status gives information on the execution of the previous request: in progress/OK/Fail/...

bmRequestType	bRequest	wValue	wIndex	wLength	Data
1010 0001b	DFU_GETSTATUS	Zero	Interface	6	Status

The device responds to the DFU_GETSTATUS request with a payload packet containing the following data:

Table 2-6. DFU_GETSTATUS Response

Offset	Field	Size	Value	Description
0	bStatus	1	Number	An indication of the status resulting from the execution of the most recent request.

Offset	Field	Size	Value	Description
1	bwPollTimeOut	3	Number	Minimum time in milliseconds that the host should wait before sending a subsequent DFU_GETSTATUS. The purpose of this field is to allow the device to dynamically adjust the amount of time that the device expects the host to wait between the status phase of the next DFU_DNLOAD and the subsequent solicitation of the device's status via DFU_GETSTATUS.
4	bState	1	Number	An indication of the state that the device is going to enter immediately following transmission of this response.
5	iString	1	Index	Index of status description in string table.

Table 2-7. *bStatus* values

Status	Value	Description
OK	0x00	No error condition is present
errTARGET	0x01	File is not targeted for use by this device
errFILE	0x02	File is for this device but fails some vendor-specific verification test
errWRITE	0x03	Device id unable to write memory
errERASE	0x04	Memory erase function failed
errCHECK_ERASED	0x05	Memory erase check failed
errPROG	0x06	Program memory function failed
errVERIFY	0x07	Programmed memory failed verification
errADDRESS	0x08	Cannot program memory due to received address that is out of range
errNOTDONE	0x09	Received DFU_DNLOAD with *wLength* = 0, but device does not think it has all the data yet.
errFIRMWARE	0x0A	Device's firmware is corrupted. It cannot return to run-time operations
errVENDOR	0x0B	*iString* indicates a vendor-specific error
errUSBR	0x0C	Device detected unexpected USB reset signaling
errPOR	0x0D	Device detected unexpected power on reset
errUNKNOWN	0x0E	Something went wrong, but the device does not know what it was
errSTALLEDPK	0x0F	Device stalled an unexpected request

Table 2-8. *bState* Values

State	Value	Description
appIDLE	0	Device is running its normal application
appDETACH	1	Device is running its normal application, has received the DFU_DETACH request, and is waiting for a USB reset
dfuIDLE	2	Device is operating in the DFU mode and is waiting for requests
dfuDNLOAD-SYNC	3	Device has received a block and is waiting for the Host to solicit the status via DFU_GETSTATUS
dfuDNBUSY	4	Device is programming a control-write block into its non volatile memories

State	Value	Description
dfuDNLOAD-IDLE	5	Device is processing a download operation. Expecting DFU_DNLOAD requests
dfuMANIFEST-SYNC	6	Device has received the final block of firmware from the Host and is waiting for receipt of DFU_GETSTATUS to begin the Manifestation phase or device has completed the Manifestation phase and is waiting for receipt of DFU_GETSTATUS.
dfuMANIFEST	7	Device is in the Manifestation phase.
dfuMANIFEST-WAIT-RESET	8	Device has programmed its memories and is waiting for a USB reset or a power on reset.
dfuUPLOAD-IDLE	9	The device is processing an upload operation. Expecting DFU_UPLOAD requests.
dfuERROR	10	An error has occurred. Awaiting the DFU_CLRSTATUS request.

2.4.2 Clear Status

Any time the device detects an error and reports an error indication status to the host in the response to a DFU_GETSTATUS request, it enters the dfuERROR state. The device cannot transition from the dfuERROR state, after reporting any error status, until after it has received a DFU_CLRSTATUS request. Upon receipt of DFU_CLRSTATUS, the device sets a status of OK and transitions to the dfuIDLE state. Only then is it able to transition to other states.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
0010 0001b	DFU_CLRSTATUS	Zero	Interface	0	None

2.4.3 Device State

This request solicits a report about the state of the device. The state reported is the current state of the device with no change in state upon transmission of the response. The values specified in the *bState* field are identical to those reported in DFU_GETSTATUS.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
1010 0001b	DFU_GETSTATE	Zero	Interface	1	State

2.4.4 DFU_ABORT request

The DFU_ABORT request enables the device to exit from certain states and return to the DFU_IDLE state. The device sets the OK status on receipt of this request. For more information, see the corresponding state transition summary.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
1010 0001b	DFU_ABORT	Zero	Interface	0	None

3. Atmel® DFU Protocol

The Atmel DFU protocol is generic and may support other physical layers than USB. For convenience the following section describes the protocol in the USB DFU environment.

3.1 Selecting a Memory

Prior to any read or program operation, a memory target must be selected as well as the page offset inside this memory.

This is achieved by sending the SELECT_MEMORY_UNIT command and the SELECT_MEMORY_PAGE command.

3.1.1 Selecting Memory Unit

3.1.1.1 Command

The SELECT_MEMORY_UNIT command is 4 bytes long as detailed in Table 3-1.

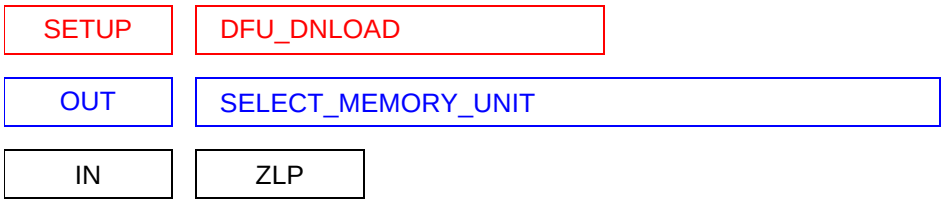
Table 3-1. SELECT_MEMORY_UNIT Command Format

Command Identifier	data[0]	data[1]	data[2]	data[3]	data[4]	Comment
	CMD_GROUP_SELECT					
	CMD_SELECT_MEMORY					
		MEMORY_UNIT				
			0x00			Flash Memory
			0x01			Reserved for future use
0x06	0x03	0x00	0x02			Security Memory
			0x03			Configuration Memory
			0x04			Bootloader Memory
			0x05			Signature Memory
			0x06			User Page Memory

For more information on the selected Memory content, refer to the AVR32 UC3 USB DFU Bootloader datasheet.

3.1.1.2 Request From Host

Figure 3-1. SELECT_MEMORY_UNIT Request



3.1.1.3 Request Status

After sending the SELECT_MEMORY_UNIT command, the host can get the request status with a DFU_GETSTATUS request (see “Status Handling” on page 16). Possible status are detailed in Table 3-2:

Table 3-2. SELECT_MEMORY_UNIT Status Values

Status	Value	Description
OK	0x00	Memory selected
errADDRESS	0x08	Unsupported memory

3.1.2 Selecting Memory Page

3.1.2.1 Command

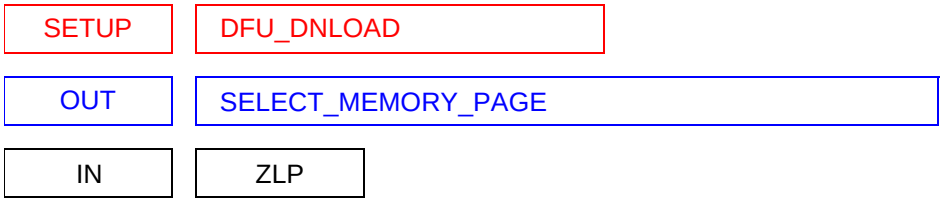
The SELECT_MEMORY_PAGE command is 5 bytes long as detailed in Table 3-3. The page address consists of two bytes (MSB first) that give a 16-bit page address.

Table 3-3. SELECT_MEMORY_PAGE Command Format

Command Identifier	data[0]	data[1]	data[2]	data[3]	data[4]	Comment
0x06	**CMD_GROUP_SELECT**					
	0x03	CMD_SELECT_MEMORY				
		0x01	MEMORY_PAGE			
			PP (MSB)	PP (LSB)		64 kB Memory page number

3.1.2.2 Request From Host

Figure 3-2. Select Memory Page Request



3.1.2.3 Request Status

After sending the SELECT_MEMORY_PAGE command, the host can get the request status with a DFU_GETSTATUS request (see “Status Handling” on page 16). Possible status are detailed in Table 3-4:

Table 3-4. SELECT_MEMORY_PAGE Status Values

Status	Value	Description
OK	0x00	Page selected
errADDRESS	0x08	Page value is out of range

3.2 Programming the Selected Memory

The memory data is downloaded via control-write transfers initiated by the DFU_DNLOAD class-specific request.

As described in the USB DFU Specification, “Data images for specific devices are, by definition, vendor specific. It is therefore required that target addresses, record sizes, and all other information relative to supporting an upgrade are encapsulated within the data image file. It is the responsibility of the device manufacturer and the firmware developer to ensure that their devices

can process these encapsulated data. With the exception of the DFU file suffix, the content of the data image file is irrelevant to the host.”

The data image is composed of:

- Command
- Data payload
- DFU Suffix on 16 Bytes. This suffix is reserved for future use.

3.2.1 Program Start Command

3.2.1.1 Command

The PROGRAM_START command is 6 bytes long as detailed in Table 3-5.

Table 3-5. PROGRAM_START Command Format

Command Identifier	data[0]	data[1]	data[2]	data[3]	data[4]	Comment
0x01	**CMD_GROUP_DOWNLOAD**					
	0x00	CMD_PROGRAM_START				
		start_address		end_address		Memory Address Range

3.2.1.2 Data Payload

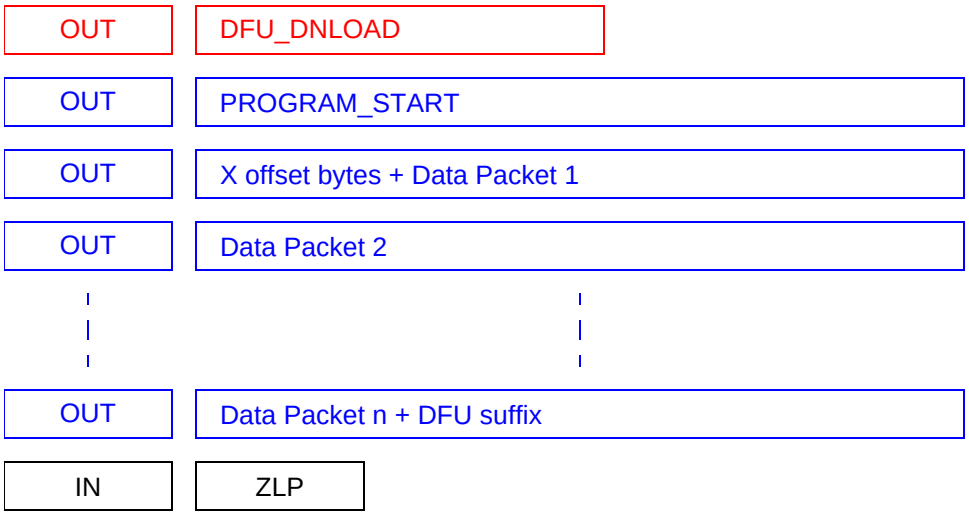
In order to be in accordance with the memory write entity (page size), X non-significant bytes may be added before the first byte to program. The X number is calculated to align the beginning of the firmware with the memory write entity.

3.2.1.3 DFU Suffix

The DFU suffix of 16 bytes is added just after the last byte to program.It is not used in the current version of the bootloader.

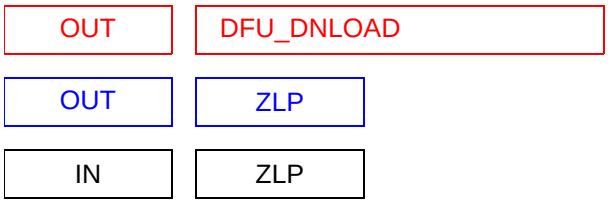
3.2.1.4 Request From Host

Figure 3-3. PROGRAM_START Request



After this request, the host sends a DFU_DNLOAD request with Zero Length Packet (ZLP) to indicate that it has completed transferring the data image file. This is the final payload packet of a download operation.

Figure 3-4. End of PROGRAM_START Request



3.2.1.5 *Request Status*

After sending the PROGRAM_START command, the host may get the request status with a DFU_GETSTATUS request (see "Status Handling" on page 16). Possible status are detailed in Table 3-6:

Table 3-6. PROGRAM_START Status Values

Status	Value	Description
OK	0x00	Programming done
errWRITE	0x03	Memory is not writable
errADDRESS	0x08	Memory address is out of range

3.3 Reading the Selected Memory

This group of commands allows to read the content as well as checking the blank state of the selected memory.

In case of reading, the memory data is uploaded via control-read transfers initiated by the DFU_UPLOAD class-specific request.

3.3.1 Read Memory

This operation is performed in 2 steps:

- DFU_DNLOAD request with the GROUP_UPLOAD command (6 bytes)
- DFU_UPLOAD request which correspond to the previous command.

3.3.1.1 *Command*

The READ_MEMORY command is 6 bytes long as detailed in Table 3-7.

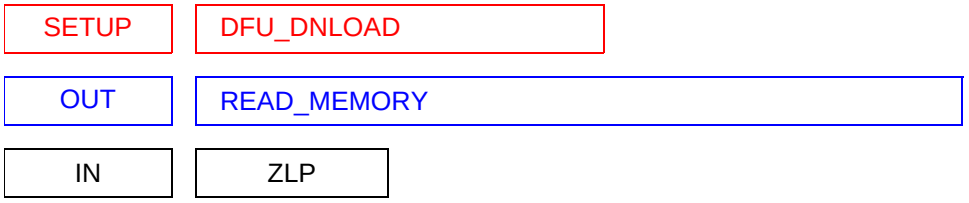
Table 3-7. READ_MEMORY Command Format

Command Identifier	data[0]	data[1]	data[2]	data[3]	data[4]	Description
0x03	**CMD_GROUP_UPLOAD**					
	0x00	CMD_READ_MEMORY				
		start_address		end_address		Memory Address Range

3.3.1.2 *First Request from Host*

The host sends a DFU Download request with a READ_MEMORY command in the data field.

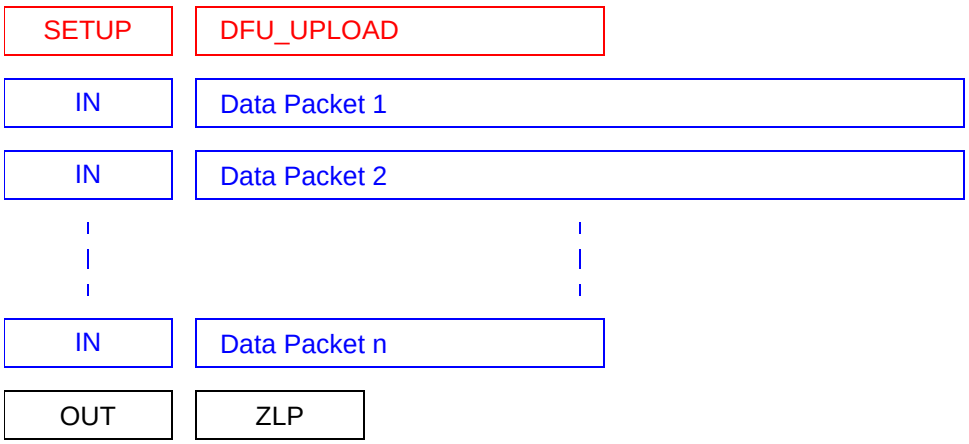
Figure 3-5. READ_MEMORY Request



3.3.1.3 *Second Request from Host*

The host sends a DFU Upload request. The device then sends the memory content from the specified start address to the specified end address of the selected memory page.

Figure 3-6. Data Payload Request



3.3.1.4 *Request Status*

After sending the READ_MEMORY command, the host may get the request status with a DFU_GETSTATUS request (see “Status Handling” on page 16). Possible status are detailed in Table 3-8:

Table 3-8. READ_MEMORY Status Values

Status	Value	Description
OK	0x00	Reading memory done
errVERIFY	0x07	Memory not readable (implemented for future memory extension)
errADDRESS	0x08	Memory address is out of range

3.3.2 Blank Check

3.3.2.1 *Command*

The BLANK_CHECK command is 6 bytes long as detailed in Table 3-9.

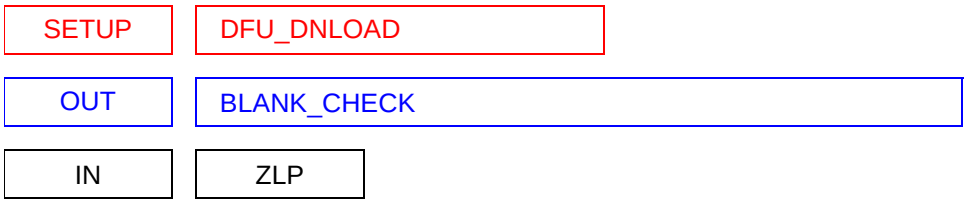
Table 3-9. BLANK_CHECK Command Format

Command Identifier	data[0]	data[1]	data[2]	data[3]	data[4]	Description
0x03	CMD_GROUP_UPLOAD					
	0x01	CMD_BLANK_CHECK				
		start_address		end_address		Memory Address Range

3.3.2.2 Request from Host

The host sends a DFU Download request with a BLANK_CHECK command in the data field.

Figure 3-7. BLANK_CHECK Request



3.3.2.3 Request Status

After sending the BLANK_CHECK command, the host controller needs to send a DFU_GETSTATUS to get the request status (see "Status Handling" on page 16). Once internal blank check has been completed, the device sends its status. Possible status are detailed in Table 3-10:

Table 3-10. BLANK_CHECK Status Values

Status	Value	Description
OK	0x00	Memory blanked
errCHECK_ERASED	0x05	Memory not blanked(1)
errADDRESS	0x08	Memory address is out of range

Note: 1. In case the device memory is not blank. The device waits for a DFU_UPLOAD request to send the first failed address.

3.4 Erasing the Flash

The Full Chip erase command erases the whole Flash Memory.

3.4.1 Chip Erase

3.4.1.1 Command

The ERASE command is 3 bytes long as detailed in Table 3-11.

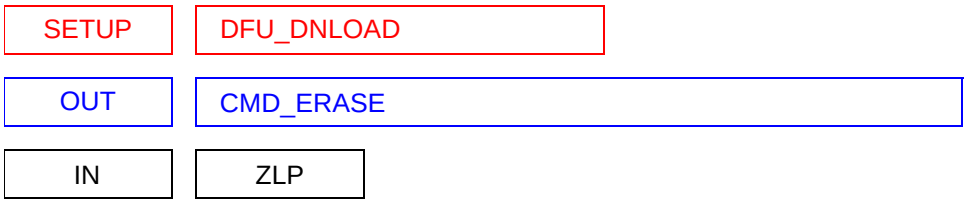
Table 3-11. ERASE Command Format

Command Identifier	data[0]	data[1]	data[2]	data[3]	data[4]	Description
0x04	CMD_GROUP_EXEC					
	0x00	CMD_ERASE				
		FFh				Full chip Erase

3.4.1.2 *Request from Host*

To start the erasing operation, the host sends a DFU_DNLOAD request with a CMD_ERASE command in the data field.

Figure 3-8. ERASE Request



3.4.1.3 *Request Status*

After sending the ERASE command, the host can get the request status with a DFU_GETSTATUS request (see "Status Handling" on page 16). Possible status are detailed in Table 3-12:

Table 3-12. ERASE Status Values

Status	Value	Description
OK	0x00	Erase Done

3.5 Starting the Application

The flow described below allows to start the application directly from the bootloader upon a specific command reception.

Two options are possible:

- Start the application with an internal hardware reset using watchdog.
- Start the application without reset.

3.5.1 Start Application with Reset

When the device receives this command the watchdog is enabled and the bootloader enters a waiting loop until the watchdog timer expires and resets the device.

3.5.1.1 *Command*

The START_APPLICATION_RESET command is 3 bytes long as detailed in Table 3-13.

Table 3-13. START_APPLI_RESET Command Format

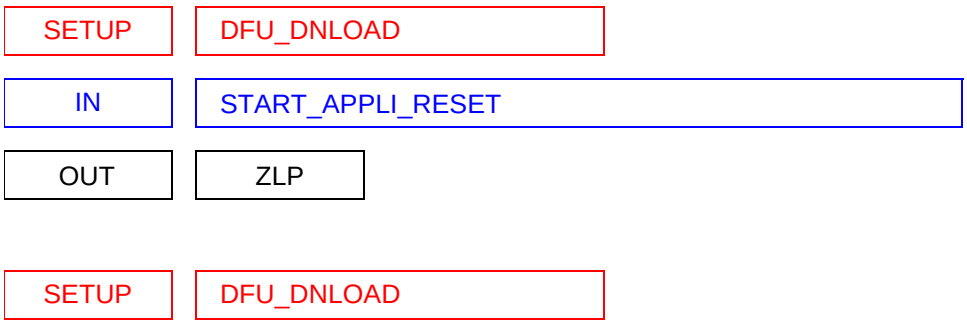
Command Identifier	data[0]	data[1]	data[2]	data[3]	data[4]	Description
0x04	CMD_GROUP_EXEC					
	0x03	CMD_START_APPLI				
		0x00	START_APPLI_RESET			
						Hardware reset

3.5.1.2 *Request From Host*

To start the application, the host sends a DFU_DNLOAD request with the START_APPLI_NO_RESET command.

This request is immediately followed by a second DFU_DNLOAD request with no data field to start the application.

Figure 3-9. START_APPLI_RESET Request



3.5.1.3 Request Status

No status is returned after the START_APPLI_RESET command.

3.5.2 Start Application without Reset

When the device receives this command a jump at first address of the Flash memory is used to start the application without reset.

3.5.2.1 Command

The START_APPLI_NO_RESET command is 3 bytes long as detailed in Table 3-14.

Table 3-14. START_APPLI_NO_RESET Command Format

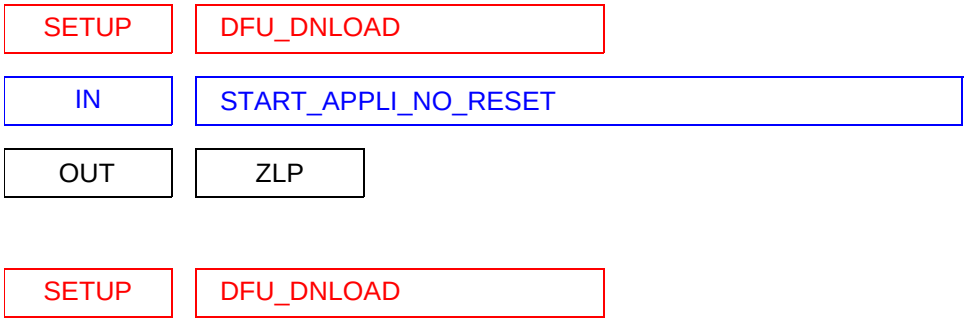
Command Identifier	data[0]	data[1]	data[2]	data[3]	data[4]	Description
0x04	**CMD_GROUP_EXEC**					
	0x03	CMD_START_APPLI				
		0x01	START_APPLI_NO_RESET			
						Software jump

3.5.2.2 Request From Host

To start the application, the host sends a DFU_DNLOAD request with the START_APPLI_NO_RESET command.

This request is immediately followed by a second DFU_DNLOAD request with no data field to start the application.

Figure 3-10. START_APPLI_NO_RESET Request



3.5.2.3 Request Status

No status is returned after the START_APPLI_NO_RESET command.

3.6 Status Handling

3.6.1 Get Status

The Host employs the DFU_GETSTATUS request (see “Device Status” on page 5) to facilitate synchronization with the device. The reported status gives information on the execution of the previous request. The possible status values are reported in the bStatus field of the request and are detailed in Table 3-15.

Table 3-15. Generic Error Values

Status	Value	Description
OK	0x00	No error condition is present
errWRITE	0x03	Device id unable to write memory
errCHECK_ERASED	0x05	Memory erase check failed
errVERIFY	0x07	Programmed memory failed verification
errADDRESS	0x08	Cannot program memory due to received address that is out of range
errSTALLEDPK	0x0F	Device stalled an unsupported or unexpected request

3.6.2 Clear Status

Each time the device detects and reports an error indication status to the host in response to a DFU_GETSTATUS request, it enters the dfuERROR state. After reporting any error status, the device can not leave the dfuERROR state, until it has received a DFU_CLRSTATUS request. Upon receipt of DFU_CLRSTATUS, the device sets status to OK and move to the dfuIDLE state.

4. Appendix-A

Table 4-1. Summary of DFU Bootloader Commands

Command Identifier	data[0]	data[1]	data[2]	data[3]	data[4]	Comment
0x01	**CMD_GROUP_DOWNLOAD**					
	0x00	CMD_PROGRAM_START				
		start_address		end_address		Memory Address Range
0x03	**CMD_GROUP_UPLOAD**					
	0x00	CMD_READ_MEMORY				
		start_address		end_address		Memory Address Range
	0x01	CMD_BLANK_CHECK				
		start_address		end_address		Memory Address Range
0x04	**CMD_GROUP_EXEC**					
	0x00	CMD_ERASE				
		FFh				Full chip Erase
	0x03	CMD_START_APPLI				
		0x00	START APPLI RESET			
						Hardware reset
		0x01	START APPLI NO RESET			
						Software jump
0x06	**CMD_GROUP_SELECT**					
	0x03	CMD_SELECT_MEMORY				
		0x00	MEMORY UNIT			
			0x00			Flash Memory
			0x01			Reserved for future use
			0x02			Security Memory
			0x03			Configuration Memory
			0x04			Bootloader Memory
			0x05			Signature Memory
			0x06			User Page Memory
		0x01	MEMORY PAGE			
			PP (MSB)	PP (LSB)		64 kB Memory page number

Headquarters

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

International

Atmel Asia
Unit 1-5 & 16, 19/F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
Hong Kong
Tel: (852) 2245-6100
Fax: (852) 2722-1369

Atmel Europe
Le Krebs
8, Rue Jean-Pierre Timbaud
BP 309
78054 Saint-Quentin-en-
Yvelines Cedex
France
Tel: (33) 1-30-60-70-00
Fax: (33) 1-30-60-71-11

Atmel 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

Product Contact

Web Site
www.atmel.com

Technical Support
avr32@atmel.com

Sales Contact
www.atmel.com/contacts

Literature Requests
www.atmel.com/literature