AVR079: STK600 Communication Protocol

Features

- **Supported Commands and Command options**
- **Command and Answer package formats**

1 Introduction

This document describes the STK®600 protocol. The firmware is distributed with AVR Studio® 4.14 or later.

The definition of all commands, responses, parameters and other defined values can be found in the file “command.h”. This file can be downloaded from the Atmel® web site.

All device specific values can be found in the xml files for each part. The xml files are distributed with AVR Studio. Download the latest AVR Studio 4 from the Atmel web site, http://www.atmel.com/products/AVR/. The xml file format is described in chapter 12.

8-bit AVR® Microcontrollers

Application Note

Rev. 8133A-AVR-04/08

2 Overview

2.1 USB Communication

The STK600 communicates with the PC through its USB interface. The USB interface utilizes two bulk endpoints; one IN and one OUT. The USB descriptors can be found in the Appendix.

2.2 Packet Format

The PC sends *commands* to the STK600, which responds with an *answer*. Each command will generate an answer.

Both commands and answers can be larger than the maximum packet size for the bulk endpoints, so a command or answer can be split into several IN/OUT packets. A *short packet* indicates the end of a command or answer.

2.3 USB Driver

In order to communicate with the STK600, a driver must be installed on the host computer. A driver can be written from scratch or by using a driver development kit.

AVR Studio 4 bundles a USB driver licensed from Jungo (www.jungo.com). By obtaining a license from Jungo, 3rd party software can access the same driver as AVR Studio. The user can then use both AVR Studio and other tools without changing drivers.

Note: Firmware upgrades for STK600 can only be uploaded with the dedicated upgrade software bundled with AVR Studio. This requires that the driver supplied with AVR Studio to be installed.

2.4 Command format

This section describes all commands that can be entered to the STK600, and all the possible responses that each command can give back to the host.

For all commands, the STK600 will return an answer with an answer ID that is equal to the command ID. The first byte in a command is always the command ID, the first byte in an answer is always the answer ID.

3 General Commands

These commands are not related to a specific programming mode.

3.1 CMD_SIGN_ON

This command returns a unique signature string for the STK600 with this implementation of the protocol.

Table 3-1. Command format.

Field	Size	Values	Description
Command ID	1 byte	CMD_SIGN_ON	Command id

Table 3-2. Answer format.

Field	Size	Values	Description
Answer ID	1 byte	CMD_SIGN_ON	Answer id
Status	1 byte	STATUS_CMD_OK	
Signature length	1 byte	6	Length of signature string
	8 bytes	“STK600”	The signature string (not null-terminated)

3.2 CMD_SET_PARAMETER

The host can set a multitude of parameters in the STK600. See the *11 Parameters* for a description of each parameter.

Table 3-3. Command format for one byte values.

Field	Size	Values	Description
Command ID	1 byte	CMD_SET_PARAMETER	Command id
Parameter ID	1 byte		Which parameter to set
Value	1 byte		Parameter new value

Table 3-4. Command format for two byte values.

Field	Size	Values	Description
Command ID	1 byte	CMD_SET_PARAMETER	Command id
Parameter ID	1 byte		Which parameter to set
Value	1 byte		Parameter new value high byte
Value	1 byte		Parameter new value low byte

Table 3-5. Command format for two byte values.

Field	Size	Values	Description
Answer ID	1 byte	CMD_SET_PARAMETER	Answer id
Status	1 byte	STATUS_CMD_OK or STATUS_CMD_FAILED	A status value indicating the result of the operation

3.3 CMD_GET_PARAMETER

The host can also read different parameters from the STK600.

Table 3-6. Command format.

Field	Size	Values	Description
Command ID	1 byte	CMD_GET_PARAMETER	Command id
Parameter ID	1 byte		Which parameter to get

Table 3-7. Answer format for one byte values if command succeeded.

Field	Size	Values	Description
Answer ID	1 byte	CMD_GET_PARAMETER	Answer id
Status	1 byte	STATUS_CMD_OK	A status value indicating success
Parameter value	1 byte		The parameter value

Table 3-8. Answer format for two byte values if command succeeded.

Field	Size	Values	Description
Answer ID	1 byte	CMD_GET_PARAMETER	Answer id
Status	1 byte	STATUS_CMD_OK	A status value indicating success
Parameter value	1 byte		The parameter value high byte
Parameter value	1 byte		The parameter value low byte

Table 3-9. Answer format if command fails.

Field	Size	Values	Description
Answer ID	1 byte	CMD_GET_PARAMETER	Answer id
Status	1 byte	STATUS_CMD_ FAILED	A status value indicating that the operation failed.

The only reason for the operation to fail is that an illegal parameter is requested.

3.4 CMD_OSCCAL

This command performs a calibration sequence as described in application note AVR053

Table 3-10. Command format.

Field	Size	Values	Description
Command ID	1 byte	CMD_OSCCAL	Command id

Table 3-11. Answer format if command succeeded.

Field	Size	Values	Description
Answer ID	1 byte	CMD_OSCCAL	Answer id
Status	1 byte	STATUS_CMD_OK or STATUS_CMD_FAILED	A status value indicating success or failure.

3.5 CMD_LOAD_ADDRESS

This command will load an address into the STK600. The next Program Flash, Read Flash, Program EEPROM or Read EEPROM command will operate from the address set with this command. The command is used in conjunction with high voltage parallel programming, high voltage serial programming and low voltage serial programming. All the abovementioned commands will increment an internal address counter, so this command needs only to be sent once.

Table 3-12. Command format.

Field	Size	Values	Description
Command ID	1 byte	CMD_LOAD_ADDRESS	Command id
Address	4 byte		The address, four bytes, MSB first

For word-addressed memories (program flash), the Address parameter is the word address.

If bit 31 is set, this indicates that the following read/write operation will be performed on a memory that is larger than 64KBytes. This is an indication to STK600 that a *load extended address* must be executed. See datasheet for devices with memories larger than 64KBytes.

Table 3-13. Answer format if command succeeded.

Field	Size	Values	Description
Answer ID	1 byte	CMD_LOAD_ADDRESS	Answer id
Status	1 byte	STATUS_CMD_OK	A status value indicating success

3.6 CMD_FIRMWARE_UPGRADE

When the host is trying to connect to the programmer, it checks the firmware version. A firmware upgrade is initiated if a newer version is available on the PC.

The STK600 can “reboot” into upgrade mode by using this command

Table 3-14. Command format.

Field	Size	Values	Description
Command ID	1 byte	CMD_FIRMWARE_UPGRADE	Command id
Parameter ID	10 bytes	”fwupgrade”	String to enable upgrade mode

Table 3-15. Answer format.

Field	Size	Values	Description
Answer ID	1 byte	CMD_SET_PARAMETER	Answer id
Status	1 byte	STATUS_CMD_OK or STATUS_CMD_FAILED	A status value indicating the result of the operation

If the status returned is STATUS_CMD_OK, the STK600 will disconnect and enter upgrade mode.

3.7 CMD_LOAD_RC_ID_TABLE

This command sends the Routing Card boardID table to the STK600 FW. Each row in this table holds first the RC_id, then the SC_id (allowed to use with that RC) and then the MAX_VTG allowed for that routingcard.

The two first bytes of the table hold the revision of the table. MSB first.

Table 3-16. Command format.

Field	Size	Values	Description
Command ID	1 byte	CMD_LOAD_RC_ID_TABLE	Command id
NumBytes			Total number of bytes to send (including the two revision bytes), MSB first
RC_ID_table	1 byte		MSB rev byte
...	1 byte		LSB rev byte
...	1 byte		ID of a Routing Card
...	1 byte		ID of SC allowed to use with the RC
...	1 byte		Max VTG for that RC

The three last lines above will be repeated until the whole table is sent.

Table 3-17. Answer format.

Field	Size	Values	Description
Answer ID	1 byte	CMD_LOAD_RC_ID_TABLE	Answer id
Status	1 byte	STATUS_CMD_OK	

3.8 CMD_LOAD_EC_ID_TABLE

This command sends the Expansion Card boardID table to the STK600 FW. Each row in this table holds first the EC_id, and then the MAX_VTG allowed for that EC.

The two first bytes of the table hold the revision of the table. MSB first.

Table 3-18. Command format.

Field	Size	Values	Description
Command ID	1 byte	CMD_LOAD_EC_ID_TABLE	Command id
NumBytes			Total number of bytes to send (including the two revision bytes), MSB first
EC_ID_table	1 byte		rev byte MSB
...	1 byte		rev byte LSB
...	1 byte		ID of a Expansion Card
...	1 byte		Max VTG for that EC

The two last lines above will be repeated until the whole table is sent.

Table 3-19. Answer format.

Field	Size	Values	Description
Answer ID	1 byte	CMD_LOAD_EC_ID_TABLE	Answer id
Status	1 byte	STATUS_CMD_OK	

3.9 CMD_CHECK_TARGET_CONNECTION

Table 3-20. Command format.

Field	Size	Values	Description
Command ID	1 byte	CMD_CHECK_TARGET_CONNECTION	Command id

Table 3-21. Answer format.

Field	Size	Values	Description
Answer ID	1 byte	CMD_LOAD_EC_ID_TABLE	Answer id
Answer value	1 byte	Se table below.	Status of target connection
Status	1 byte	STATUS_CMD_OK	

This command runs a target connection check and returns the status of the target connection.

The value is 6bits, where bit 0-2 indicates the status of the short circuit protection system and bit 4-5 indicates if the ISP is connected correctly to the target system.

Bit 4-5 will only indicate correct target connection if the target is powered.

Bit 3 is “Don’t care”.

Table 3-22. bit descriptions.

Bit #	Status
0	STATUS_CONN_FAIL_MOSI
1	STATUS_CONN_FAIL_RST
2	STATUS_CONN_FAIL_SCK
4	STATUS_ISP_READY
5	STATUS_TGT_REVERSE_INSERTED

The corresponding bit will be set ‘1’ to indicate an error.

That is, if a line is short circuited, if target is not detected or the plug is inserted with a reverse orientation.

If the value 0x00 is returned it means the connection is ok.

The parameter should be checked before starting a programming sequence to check if target connection is correct (bit 4-5).

It should also be checked after a programming sequence if the command failed, to check if the operation failed because of a short circuit.

A short circuit can only be detected after the command Enter Progmode has been issued, because the control circuits of the STK600 is isolated via switches when the STK600 is in idle mode.

4 ISP Programming Commands

These commands handles FLASH, EEPROM, fuse bytes, lock bits, signature and oscillator calibration programming in ISP mode.

4.1 CMD_ENTER_PROGMODE_ISP

This command will make the target device enter programming mode.

XML path: /AVRPART/ICE_SETTINGS/STK600/IspEnterProgMode/

Table 4-1. Command format.

Field	Size	Values	Description
Command ID	1 byte	CMD_ENTER_ PROGMODE_ISP	Command id
timeout	1 byte	XML: timeout	Command time-out (in ms)
stabDelay	1 byte	XML: stabDelay	Delay (in ms) used for pin stabilization
cmdexeDelay	1 byte	XML: cmdexeDelay	Delay (in ms) in connection with the EnterProgMode command execution
synchLoops	1 byte	XML: synchLoops	Number of synchronization loops
byteDelay	1 byte	XML: byteDelay	Delay (in ms) between each byte in the EnterProgMode command.
pollValue (1)	1 byte	XML: pollValue	Poll value: 0x53 for AVR®
pollIndex	1 byte	XML: pollIndex	Start address, received byte: 0 = no polling, 3 = AVR
cmd1(2)	1 byte		Command Byte # 1 to be transmitted
cmd2(2)	1 byte		Command Byte # 2 to be transmitted
cmd3(2)	1 byte		Command Byte # 3 to be transmitted
cmd4(2)	1 byte		Command Byte # 4 to be transmitted

Notes:
1. The pollValue parameter indicates after which of the transmitted bytes on the SPI interface to store the return byte, as the SPI interface is implemented as a ring buffer (one byte out, one byte in)
2. cmd1, cmd2, cmd3 and cmd4 are the instruction bytes found in the SPI Serial Programming Instruction Set found in the device datasheet.

Table 4-2. Answer format.

Field	Size	Values	Description
Answer ID	1 byte	CMD_SET_PARAMETER	Answer id
Status	1 byte	STATUS_CMD_TOUT, STATUS_CMD_OK or STATUS_CMD_FAILED	A status value indicating the result of the operation

4.2 CMD_LEAVE_PROGMODE_ISP

This command will make STK600 leave programming mode. The device will be put into normal operating mode.

XML path: /AVRPART/ICE_SETTINGS/STK600_2/IspLeaveProgMode/

Table 4-3. Command format.

Field	Size	Values	Description
Command ID	1 byte	CMD_LEAVE_PROGMODE_ISP	Command id
preDelay	1 byte	XML: preDelay	Pre-delay (in ms)
PostDelay	1 byte	XML: postDelay	Post-delay (in ms)

Table 4-4. Answer format.

Field	Size	Values	Description
Answer ID	1 byte	CMD_LEAVE_PROGMODE_ISP	Answer id
Status	1 byte	STATUS_CMD_OK	This command will always return STATUS_CMD_OK

4.3 CMD_CHIP_ERASE_ISP

This command will perform a chip erase on the target device.

XML path: /AVRPART/ICE_SETTINGS/STK600/IspChipErase/

Table 4-5. Command format.

Field	Size	Values	Description
Command ID	1 byte	CMD_CHIP_ERASE_ISP	Command id
eraseDelay	1 byte	XML: eraseDelay	Delay (in ms) to ensure that the erase of the device is finished
pollMethod	1 byte	XML: pollMethod	Poll method, 0 = use delay 1= use RDY/BSY command
cmd1(1)	1 byte		Chip erase command byte #1
cmd2(1)	1 byte		Chip erase command byte #2
cmd3(1)	1 byte		Chip erase command byte #3
cmd4(1)	1 byte		Chip erase command byte #4

Notes: 1. cmd1, cmd2, cmd3 and cmd4 are the instruction bytes found in the SPI Serial Programming Instruction Set found in the device datasheet.

Table 4-6. Answer format.

Field	Size	Values	Description
Answer ID	1 byte	CMD_CHIP_ERASE_ISP	Answer id
Status	1 byte	STATUS_CMD_OK or STATUS_CMD_TOUT	A status value indicating the result of the operation

4.4 CMD_PROGRAM_FLASH_ISP

This command will program data into the FLASH memory of the target device if it succeeds.

XML path: /AVRPART/ICE_SETTINGS/STK600/IspProgramFlash/

Table 4-7. Command format.

Field	Size	Values	Description
Command ID	1 byte	CMD_PROGRAM_FLASH_ISP	Command id
NumBytes	2 byte		Total number of bytes to program, MSB first
mode	1 byte	XML: mode *	Mode byte*
delay	1 byte	XML: delay	Delay, used for different types of programming termination, according to mode byte
cmd1	1 byte		Command 1 (Load Page, Write Program Memory)
cmd2	1 byte		Command 2 (Write Program Memory Page)
cmd3	1 byte		Command 3 (Read Program Memory)
poll1	1 byte	XML: pollVal1	Poll Value #1
poll2	1 byte	XML: pollVal2	Poll Value #2 (not used for flash programming)
Data	N bytes		N data

Mode byte

The *mode* parameter is essential for how this command works. The bits in the mode byte have the following meanings:

Table 4-8. Mode byte, bit descriptions.

Bit #	Description	Mode
0	Word/Page Mode (0 = word, 1 = page)	
1	Timed delay	Word Mode
2	Value polling	
3	RDY/BSY polling	
4	Timed delay	Page Mode
5	Value polling	
6	RDY/BSY polling	
7	Write page	

The *Word/Page Mode* bit selects if the device supports page programming or not.

The command bytes are different for word and page mode. In word mode, the ISP commands *Write Program Memory* and *Read Program Memory* are used. In page mode, *Load Page*, *Write Program Memory Page* and *Read Program Memory* are used. The read instruction is used if *Value Polling* is specified in the mode bit. The Low/High byte selection bit (3rd bit in the Load Page, Write Program Memory commands) is handled by STK600, so leave this bit cleared. The instruction values are found in the SPI Serial Programming Instruction Set found in the device datasheet.

According to the mode, different termination methods are selected – *Timed delay*, *Value polling* or *RDY/BSY polling*.

For paged operation, the *Write page* bit decides if a *Write Program Memory Page* command should be issued after the data has been loaded into the page buffer. For

devices with page size bigger than what can be transferred to STK600 in one command, several CMD_PROGRAM_FLASH_ISP commands must be issued. In such a case, only the last command should have the *Write Page* mode bit set.

NOTE: Only bit 0-6 are set in the XML file, because bit 7 is not constant and must be controlled by the PC software.

When *value polling* is used to determine when a programming operation is complete, *poll1* must be supplied. This value indicates which value will be read from the device until the programmed value is read. This indicates end of programming. *poll2* is used only for EEPROM programming.

Table 4-9. Answer format.

Field	Size	Values	Description
Answer ID	1 byte	CMD_PROGRAM_FLASH_ISP	Answer id
Status	1 byte	STATUS_CMD_OK, STATUS_CMD_TOUT, STATUS_RDY_BSY_TOUT	A status value indicating the result of the operation

4.5 CMD_READ_FLASH_ISP

This command will read data from the FLASH memory of the target device if it succeeds.

XML path: /AVRPART/ICE_SETTINGS/ STK600/IspReadFlash/

Table 4-10. Command format.

Field	Size	Values	Description
Command ID	1 byte	CMD_READ_FLASH_ISP	Command id
NumBytes	2 byte	XML: blockSize	Total number of bytes to read, MSB first
cmd1	1 byte		Read Program Memory command byte #1. Low/High byte selection bit (3rd bit) is handled in the FIRMWARE.

Table 4-11. Answer format if the command was executed.

Field	Size	Values	Description
Answer ID	1 byte	CMD_PROGRAM_FLASH_ISP	Answer id
Status1	1 byte	STATUS_CMD_OK	Indicates success. Will always read OK
Data	N bytes		The data read from the device
Status2	1 byte	STATUS_CMD_OK	A status value indicating the result of the operation. Will always read OK

Table 4-12. Answer format if the command was executed.

Field	Size	Values	Description
Answer ID	1 byte	CMD_PROGRAM_FLASH_ISP	Answer id
Status	1 byte	STATUS_CMD_FAILED	Indicates failure

4.6 CMD_PROGRAM_EEPROM_ISP

See the CMD_PROGRAM_FLASH_ISP command.

XML path: /AVRPART/ICE_SETTINGS/STK600/IspProgramEeprom/

4.7 CMD_READ_EEPROM_ISP

See the CMD_READ_FLASH_ISP command.

XML path: /AVRPART/ICE_SETTINGS/STK600/IspReadEeprom/

4.8 CMD_PROGRAM_FUSE_ISP

This command programs the fuses of the target device.

Table 4-13. Command format.

Field	Size	Values	Description
Command ID	1 byte	CMD_PROGRAM_FUSE_ISP	Command id
cmd1[1]	1 byte		Command Byte #1
cmd2[1]	1 byte		Command Byte #2
cmd3[1]	1 byte		Command Byte #3
cmd4[1]	1 byte		Command Byte #4

Notes: 1. cmd1, cmd2, cmd3 and cmd4 are the instruction bytes found in the SPI Serial Programming Instruction Set found in the device datasheet.

Table 4-14. Answer format.

Field	Size	Values	Description
Answer ID	1 byte	CMD_PROGRAM_FUSE_ISP	Answer id
Status1	1 byte	STATUS_CMD_OK	Will always read OK
Status2	1 byte	STATUS_CMD_OK	Will always read OK

4.9 CMD_READ_FUSE_ISP

This command reads the fuses of the target device.

XML path: /AVRPART/ICE_SETTINGS/STK600/IspReadFuse/

Table 4-15. Command format.

Field	Size	Values	Description
Command ID	1 byte	CMD_READ_FUSE_ISP	Command id
RetAddr[1]	1 byte	XML: pollIndex	Return address
cmd1[2]	1 byte		Command Byte #1
cmd2[2]	1 byte		Command Byte #2
cmd3[2]	1 byte		Command Byte #3
cmd4[2]	1 byte		Command Byte #4

Field	Size	Values	Description

Notes:
1. RetAddr indicates after which of the transmitted bytes on the SPI interface to store the return byte, as the SPI interface is implemented as a ring buffer (one byte out, one byte in).
2. cmd1, cmd2, cmd3 and cmd4 are the instruction bytes found in the SPI Serial Programming Instruction Set found in the device datasheet.

Table 4-16. Answer format.

Field	Size	Values	Description
Answer ID	1 byte	CMD_READ_FUSE_ISP	Answer id
Status1	1 byte	STATUS_CMD_OK	A status value indicating the result of the operation, always OK
Data	1 byte		The fuse byte read from the device
Status2	1 byte	STATUS_CMD_OK	A status value indicating the result of the operation, always OK

4.10 CMD_PROGRAM_LOCK_ISP

See CMD_PROGRAM_FUSE. This command is basically the same as the program fuse command, only that ISP commands for programming the lock byte must be supplied.

4.11 CMD_READ_LOCK_ISP

See CMD_READ_FUSE. This command is basically the same as the read fuse command, only that ISP commands for reading the lock byte must be supplied.

XML path: /AVRPART/ICE_SETTINGS/STK600/IspReadLock/

4.12 CMD_READ_SIGNATURE_ISP

See CMD_READ_FUSE. This command is basically the same as the read fuse command, only that ISP commands for reading one of the signature bytes must be supplied.

XML path: /AVRPART/ICE_SETTINGS/STK600/IspReadSign/

4.13 CMD_READ_OSCCAL_ISP

See CMD_READ_FUSE. This command is basically the same as the read fuse command, only that ISP commands for reading the OSCCAL byte must be supplied.

XML path: /AVRPART/ICE_SETTINGS/STK600/IspReadOsccal/

4.14 CMD_SPI_MULTI

This is a generic command that can be used to execute any of the ISP commands. The command writes a number of bytes to the SPI bus, and returns a number of bytes.

Table 4-17. Command format.

Field	Size	Values	Description
Command ID	1 byte	CMD_SPI_MULTI	Command ID

Field	Size	Values	Description
NumTx	1 byte	0-255	Number of bytes to transmit
NumRx	1 byte	0-255	Number of bytes to receive
RxStartAddr	1 byte		Start address of returned data. Specifies on what transmitted byte the response is to be stored and returned.
TxData	0-255 bytes		The data to be transmitted. The size is specified by NumTx

If the number of bytes to receive is greater than number of bytes to transmit, then the firmware will pad with the necessary 0x00 bytes. This is in order to save time-consuming transfer from PC to the programmer.

Table 4-18. Answer format.

Field	Size	Values	Description
Answer ID	1 byte	CMD_SPI_MULTI	Answer id
Status1	1 byte	STATUS_CMD_OK	Will always read OK
data	0-255 bytes		The data read from the ISP bus as indicated in the command
Status2	1 byte	STATUS_CMD_OK	Will always read OK

5 Parallel Programming Mode Commands

5.1 CMD_ENTER_PROGMODE_PP

This command will make the target device enter programming mode if it succeeds.

XML PATH: /AVRPART/ICE_SETTINGS/STK600/PpEnterProgMode/

Table 5-1. Command format.

Field	Size	Values	Description
Command ID	1 byte	CMD_ENTER_PROGMODE_PP	Command id
stabDelay	1 byte	XML: stabDelay	Delay (in ms) used for pin stabilization
progModeDelay	1 byte	XML: progModeDelay	Delay (in ms) in connection with the EnterProgMode command execution
latchCycles	1 byte	XML: latchCycles	Number of xtal cycles used to latch OSCCAL
toggleVtg	1 byte	XML: toggleVtg	Toggle Vtg when entering prog.mode (0=no, 1=yes). For parts with RSTDSBL functionality
powerOffDelay	1 byte	XML: powerOffDelay	Power-off delay. Additional delay (in ms) after Vtg is turned off in order to make sure the Vtg is low enough
resetDelayMs	1 byte	XML: resetDelayMs	RSTDELAY #1 (in ms) Additional delay between Vtg is turned on and reset goes high.
resetDelayUs	1 byte	XML: resetDelayUs	RSTDELAY #2(in us x 10) Additional delay between Vtg is turned on and reset goes high. Total delay is RSTDELAY #1 (ms) + RSTDELAY #2 (us x 10)

Table 5-2. Answer format (same for all results).

Field	Size	Values	Description
Answer ID	1 byte	CMD_ENTER_PROGMODE_PP	Answer id
Status	1 byte	See table below	A *Result Value* indicating the result of the operation

Table 5-3. Valid Result Values for the answer to this command.

Value	Description
STATUS_CMD_OK	Operation succeeded
STATUS_CMD_ FAILED	Operation failed

5.2 CMD_LEAVE_PROGMODE_PP

This command will make the target device leave programming mode if it succeeds.

XML PATH: /AVRPART/ICE_SETTINGS/STK600/PpLeaveProgMode/

Table 5-4. Command format.

Field	Size	Values	Description
Command ID	1 byte	CMD_LEAVE_PROGMODE_PP	Command id
StabDelay	1 byte	XML: stabDelay	Delay (in ms) used for pin stabilization
ResetDelay	1 byte	XML: resetDelay	Delay (ms) for holding RESET low

Table 5-5. Answer format.

Field	Size	Values	Description
Answer ID	1 byte	CMD_LEAVE_PROGMODE_PP	Answer id
Status	1 byte	STATUS_CMD_OK	A *Result Value* indicating the result of the operation

5.3 CMD_CHIP_ERASE_PP

This command will perform a chip erase on the target device if it succeeds.

XML PATH: /AVRPART/ICE_SETTINGS/STK600/PpChipErase/

Table 5-6. Command format.

Field	Size	Values	Description
Command ID	1 byte	CMD_CHIP_ERASE_PP	Command id
pulseWidth	1 byte	XML: pulseWidth	Width (in ms) of the /WR pulse. 0 => 1 cycle
pollTimeout	1 byte	XML: pollTimeout	Timeout period (in ms) to wait for RDY/BSY flag to rise. If 0, RDY/BSY flag is NOT used.

Table 5-7. Answer format (same for all results).

Field	Size	Values	Description

Field	Size	Values	Description
Answer ID	1 byte	CMD_CHIP_ERASE_PP	Answer id
Status	1 byte	See table below	A *Result Value* indicating the result of the operation

Table 5-8. Valid Result Values for the answer to this command.

Value	Description
STATUS_CMD_OK	Operation succeeded
STATUS_RDY_BSY_TOUT	No response from target device within specified timeframe

5.4 CMD_PROGRAM_FLASH_PP

This command will program data into the FLASH memory of the target device if it succeeds. For devices with the FLASH organized in pages, the data address and size used with this command must confirm to that of the device. I.e. one ProgramFlash command is used to program one page in the target device.

XML PATH: /AVRPART/ICE_SETTINGS/STK600/PpProgramFlash/

Table 5-9. Command format.

Field	Size	Values	Description
Command ID	1 byte	CMD_PROGRAM_FLASH_PP	Command id
Nmb bytes (MSB	1 byte		Total number of bytes to program (MSB)
Nmb bytes (LSB)	1 byte		Total number of bytes to program (LSB)
Mode(1)	1 byte	XML: mode, *see description below	Mode byte, *see description below
pollTimeout	1 byte	XML: pollTimeout	pollTimeout (in ms)
Data	1 byte	Data 1	
Data	1 byte	...	
Data	1 byte	Data N	

Notes: 1. See details in list below.

Mode byte description

- Bit 0: This bit indicates whether to use byte ‘0’ or page ‘1’ programming.
- Bit 1-3 are the pagesize bits, pagesize are given in bytes not words, see table below.
- Bit 4-5 are not in use.
- Bit 6 must be set to ‘1’ when it is the very last page to be programmed, otherwise ‘0’.
- Bit 7 indicates if a page write should be issued (*Transfer data to flash*). Normally it should always be set ‘1’. However, if the page size of the target device is too large to be covered by one Program Flash command (because the amount of available SRAM in STK600 is limited) this can be used to let 2 or more commands fill the page buffer of the target device. The *transfer data to flash* flag should then only be set on the last command.

Table 5-10. Mode byte description.

Pagesize \ bit	3	2	1
256	0	0	0
2	0	0	1
4	0	1	0
8	0	1	1
16	1	0	0
32	1	0	1
64	1	1	0
128	1	1	1

NOTE: that only Bit 0-3 are set in the XML file, cause Bit 6-7 are not static and must be controlled by the PC Frontend.

Table 5-11. Answer format /(same for all results).

Field	Size	Values	Description
Answer ID	1 byte	CMD_PROGRAM_FLASH_PP	Answer id
Status	1 byte	See table below	A *Result Value* indicating the result of the operation

Table 5-12. Valid Result Values for the answer to this command.

Value	Description
STATUS_CMD_OK	Operation succeeded
STATUS_RDY_BSY_TOUT	No response from target device within specified timeframe

5.5 CMD_READ_FLASH_PP

This command will read data from the FLASH memory of the target device if it succeeds. For devices with the FLASH organized in pages, the data address and size used with this command must confirm to that of the device. I.e. one ReadFlash command is used to read one page in the target device.

XML PATH: /AVRPART/ICE_SETTINGS/STK600/PpReadFlash/

Table 5-13. Command format.

Field	Size	Values	Description
Command ID	1 byte	CMD_READ_FLASH_PP	Command id
Nmb bytes (MSB)	1 byte		Total number of bytes to read (MSB)
Nmb bytes (LSB)	1 byte		Total number of bytes to read (LSB)

Table 5-14. Answer format if the command is executed.

Field	Size	Values	Description
Answer ID	1 byte	CMD_READ_FLASH_PP	Answer id

Field	Size	Values	Description
Status	1 byte	STATUS_CMD_OK	
Data	Nmb bytes		Data read from device. Will be padded with 0's if errors occurred during read-out.
Status	1 byte	STATUS_CMD_OK	A *Result Value* indicating the result of the operation

5.6 CMD_PROGRAM_EEPROM_PP

This command programs one page the EEPROM memory of the target device if it succeeds.

XML PATH: /AVRPART/ICE_SETTINGS/STK600/PpReadEeprom/

Command format: See CMD_PROGRAM_FLASH_PP.

5.7 CMD_READ_EEPROM_PP

This command will read data from the EEPROM memory of the target device if it succeeds.

XML PATH: /AVRPART/ICE_SETTINGS/STK600/PpReadEeprom/

Command format: See CMD_READ_FLASH_PP

5.8 CMD_PROGRAM_FUSE_PP

XML PATH: /AVRPART/ICE_SETTINGS/STK600/PpProgramFuse/

Table 5-15. Command format.

Field	Size	Values	Description
Command ID	1 byte	CMD_PROGRAM_FUSE_PP	Command id
Address	1 byte		address of fuse byte to program (low, high, ext, ext2)
Data	1 byte		fuse byte value to be programmed
pulseWidth	1 byte	XML: pulseWidth	width (ms) of /WR pulse (0 => 1 cycle)
pollTimeout	1 byte	XML: pollTimeout	Time-out (ms) for polling RDY/BSY (0 = don't poll)

Table 5-16. Answer format.

Field	Size	Values	Description
Answer ID	1 byte	CMD_PROGRAM_FUSE_PP	Answer id
Status	1 byte	See table below	A *Result Value* indicating the result of the operation

Table 5-17. Valid Result Values for the answer to this command.

Value	Description
STATUS_CMD_OK	Operation succeeded

Value	Description
STATUS_RDY_BSY_TOUT	No response from target device within specified timeframe

5.9 CMD_READ_FUSE_PP

XML PATH: /AVRPART/ICE_SETTINGS/STK600/PpReadFuse/

Table 5-18. Command format.

Field	Size	Values	Description
Command ID	1 byte	CMD_READ_FUSE_PP	Command id
	1 byte		address of fuse byte to read

Table 5-19. Answer format if command was executed.

Field	Size	Values	Description
Answer ID	1 byte	CMD_ READ_FUSE	Answer id
Status	1 byte	STATUS_CMD_OK	
Data	1 byte		Data read from device. Contains the read low, high or ext fuse byte. Will be padded with 0's if errors occurred during read-out.

5.10 CMD_PROGRAM_LOCK_PP

See CMD_PROGRAM_FUSE.

Note: Address must be sent but is ignored by firmware.

XML PATH: /AVRPART/ICE_SETTINGS/STK600/PpProgramLock/

5.11 CMD_READ_LOCK_PP

See CMD_READ_FUSE_PP.

Note: Address field must be sent but is ignored by firmware.

XML PATH: /AVRPART/ICE_SETTINGS/STK600/PpReadLock/

5.12 CMD_READ_SIGNATURE_PP

See CMD_READ_FUSE.

XML PATH: /AVRPART/ICE_SETTINGS/STK600/PpReadSign/

5.13 CMD_READ_OSCCAL_PP

See CMD_READ_FUSE.

XML PATH: /AVRPART/ICE_SETTINGS/STK600/PpReadOsccal/

5.14 CMD_SET_CONTROL_STACK

This command uploads the control stack to the STK. This is used for both PP and HVSP.

Note: The Control stack must always be uploaded before performing any programming commands in high voltage mode if the STK600 has been powered down.

To check if the controller has a valid control stack: Read PARAM_CONTROLLER_INIT.

Se chapter 11.7: PARAM_CONTROLLER_INIT

Table 5-20. Command format.

Field	Size	Values	Description
Command ID	1 byte	CMD_SET_CONTROL_STACK	Command id
Data	32byte		Control stack Data

Table 5-21. Answer format if command was executed.

Field	Size	Values	Description
Answer ID	1 byte	CMD_SET_CONTROL_STACK	Answer id
Status	1 byte	STATUS_CMD_OK	A *Result Value* indicating the result of the operation

6 High Voltage Serial Programming Commands

This chapter describes the High Voltage Serial Programming (HVSP) commands. Note that the SetControlStack command is required for HVSP as for PP. Description of the SetControlStack is found in chapter 5.14 CMD_SET_CONTROL_STACK

6.1 CMD_ENTER_PROGMODE_HVSP

This command will make the target device enter programming mode if it succeeds.

XML path: /AVRPART/ICE_SETTINGS/STK600/HvspEnterProgMode/

Table 6-1. Command format.

Field	Size	Values	Description
Command ID	1 byte	CMD_ENTER_PROGMODE_HVSP	Command id
StabDelay	1 byte	XML: stabDelay	Delay (in ms) used for pin stabilization
CmdexeDelay	1 byte	XML: cmdexeDelay	Delay (in ms) in connection with the EnterProgMode command execution
SynchCycles	1 byte	XML: synchCycles	Number of synchronization clock cycles
LatchCycles	1 byte	XML: latchCycles	Number of PulseXtal1_HVSP cycles
ToggleVtg	1 byte	XML: toggleVtg	Toggle Vtg when entering prog.mode (0=no, 1=yes). For parts with RSTDSBL functionality

Field	Size	Values	Description
PowoffDelay	1 byte	XML: powoffDelay	Power-off delay. Additional delay (in ms) after Vtg is turned off in order to make sure the Vtg is low enough
resetDelay1	1 byte	XML: resetDelay1	RSTDELAY #1 (in ms) Additional delay between Vtg is turned on and reset goes high.
resetDelay2	1 byte	XML: resetDelay2	RSTDELAY #2(in us x 10) Additional delay between Vtg is turned on and reset goes high. Total delay is RSTDELAY #1 (ms) + RSTDELAY #2 (us x 10)

Table 6-2. Answer format.

Field	Size	Values	Description
Answer ID	1 byte	CMD_ENTER_PROGMODE_HVSP	Answer id
Status	1 byte	STATUS_CMD_OK or STATUS_CMD_FAILED	A *Result Value* indicating the result of the operation. If failed the target voltage is >5.5V or <4.5V

6.2 CMD_LEAVE_PROGMODE_HVSP

This command will make the target device leave programming mode if it succeeds.

XML path: /AVRPART/ICE_SETTINGS/STK600/HvspLeaveProgMode/

Table 6-3. Command format.

Field	Size	Values	Description
Command ID	1 byte	CMD_LEAVE_PROGMODE_HVSP	Command id
stabDelay	1 byte	XML: stabDelay	Delay (in ms) used for pin stabilization
resetDelay	1 byte	XML: resetDelay	Delay (in ms) in connection with the LeaveProgMode command execution and Reset_low_duration

Table 6-4. Answer format.

Field	Size	Values	Description
Answer ID	1 byte	CMD_LEAVE_PROGMODE_HVSP	Answer id
Status	1 byte	STATUS_CMD_OK	A *Result Value* indicating the result of the operation. Will always read OK

6.3 CMD_CHIP_ERASE_HVSP

This command will perform a chip erase on the target device if it succeeds.

XML path: /AVRPART/ICE_SETTINGS/STK600/HvspChipErase/

Table 6-5. Command format.

Field	Size	Values	Description
Command ID	1 byte	CMD_CHIP_ERASE_HVSP	Command id
pollTimeout	1 byte	XML: pollTimeout	Timeout period (in ms) to wait for RDY/BSY flag to rise. If 0, RDY/BSY flag is NOT used.
eraseTime	1 byte	XML: eraseTime	Delay (in ms) to ensure that the erase of the device is finished. If 0, polling will be used.

Table 6-6. Answer format.

Field	Size	Values	Description
Answer ID	1 byte	CMD_CHIP_ERASE_HVSP	Answer id
Status	1 byte	STATUS_CMD_OK or STATUS_RDY_BSY_TOUT	A *Result Value* indicating the result of the operation.

6.4 CMD_PROGRAM_FLASH_HVSP

This command will program data into the FLASH memory of the target device if it succeeds. For devices with the FLASH organized in pages, the data address and size used with this command must confirm to that of the device. I.e. one ProgramFlash command is used to program one page in the target device.

XML path: /AVRPART/ICE_SETTINGS/STK600/HvspProgramFlash/

Table 6-7. Command format.

Field	Size	Values	Description
Command ID	1 byte	CMD_PROGRAM_FLASH_HVSP	Command id
NumBytes	2 byte	XML: blockSize	Total number of bytes to program, MSB first
Mode(1)	1 byte	XML: mode	Mode byte, *see description below
pollTimeout	1 byte	XML: pollTimeout	pollTimeout (in ms)
Data 1	1 byte		Data 1
…	.....	.....	.....
Data N	1 byte		Data N

Notes: 1. See details in table below.

Mode byte description

- Bit 0: This bit indicates whether to use byte ‘0’ or page ‘1’ programming.
- Bit 1-3 are the pagesize bits, pagesize are given in bytes not words, see table below.
- Bit 4-5 are not in use.
- Bit 6 must be set to ‘1’ when it is the very last page to be programmed, otherwise ‘0’
- Bit 7 indicates if a page write should be issued (*Transfer data to flash*). Normally it should always be set ‘1’. However, if the page size of the target device is too large to be covered by one Program Flash command (because the amount of available SRAM in STK600 is limited) this can be used to let 2 or more commands fill the

page buffer of the target device. The *transfer data to flash* flag should then only be set on the last command.

Table 6-8. Mode byte description.

Pagesize \ bit	3	2	1
256	0	0	0
2	0	0	1
4	0	1	0
8	0	1	1
16	1	0	0
32	1	0	1
64	1	1	0
128	1	1	1

NOTE: that only Bit 0-3 are set in the XML file, cause Bit 6-7 are not static and must be controlled by the PC Frontend.

Table 6-9. Answer format.

Field	Size	Values	Description
Answer ID	1 byte	CMD_CHIP_ERASE_HVSP	Answer id
Status	1 byte	STATUS_CMD_OK or STATUS_RDY_BSY_TOUT	A *Result Value* indicating the result of the operation.

6.5 CMD_READ_FLASH_HVSP

This command will read data from the FLASH memory of the target device if it succeeds. For devices with the FLASH organized in pages, the data address and size used with this command must confirm to that of the device. I.e. one ReadFlash command is used to read one page in the target device.

XML path: /AVRPART/ICE_SETTINGS/STK600/HvspReadFlash/

Table 6-10. Command format.

Field	Size	Values	Description
Command ID	1 byte	CMD_READ_FLASH_HVSP	Command id
NumBytes	2 byte	XML: blockSize	Total number of bytes to read, MSB first

Table 6-11. Answer format if the command is executed.

Field	Size	Values	Description
Answer ID	1 byte	CMD_READ_FLASH_HVSP	Answer id
Status1	1 byte	STATUS_CMD_OK	Temporary status, will always read OK
Data	Byte*N		Data read from device. Will be padded with 0's if errors occurred during read-out.
Status2	1 byte	STATUS_CMD_OK	A *Result Value* indicating the result of the operation. Will always read OK

6.6 CMD_PROGRAM_EEPROM_HVSP

See the CMD_WRITE_FLASH_HVSP command.

XML path: /AVRPART/ICE_SETTINGS/STK600/HvspProgramEeprom/

6.7 CMD_READ_EEPROM_HVSP

See the CMD_READ_FLASH_HVSP command.

XML path: /AVRPART/ICE_SETTINGS/STK600/HvspReadEeprom/

6.8 CMD_PROGRAM_FUSE_HVSP

This command programs one fuse byte, addressed by the Fuse Address byte.

XML path: /AVRPART/ICE_SETTINGS/STK600/HvspProgramFuse/

Table 6-12. Command format.

Field	Size	Values	Description
Command ID	1 byte	CMD_PROGRAM_FUSE_HVSP	Command id
Fuse Address	1 byte	0, 1 or 2	Address of byte to write (0=low, 1=high, 2=ext)
Fuse Byte	1 byte		Fuse byte value (low, high or ext) to be programmed
pollTimeout	1 byte	XML: pollTimeout	Poll timeout

Table 6-13. Answer format.

Field	Size	Values	Description
Answer ID	1 byte	CMD_CHIP_ERASE_HVSP	Answer id
Status	1 byte	STATUS_CMD_OK or STATUS_RDY_BSY_TOUT	A *Result Value* indicating the result of the operation.

6.9 CMD_READ_FUSE_HVSP

This command programs one fuse byte, addressed by the Fuse Address byte.

XML path: /AVRPART/ICE_SETTINGS/STK600/HvspReadFuse/

Table 6-14. Command format.

Field	Size	Values	Description
Command ID	1 byte	CMD_ READ_FUSE_HVSP	Command id
Fuse Address	1 byte	0, 1 or 2	Address of byte to read (0=low, 1=high, 2=ext)

Table 6-15. Answer format.

Field	Size	Values	Description
Answer ID	1 byte	CMD_ READ_FUSE_HVSP	Answer id
Status	1 byte	STATUS_CMD_OK	A *Result Value* indicating the result of the operation. Will always read OK
Fuse Byte	1 byte		Data read from device. Contains the read low, high or ext fuse byte. Will be padded with 0's if errors occurred during read-out.

6.10 CMD_PROGRAM_LOCK_HVSP

See CMD_PROGRAM_FUSE_HVSP.

Note: Address is required but ignored.

6.11 CMD_READ_LOCK_HVSP

See CMD_READ_FUSE_HVSP.

Note: Address is required but ignored.

6.12 CMD_READ_SIGNATURE_HVSP

See CMD_READ_FUSE_HVSP.

6.13 CMD_READ_OSCCAL_HVSP

See CMD_READ_FUSE_HVSP.

7 AVR8 JTAG

All AVR8 jtag commands start with CMD_JTAG_AVR (0x90)

The implementation of JTAG programming in STK600 is in equal to the JTAGICE mkII (see T0152-0007 JTAGICE mkII Communication Protocol) but without the JTAGICE mkII message structure and fifo packages, i.e. only the message body is used in STK600.

Table 7-1. Overall command format.

Field	Size	Values	Description
AVR8JTAG id	1 byte	CMD_JTAG_AVR	Command id, "JTAGICE wrapper"
Ice_command	1 byte	CMND_xxxx	JTAG command
Payload	n byte		JTAG command specific

Table 7-2. Overall answer format.

Field	Size	Values	Description
Answer ID	1 byte	CMD_JTAG_AVR	Command id
ICE answer msg body	n byte		

Before issuing any AVR8 JTAG commands the following parameters must be set correctly with the CMD_SET_PARAMETER command:

- PARAM_JTAG_ALLOW_FULL_PAGE_STREAM
- PARAM_JTAG_EEPROM_PAGE_SIZE
- PARAM2_JTAG_FLASH_PAGE_SIZE
- PARAM2_JTAG_FLASH_SIZE_H
- PARAM2_JTAG_FLASH_SIZE_L
- PARAM_JTAG_DAISY_BITS_BEFORE
- PARAM_JTAG_DAISY_BITS_AFTER
- PARAM_JTAG_DAISY_UNITS_BEFORE
- PARAM_JTAG_DAISY_UNITS_AFTER

7.1 Memory Read / Write

The following memory modes are defined for CMND_WRITE_MEMORY and CMND_READ_MEMORY. (The values are the same as for jtagice mkII)

Table 7-3. Valid Result Values for the answer to this command.

Name	Value
JTAGC_MemType_FLASH_PAGE	0xB0
JTAGC_MemType_EEPROM_PAGE	0xB1
JTAGC_MemType_FUSE_BITS	0xB2
JTAGC_MemType_LOCK_BITS	0xB3
JTAGC_MemType_SIGN_JTAG (ro)	0xB4
JTAGC_MemType_OSCCAL_BYTE (ro)	0xB5

7.1.1 CMND_READ_MEMORY

This commands reads a specified amount of bytes from the chosen target MCU memory. The target MCU must be in programming mode (CMND_ENTER_PROGMODE) before issuing this command.

Table 7-4. Command format.

Field	Size	Values	Description
AVR8JTAG id	1 byte	CMD_JTAG_AVR	Command id
Message ID	1 byte	CMND_WRITE_MEMORY (0x05)	
Memory type	1 byte	JTAGC_MemType_xxxx	See table above
Byte count	4 bytes	Lsb first	Number of bytes to write
Start address	4 bytes	Lsb first	Start memory address

Table 7-5. Answer format.

Field	Size	Values	Description
Answer id	1 byte	CMD_JTAG_AVR	Answer id
Status	1 byte	STATUS_CMD_OK STATUS_CMD_FAILED	
Data	n bytes		Payload if STATUS_CMD_OK

7.1.2 CMND_WRITE_MEMORY

This commands writes a specified amount of bytes to the chosen target MCU memory. The target MCU must be in programming mode (CMND_ENTER_PROGMODE) before issuing this command.

Table 7-6. Command format.

Field	Size	Values	Description
AVR8JTAG id	1 byte	CMD_JTAG_AVR	Command id
Message ID	1 byte	CMND_WRITE_MEMORY (0x04)	
Memory type	1 byte	JTAGC_MemType_xxxx	See table above
Byte count	4 bytes	Lsb first	Number of bytes to write
Start address	4 bytes	Lsb first	Start memory address
Data	n bytes	data	Byte count bytes of data

Table 7-7. Answer format.

Field	Size	Values	Description
Answer id	1 byte	CMD_JTAG_AVR	Answer id
Status	1 byte	STATUS_CMD_OK STATUS_CMD_TOUT STATUS_CMD_FAILED	

7.1.3 CMND_RESET

This command will place the RESET instruction into the JTAG instruction register. The target MCU must be in programming mode (CMND_ENTER_PROGMODE) before issuing this command.

Table 7-8. Command format.

Field	Size	Values	Description
AVR8JTAG id	1 byte	CMD_JTAG_AVR	Command id
Message ID	1 byte	CMND_RESET (0x0B)	

Table 7-9. Answer format.

Field	Size	Values	Description
Answer id	1 byte	CMD_JTAG_AVR	Answer id
Status	1 byte	STATUS_CMD_OK	

7.1.4 CMND_ENTER_PROGMODE

This command enables communication between the STK600 and the target MCU and should be the first JTAG command sent to the STK600. After receiving this command the MCU is held in reset state and memory read and write is allowed.

Table 7-10. Command format.

Field	Size	Values	Description
AVR8JTAG id	1 byte	CMD_JTAG_AVR	Command id
Message ID	1 byte	CMND_ENTER_PROGMODE (0x14)	

Table 7-11. Answer format.

Field	Size	Values	Description
Answer id	1 byte	CMD_JTAG_AVR	Answer id
status	1 byte	STATUS_CMD_OK STATUS_CMD_FAILED	If failed the JTAG id was not read successfully from target

7.1.5 CMND_LEAVE_PROGMODE

This command releases the target MCU which will run freely from the reset vector.

Table 7-12. Command format.

Field	Size	Values	Description
AVR8JTAG id	1 byte	CMD_JTAG_AVR	Command id
Message ID	1 byte	CMND_LEAVE_PROGMODE (0x15)	

Table 7-13. Answer format.

Field	Size	Values	Description
Answer id	1 byte	CMD_JTAG_AVR	Answer id
status	1 byte	STATUS_CMD_OK	

7.1.6 CMND_CHIP_ERASE

This command erases all memory and releases the lock bits of the target MCU.

Table 7-14. Command format.

Field	Size	Values	Description
AVR8JTAG id	1 byte	CMD_JTAG_AVR	Command id
Message ID	1 byte	CMND_CHIP_ERASE (0x13)	

Table 7-15. Answer format.

Field	Size	Values	Description
Answer id	1 byte	CMD_JTAG_AVR	Answer id
status	1 byte	STATUS_CMD_OK STATUS_CMD_TOUT	

8 AVR32 JTAG

All AVR32 jtag commands except for ENTER_PROGMODE and LEAVE_PROGMODE start with CMD_JTAG_AVR32 (0x80)

The implementation of JTAG programming in STK600 is in equal to the JTAGICE mkII (see T0152-0007 JTAGICE mkII Communication Protocol) but without the JTAGICE mk II message structure and fifo packages, i.e. only the message body is used in STK600.

Before issuing any AVR32 JTAG commands the following parameters must be set correctly with the CMD_SET_PARAMETER command:

- PARAM_JTAG_DAISY_BITS_BEFORE
- PARAM_JTAG_DAISY_BITS_AFTER
- PARAM_JTAG_DAISY_UNITS_BEFORE
- PARAM_JTAG_DAISY_UNITS_AFTER

8.1 CMD_ENTER_PROGMODE_JTAG_AVR32

This command enables communication with the JTAG interface on the AVR32. This command should be the first AVR32 JTAG command issued.

Table 8-1. Command format.

Field	Size	Values	Description
AVR32JTAG id	1 byte	CMD_ENTER_PROGMODE_32 (0x81)	Command id

Table 8-2. Answer format.

Field	Size	Values	Description
Answer id	1 byte	CMD_ENTER_PROGMODE_32	Answer id
status	1 byte	STATUS_CMD_OK	

8.2 CMD_LEAVE_PROGMODE_JTAG_AVR32

This command disables the JTAG interface.

Table 8-3. Command format.

Field	Size	Values	Description
AVR32JTAG id	1 byte	CMD_LEAVE_PROGMODE_32	Command id (0x82)

Table 8-4. Answer format.

Field	Size	Values	Description
Answer id	1 byte	CMD_LEAVE_PROGMODE_32	Answer id
status	1 byte	STATUS_CMD_OK	

8.3 CMD_RESET_AVR32

This command places the given reset value into the JTAG reset register of the target MCU.

Table 8-5. Command format.

Field	Size	Values	Description
AVR32JTAG id	1 byte	CMD_JTAG_AVR32	Command id (0x80)
ICE command	1 byte	CMND_AVR32_RESET (0x2b)	
Reset value	1 byte	MCU specific	Reset value

Table 8-6. Answer format.

Field	Size	Values	Description
Answer id	1 byte	CMD_JTAG_AVR32	Answer id
status	1 byte	STATUS_CMD_OK	

8.4 CMD_SAB_WRITE_AVR32

This command writes a word to the given SAB address in the target MCU

Table 8-7. Command format.

Field	Size	Values	Description
AVR32JTAG id	1 byte	CMD_JTAG_AVR32	Command id (0x80)
ICE command	1 byte	CMND_JTAG_SAB_WRITE (0x28)	
SAB address	5 bytes	Sab address	(msb first)
Data	4 bytes		(msb first)

Table 8-8. Answer format.

Field	Size	Values	Description
Answer id	1 byte	CMD_JTAG_AVR32	Answer id
status	1 byte	RSP_OK RSP_FAILED	

8.5 CMD_SAB_READ_AVR32

This command reads a word from the given SAB address from the target MCU.

Table 8-9. Command format.

Field	Size	Values	Description
AVR32JTAG id	1 byte	CMD_JTAG_AVR32	Command id (0x80)
ICE command	1 byte	CMND_JTAG_SAB_READ (0x29)	
SAB address	5 bytes		(msb first)

Table 8-10. Answer format.

Field	Size	Values	Description
Answer id	1 byte	CMD_JTAG_AVR32	Answer id
status	1 byte	RSP_FAILED RSP_SCAN_CHAIN_READ	
data	4 bytes		If RSP_SCAN_CHAIN_READ

8.6 CMD_BLOCK_WRITE_AVR32

This command writes a block of data starting at the specified SAB address. N should be less than 64 to comply with the maximum data buffer of STK600.

Table 8-11. Command format.

Field	Size	Values	Description
AVR32JTAG id	1 byte	CMD_JTAG_AVR32	Command id (0x80)
ICE command	1 byte	CMND_JTAG_ BLOCK_WRITE (0x2d)	
count	1 byte	n	Number of words < 64
SAB address	8 bytes		0 padded, msb first
data	n * 4 bytes		

Table 8-12. Answer format.

Field	Size	Values	Description
Answer id	1 byte	CMD_JTAG_AVR32	Answer id
Status	1 byte	RSP_OK RSP_FAILED RSP_ILLEGAL_VALUE	

8.7 CMD_ BLOCK_READ_AVR32

This command reads a block of data starting at the specified SAB address. N should be less than 64 to comply with the maximum data buffer of STK600

Table 8-13. Command format.

Field	Size	Values	Description
AVR32JTAG id	1 byte	CMD_JTAG_AVR32	Command id (0x80)
ICE command	1 byte	CMND_JTAG_BLOCK_READ (0x2c)	
Count	1 byte	n	Number of words
SAB address	5 bytes		

Table 8-14. Answer format.

Field	Size	Values	Description
Answer id	1 byte	CMD_JTAG_AVR32	Answer id
Status	1 byte	RSP_FAILED RSP_ILLEGAL_VALUE RSP_SCAN_CHAIN_READ	
Data	n * 4 bytes		If RSP_SCAN_CHAIN_READ

8.8 CMD_NEXUS_WRITE_AVR32

This command writes a word to the specified nexus register. It is equivalent to a SAB write to address OCD_REGISTER + <register> * 4.

Table 8-15. Command format.

Field	Size	Values	Description
AVR32JTAG id	1 byte	CMD_JTAG_AVR32	Command id (0x80)
ICE command	1 byte	CMND_JTAG_NEXUS_WRITE (0x26)	
register	1 byte		
Data	4 bytes		

Table 8-16. Answer format.

Field	Size	Values	Description
Answer id	1 byte	CMD_JTAG_AVR32	Answer id
status	1 byte	RSP_OK RSP_FAILED	

8.9 CMD_NEXUS_READ_AVR32

This command reads a word to the specified nexus register. It is equivalent to a SAB read from address OCD_REGISTER + <register> * 4.

Table 8-17. Command format.

Field	Size	Values	Description
AVR32JTAG id	1 byte	CMD_JTAG_AVR32	Command id (0x80)
ICE command	1 byte	CMND_JTAG_NEXUS_READ (0x27)	
Register	1 byte		

Table 8-18. Answer format.

Field	Size	Values	Description
Answer id	1 byte	CMD_JTAG_AVR32	Answer id
status	1 byte	RSP_FAILED RSP_SCAN_CHAIN_READ	
Data	4 bytes		If RSP_SCAN_CHAIN_READ

9 XPROG protocol

The XPROG programming protocol is used with the ATxmega devices, and can use both JTAG and PDI as the physical interface.

All multi-byte values (e.g. address and length fields) are big endian.

9.1 CMD_XPROG_SETMODE

Before using any of the XPROG commands, the descried XPROG interface must be selected. This can either be PDI or JTAG.

Table 9-1. Command format.

Offset	Field	Size	Values
0	Command ID(1)	1 byte	CMD_XPROG_SETMODE
1	Mode(2)	1 byte	0 = PDI mode, 1 = JTAG mode

Notes: 1. The command identifier.
2. Set which of the physical interfaces to use: 0 = PDI, 1 = JTAG.

Table 9-2. Answer format.

Offset	Field	Size	Values
0	Command ID(1)	1 byte	CMD_XPROG_SETMODE
1	Status(2)	1 byte	

Notes: 1. The command identifier.
2. One of the defined error codes. XPRG_ERR_OK indicates success.

9.2 CMD_XPROG

This is STK600's wrapper command for all XPROG specific commands.

Table 9-3. Command format.

Offset	Field	Size	Values
0	Command ID(1)	1 byte	CMD_XPROG
1	XPROG Command(2)	n bytes	Any XPROG command

Notes: 1. The command identifier.
2. This part of the STK600 command contains the XPROG command.

Table 9-4. Answer format.

Offset	Field	Size	Values
0	Command ID(1)	1 byte	CMD_XPROG
1	XPROG answer(2)	n byte	Any XPROG command's answer

Notes: 1. The command identifier.
2. The answer for the XPROG command is placed in this filed. The length of the field is depending on the XPROG command.

The XPROG commands are described in the following section. The offsets indicated are offsets in the XPROG part of the CMD_XPROG command. To find the absolute offset in the STK600 command, add one.

9.2.1 XPRG_ENTER_PROGMODE

This command enables programming mode in the device by enabling the programming interface hardware and sending the programming key.

Note: Before using this instruction, the programming interface (PDI or JTAG) must have been set by the programmer specific 'set mode' command.

Table 9-5. Command format.

Offset	Field	Size	Values
0	Command ID[1]	1 byte	XPRG_ENTER_PROGMODE

Notes: 1. The command identifier

Table 9-6. Answer format.

Offset	Field	Size	Values
0	Command ID[1]	1 byte	XPRG_ENTER_PROGMODE
1	Status[2]	1 byte	

Notes: 1. The command identifier.
2. One of the defined error codes.

9.2.2 XPRG_LEAVE_PROGMODE

This command leaves the device's programming mode by clearing the reset flags and disabling the programming interface in hardware.

Table 9-7. Command format.

Offset	Field	Size	Values
0	Command ID[1]	1 byte	XPRG_LEAVE_PROGMODE

Notes: 1. The command identifier

Table 9-8. Answer format.

Offset	Field	Size	Values
0	Command ID[1]	1 byte	XPRG_ LEAVE _PROGMODE
1	Status[2]	1 byte	

Notes: 1. The command identifier.
2. XPRG_ERR_OK indicates success.

9.2.3 XPRG_SET_PARAMETER

In order to work correctly, some parameters must be set in the programmer. Use the following command to initialize the XPROG interface:

Table 9-9. Command format.

Offset	Field	Size	Values
0	Command ID(1)	1 byte	XPRG_ENTER_PROGMODE
1	Parameter(2)	1 byte	One of the XPRG_PARAM parameters
2	Value(3)	1-4	Depends on the parameter.

Notes: 1. The command identifier
2. The parameter to set. See the appendix for the values.
3. The value's size is depending on which parameter that is to be set.

Table 9-10. Answer format.

Offset	Field	Size	Values
0	Command ID(1)	1 byte	XPRG_ENTER_PROGMODE
1	Status(2)	1 byte	

Notes: 1. The command identifier.
2. One of the defined error codes.

All parameters listed in the appendix must be set in order to initialize the interface.

NOTE: Later revisions of the XPROG programming protocol may introduce new parameters.

9.2.4 XPRG_ERASE

The XMega memories, fuses and lockbits can be erased at different levels: chip erase (which clears all memories and the lockbits), application erase, boot erase and EEPROM erase. There is also possible to erase single pages in flash and EEPROM

Table 9-11. Command format.

Offset	Field	Size	Values
0	Command ID[1]	1 byte	XPRG_ERASE
1	Erase mode[2]	1 byte	XPRG_ERASE_CHIP, XPRG_ERASE_APP, XPRG_ERASE_BOOT, XPRG_ERASE_EEPROM, XPRG_ERASE_APP_PAGE, XPRG_ERASE_BOOT_PAGE, XPRG_ERASE_EEPROM_PAGE or XPRG_ERASE_USERSIG
2	Address[3]	4 bytes	Any

Notes:
1. The command identifier
2. XPRG_ERASE_CHIP: Erases the whole chip and clears lockbits
   XPRG_ERASE_APP: Erases the application section
   XPRG_ERASE_BOOT: Erase the boot section
   XPRG_ERASE_EEPROM: Erases the whole EEPROM
   XPRG_ERASE_APP_PAGE: Erases one application page
   XPRG_ERASE_BOOT_PAGE: Erase one boot page
   XPRG_ERASE_EEPROM_PAGE: Erases one EEPROM page
   XPRG_ERASE_USERSIG: Erases the user signature
3. For the other modes, the Address parameter must point to a location inside the memory area to be erased. The address is a TIF space address.
3. The address parameter is ignored if erase mode is XPRG_ERASE_CHIP.

Table 9-12. Answer format.

Offset	Field	Size	Values
0	Command ID[1]	1 byte	XPRG_ERASE
1	Status[2]	1 byte	

Notes:
1. The command identifier.
2. One of the defined error codes.

9.2.5 XPRG_WRITE_MEM

This command handles programming of the different XMega memories: application, boot and eeprom. Fuses, lockbits and user signatures are also programmed with this command.

Table 9-13. Command format.

Offset	Field	Size	Values
0	Command ID(1)	1 byte	XPRG_ERASE
1	Memory type(2)	1 byte	Application, Boot, EEPROM, Fuse, Lockbits…
2	PageMode(3)	1 byte	Bitfield, see description below
3	Address(4)	4 bytes	Any address
7	Length(5)	2 bytes	1 to 512
9	Data(6)	N bytes	N data bytes, size is given by the *Length* parameter

Notes:
1. The command identifier
2. XPRG_MEM_TYPE_APPL
   XPRG_MEM_TYPE_BOOT
   XPRG_MEM_TYPE_EEPROM
   XPRG_MEM_TYPE_FUSE
   XPRG_MEM_TYPE_LOCKBITS
   XPRG_MEM_TYPE_USERSIG
3. If *Memory type* is XPRG_MEM_TYPE_APPL, XPRG_MEM_TYPE_BOOT or XPRG_MEM_TYPE_EEPROM:
   Bit 0: Write page
   Bit 1: Erase page
4. The start address of the data to be written. The address is in the TIF address space
5. Can be any value between 1 and 512. If page programming, and the actual page size is bigger than 256, the operation must be split into two or more XPRG_WRITE_MEM operations, where only the last operation has the *Write page* bit set.
   Note: Only APP, BOOT and EEPROM handles page operations, for any other memory type, the *Length* field must be set to 1.
4. The data to be written. The size is indicated by the *Length* field.

Table 9-14. Answer format.

Offset	Field	Size	Values
0	Command ID(1)	1 byte	XPRG_WRITE_MEM
1	Status(2)	1 byte	

Notes:
1. The command identifier.
2. One of the defined error codes.

9.2.6 XPRG_READ_MEM

This command handles reading of the different XMega memories: application, boot and eeprom, signatures, fuses, lockbits and factory calibration values.

Table 9-15. Command format.

Offset	Field	Size	Values
0	Command ID[1]	1 byte	XPRG_READ_MEM
1	Memory type[2]	1 byte	Application, Boot, EEPROM, Fuse, Lockbits…
2	Address[3]	4 bytes	Any address
6	Length[4]	2 bytes	1 to 256

Notes:
1. The command identifier
2. XPRG_MEM_TYPE_APPL
   XPRG_MEM_TYPE_BOOT
   XPRG_MEM_TYPE_EEPROM
   XPRG_MEM_TYPE_FUSE
   XPRG_MEM_TYPE_LOCKBITS
   XPRG_MEM_TYPE_USERSIG
   XPRG_MEM_TYPE_FACTORY_CALIBRATION
3. The start address of the data to be read. The address is in the TIF address space.
4. How many bytes to be read. It can be any value between 1 and 256.

Table 9-16. Answer format.

Offset	Field	Size	Values
0	Command ID[1]	1 byte	XPRG_READ_MEM
1	Status[2]	1 byte	
2	Data[3]	N bytes	

Notes:
1. The command identifier.
2. One of the defined error codes.
3. The requested memory area, lowest address first.

9.2.7 XPRG_READ_CRC

This command starts the CRC generator and returns a three-byte CRC.

Table 9-17. Command format.

Offset	Field	Size	Values
0	Command ID[1]	1 byte	XPRG_READ_CRC
1	CRC type[2]	1 byte	Application, Boot, or entire flash

Notes:
1. The command identifier
2. XPRG_CRC_APP,
   XPRG_CRC_BOOT or
   XPRG_CRC_FLASH, see appendix.

Table 9-18. Answer format.

Offset	Field	Size	Values
0	Command ID[1]	1 byte	XPRG_READ_CRC
1	Status[2]	1 byte	
2	Data[3]	3 bytes	CRC value

Notes: 1. The command identifier.
2. One of the defined error codes.
3. The CRC value calculated by the device itself.

10 Return Values

This section describes all possible return values and their meaning in detail.

10.1 Success

Table 10-1. Return values

Value	Meaning
STATUS_CMD_OK	Command executed OK

10.2 Warnings

All warnings have MSB set to 1 and MSB-1 set to 0.

Table 10-2. Return values

Value	Meaning
STATUS_CMD_TOUT	Command timed out
STATUS_RDY_BSY_TOUT	Sampling of the RDY/nBSY pin timed out
STATUS_SET_PARAM_MISSING	The ‘Set Device Parameters’ have not been executed in advance of this command

10.3 Errors

All errors have MSB and MSB-1 set to 1.

Table 10-3. Return values

Value	Meaning
STATUS_CMD_FAILED	Command failed
STATUS_CKSUM_ERROR	Checksum Error
STATUS_CMD_UNKNOWN	Unknown command

11 Parameters

The following parameters can be read and/or written by the CMD_GET_PARAM and CMD_SET_PARAM commands:

The parameter values are either one or two bytes. For two-byte values, the high byte is read/written first.

Table 11-1. One-byte parameters

Value	Meaning	R/W
PARAM_HW_VER	Hardware version	R
PARAM_SW_MAJOR	Firmware version number, master mcu	R
PARAM_SW_MINOR	Firmware version number, master mcu	R
PARAM_VTARGET	Target Voltage	RW
PARAM_STATUS	Returns status register	R
PARAM_STATUS_TGT_CONN	Status of target connection	R
PARAM_CONTROLLER_INIT	Indicates if board has been powered down	R
PARAM_DISCHARGEDELAY	Delay with higher resistance of reset line	W
PARAM_SOCKETCARD_ID	Identification value for socketcard	R
PARAM_ROUTINGCARD_ID	Identification value for routingcard	R
PARAM_EXPCARD_ID	Identification value for expansioncard	R
PARAM_SW_MAJOR_SLAVE1	Firmware version number, slave 1 mcu	R
PARAM_SW_MINOR_SLAVE1	Firmware version number, slave 1 mcu	R
PARAM_SW_MAJOR_SLAVE2	Firmware version number, slave 2 mcu	R
PARAM_SW_MINOR_SLAVE2	Firmware version number, slave 2 mcu	R/W
PARAM_BOARD_ID_STATUS	Returns status for valid combination of routing and socketcard	R
PARAM_RESET	Set reset high or low	R
PARAM_JTAG_ALLOW_FULL_PAGE_STREAM	Boolean: If target allows full page bit stream for memory read/write	R
PARAM_JTAG_EEPROM_PAGE_SIZE	Page size of target eeprom	RW
PARAM_JTAG_DAISY_BITS_BEFORE	Instruction register bits before	R

Table 11-2. Two-byte parameters

Value	Meaning	R/W
PARAM2_SCK_DURATION	ISP SCK duration	RW
PARAM2_AREF0	ADC reference voltage	RW
PARAM2_AREF1	ADC reference voltage	RW
PARAM2_CLOCK_CONF	Register values for programmable oscillator	RW
PARAM2_RC_ID_TABLE_REV	Routing and socketcard table revision	R
PARAM2_EC_ID_TABLE_REV	Expansioncard table revision	R
PARAM2_JTAG_FLASH_PAGE_SIZE	Page size of target flash	RW
PARAM2_JTAG_FLASH_SIZE_H	High word of flash size	RW
PARAM2_JTAG_FLASH_SIZE_L	Low word of flash size	RW

11.1 PARAM_HW_VER

Returns a hardware revision number.

11.2 PARAM_SW_MAJOR

The PARAM_SW_MAJOR and PARAM_SW_MINOR returns the firmware version of the master mcu.

11.3 PARAM_SW_MINOR

See PARAM_SW_MAJOR.

11.4 PARAM_VTARGET

The parameter value is voltage in volts x10, i.e. a parameter value of 42 (decimal) corresponds to 4.2V.

Note: This parameter cannot be set when the STK600 is in programming mode.

The maximum target voltage level is dependant on what target routing/socket board is mounted.

To find the maximum allowable target voltage the board-id must be read before setting a new voltage.

The maximum voltage for the board is found in “targetboard.xml”

11.5 PARAM_STATUS_TGT_CONN

This parameter returns the same status value as CMD_CHECK_TARGET_CONNECTION.

See chapter 3.9

The difference between using the command and the GET_PARAM is that bits 4-5 are only updated after the command has been run.

11.6 PARAM2_SCK_DURATION

This is a two-byte value, which sets the ISP frequency.

When using the ISP programming interface, the ISP clock frequency must not exceed what the target device supports. (The maximum ISP clock frequency depends on the device system clock, internal clock division etc.)

The STK600 supports ISP frequencies from 1953 Hz up to 8.0 MHz. The value for PARAM_SCK_DURATION can be found using the following algorithm:

```
unsigned int CalcSckDur(int freq)
{
    sck_dur = ceil((16e6/(2*freq))-1);

    return __min(4096, sck_dur);     // 4096 is an illegal value
}
```

11.7 PARAM_CONTROLLER_INIT

This parameter is internally set to 0 when the programmer MCU resets. The host software can write any value it wants to this parameter, and it can be read back later.

This parameter is intended to be used as a way of telling if the power on STK600 has been lost or turned off and then back again.

This way, the host software can tell if the programmer needs to be initialized again before continuing with its operation.

If the returned value is 0 the CMD_SET_CONTROL_STACK must be executed before continuing with any high voltage programming commands.

11.8 PARAM_DISCHARGEDELAY

This parameter sets a time period for which the reset line has a higher resistance for each time it is toggled.

The purpose is to reduce the maximum current caused by the discharge/recharge of a decoupling capacitor connected to the reset pin.

When the reset is toggled a resistor of 510ohm will be switched in, which reduces the peak current to an acceptable level for the internal components of the STK600.

The delay should be set to: t>510ohm*C

If no capacitor is connected this parameter could be set to 0.

11.9 PARAM2_AREF0

The parameter value is voltage in volts x100, i.e. a parameter value of 449 (decimal) corresponds to 4.49V.

The maximum value is 5.5V, or parameter value 550.

11.10 PARAM2_AREF1

Same as for AREF0.

11.11 PARAM2_CLOCK_CONF

This is a two-byte value, which configures the chip oscillator.

Table 11-3. Two-byte value.

D15	D14	D13	D12	D11	D10	D9	D8
OCT3	OCT2	OCT1	OCT0	DAC9	DAC8	DAC7	DAC6

Table 11-4. Two-byte value.

D7	D6	D5	D4	D3	D2	D1	D0
DAC5	DAC4	DAC3	DAC2	DAC1	DAC0	X	X

The frequency is given by the formula:

Equation 11-1. Frequency

$$f = 2^{OCT} \cdot \frac{2078}{2 - \frac{DAC}{1024}}$$

Equation 11-2. OCT

$$OCT = 3.322 \log\left(\frac{f}{1039}\right)$$

Equation 11-3. DAC

$$DAC = 2048 - \frac{2078 \cdot 2^{(10+OCT)}}{f}$$

11.12 PARAM_SOCKETCARD_ID

Returns a byte that identifies the socket card.

The file “targetboard.xml” contains a list of the cards with id values.

11.13 PARAM_ROUTINGCARD_ID

Returns a byte that identifies the routing card.

The file “targetboard.xml” contains a list of the cards with id values.

11.14 PARAM_EXPCARD_ID

Returns a byte that identifies the expansion card.

The file “targetboard.xml” contains a list of the cards with id values.

11.15 PARAM_SW_MAJOR_SLAVE1

The PARAM_SW_MAJOR_SLAVE1 and PARAM_SW_MINOR_SLAVE1 returns the firmware version of the slave 1 mcu.

11.16 PARAM_SW_MINOR_SLAVE1

See chapter 11.15

11.17 PARAM_SW_MAJOR_SLAVE2

The PARAM_SW_MAJOR_SLAVE2 and PARAM_SW_MINOR_SLAVE2 returns the firmware version of the slave 2 mcu.

11.18 PARAM_SW_MINOR_SLAVE2

See chapter 11.17

11.19 PARAM2_RC_ID_TABLE_REV

Routingcard Board ID table revision

11.20 PARAM2_EC_ID_TABLE_REV

Expansioncard Board ID table revision

11.21 PARAM_BOARD_ID_STATUS

0x00 = OK

0x01 = BoardID mismatch (VTG set to 0V)

0x02 = BoardID changed while power on STK600 (VTG set to 0V)

11.22 PARAM_RESET

Used to set the reset line high or low. Will only work when the board is idle.

If used when the kit is in programming mode the parameter will be ignored.

11.23 PARAM_RESET_POLARITY

For backward compatability with the STK500, this parameter sets the reset polarity.

11.24 PARAM_JTAG_ALLOW_FULL_PAGE_STREAM

Boolean value to indicate if the target MCU supports streaming a full flash page of data, or if each byte must be followed by a return to idle (jtag state). XML:AVRPART\ICE_SETTINGS\JTAGICEmkII\ucAllowFullPageBitstream

11.25 PARAM_JTAG_EEPROM_PAGE_SIZE

Used to set the EEprom page size (in bytes)

11.26 PARAM2_JTAG_FLASH_PAGE_SIZE

Used to set the flash page size (in bytes)

11.27 PARAM2_JTAG_FLASH_SIZE_H

Used to set the flash size (in bytes). As this value can extend 0xffff (64k) the value is split into two parameter registers.

11.28 PARAM2_JTAG_FLASH_SIZE_L

See PARAM2_JTAG_FLASH_SIZE_H

11.29 PARAM_JTAG_DAISY_BITS_BEFORE

When the target MCU is part of a JTAG daisy chain the STK600 must know how many bits it should skip before and after the target. The PARAM_JTAG_DAISY_BITS_BEFORE and AFTER values specifies how many instruction register bits to skip.

In the JTAG data phase all non targeted devices have the SKIP register selected. The SKIP register is one bit. The PARAM_JTAG_DAISY_UNITS_BEFORE and AFTER parameters specify the number of devices before and after the target device, and hence the total SKIP register length to ignore.

Example: If the targeted AVR is placed second, in a 4 device chain with instruction register lengths 2, 2 and 5 respectively,

Table 11-5.

Name	Value

Name	Value
PARAM_JTAG_DAISY_BITS_BEFORE	2
PARAM_JTAG_DAISY_BITS_AFTER	7
PARAM_JTAG_DAISY_UNITS_BEFORE	1
PARAM_JTAG_DAISY_UNITS_AFTER	2

11.30 PARAM_JTAG_DAISY_BITS_AFTER

See PARAM_JTAG_DAISY_BITS_BEFORE.

11.31 PARAM_JTAG_DAISY_UNITS_BEFORE

See PARAM_JTAG_DAISY_BITS_BEFORE.

11.32 PARAM_JTAG_DAISY_UNITS_AFTER

See PARAM_JTAG_DAISY_BITS_BEFORE.

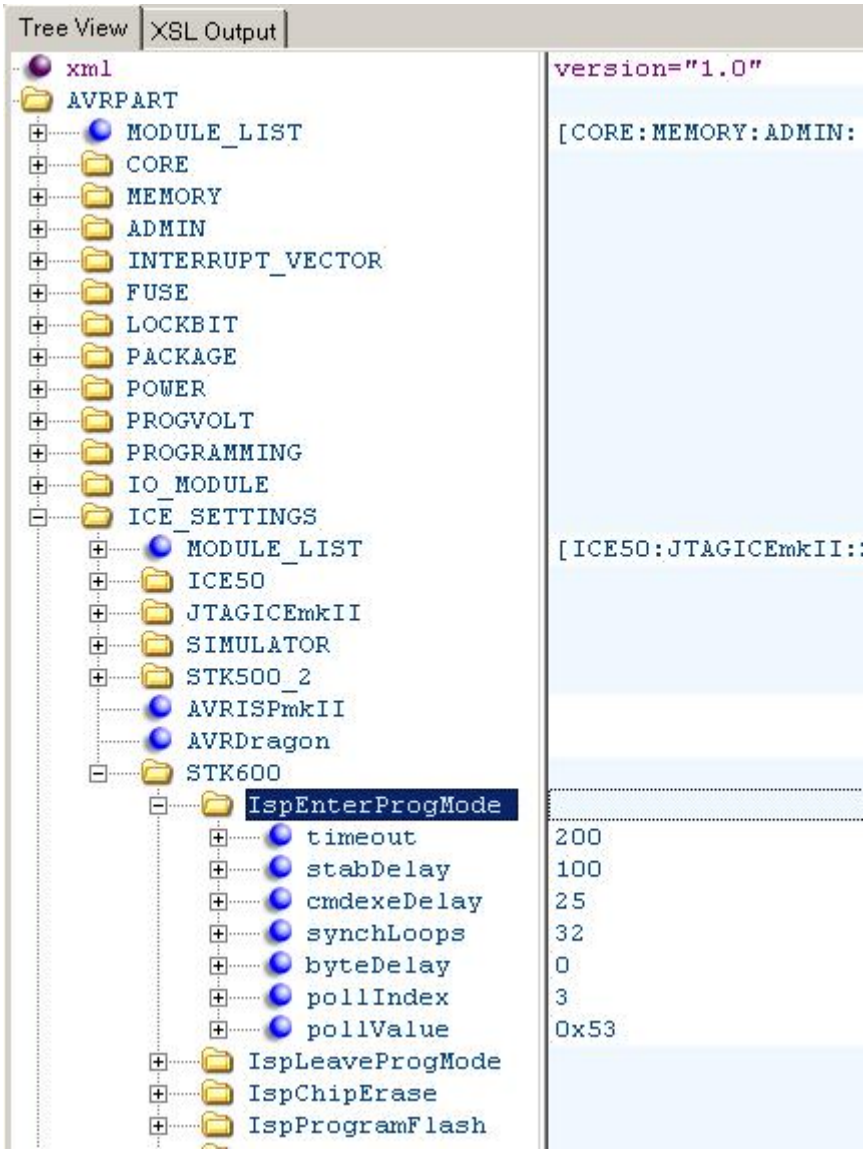
12 XML Parameter Values

A parameter set is specific for a certain AVR device, thus the parameter settings belong in the XML part description file. Below is a description where to find the STK600 parameter values in the device specific XML file.

After installing AVR Studio 4, all xml files can be found at:

“\Program Files\Atmel\AVR Tools\PartDescriptionFiles\”.

Figure 12-1. XML file viewed in XML notepad



Open the XML file in xml editor/viewer (e.g XML Notepad or Internet Explorer). All device specific values for STK600 are located under STK600 node. For parameters for the CMD_ENTER_PROGMODE_ISP command, look in /AVRPART/ICE_SETTINGS/STK600/IspEnterProgMode.

13 Command Sequence Example

This chapter contains examples of how to connect to the STK600 from the PC front-end and how to read signature from a device.

See chapter 2.4 Command for the description of the commands and parameters.

13.1 Connect

The sequence of commands and parameters sent from PC front-end to the STK600 in order to connect is listed below.

- CMD_SIGN_ON
- CMD_GET_PARAMETER, PARAM_HW_VER
- CMD_GET_PARAMETER, PARAM_SW_MAJOR
- CMD_GET_PARAMETER, PARAM_SW_MINOR
- CMD_GET_PARAMETER, PARAM_SW_MAJOR_S1
- CMD_GET_PARAMETER, PARAM_SW_MINOR_S1
- CMD_GET_PARAMETER, PARAM_SW_MAJOR_S2
- CMD_GET_PARAMETER, PARAM_SW_MINOR_S2

13.2 Read Signature

The sequence of commands and parameters sent from PC front-end to the STK600 in order to read the device signature through ISP is listed below. Note that one already has to be connected to do this.

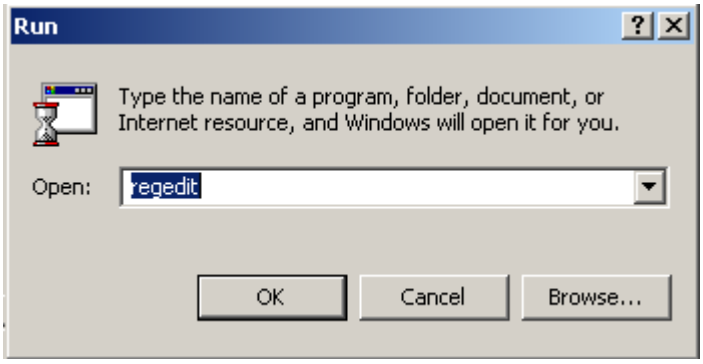
- CMD_CHECK_STATUS_TGT_CONN
- CMD_ENTER_PROGMODE_ISP
- CMD_READ_SIGNATURE_ISP
- CMD_READ_SIGNATURE_ISP
- CMD_READ_SIGNATURE_ISP
- CMD_LEAVE_PROGMODE_ISP

13.3 STK600 Communication Logging

For further details and examples of the communication between AVR Studio and STK600 one can set up logging of all communication to a text file. This can be done by adding a register key in the Registry as described below.

1. Open the Registry by running “regedit”:

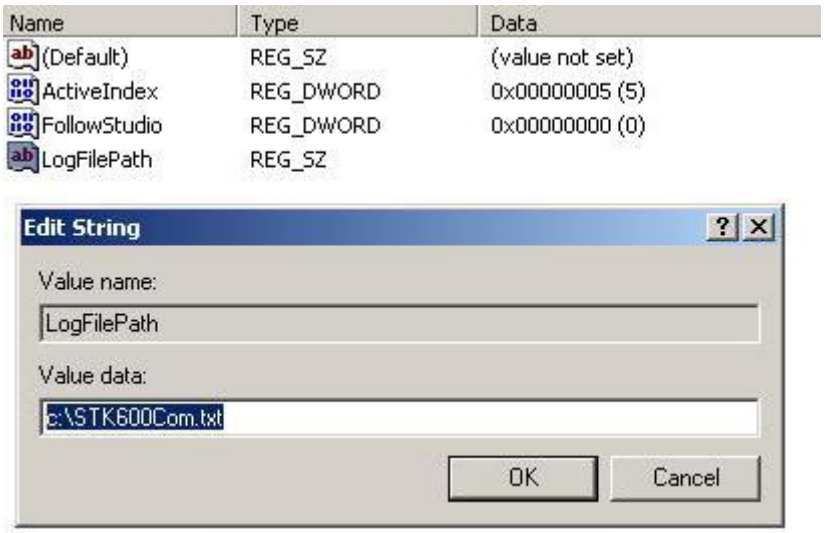
Figure 13-1. Start regedit from run on the windows menu



2. Browse to the path: HKEY_CURRENT_USER\Software\Atmel\AVRTools\STK500\

3. Make a new String Value (right-click > New > String Value) named “LogFilePath”
4. Enter Data e.g. ”c:\STK600Com.txt” (right-click ”LogFilePath” > Modify > enter Value Data)

Figure 13-2. Adding registry string



The communication log for an ISP read signature of an ATmega2560 is shown below.

Compare the command and answer data with the format tables in chapter 4.9

Figure 13-3. Communication log of ISP read signature of ATmega2560

```
Sending packet 01/04/2008 09:13:18.109
CMD_ENTER_PROGMODE_ISP
(1200 ms, 12 bytes) > 10 C8 64 19 20 00 53 03 AC 53 00 00

Receiving packet 01/04/2008 09:13:18.171
CMD_ENTER_PROGMODE_ISP
(1200 ms, 2 bytes) < 10 00
Returned status: Command succeeded

Sending packet 01/04/2008 09:13:18.187
CMD_READ_SIGNATURE_ISP
(1000 ms, 6 bytes) > 1B 04 30 00 00 00

Receiving packet 01/04/2008 09:13:18.187
CMD_READ_SIGNATURE_ISP
(1000 ms, 4 bytes) < 1B 00 1E 00
Returned status: Command succeeded

Sending packet 01/04/2008 09:13:18.187
CMD_READ_SIGNATURE_ISP
(1000 ms, 6 bytes) > 1B 04 30 00 01 00

Receiving packet 01/04/2008 09:13:18.187
CMD_READ_SIGNATURE_ISP
(1000 ms, 4 bytes) < 1B 00 98 00
Returned status: Command succeeded

Sending packet 01/04/2008 09:13:18.187
CMD_READ_SIGNATURE_ISP
(1000 ms, 6 bytes) > 1B 04 30 00 02 00

Receiving packet 01/04/2008 09:13:18.187
CMD_READ_SIGNATURE_ISP
(1000 ms, 4 bytes) < 1B 00 01 00
Returned status: Command succeeded

Sending packet 01/04/2008 09:13:18.218
CMD_LEAVE_PROGMODE_ISP
(1000 ms, 3 bytes) > 11 01 01

Receiving packet 01/04/2008 09:13:18.218
CMD_LEAVE_PROGMODE_ISP
(1000 ms, 2 bytes) < 11 00
Returned status: Command succeeded
Port closed
Returned status: Command succeeded
```

Command to STK600

Answer from STK600

Appendix

A.1 Commands and parameters

```
// *** General command constants ***

#define CMD_SIGN_ON                    0x01
#define CMD_SET_PARAMETER              0x02
#define CMD_GET_PARAMETER              0x03
#define CMD_OSCCAL                     0x05
#define CMD_LOAD_ADDRESS               0x06
#define CMD_FIRMWARE_UPGRADE           0x07
#define CMD_RESET_PROTECTION           0x0A
#define CMD_CHECK_TARGET_CONNECTION    0x0D
#define CMD_LOAD_RC_ID_TABLE           0x0E
#define CMD_LOAD_EC_ID_TABLE           0x0F
#define CMD_CLEAR_RC_ID_TABLE          0x09


// *** ISP command constants ***

#define CMD_ENTER_PROGMODE_ISP         0x10
#define CMD_LEAVE_PROGMODE_ISP         0x11
#define CMD_CHIP_ERASE_ISP             0x12
#define CMD_PROGRAM_FLASH_ISP          0x13
#define CMD_READ_FLASH_ISP             0x14
#define CMD_PROGRAM_EEPROM_ISP         0x15
#define CMD_READ_EEPROM_ISP            0x16
#define CMD_PROGRAM_FUSE_ISP           0x17
#define CMD_READ_FUSE_ISP              0x18
#define CMD_PROGRAM_LOCK_ISP           0x19
#define CMD_READ_LOCK_ISP              0x1A
#define CMD_READ_SIGNATURE_ISP         0x1B
#define CMD_READ_OSCCAL_ISP            0x1C
#define CMD_SPI_MULTI                  0x1D


// *** PP command constants ***

#define CMD_ENTER_PROGMODE_PP          0x20
#define CMD_LEAVE_PROGMODE_PP          0x21
#define CMD_CHIP_ERASE_PP              0x22
#define CMD_PROGRAM_FLASH_PP           0x23
#define CMD_READ_FLASH_PP              0x24
#define CMD_PROGRAM_EEPROM_PP          0x25
#define CMD_READ_EEPROM_PP             0x26
```

```
#define CMD_PROGRAM_FUSE_PP          0x27
#define CMD_READ_FUSE_PP             0x28
#define CMD_PROGRAM_LOCK_PP          0x29
#define CMD_READ_LOCK_PP             0x2A
#define CMD_READ_SIGNATURE_PP        0x2B
#define CMD_READ_OSCCAL_PP           0x2C
#define CMD_SET_CONTROL_STACK        0x2D


// *** STK HVSP command constants ***

#define CMD_ENTER_PROGMODE_HVSP      0x3D
#define CMD_LEAVE_PROGMODE_HVSP      0x3E
#define CMD_CHIP_ERASE_HVSP          0x32
#define CMD_PROGRAM_FLASH_HVSP       0x33
#define CMD_READ_FLASH_HVSP          0x34
#define CMD_PROGRAM_EEPROM_HVSP      0x35
#define CMD_READ_EEPROM_HVSP         0x36
#define CMD_PROGRAM_FUSE_HVSP        0x37
#define CMD_READ_FUSE_HVSP           0x38
#define CMD_PROGRAM_LOCK_HVSP        0x39
#define CMD_READ_LOCK_HVSP           0x3A
#define CMD_READ_SIGNATURE_HVSP      0x3B
#define CMD_READ_OSCCAL_HVSP         0x3C


// *** XPROG command constants ***

#define CMD_XPROG                    0x50
#define CMD_XPROG_SETMODE            0x51


// *** AVR32 JTAG Programming command ***

#define CMD_JTAG_AVR32               0x80
#define CMD_ENTER_PROGMODE_JTAG_AVR32 0x81
#define CMD_LEAVE_PROGMODE_JTAG_AVR32 0x82


// *** AVR JTAG Programming command ***

#define CMD_JTAG_AVR                 0x90

// *** Status constants ***

// Success
```

```
#define STATUS_CMD_OK                0x00

// Warnings
#define STATUS_CMD_TOUT              0x80
#define STATUS_RDY_BSY_TOUT          0x81
#define STATUS_SET_PARAM_MISSING     0x82

// Errors
#define STATUS_CMD_FAILED            0xC0
#define STATUS_CMD_UNKNOWN           0xC9
#define STATUS_CMD_ILLEGAL_PARAMETER 0xCA

// Status
#define STATUS_CONN_FAIL_MOSI        0x01
#define STATUS_CONN_FAIL_RST         0x02
#define STATUS_CONN_FAIL_SCK         0x04
#define STATUS_TGT_NOT_DETECTED      0x00
#define STATUS_ISP_READY             0x10
#define STATUS_TGT_REVERSE_INSERTED  0x20

// hw_status
// Bits in status variable
// Bit 0-3: Slave MCU
// Bit 4-7: Master MCU

#define STATUS_AREF_ERROR            0
// Set to '1' if AREF is short circuited

#define STATUS_VTG_ERROR4
// Set to '1' if VTG is short circuited

#define STATUS_RC_CARD_ERROR         5
// Set to '1' if board id changes when board is powered

#define STATUS_PROGMODE              6
// Set to '1' if board is in programming mode

#define STATUS_POWER_SURGE           7
// Set to '1' if board draws excessive current


// *** Parameter constants ***

#define PARAM_HW_VER                        0x90
#define PARAM_SW_MAJOR                      0x91
#define PARAM_SW_MINOR                      0x92
```

```
#define PARAM_VTARGET                       0x94
#define PARAM_RESET_POLARITY                0x9E
#define PARAM_CONTROLLER_INIT               0x9F
#define PARAM_STATUS_TGT_CONN               0xA1
#define PARAM_DISCHARGEDELAY                0xA4
#define PARAM_SOCKETCARD_ID                 0xA5
#define PARAM_ROUTINGCARD_ID                0xA6
#define PARAM_EXPCARD_ID                    0xA7
#define PARAM_SW_MAJOR_SLAVE1               0xA8
#define PARAM_SW_MINOR_SLAVE1               0xA9
#define PARAM_SW_MAJOR_SLAVE2               0xAA
#define PARAM_SW_MINOR_SLAVE2               0xAB
#define PARAM_BOARD_ID_STATUS               0xAD
#define PARAM_RESET                         0xB4

#define PARAM_JTAG_ALLOW_FULL_PAGE_STREAM   0x50
#define PARAM_JTAG_EEPROM_PAGE_SIZE         0x52
#define PARAM_JTAG_DAISY_BITS_BEFORE        0x53
#define PARAM_JTAG_DAISY_BITS_AFTER         0x54
#define PARAM_JTAG_DAISY_UNITS_BEFORE       0x55
#define PARAM_JTAG_DAISY_UNITS_AFTER        0x56


// *** Parameter constants for 2 byte values ***

#define PARAM2_SCK_DURATION                 0xC0
#define PARAM2_CLOCK_CONF                   0xC1
#define PARAM2_AREF0                        0xC2
#define PARAM2_AREF1                        0xC3

#define PARAM2_JTAG_FLASH_SIZE_H            0xC5
#define PARAM2_JTAG_FLASH_SIZE_L            0xC6
#define PARAM2_JTAG_FLASH_PAGE_SIZE         0xC7
#define PARAM2_RC_ID_TABLE_REV              0xC8
#define PARAM2_EC_ID_TABLE_REV              0xC9
```

A.2 XPROG Commands and Parameters

```
// XPROMG commands
#define XPRG_CMD_ENTER_PROGMODE             0x01
#define XPRG_CMD_LEAVE_PROGMODE             0x02
#define XPRG_CMD_ERASE                      0x03
#define XPRG_CMD_WRITE_MEM                  0x04
#define XPRG_CMD_READ_MEM                   0x05
#define XPRG_CMD_CRC                        0x06
#define XPRG_CMD_SET_PARAM                  0x07

// Memory types
#define XPRG_MEM_TYPE_APPL                  1
#define XPRG_MEM_TYPE_BOOT                  2
#define XPRG_MEM_TYPE_EEPROM                3
#define XPRG_MEM_TYPE_FUSE                  4
#define XPRG_MEM_TYPE_LOCKBITS              5
#define XPRG_MEM_TYPE_USERSIG               6
#define XPRG_MEM_TYPE_FACTORY_CALIBRATION   7

// Erase types
#define XPRG_ERASE_CHIP                     1
#define XPRG_ERASE_APP                      2
#define XPRG_ERASE_BOOT                     3
#define XPRG_ERASE_EEPROM                   4
#define XPRG_ERASE_APP_PAGE                 5
#define XPRG_ERASE_BOOT_PAGE                6
#define XPRG_ERASE_EEPROM_PAGE              7
#define XPRG_ERASE_USERSIG                  8

// Write mode flags
#define XPRG_MEM_WRITE_ERASE                0
#define XPRG_MEM_WRITE_WRITE                1

// CRC types
#define XPRG_CRC_APP                        1
#define XPRG_CRC_BOOT                       2
#define XPRG_CRC_FLASH                      3

// Error codes
#define XPRG_ERR_OK                         0
#define XPRG_ERR_FAILED                     1
#define XPRG_ERR_COLLISION                  2
#define XPRG_ERR_TIMEOUT                    3

// XPROG parameters of different sizes
```

```
// 4-byte address
#define XPRG_PARAM_NVMBASE          0x01
// 2-byte page size
#define XPRG_PARAM_EEPPAGESIZE      0x02
```

A.3 USB Descriptors

Table 13-1. Device descriptor

Name	Value	Hex
bLength	Valid	0x12
bDescriptorType	DEVICE	0x01
bcdUSB	2.0	0x0200
bDeviceClass	Vendor-specific	0xFF
bDeviceSubClass	Vendor-specific	0x00
bDeviceProtocol	None	0x00
bMaxPacketSize0	64	0x40
idVendor	Atmel Corporation	0x03EB
idProduct	0x2106	0x2106
bcdDevice	2.0	0x0200
iManufacturer	1	0x01
iProduct	2 "STK600"	0x02
iSerialNumber	3	0x03
bNumConfigurations	1	0x01

Table 13-2. Configuration descriptor

Name	Value	Hex
bLength	Valid	0x09
bDescriptorType	CONFIGURATION	0x02
wTotalLength	32 bytes	0x0020
bNumInterface	1	0x01
bConfigurationValue	1	0x01
iConfiguration	0	0x00
bmAttributes	0x80	0x80
bMaxPower	500 mA	0xFA

Table 13-3. Interface descriptor

Name	Value	Hex
bLength	Valid	0x09
bDescriptorType	INTERFACE	0x04
bInterfaceNumber	0	0x00
bAlternateSetting	0	0x00
bNumEndpoints	2	0x02
bInterfaceClass	Vendor-specific	0xFF
bInterfaceSubClass	Vendor-specific	0x00

Name	Value	Hex
bInterfaceProtocol	None	0x00
iInterface	0	0x00

Table 13-4. Endpoint descriptor

Name	Value	Hex
bLength	Valid	0x07
bDescriptorType	ENDPOINT	0x05
bEndpointAddress	3 IN	0x83
bmAttributes	Types - Pkt Size Adjust : No	0x02
wMaxPacketSize	64 bytes	0x0040
bInterval	Ignored for Bulk endpoints	0x0A

Table 13-5. Endpoint descriptor

Name	Value	Hex
bLength	Valid	0x07
bDescriptorType	ENDPOINT	0x05
bEndpointAddress	2 OUT	0x02
bmAttributes	Types - Pkt Size Adjust:No	0x02
wMaxPacketSize	64 bytes	0x0040
bInterval	Ignored for Bulk endpoints	0x00

14 Table of Contents

Features *1*
1 Introduction *1*
2 Overview *2*
2.1 USB Communication 2
2.2 Packet Format 2
2.3 USB Driver 2
2.4 Command format 2
3 General Commands *2*
3.1 CMD_SIGN_ON 2
3.2 CMD_SET_PARAMETER 3
3.3 CMD_GET_PARAMETER 3
3.4 CMD_OSCCAL 4
3.5 CMD_LOAD_ADDRESS 4
3.6 CMD_FIRMWARE_UPGRADE 5
3.7 CMD_LOAD_RC_ID_TABLE 5
3.8 CMD_LOAD_EC_ID_TABLE 6
3.9 CMD_CHECK_TARGET_CONNECTION 7
4 ISP Programming Commands *7*
4.1 CMD_ENTER_PROGMODE_ISP 8
4.2 CMD_LEAVE_PROGMODE_ISP 8
4.3 CMD_CHIP_ERASE_ISP 9
4.4 CMD_PROGRAM_FLASH_ISP 9
4.5 CMD_READ_FLASH_ISP 11
4.6 CMD_PROGRAM_EEPROM_ISP 12
4.7 CMD_READ_EEPROM_ISP 12
4.8 CMD_PROGRAM_FUSE_ISP 12
4.9 CMD_READ_FUSE_ISP 12
4.10 CMD_PROGRAM_LOCK_ISP 13
4.11 CMD_READ_LOCK_ISP 13
4.12 CMD_READ_SIGNATURE_ISP 13
4.13 CMD_READ_OSCCAL_ISP 13
4.14 CMD_SPI_MULTI 13
5 Parallel Programming Mode Commands *14*
5.1 CMD_ENTER_PROGMODE_PP 14

5.2 CMD_LEAVE_PROGMODE_PP 15
5.3 CMD_CHIP_ERASE_PP 15
5.4 CMD_PROGRAM_FLASH_PP 16
5.5 CMD_READ_FLASH_PP 17
5.6 CMD_PROGRAM_EEPROM_PP 18
5.7 CMD_READ_EEPROM_PP 18
5.8 CMD_PROGRAM_FUSE_PP 18
5.9 CMD_READ_FUSE_PP 19
5.10 CMD_PROGRAM_LOCK_PP 19
5.11 CMD_READ_LOCK_PP 19
5.12 CMD_READ_SIGNATURE_PP 19
5.13 CMD_READ_OSCCAL_PP 19
5.14 CMD_SET_CONTROL_STACK 20
6 High Voltage Serial Programming Commands 20
6.1 CMD_ENTER_PROGMODE_HVSP 20
6.2 CMD_LEAVE_PROGMODE_HVSP 21
6.3 CMD_CHIP_ERASE_HVSP 21
6.4 CMD_PROGRAM_FLASH_HVSP 22
6.5 CMD_READ_FLASH_HVSP 23
6.6 CMD_PROGRAM_EEPROM_HVSP 24
6.7 CMD_READ_EEPROM_HVSP 24
6.8 CMD_PROGRAM_FUSE_HVSP 24
6.9 CMD_READ_FUSE_HVSP 24
6.10 CMD_PROGRAM_LOCK_HVSP 25
6.11 CMD_READ_LOCK_HVSP 25
6.12 CMD_READ_SIGNATURE_HVSP 25
6.13 CMD_READ_OSCCAL_HVSP 25
7 AVR8 JTAG 25
7.1 Memory Read / Write 26
7.1.1 CMND_READ_MEMORY 26
7.1.2 CMND_WRITE_MEMORY 27
7.1.3 CMND_RESET 27
7.1.4 CMND_ENTER_PROGMODE 28
7.1.5 CMND_LEAVE_PROGMODE 28
7.1.6 CMND_CHIP_ERASE 28
8 AVR32 JTAG 29
8.1 CMD_ENTER_PROGMODE_JTAG_AVR32 29
8.2 CMD_LEAVE_PROGMODE_JTAG_AVR32 29

8.3 CMD_RESET_AVR32 30
8.4 CMD_SAB_WRITE_AVR32 30
8.5 CMD_SAB_READ_AVR32 30
8.6 CMD_BLOCK_WRITE_AVR32 31
8.7 CMD_ BLOCK_READ_AVR32 31
8.8 CMD_NEXUS_WRITE_AVR32 32
8.9 CMD_NEXUS_READ_AVR32 32
9 XPROG protocol 33
9.1 CMD_XPROG_SETMODE 33
9.2 CMD_XPROG 33
9.2.1 XPRG_ENTER_PROGMODE 34
9.2.2 XPRG_LEAVE_PROGMODE 34
9.2.3 XPRG_SET_PARAMETER 34
9.2.4 XPRG_ERASE 35
9.2.5 XPRG_WRITE_MEM 36
9.2.6 XPRG_READ_MEM 37
9.2.7 XPRG_READ_CRC 38
10 Return Values 39
10.1 Success 39
10.2 Warnings 39
10.3 Errors 39
11 Parameters 39
11.1 PARAM_HW_VER 40
11.2 PARAM_SW_MAJOR 41
11.3 PARAM_SW_MINOR 41
11.4 PARAM_VTARGET 41
11.5 PARAM_STATUS_TGT_CONN 41
11.6 PARAM2_SCK_DURATION 41
11.7 PARAM_CONTROLLER_INIT 41
11.8 PARAM_DISCHARGEDELAY 42
11.9 PARAM2_AREF0 42
11.10 PARAM2_AREF1 42
11.11 PARAM2_CLOCK_CONF 42
11.12 PARAM_SOCKETCARD_ID 43
11.13 PARAM_ROUTINGCARD_ID 43
11.14 PARAM_EXPCARD_ID 43
11.15 PARAM_SW_MAJOR_SLAVE1 43
11.16 PARAM_SW_MINOR_SLAVE1 43

11.17 PARAM_SW_MAJOR_SLAVE2 43
11.18 PARAM_SW_MINOR_SLAVE2 43
11.19 PARAM2_RC_ID_TABLE_REV 43
11.20 PARAM2_EC_ID_TABLE_REV 43
11.21 PARAM_BOARD_ID_STATUS 44
11.22 PARAM_RESET 44
11.23 PARAM_RESET_POLARITY 44
11.24 PARAM_JTAG_ALLOW_FULL_PAGE_STREAM 44
11.25 PARAM_JTAG_EEPROM_PAGE_SIZE 44
11.26 PARAM2_JTAG_FLASH_PAGE_SIZE 44
11.27 PARAM2_JTAG_FLASH_SIZE_H 44
11.28 PARAM2_JTAG_FLASH_SIZE_L 44
11.29 PARAM_JTAG_DAISY_BITS_BEFORE 44
11.30 PARAM_JTAG_DAISY_BITS_AFTER 45
11.31 PARAM_JTAG_DAISY_UNITS_BEFORE 45
11.32 PARAM_JTAG_DAISY_UNITS_AFTER 45
12 XML Parameter Values 45
13 Command Sequence Example 46
13.1 Connect 47
13.2 Read Signature 47
13.3 STK600 Communication Logging 47
A.1 Commands and parameters 50
A.2 XPROG Commands and Parameters 54
A.3 USB Descriptors 55
14 Table of Contents 57
Disclaimer 61



Headquarters

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

International

Atmel Asia
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimshatsui
East Kowloon
Hong Kong
Tel: (852) 2721-9778
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
avr@atmel.com

Sales Contact
www.atmel.com/contacts

Literature Request
www.atmel.com/literature

Disclaimer: The information in this document is provided in connection with Atmel products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Atmel products. **EXCEPT AS SET FORTH IN ATMEL'S TERMS AND CONDITIONS OF SALE LOCATED ON ATMEL'S WEB SITE, ATMEL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ATMEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.** Atmel makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and product descriptions at any time without notice. Atmel does not make any commitment to update the information contained herein. Unless specifically provided otherwise, Atmel products are not suitable for, and shall not be used in, automotive applications. Atmel's products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.

© 2008 Atmel Corporation. All rights reserved. Atmel®, logo and combinations thereof, AVR®, AVR Studio®, STK®, and others, are the registered trademarks or trademarks of Atmel Corporation or its subsidiaries. Other terms and product names may be trademarks of others.

8133A-AVR-04/08