AVR061: STK500 Communication Protocol

Introduction

This document describes the protocol for the STK500 starterkit. This protocol is based on earlier protocols made for other AVR tools and is fully compatible with them in that there should not be any overlapping or redefined commands.

The following sections describes each part of the protocol in detail. All commands (both commands and responses) are standard ASCII characters between 0x00 - 0x7F. Data can be any character with value between 0x00 - 0xFF.

The definition of all commands, responses, parameters and other defined values can be found in the file "command.h". The device codes can be found in the file "devices.h". These files are located in the software section on Atmel web site, www.atmel.com.

At the end of this document, there is an overview of device codes, signatures, Fuse bits and Lock bits for each currently supported device.

8-bit AVR®

STK500 Communication Protocol

Application Note

Rev. 2525B–AVR–04/03

Response Definitions

This section describes the meaning of the valid responses from the STK500 starterkit. Some of the responses will be part of more complex responses also containing data, depending on the command.

OK

Response Resp_STK_OK is sent after a valid command has been executed.

Failed

Response Resp_STK_FAILED is sent if a command execution fails.

Insync

Response Resp_STK_INSYNC is sent after Sync_CRC_EOP has been received.

Nosync

Response Resp_STK_NOSYNC is sent if Sync_CRC_EOP is not received after a command.

Unknown

Response Resp_STK_UNKNOWN is sent as a response to an unknown command, if the unknown command is directly followed by Sync_CRC_EOP.

Nodevice

Response Resp_STK_NODEVICE is sent as a response to a Cmnd_STK_ENTER_PROGMODE command if the proper device parameters has not been set with Cmnd_STK_SET_DEVICE first.

Command Definitions

This section defines the commands that are understood by the STK500 starterkit.

Check if Starterkit Present

The PC sends this command to check if the starterkit is present on the communication channel.

Command

Cmnd_STK_GET_SIGN_ON, Sync_CRC_EOP

Command Value

0x31

Response

Resp_STK_INSYNC, sign_on_message, Resp_STK_OK

or

Resp_STK_NOSYNC (If no Sync_CRC_EOP received)

Table 1. Parameters

Parameter Name	Field Usage	Field Format
sign_on_message	Text string. Always “AVR STK”	7 bytes

Get Synchronization

Use this command to try to regain synchronization when sync is lost. Send this command until Resp_STK_INSYNC is received.

Command

Cmnd_STK_GET_SYNC, Sync_CRC_EOP

Command Value

0x30

Response

Resp_STK_INSYNC, Resp_STK_OK

or

Resp_STK_NOSYNC (If no Sync_CRC_EOP received)

Table 2. Parameters

Parameter Name	Field Usage	Field Format
(No parameters)	–	–

Get Parameter Value

Get the value of a valid parameter from the STK500 starterkit. If the parameter is not used, the same parameter will be returned together with a Resp_STK_FAILED response to indicate the error. See the parameters section for valid parameters and their meaning.

Command

Cmnd_STK_GET_PARAMETER, parameter, Sync_CRC_EOP

Command Value

0x41

Response

Resp_STK_INSYNC, value, Resp_STK_OK

or

Resp_STK_INSYNC, parameter, Resp_STK_FAILED

or

Resp_STK_NOSYNC (If no Sync_CRC_EOP received)

Table 3. Parameters

Parameter Name	Field Usage	Field Format
parameter	Any valid parameter as defined in “command.h”	1 byte (0x00 - 0x7F)
value	Any 8-bit value	1 byte (0x00 - 0xFF)

Set Parameter Value

Set the value of a valid parameter in the STK500 starterkit. See the parameters section for valid parameters and their meaning.

Command

Cmnd_STK_SET_PARAMETER, parameter, value, Sync_CRC_EOP

Command Value

0x40

Response

Resp_STK_INSYNC, Resp_STK_OK

or

Resp_STK_INSYNC, parameter, Resp_STK_FAILED

or

Resp_STK_NOSYNC (If no Sync_CRC_EOP received)

Table 4. Parameters

Parameter Name	Field Usage	Field Format
parameter	Any valid parameter as defined in COMMON.H	1 byte (0x00 - 0x7F)
value	Any 8-bit value	1 byte (0x00 - 0xFF)

Set Device Programming Parameters

Set the device Programming parameters for the current device. These parameters must be set before the starterkit can enter Programming mode.

Command

Cmnd_STK_SET_DEVICE, devicecode, revision, progtype, parmode, polling, selftimed, lockbytes, fusebytes, flashpollval1, flashpollval2, eeprompollval1, eeprompollval2, pagesizehigh, pagesizelow, eepromsizehigh, eepromsizelow, flashsize4, flashsize3, flashsize2, flashsize1, Sync_CRC_EOP

Command Value

0x42

Response

Resp_STK_INSYNC, Resp_STK_OK

or

Resp_STK_NOSYNC (If no Sync_CRC_EOP received)

Table 5. Parameters

Parameter Name	Field Usage	Field Format
devicecode	Device code as defined in “devices.h”	1 byte
revision	Device revision. Currently not used. Should be set to 0.	1 byte
progtype	Defines which Program modes is supported: “0” – Both Parallel/High-voltage and Serial mode “1” – Only Parallel/High-voltage mode	1 byte
parmode	Defines if the device has a full parallel interface or a pseudo parallel programming interface: “0” – Pseudo parallel interface “1” – Full parallel interface	1 byte
polling	Defines if polling may be used during SPI access: “0” – No polling may be used “1” – Polling may be used	1 byte
selftimed	Defines if programming instructions are self timed: “0” – Not self timed “1” – Self timed	1 byte
lockbytes	Number of Lock bytes. Currently not used. Should be set to actual number of Lock bytes for future compability.	1 byte
fusebytes	Number of Fuse bytes. Currently not used. Should be set to actual number of Fuse bytes for future caompability.	1 byte
flashpollval1	FLASH polling value. See Data Sheet for the device.	1 byte
flashpollval2	FLASH polling value. Same as “flashpollval1”	1 byte
eeprompollval1	EEPROM polling value 1 (P1). See data sheet for the device.	1 byte
eeprompollval2	EEPROM polling value 2 (P2). See data sheet for device.	1 byte
pagesizehigh	Page size in bytes for pagemode parts, High Byte of 16-bit value.	1 byte
pagesizelow	Page size in bytes for pagemode parts, Low Byte of 16-bit value.	1 byte
eepromsizehigh	EEPROM size in bytes, High Byte of 16-bit value.	1 byte
eepromsizelow	EEPROM size in bytes, Low Byte of 16-bit value.	1 byte

Table 5. Parameters (Continued)

Parameter Name	Field Usage	Field Format
flashsize4	FLASH size in bytes, byte 4 (High Byte) of 32-bit value.	1 byte
flashsize3	FLASH size in bytes, byte 3 of 32-bit value.	1 byte
flashsize2	FLASH size in bytes, byte 2 of 32-bit value.	1 byte
flashsize1	FLASH size in bytes, byte 1 (Low Byte) of 32-bit value.	1 byte

Set Extended Device Programming Parameters

Set extended programming parameters for the current device.

Command

Cmnd_SET_DEVICE_EXT, commandsize, eeprompagesize, signalpagel, signalbs2, Synch_CRC_EOP

Command Value

0x45

Response

Resp_STK_INSYNC, Resp_STK_OK

or

Resp_STK_NOSYNC (if no Sync_CRC_EOP received)

Table 6. Parameters

Parameter Name	Field Usage	Field Format
commandsize	Defines how many bytes of additional parameters the command contains. In this case it's value should be 4 (for the eepromsize, signalpagel and signalbs2 parameters). The STK500 may accept more parameters in later revisions.	1 byte
eeprompagesize	EEPROM page size in bytes.	1 byte
signalpagel	Defines to which port pin the PAGEL signal should be mapped. Example: signalpagel = 0xD7. In this case PAGEL should be mapped to PORTD7.	1 byte
signalbs2	Defines to which port pin the BS2 signal should be mapped. See signalpagel.	1 byte
ResetDisable	Defines whether a part has RSTDSBL Fuse (value = 1) or not (value = 0).	1 byte

Enter Program Mode

Enter Programming mode for the selected device. The Programming mode and device programming parameters must have been set by Cmnd_STK_SET_DEVICE prior to calling this command, or the command will fail with a Resp_STK_NODEVICE response.

Command

Cmnd_STK_ENTER_PROGMODE, Sync_CRC_EOP

Command Value

0x50

Response

Resp_STK_INSYNC, Resp_STK_OK

or

Resp_STK_INSYNC, Resp_STK_NODEVICE

or

Resp_STK_NOSYNC (If no Sync_CRC_EOP received)

Table 7. Parameters

Parameter Name	Field Usage	Field Format
(No parameter)	–	–

Leave Program Mode

Leave programming mode.

Command

Cmnd_STK_LEAVE_PROGMODE, Sync_CRC_EOP

Command Value

0x51

Response

Resp_STK_INSYNC, Resp_STK_OK

or

Resp_STK_NOSYNC (If no Sync_CRC_EOP received)

Parameters:

Table 8. Parameters

Parameter Name	Field Usage	Field Format
(No parameter)	–	–

Chip Erase

Erase device.

Command

Cmnd_STK_CHIP_ERASE, Sync_CRC_EOP

Command Value

0x52

Response

Resp_STK_INSYNC, Resp_STK_OK

or

Resp_STK_NOSYNC (If no Sync_CRC_EOP received)

Table 9. Parameters

Parameter Name	Field Usage	Field Format
(No parameter)	–	–

Check for Address Autoincrement

Check if the write/read address is automatically incremented while using the Cmnd_STK_PROG/READ_FLASH/EEPROM commands. Since STK500 always auto-increments the address, this command will always be successful.

Command Cmnd_STK_CHECK_AUTOINC, Sync_CRC_EOP

Command Value 0x53

Response Resp_STK_INSYNC, Resp_STK_OK

or

Resp_STK_NOSYNC (If no Sync_CRC_EOP received)

Table 10. Parameters

Parameter Name	Field Usage	Field Format
(No parameter)	–	–

Load Address

Load 16-bit address down to starterkit. This command is used to set the address for the next read or write operation to FLASH or EEPROM. Must always be used prior to Cmnd_STK_PROG_PAGE or Cmnd_STK_READ_PAGE.

Command Cmnd_STK_LOAD_ADDRESS, addr_low, addr_high, Sync_CRC_EOP

Command Value 0x55

Response Resp_STK_INSYNC, Resp_STK_OK

or

Resp_STK_NOSYNC (If no Sync_CRC_EOP received)

Table 11. Parameters

Parameter Name	Field Usage	Field Format
addr_low	LSB byte of address	1 byte
addr_high	MSB byte of address	1 byte

Program Flash Memory

Program one word in FLASH memory.

Command Cmnd_STK_PROG_FLASH, flash_low, flash_high, Sync_CRC_EOP

Command Value 0x60

Response Resp_STK_INSYNC, Resp_STK_OK

or

Resp_STK_NOSYNC (If no Sync_CRC_EOP received)

Table 12. Parameters

Parameter Name	Field Usage	Field Format
flash_low	Low byte of word to program	1 byte
flash_high	High byte of word to program	1 byte

Program Data Memory

Program one byte in EEPROM memory.

Command

Cmnd_STK_PROG_DATA, data, Sync_CRC_EOP

Command Value

0x61

Response

Resp_STK_INSYNC, Resp_STK_OK

or

Resp_STK_NOSYNC (If no Sync_CRC_EOP received)

Table 13. Parameters

Parameter Name	Field Usage	Field Format
data	Byte to program	1 byte

Program Fuse Bits

Program Fuse bits. The Fuse bit mapping for the currently supported devices is described in the appendix.

Note: For ISP Programming, use the Cmnd_STK_UNIVERSAL command with the appropriate ISP command bytes (found in the device data sheet).

Command

Cmnd_STK_PROG_FUSE, fuse_low, fuse_high, Sync_CRC_EOP

Command Value

0x62

Response

Resp_STK_INSYNC, Resp_STK_OK

or

Resp_STK_NOSYNC (If no Sync_CRC_EOP received)

Table 14. Parameters

Parameter Name	Field Usage	Field Format
fuse_low	Low byte of Fuse bits	1 byte
fuse_high	High byte of Fuse bits	1 byte

Program Fuse Bits Extended

Program Extended Fuse bits. The Fuse bit mapping for the currently supported devices is described in the appendix.

Note: For ISP Programming, use the Cmnd_STK_UNIVERSAL command with the appropriate ISP command bytes (found in the device data sheet).

Command

Cmnd_STK_PROG_FUSE_EXT, fuse_low, fuse_high, fuse_ext, Sync_CRC_EOP

Command Value

0x65

Response

Resp_STK_INSYNC, Resp_STK_OK

or

Resp_STK_NOSYNC (If no Sync_CRC_EOP received)

Table 15. Parameters

Parameter Name	Field Usage	Field Format
fuse_low	Low byte of Fuse bits	1 byte
fuse_high	High byte of Fuse bits	1 byte
fuse_ext	Extended byte of Fuse bits	1 byte

Program Lock Bits

Program Lock bits. The Lock bit mapping for the currently supported devices is described in the appendix.

Note: For ISP Programming, use the Cmnd_STK_UNIVERSAL command with the appropriate ISP command bytes (found in the device data sheet).

Command

Cmnd_STK_PROG_LOCK, Synch_CRC_EOP

Command Value

0x63

Response

Resp_STK_INSYNC, Resp_STK_OK

or

Resp_STK_NOSYNC (If no Sync_CRC_EOP received)

Table 16. Parameters

Parameter Name	Field Usage	Field Format
lock	Lock bits as defined by datasheets	[BYTE]

Program Page

Download a block of data to the starterkit and program it in FLASH or EEPROM of the current device. The data block size should not be larger than 256 bytes.

Command

Cmnd_STK_PROG_PAGE, bytes_high, bytes_low, memtype, data, Sync_CRC_EOP

Command Value

0x64

Response

Resp_STK_INSYNC, Resp_STK_OK

or

Resp_STK_NOSYNC (If no Sync_CRC_EOP received)

Table 17. Parameters

Parameter Name	Field Usage	Field Format
bytes_high	High byte of data block size	1 byte
bytes_low	Low byte of data block size	1 byte
memtype	Memory type: “E” – EEPROM, “F” – FLASH	1 byte
data	Data to program into FLASH or EEPROM. If programming to FLASH, the sequence of data is lowbyte:highbyte.	((bytes_high << 8) | bytes_low) bytes.

Read Flash Memory

Read one word from FLASH memory.

Command

Cmnd_STK_READ_FLASH, Sync_CRC_EOP

Command Value

0x70

Response

Resp_STK_INSYNC, flash_low, flash_high, Resp_STK_OK

or

Resp_STK_NOSYNC (If no Sync_CRC_EOP received)

Table 18. Parameters

Parameter Name	Field Usage	Field Format
flash_low	Low byte	1 byte
flash_high	High byte	1 byte

Read Data Memory

Read one byte from EEPROM memory.

Command

Cmnd_STK_READ_DATA, Sync_CRC_EOP

Command Value

0x71

Response

Resp_STK_INSYNC, data, Resp_STK_OK

or

Resp_STK_NOSYNC (If no Sync_CRC_EOP received)

Table 19. Parameters

Parameter Name	Field Usage	Field Format
data	Data byte	1 byte

Read Fuse Bits

Read Fuse bits. The Fuse bit mapping for the currently supported devices is described in the appendix.

Note: For ISP Programming, use the Cmnd_STK_UNIVERSAL command with the appropriate ISP command bytes (found in the device data sheet).

Note that some devices combine Lock bits and Fuse bits in way that actually requires using the Cmnd_STK_READ_LOCK to retrieve the Fuse byte(s). This is described closer in the appendix.

Command

Cmnd_STK_READ_FUSE, Sync_CRC_EOP

Command Value

0x72

Response

Resp_STK_INSYNC, fuse_low, fuse_high, Resp_STK_OK

or

Resp_STK_NOSYNC (If no Sync_CRC_EOP received)

Table 20. Parameters

Parameter Name	Field Usage	Field Format
fuse_low	Low byte of Fuse bits	1 byte
fuse_high	High byte of Fuse bits	1 byte

Read Fuse Bits Extended

Read Extended Fuse bits. The Fuse bit mapping for the currently supported devices is described in the appendix.

Note: For ISP Programming, use the Cmnd_STK_UNIVERSAL command with the appropriate ISP command bytes (found in the device data sheet).

Note that some devices combine Lock bits and Fuse bits in way that actually requires using the Cmnd_STK_READ_LOCK to retrieve the Fuse byte(s). This is described closer in the appendix.

Command

Cmnd_STK_READ_FUSE_EXT

Command Value

0x77

Response

Resp_STK_INSYNC, fuse_low, fuse_high, fuse_ext, Resp_STK_OK

or

Resp_STK_NOSYNC (If no Sync_CRC_EOP received)

Table 21. Parameters

Parameter Name	Field Usage	Field Format
fuse_low	Low byte of Fuse bits	1 byte
fuse_high	High byte of Fuse bits	1 byte
fuse_ext	Extended byte of Fuse bits	1 byte

Read Lock Bits

Read Lock bits. The Lock bit mapping for the currently supported devices is described in the appendix.

Note: For ISP Programming, use the Cmnd_STK_UNIVERSAL command with the appropriate ISP command bytes (found in the device data sheet).

Note that some device combine Lock bits and Fuse bits in way that requires some modification of the read-back fuse before it can be interpreted normally. This is described closer in the appendix.

Command

Cmnd_STK_READ_LOCK, Sync_CRC_EOP

Command Value

0x73

Response

Resp_STK_INSYNC, lock, Resp_STK_OK

or

Resp_STK_NOSYNC (If no Sync_CRC_EOP received)

Table 22. Parameters

Parameter Name	Field Usage	Field Format
lock	Lock bits	1 byte

Read Page

Read a block of data from FLASH or EEPROM of the current device. The data block size should not be larger than 256 bytes.

Command

Cmnd_STK_READ_PAGE, bytes_high, bytes_low, memtype, Sync_CRC_EOP

Command Value

0x74

Response

Resp_STK_INSYNC, data, Resp_STK_OK

or

Resp_STK_NOSYNC (If no Sync_CRC_EOP received)

Table 23. Parameters

Parameter Name	Field Usage	Field Format
bytes_high	High byte of data block size	1 byte
bytes_low	Low byte of data block size	1 byte
memtype	Memory type: “E” – EEPROM, “F” – FLASH	1 byte
data	Data to program into FLASH or EEPROM. If programming to FLASH, the sequence of data is lowbyte:highbyte.	((bytes_high << 8) | bytes_low) bytes.

Read Signature Bytes

Read signature bytes.

Command

Cmnd_STK_READ_SIGN, Sync_CRC_EOP

Command Value

0x75

Response

Resp_STK_INSYNC, sign_high, sign_middle, sign_low, Resp_STK_OK

or

Resp_STK_NOSYNC (If no Sync_CRC_EOP received)

Table 24. Parameters

Parameter Name	Field Usage	Field Format
sigh_high	High byte of signature	1 byte
sign_middle	Middle byte of signature	1 byte
sign_low	Low byte of signature	1 byte

Read Oscillator Calibration Byte

Read Oscillator calibration byte.

Command

Cmnd_STK_READ_OSCCAL, Sync_CRC_EOP

Command Value

0x76

Response

Resp_STK_INSYNC, osc_cal_byte, Resp_STK_OK

or

Resp_STK_NOSYNC (If no Sync_CRC_EOP received)

Table 25. Parameters

Parameter Name	Field Usage	Field Format
osc_cal_byte	Oscillator callibration byte	1 byte

Read Oscillator Calibration Byte Extended

Read Oscillator calibration byte.

Command

Cmnd_STK_READ_OSCCAL_EXT, address, Sync_CRC_EOP

Command Value

0x78

Response

Resp_STK_INSYNC, osc_cal_byte, Resp_STK_OK

or

Resp_STK_NOSYNC (If no Sync_CRC_EOP received)

Table 26. Parameters

Parameter Name	Field Usage	Field Format
address	Address of the oscillator callibration byte to read	1 byte
osc_cal_byte	Oscillator callibration byte	1 byte

Universal Command

Universal command is used to send a generic 32-bit data/command stream directly to the SPI interface of the current device. Shifting data into the SPI interface at the same time shifts data out of the SPI interface. The response of the last eight bits that are shifted out are returned.

Currently this command is used to read and write Fuse and Lock bits in Serial/High-voltage mode. For more information, see the appendix.

Command

Cmnd_STK_UNIVERSAL, byte1, byte2, byte3, byte4, Sync_CRC_EOP

Command Value

0x56

Response

Resp_STK_INSYNC, byte4_out, Resp_STK_OK

or

Resp_STK_NOSYNC (If no Sync_CRC_EOP received)

Table 27. Parameters

Parameter Name	Field Usage	Field Format
byte1	First byte of SPI data/command	1 byte
byte2	Second byte of SPI data/command	1 byte
byte3	Third byte of SPI data/command	1 byte
byte4	Fourth byte of SPI data/command	1 byte
byte4_out	The response of the last eight bits to be shifted out from the SPI interface	1 byte

Extended Universal Command

Extended SPI universal command used to clock any number of bytes to the SPI interface of the current device.

Command

Cmnd_STK_UNIVERSAL_MULTI, number_of_bytes, data bytes, Sync_CRC_EOP.

Table 28. Parameters

Parameter Name	Field Usage	Field Format
number_of_bytes	0 - 255 equals 1 - 256 bytes to be sent	1 byte

Command Value

0x57

Response

Resp_STK_INSYNC, Resp_STK_OK

or

Resp_STK_NOSYNC (if no Sync_CRC_EOP received)

Parameter Definitions

Hardware Version

This parameter defines the version of the starterkit hardware.

Parameter

Parm_STK_HW_VER

Access: Read only

Table 29.

Value	Description
–	–

Parameter Value

0x80

Software Version Major

The major version of the starterkit MCU software.

Parameter

Parm_STK_SW_MAJOR

Access: Read Only

Table 30.

Value	Description
–	–

Parameter Value

0x81

Software Version Minor

The minor version of the starterkit MCU software.

Parameter

Parm_STK_SW_MINOR

Access: Read only

Table 31.

Value	Description
–	–

Parameter Value

0x82

Status LED

Sets or retrieves the current setting of the status LED on the starterkit. The status LED on the starterkit is a dual LED which can emit red light, green light or both (yellow).

Parameter

Parm_STK_LEDS

Access: Read/Write

Table 32.

Value	Description
0	Status LED off
1	Green on
2	Red on
3	Green and red both on

Parameter Value

0x83

Target Voltage (VTARGET)

Sets or retrieves the current target voltage (VTARGET). The actual voltage is a function of the parameter:

Real target voltage = [Parm_STK_VTARGET]/10.0

Parameter

Parm_STK_VTARGET

Access: Read/Write

Table 33.

Value	Description
–	–

Parameter Value

0x84

Adjustable Voltage (AREF)

Sets or retrieves the current adjustable voltage (AREF). The actual voltage is a function of the parameter:

Real adjustable voltage = [Parm_STK_VADJUST]/10.0

Parameter

Parm_STK_VADJUST

Access: Read/Write

Table 34.

Value	Description
–	–

Parameter Value

0x85

Oscillator Timer Prescaler Value

Sets or retrieves the current prescaler value for the timer in the mcu that is used to generate the adjustable Oscillator. The actual Oscillator frequency is a function of the Parm_STK_OSC_PSCALE and the Parm_STK_OSC_CMATCH parameters. See the Table 35 below.

Parameter Parm_STK_OSC_PSCALE

Access: Read/Write

Table 35.

Value	Oscillator Frequency
0	Oscillator off
1	mcu_freq/(([Parm_STK_OSC_CMATCH] + 1) * 1 * 2)
2	mcu_freq/(([Parm_STK_OSC_CMATCH] + 1) * 8 * 2)
3	mcu_freq/(([Parm_STK_OSC_CMATCH] + 1) * 32 * 2)
4	mcu_freq/(([Parm_STK_OSC_CMATCH] + 1) * 64 * 2)
5	mcu_freq/(([Parm_STK_OSC_CMATCH] + 1) * 128 * 2)
6	mcu_freq/(([Parm_STK_OSC_CMATCH] + 1) * 256 * 2)
7	mcu_freq/(([Parm_STK_OSC_CMATCH] + 1) * 1024 * 2)

Parameter Value 0x86

Oscillator Timer Compare Match Value

Sets or retrieves the current compare match value for the timer in the mcu that is used to generate the adjustable Oscillator. See the description of the Parm_STK_OSC_PSCALE parameter.

Parameter Parm_STK_OSC_CMATCH

Access: Read/Write

Table 36.

Value	Description
–	–

Parameter Value 0x87

ISP SCK Duration

Sets or retrieves the parameter for the current ISP SCK half-period duration. To ensure correct programming, this parameter should be set to the following criteria:

[Parm_STK_SCK_DURATION] > (8*10^6 / target_freq) - 2

Parameter Parm_STK_SCK_DURATION

Access: Read/Write

Table 37.

Value	Description
–	–

Parameter Value 0x89

Buffer Size

These parameters retrieve the size of the communication buffer in the starterkit MCU. The parameters form a 16-bit value, where Parm_STK_BUFSIZEH is the high byte and Parm_STK_BUFSIZEL is the Low byte.

Parameter

Parm_STK_BUFSIZEH:Parm_STK_BUFSIZEL

Access: Read only

Table 38.

Value	Description
–	–

Parameter Value

0x91:0x90

Topcard Detect

Retrieves information about whether there is a top-card mounted on the STK500 or not.

Parameter

Param_STK500_TOPCARD_DETECT

Access: Read only

Table 39.

Value	Description
0	Reserved for future use
1	STK502 detected
2	STK501 detected
3	No top-card detected

Parameter Value

0x98

Appendix

Reserved Commands

There are several commands (or command characters) that are reserved and should not be used for this product. They are either used for other AVR products or are reserved for future use. All reserved commands will return a NACK response (character “?”) to indicate that they are not used. The reserved commands are listed below without further comments:

- “%”
- “:”
- “<“
- “>”
- “I”
- “i”
- “Q”
- “q”
- “U”
- “u”
- “Z”

Device Codes

Table 40. Device Codes

Revision	A	B	C	D	E	F	G	H
ATtiny11	0x11	–	–	–	–	–	–	–
ATtiny12	0x12	–	–	–	–	–	–	–
ATtiny15	0x13	–	–	–	–	–	–	–
ATtiny22	0x20	–	–	–	–	–	–	–
ATtiny26	0x21	–	–	–	–	–	–	–
ATtiny28	0x28	–	–	–	–	–	–	–
AT90S1200	0x33	–	–	–	–	–	–	–
AT90S2313	0x40	–	–	–	–	–	–	–
AT90S2323	0x41	–	–	–	–	–	–	–
AT90S2333	0x42	–	–	–	–	–	–	–
AT90S2343	0x43	–	–	–	–	–	–	–
AT90S4414	0x50	–	–	–	–	–	–	–
AT90S4433	0x51	–	–	–	–	–	–	–
AT90S4434	0x52	–	–	–	–	–	–	–
AT90S8515	0x60	–	–	–	–	–	–	–
AT90S8535	0x61	–	–	–	–	–	–	–
ATmega8	0x62	–	–	–	–	–	–	–
ATmega8515	0x63	–	–	–	–	–	–	–
ATmega8535	0x64	–	–	–	–	–	–	–
ATmega161	0x80	–	–	–	–	–	–	–
ATmega163	0x81	–	–	–	–	–	–	–
ATmega16	0x82	–	–	–	–	–	–	–
ATmega162	0x83	–	–	–	–	–	–	–
ATmega169	0x84	–	–	–	–	–	–	–
ATmega323	0x90	–	–	–	–	–	–	–
ATmega32	0x91	–	–	–	–	–	–	–
ATmega64	0xA0	–	–	–	–	–	–	–
ATmega103	0xB1	–	–	–	–	–	–	–
ATmega128	0xB2	–	–	–	–	–	–	–
AT89551	0xE1	–	–	–	–	–	–	–
AT89552	0xE2	–	–	–	–	–	–	–
AT86RF401	0xD0	–	–	–	–	–	–	–

Signature Bytes

Table 41. Signature Bytes

Address	–	–	–	–	–	0x002	0x001	0x000
ATtiny11	–	–	–	–	–	0x05	0x90	0x1E
ATtiny12	–	–	–	–	–	0x05	0x90	0x1E
ATtiny15	–	–	–	–	–	0x06	0x90	0x1E
ATtiny22	–	–	–	–	–	0x06	0x91	0x1E
ATtiny26	–	–	–	–	–	0x09	0x91	0x1E
ATtiny28	–	–	–	–	–	0x07	0x91	0x1E
AT90S1200	–	–	–	–	–	0x01	0x90	0x1E
AT90S2313	–	–	–	–	–	0x01	0x91	0x1E
AT90S2323	–	–	–	–	–	0x02	0x91	0x1E
AT90S2333	–	–	–	–	–	0x05	0x91	0x1E
AT90S2343	–	–	–	–	–	0x03	0x91	0x1E
AT90S4414	–	–	–	–	–	0x01	0x92	0x1E
AT90S4433	–	–	–	–	–	0x03	0x92	0x1E
AT90S4434	–	–	–	–	–	0x02	0x92	0x1E
AT90S8515	–	–	–	–	–	0x01	0x93	0x1E
AT90S8535	–	–	–	–	–	0x03	0x93	0x1E
ATmega8	–	–	–	–	–	0x07	0x93	0x1E
ATmega8515	–	–	–	–	–	0x06	0x93	0x1E
ATmega8535	–	–	–	–	–	0x08	0x93	0x1E
ATmega161	–	–	–	–	–	0x01	0x94	0x1E
ATmega163	–	–	–	–	–	0x02	0x94	0x1E
ATmega16	–	–	–	–	–	0x03	0x94	0x1E
ATmega162	–	–	–	–	–	0x04	0x94	0x1E
ATmega169	–	–	–	–	–	0x05	0x94	0x1E
ATmega323	–	–	–	–	–	0x01	0x95	0x1E
ATmega32	–	–	–	–	–	0x02	0x95	0x1E
ATmega64	–	–	–	–	–	0x02	0x96	0x1E
ATmega103	–	–	–	–	–	0x01	0x97	0x1E
ATmega128	–	–	–	–	–	0x02	0x97	0x1E
AT89551	–	–	–	–	–	0x06	0x51	0x1E
AT89552	–	–	–	–	–	0x06	0x52	0x1E
AT86RF401	–	–	–	–	–	0x81	0x91	0x1E

Lock Bits

For explanation of the Lock bits, read the data sheet for the particular device. Unprogrammed bits should be set to “1”.

Table 42. Lock Bits

Lock Bits	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ATtiny11	1	1	1	1	1	LB2	LB1	1
ATtiny12	1	1	1	1	1	LB2	LB1	1
ATtiny15	1	1	1	1	1	LB2	LB1	1
ATtiny22	1	1	1	1	1	LB2	LB1	1
ATtiny26	1	1	1	1	1	LB2	LB1	1
ATtiny28	1	1	1	1	1	LB2	LB1	1
AT90S1200	1	1	1	1	1	LB1	LB2	1
AT90S2313	1	1	1	1	1	LB1	LB2	1
AT90S2323	1	1	1	1	1	LB2	LB1	1
AT90S2333	1	1	1	1	1	LB2	LB1	1
AT90S2343	1	1	1	1	1	LB2	LB1	1
AT90S4414	1	1	1	1	1	LB2	LB1	1
AT90S4433	1	1	1	1	1	LB2	LB1	1
AT90S4434	1	1	1	1	1	LB2	LB1	1
AT90S8515	1	1	1	1	1	LB2	LB1	1
AT90S8535	1	1	1	1	1	LB2	LB1	1
ATmega8	1	1	BLB12	BLB11	BLB02	BLB01	LB2	LB1
ATmega8515	1	1	BLB12	BLB11	BLB02	BLB01	LB2	LB1
ATmega8535	1	1	BLB12	BLB11	BLB02	BLB01	LB2	LB1
ATmega161	1	1	BLB12	BLB11	BLB02	BLB01	LB2	LB1
ATmega163	1	1	BLB12	BLB11	BLB02	BLB01	LB2	LB1
ATmega16	1	1	BLB12	BLB11	BLB02	BLB01	LB2	LB1
ATmega162	1	1	BLB12	BLB11	BLB02	BLB01	LB2	LB1
ATmega169	1	1	BLB12	BLB11	BLB02	BLB01	LB2	LB1
ATmega323	1	1	BLB12	BLB11	BLB02	BLB01	LB2	LB1
ATmega32	1	1	BLB12	BLB11	BLB02	BLB01	LB2	LB1
ATmega64	1	1	BLB12	BLB11	BLB02	BLB01	LB2	LB1
ATmega103	1	1	1	1	1	LB2	LB1	1
ATmega128	1	1	BLB12	BLB11	BLB02	BLB01	LB2	LB1
AT89551[1]	–							
AT89552[2]	–							
AT86RF401	1	1	1	1	1	LB2	LB1	1

Notes: 1. See the AT89551 Data Sheet for more information.
2. See the AT89552 Data Sheet for more information.

Fuse Bits, Low Byte

For explanation of the Fuse bits, read the data sheet for the particular device. Unprogrammed bits should be set to “1”.

Table 43. Fuse Bits, Low Byte

Fuse Bits	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ATtiny11	0	0	0	FSTRT	RSTDSBL	CKSEL2	CKSEL1	CKSEL0
ATtiny12	BODLVL	BODEN	SPIEN	RSTDSBL	CKSEL3	CKSEL2	CKSEL1	CKSEL0
ATtiny15	BODLVL	BODEN	SPIEN	RTDSBL	1	1	CKSEL1	CKSEL0
ATtiny22	1	1	SPIEN	1	1	1	1	RCEN
ATtiny26	PLLCK	CKOPT	SUT1	SUT0	CKSEL3	CKSEL2	CKSEL1	CKSEL0
ATtiny28	1	1	1	INTCAP	CKSEL3	CKSEL2	CKSEL1	CKSEL0
AT90S1200	1	1	SPIEN	1	1	1	1	RCEN
AT90S2313	1	1	SPIEN	1	1	1	1	FSTRT
AT90S2323	1	1	SPIEN	1	1	1	1	FSTRT
AT90S2333	1	1	SPIEN	BODLVL	BODEN	CKSEL2	CKSEL1	CKSEL0
AT90S2343	1	1	SPIEN	1	1	1	1	RCEN
AT90S4414	1	1	SPIEN	1	1	1	1	FSTRT
AT90S4433	1	1	SPIEN	BODLVL	BODEN	CKSEL2	CKSEL1	CKSEL0
AT90S4434	1	1	SPIEN	1	1	1	1	FSTRT
AT90S8515	1	1	SIEN	1	1	1	1	FSTRT
AT90S8535	1	1	SPIEN	1	1	1	1	FSTRT
ATmega8	BODLEVEL	BODEN	SUT1	SUT0	CKSEL3	CKSEL2	CKSEL1	CKSEL0
ATmega8515	BODLEVEL	BODEN	SUT1	SUT0	CKSEL3	CKSEL2	CKSEL1	CKSEL0
ATmega8535	BODLEVEL	BODEN	SUT1	SUT0	CKSEL3	CKSEL2	CKSEL1	CKSEL0
ATmega161	1	BOOTRST	SPIEN	BODLVL	BODEN	CKSEL2	CKSEL1	CKSEL0
ATmega163	BODLEV	BODEN	SPIEN	1	CKSEL3	CKSEL2	CKSEL1	CKSEL0
ATmega16	BODLEVEL	BODEN	SUT1	SUT0	CKSEL3	CKSEL2	CKSEL1	CKSEL0
ATmega162	CKDIV8	CKOUT	SUT1	SUT0	CKSEL3	CKSEL2	CKSEL1	CKSEL0
ATmega169	CKDIV8	CKOUT	SUT1	SUT0	CKSEL3	CKSEL2	CKSEL1	CKSEL0
ATmega323	BODLEV	BODEN	1	1	CKSEL3	CKSEL2	CKSEL1	CKSEL0
ATmega32	BODLEVEL	BODEN	SUT1	SUT0	CKSEL3	CKSEL2	CKSEL1	CKSEL0
ATmega64	BODLEVEL	BODEN	SUT1	SUT0	CKSEL3	CKSEL2	CKSEL1	CKSEL0
ATmega103	1	1	SPIEN	1	EESAVE	1	SUT1	SUT0
ATmega128	BODLEVEL	BODEN	SUT1	SUT0	CKSEL3	CKSEL2	CKSEL1	CKSEL0
AT89551	1	1	1	1	1	1	1	1
AT89552	1	1	1	1	1	1	1	1
AT86RF401	1	1	1	1	1	1	1	1

Fuse Bits, High Byte

For explanation of the Fuse bits, read the data sheet for the particular device. Unprogrammed bits should be set to “1”.

Table 44. Fuse Bits, High Byte

Fuse Bits	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ATtiny11	1	1	1	1	1	1	1	1
ATtiny12	1	1	1	1	1	1	1	1
ATtiny15	1	1	1	1	1	1	1	1
ATtiny22	1	1	1	1	1	1	1	1
ATtiny26	1	1	1	RSTDISBL	SPIEN	EESAVE	DOBLEVEL	BODEN
ATtiny28	1	1	1	1	1	1	1	1
AT90S1200	1	1	1	1	1	1	1	1
AT90S2313	1	1	1	1	1	1	1	1
AT90S2323	1	1	1	1	1	1	1	1
AT90S2333	1	1	1	1	1	1	1	1
AT90S2343	1	1	1	1	1	1	1	1
AT90S4414	1	1	1	1	1	1	1	1
AT90S4433	1	1	1	1	1	1	1	1
AT90S4434	1	1	1	1	1	1	1	1
AT90S8515	1	1	1	1	1	1	1	1
AT90S8535	1	1	1	1	1	1	1	1
ATmega8	RSTDISBL	WDTON	SPIEN	CKOPT	EESAVE	BOOTSZ1	BOOTSz0	BOOTRST
ATmega8515	S8515C	WDTON	SPIEN	CKOPT	EESAVE	BOOTSZ1	BOOTSz0	BOOTRST
ATmega8535	S8535C	WDTON	SPIEN	CKOPT	EESAVE	BOOTSZ1	BOOTSz0	BOOTRST
ATmega161	1	1	1	1	1	1	1	1
ATmega163	1	1	1	1	1	BOOTSZ1	TOOTSZ0	BOOTRST
ATmega16	OCDEN	JTAGEN	SPIEN	CKOPT	EESAVE	BOOTSZ1	BOOTSz0	BOOTRST
ATmega162	OCDEN	JTAGEN	SPIEN	WDTON	EESAVE	BOOTSZ1	BOOTSz0	BOOTRST
ATmega169	OCDEN	JTAGEN	SPIEN	WDTON	EESAVE	BOOTSZ1	BOOTSz0	BOOTRST
ATmega323	OCDEN	JTAGEN	SPIEN	1	EESAVE	BOOTSZ1	TOOTSZ0	BOOTRST
ATmega32	OCDEN	JTAGEN	SPIEN	CKOPT	EESAVE	BOOTSZ1	BOOTSz0	BOOTRST
ATmega64	OCDEN	JTAGEN	SPIEN	CKOPT				
ATmega103	1	1	1	1	1	1	1	1
ATmega128	OCDEN	JTAGEN	SPIEN	CKOPT				
AT89551	1	1	1	1	1	1	1	1
AT89552	1	1	1	1	1	1	1	1
AT86RF401	1	1	1	1	1	1	1	1

Fuse Bits, Extended Byte

For explanation of the Fuse bits, read the data sheet for the particular device. Unprogrammed bits should be set to “1”.

Table 45. Fuse Bits, Extended Byte

Fuse Bits	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ATtiny11	–	–	–	–	–	–	–	–
ATtiny12	–	–	–	–	–	–	–	–
ATtiny15	–	–	–	–	–	–	–	–
ATtiny22	–	–	–	–	–	–	–	–
ATtiny26	–	–	–	–	–	–	–	–
ATtiny28	–	–	–	–	–	–	–	–
AT90S1200	–	–	–	–	–	–	–	–
AT90S2313	–	–	–	–	–	–	–	–
AT90S2323	–	–	–	–	–	–	–	–
AT90S2333	–	–	–	–	–	–	–	–
AT90S2343	–	–	–	–	–	–	–	–
AT90S4414	–	–	–	–	–	–	–	–
AT90S4433	–	–	–	–	–	–	–	–
AT90S4434	–	–	–	–	–	–	–	–
AT90S8515	–	–	–	–	–	–	–	–
AT90S8535	–	–	–	–	–	–	–	–
ATmega8	–	–	–	–	–	–	–	–
ATmega8515	–	–	–	–	–	–	–	–
ATmega8535	–	–	–	–	–	–	–	–
ATmega161	–	–	–	–	–	–	–	–
ATmega163	–	–	–	–	–	–	–	–
ATmega16	–	–	–	–	–	–	–	–
ATmega162	1	1	1	M161C	BODLEVEL2	BODLEVEL1	BODLEVEL0	1
ATmega169	1	1	1	1	BODLEVEL2	BODLEVEL1	BODLEVEL0	RSTDISBL
ATmega323	–	–	–	–	–	–	–	–
ATmega32	–	–	–	–	–	–	–	–
ATmega64	1	1	1	1	1	1	M103C	WDTON
ATmega103	–	–	–	–	–	–	–	–
ATmega128	1	1	1	1	1	1	M103C	WDTON
AT89551	–	–	–	–	–	–	–	–
AT89552	–	–	–	–	–	–	–	–
AT86RF401	–	–	–	–	–	–	–	–

Devices Combining the Fuse and Lock Bits on Readback

Some devices combine readback of the the Fuse and Lock bits in the same command and response. The proper command to use is the Cmnd_STK_READ_LOCK command. The parts currently using this combined approach are:

- AT90S1200
- AT90S2313
- AT90S2323
- AT90S2343
- AT90S4414
- AT90S8515
- AT90S4434
- AT90S8535
- ATtiny22

Check the pseudo-code in the following sections for comments on how to extract the Fuse bits and Lock bits from the returned value.

Reading and Writing the Fuse and Lock Bits in Serial Mode

Writing and reading the Lock bits and Fuse bits in Serial mode is currently done using the Cmnd_STK_UNIVERSAL command. This command requires that the ISP commands is properly formatted from the PC. These commans are not all of the simple kind.

The following pseudo-code sections illustrates how the Cmnd_STK_UNIVERSAL command may be used. The function:

```
UniversalCommand(UCHAR byte1, UCHAR byte2, UCHAR byte3, UCHAR *byte4)
```

is used in the pseudo-code. byte1 - byte4 should be set to contain the bitstream giving the command and data. On return, byte4 will contain the last eight bit of the clocked-out bitstream from the ISP interface. See the description of the Cmnd_STK_UNIVERSAL command in earlier sections for more details.

Reading the Lock Bits

```
if (m_pDevice->ucLockBytes >= 1)
  UniversalCommandNew(0x58, 0, 0, &data)

// Some parts combine Fuses and Locks in same byte on readback. To get the
// correct Lock bits from these parts, we need to mask away the bits that are
// not Lock bits. We also shift the Lock bits to the position described in the
// lock bits appendix section.
// Note: The same masking of the data must be done when reading the Lock bits
// in Parallell mode.

if (part does combine Fuse and Lock bits)
{
  lock_byte = 0xff;
  if (! (data & 0x80))
    lock_byte &= 0xfd;

  if (! (data & 0x40))
    lock_byte &= 0xfb;
}
```

Writing the Lock Bits

```
if (number of lock bytes >= 1)
{
  // Devices with selftimed SPI has another programming
// format for low lock bits than those without, apart from for ATtiny12 &
ATtiny15

  if ((device supports self-timing programming) && (device is not Attiny12 or
Attiny15))
    UniversalCommand(0xAC, 0xE0, 0, &lock_byte))
  else
    UniversalCommand(0xAC, lock_byte | 0xF9, 0, &dummy_var))

  Sleep(50); // Wait for lock bit write operation in device to complete
}
```

Reading the Fuse Bits

```
// As mentioned, some parts combine Fuses and Locks in same byte on readback.
// They have another programming format than those which don't.

if (part does not combine fuse and lock bits)
{
  if (number of fuse bytes >= 1)
    UniversalCommandNew(0x50, 0, 0, &low_fuse_byte);

  if (number of fuse bytes >= 2)
    UniversalCommandNew(0x58, 0x08, 0, &high_fuse_byte);
}

// Common Fuse and Lock bits. This is exactly like reading Lock bits, execept
// that we pick another mask for getting the Fuse bits from the returned data.
// Note: The same masking of the data must be done when reading the Fuses
// in Parallell mode.

else if (part does combine fuse and lock bits)
{
  if (number of lock bits are >= 1)
    UniversalCommandNew(0x58, 0, 0, &low_fuse_byte);

  low_fuse_byte |= 0xC0;// Set non-fuse bits to 1
}
```

Writing the Fuse Bits

```
if (number of fuse bytes >= 1)
{
// Devices with selftimed SPI have another Programming
// format for low fuses than those without

  if (device supports self-timing programming)
    UniversalCommand(0xAC, 0xA0, 0, &low_fuse_byte);
  else
    UniversalCommand(0xAC, (low_fuse_byte & 0x1F) | 0xA0, 0, &low_fuse_byte);

  Sleep(50); // Wait for fuse write operation in device to complete
}

if (number of fuse bytes >= 2)
{
  UniversalCommandNew(0xAC, 0xA8, 0, &high_fuse_byte);

  Sleep(50); // Wait for fuse write operation in device to complete
}
```

Atmel Corporation

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

Regional Headquarters

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

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

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

Atmel Operations

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

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

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

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

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

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

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

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

Biometrics/Imaging/Hi-Rel MPU/
High Speed Converters/RF Datacom
Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex, France
Tel: (33) 4-76-58-30-00
Fax: (33) 4-76-58-34-80

e-mail
literature@atmel.com

Web Site
http://www.atmel.com

2525B–AVR–04/03 0M