Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!

Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832
Email & Skype: info@chipsmall.com Web: www.chipsmall.com
Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China

AVR069: AVRISP mkII Communication Protocol

Features

- **General commands**
- **ISP commands**
- **Return values**
- **Parameters**

1 Introduction

This document describes the AVRISP mkII protocol. The firmware is distributed with AVR Studio 4.12 or later. Download the latest AVR Studio from the Atmel web site, http://www.atmel.com/products/AVR/.

The definition of all commands, responses, parameters and other defined values can be found in chapter 6.1.

All device specific values can be found in the XML part description files. See chapter 4 for how to find the parameter values for AVRISP mkII.

Figure 1-1. AVRISP mkII

8-bit AVR® Microcontrollers

Application Note

Rev. 8015B-AVR-02/06

2 USB Communication

The communication between the AVRISP mkII and the PC is done through its USB interface. The USB interface utilizes two bulk endpoints; one IN and one OUT. The USB descriptors can be found in chapter 6.2.

2.1 Packet Format

The PC sends commands to the AVRISP mkII, 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.

The commands and their respective answers are described in chapter 3.

2.2 USB Driver

In order to communicate with the AVRISP mkII, 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 AVRISP mkII 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.

3 Commands

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

For all commands, the AVRISP mkII 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.

3.1 General Commands

3.1.1 CMD_SIGN_ON

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

Table 3-1. Command format

Field	Size	Value	Description
Command ID	1 byte	CMD_SIGN_ON	Command id

Table 3-2. Answer format

Field	Size	Value	Description
Answer ID	1 byte	CMD_SIGN_ON	Answer id
Status	1 byte	STATUS_CMD_OK	This command will always return STATUS_CMD_OK
Signature length	1 byte	10	Length of signature string
	10 bytes	“AVRISP_MK2”	The signature string (not null terminated)

3.1.2 CMD_SET_PARAMETER

The host can set a multitude of parameters in the AVRISP mkII. See the 3.4: Parameters for a description of each parameter. All parameters are one-byte values.

Table 3-3. Command format

Field	Size	Value	Description
Command ID	1 byte	CMD_SET_PARAMETER	Command id
Parameter ID	1 byte		
Value	1 byte		

Table 3-4. Answer format

Field	Size	Value	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

3.1.3 CMD_GET_PARAMETER

The host can also read different parameters from the AVRISP mkII.

Table 3-5. Command format

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

Table 3-6. Answer format if command succeeds

Field	Size	Value	Description
Answer ID	1 byte	CMD_GET_PARAMETER	Answer id
Status	1 byte	STATUS_CMD_OK	A status value indicating success
Signature length	1 byte		The parameter value

Table 3-7. Answer format if fails

Field	Size	Value	Description
Answer ID	1 byte	CMD_GET_PARAMETER	Answer id

Field	Size	Value	Description
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.

3.1.4 CMD_OSCCAL

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

Table 3-8. Command format

Field	Size	Value	Description
Command ID	1 byte	CMD_OSCCAL	Command id

Table 3-9. Answer format

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

3.1.5 CMD_LOAD_ADDRESS

This command will load an address into the AVRISP mkII. 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 all programming modes. All the abovementioned commands will increment an internal address counter, so this command needs only to be sent once.

Table 3-10. Command format

Field	Size	Value	Description
Command ID	1 byte	CMD_LOAD_ADDRESS	Command id
Address	4 bytes		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 AVRISP mkII that a *load extended address* must be executed. See datasheet for devices with memories larger than 64KBytes.

Table 3-11. Answer format.

Field	Size	Value	Description
Answer ID	1 byte	CMD_LOAD_ADDRESS	Answer id
Status	1 byte	STATUS_CMD_OK	This command will always return STATUS_CMD_OK

3.1.6 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 AVRISP mkII can “reboot” into upgrade mode by using this command.

Table 3-12. Command format

Field	Size	Value	Description
Command ID	1 byte	CMD_FIRMWARE_UPGRADE	Command id
Parameter ID	9 bytes	”fwupgrade”	String to enable upgrade mode (not null terminated)

Table 3-13. Answer format

Field	Size	Value	Description
Answer ID	1 byte	CMD_FIRMWARE_UPGRADE	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 AVRISP mkII will disconnect and enter upgrade mode.

3.1.7 CMD_RESET_PROTECTION

This command resets the short circuit protection system after a short circuit state has been returned by the PARAM_STATUS_TGT_CONN.

Table 3-14. Command format

Field	Size	Value	Description
Command ID	1 byte	CMD_RESET_PROTECTION	Command id

Table 3-15. Answer format

Field	Size	Value	Description
Answer ID	1 byte	CMD_RESET_PROTECTION	Answer id
Status	1 byte	STATUS_CMD_OK	This command will always return STATUS_CMD_OK

3.2 ISP Programming Commands

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

3.2.1 CMD_ENTER_PROGMODE_ISP

This command will make the target device enter programming mode.

XML path: `/AVRPART/ICE_SETTINGS/STK500_2/IspEnterProgMode/`

Table 3-16. Command form

Field	Size	Value	Description

Field	Size	Value	Description
Command ID	1 byte	CMD_ENTER_PROG MODE_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, 0x69 for AT89xx
pollIndex	1 byte	XML: pollIndex	Start address, received byte: 0 = no polling, 3 = AVR, 4 = AT89xx
cmd1	1 byte		Command Byte # 1 to be transmitted
cmd2	1 byte		Command Byte # 2 to be transmitted
cmd3	1 byte		Command Byte # 3 to be transmitted
cmd4	1 byte		Command Byte # 4 to be transmitted

Note: 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)

Table 3-17. Answer format

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

3.2.2 CMD_LEAVE_PROGMODE_ISP

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

XML path: /AVRPART/ICE_SETTINGS/AVRISP mkII_2/IspLeaveProgMode/

Table 3-18. Command format

Field	Size	Value	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 3-19. Answer format

Field	Size	Value	Description

Field	Size	Value	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

3.2.3 CMD_CHIP_ERASE_ISP

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

XML path: /AVRPART/ICE_SETTINGS/STK500_2/IspChipErase/

Table 3-20. Command form

Field	Size	Value	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 delay1= use RDY/BSY command
cmd1	1 byte		Chip erase command byte #1
cmd2	1 byte		Chip erase command byte #2
cmd3	1 byte		Chip erase command byte #3
cmd4	1 byte		Chip erase command byte #4

Table 3-21. Answer format

Field	Size	Value	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

3.2.4 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/STK500_2/IspProgramFlash/`

Table 3-22. Command form

Field	Size	Value	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)

Field	Size	Value	Description
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:

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 AVRISP mkII, so leave this bit cleared.

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 AVRISP mkII 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 3-23. Answer format

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

3.2.5 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/ STK500_2/IspReadFlash/`

Table 3-24. Command format

Field	Size	Value	Description
Command ID	1 byte	CMD_READ_FLASH_ISP	Command id
NumBytes	2 bytes	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 3-25. Answer format if the command is executed

Field	Size	Value	Description
ANSWER ID	1 byte	CMD_READ_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 3-26. Answer format if the command was not executed

Field	Size	Value	Description
ANSWER ID	1 byte	CMD_READ_FLASH_ISP	Answer id
STATUS	1 byte	STATUS_CMD_FAILED	Indicates failure

3.2.6 CMD_PROGRAM_EEPROM_ISP

See the CMD_PROGRAM_FLASH_ISP command.

3.2.7 CMD_READ_EEPROM_ISP

See the CMD_READ_FLASH_ISP command.

3.2.8 CMD_PROGRAM_FUSE_ISP

This command programs the fuses of the target device.

Table 3-27. Command form

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

Note: cmd1, cmd2, cmd3 and cmd4 are the four bytes of the low-level program fuse ISP command.

Table 3-28. Answer format

Field	Size	Value	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

3.2.9 CMD_READ_FUSE_ISP

This command reads the fuses of the target device.

Table 3-29. Command form

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

Note: 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)

Table 3-30. Answer format

Field	Size	Value	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

3.2.10 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.

3.2.11 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.

3.2.12 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.

3.2.13 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.

3.2.14 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 3-31. Command format

Field	Size	Value	Description
Command ID	1 byte	CMD_SPI_MULTI	Command ID
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 be transmitted. The size is specified by NumTx

Note:

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 3-32. Answer format

Field	**Size**	**Value**	**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

3.3 Return Values

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

3.3.1 Success

Table 3-33. Success

Value	Meaning
STATUS_CMD_OK	Command executed OK

3.3.2 Warnings

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

Table 3-34. Success

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

3.3.3 Errors

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

Table 3-35. Success

Value	Meaning
STATUS_CMD_FAILED	Command failed
STATUS_CMD_UNKNOWN	Unknown command

3.4 Parameters

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

Table 3-36. Success

Value	Meaning	R/W
PARAM_BUILD_NUMBER_LOW	Firmware build number, high byte	R
PARAM_BUILD_NUMBER_HIGH	Firmware build number, low byte	R
PARAM_HW_VER	Hardware version	R
PARAM_SW_MAJOR	Firmware version number, major byte	R
PARAM_SW_MINOR	Firmware version number, minor byte	R
PARAM_VTARGET	Target Voltage	R
PARAM_SCK_DURATION	ISP SCK duration	R/W
PARAM_RESET_POLARITY	Active low or active high RESET handling	W
PARAM_STATUS_TGT_CONN	Status of target connection	R
PARAM_DISCHARGEDELAY	Delay with higher resistance of reset line	W

3.4.1 PARAM_BUILD_NUMBER_LOW

The PARAM_BUILD_NUMBER_LOW and PARAM_BUILD_NUMBER_HIGH together return a number that is incremented for each build of the firmware. This number is mainly for ATMEL internal use.

3.4.2 PARAM_BUILD_NUMBER_HIGH

See PARAM_BUILD_NUMBER_LOW.

3.4.3 PARAM_HW_VER

Returns a hardware revision number.

3.4.4 PARAM_SW_MAJOR

The PARAM_SW_MAJOR and PARAM_SW_MINOR returns the firmware version.

3.4.5 PARAM_SW_MINOR

See PARAM_SW_MAJOR.

3.4.6 PARAM_VTARGET

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

3.4.7 PARAM_SCK_DURATION

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 AVRISP mkII supports ISP frequencies from 51 Hz up to 8.0 MHz. The value for PARAM_SCK_DURATION can be found using the algorithm shown in chapter 6.3.

3.4.8 PARAM_RESET_POLARITY

The AVRISP mkII can program both AT90 (AVR) family and AT89 (8051) family of microcontrollers. They have different RESET pin polarity. The AVR has active low reset, while the AT89 has active high.

This parameter sets the polarity of the reset signal. Set the parameter to 1 when programming AVRs, and 0 when programming AT89 controllers.

NOTE: AVRISP mkII stores this parameter in EEPROM, so they are available the next time power is applied to the programmers.

3.4.9 PARAM_STATUS_TGT_CONN

This parameter returns the status of the target connection. Each bit has a separate status. See table below:

Table 3-37. Success

Bit value	Status
0x00	STATUS_ISP_READY
0x01	STATUS_CONN_FAIL_MOSI
0x02	STATUS_CONN_FAIL_RST

Bit value	Status
0x04	STATUS_CONN_FAIL_SCK
0x10	STATUS_TGT_NOT_DETECTED
0x20	STATUS_TGT_REVERSE_INSERTED
0x00	STATUS_ISP_READY

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.

If any *_CONN_FAIL_* bit is set, the command CMD_RESET_PROTECTION must be issued.

The parameter should be checked before starting a programming sequence to check if target connection is correct.

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 AVRISP mkII is isolated via switches when the AVRISP mkII is in idle mode.

3.4.10 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 AVRISP mkII.

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

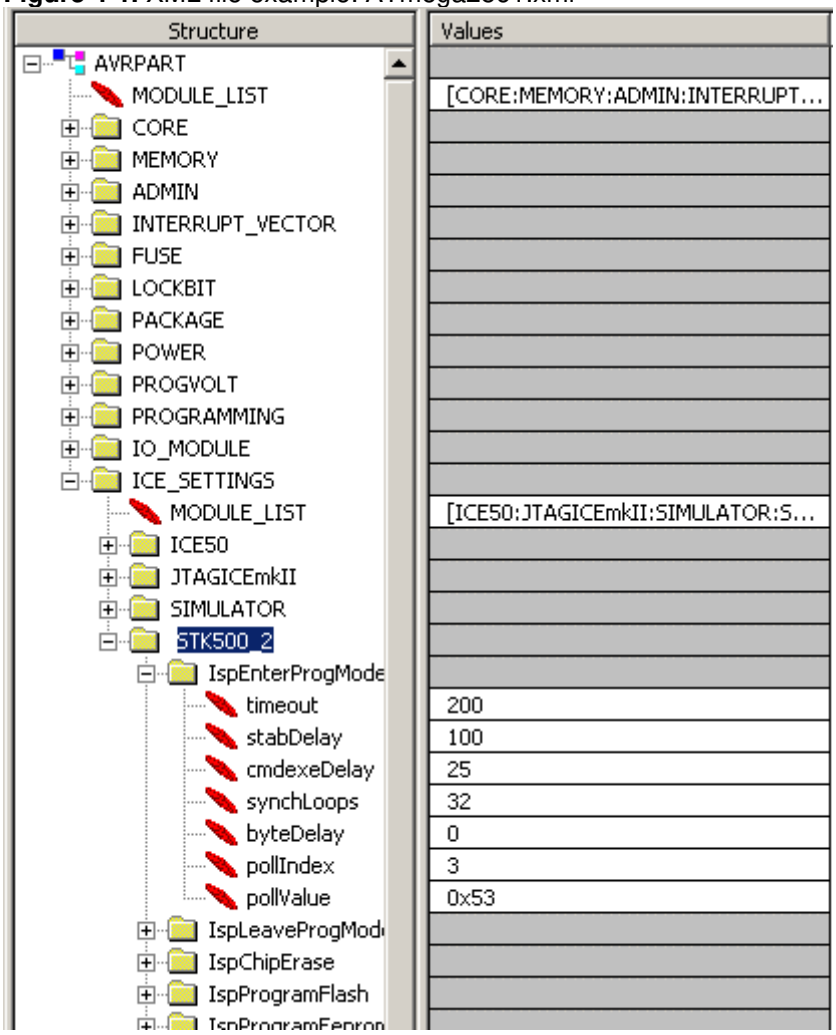
If no capacitor is connected this parameter could be set to 0.

4 XML Parameter Values

The AVRISP mkII firmware uses parameters extensively for its programming algorithms. All AVR devices have their own set of parameters. They can be found in part description files installed with AVR Studio. The part description files are XML files and can be found in the folder

```
… \Atmel\AVR Tools\PartDescriptionFiles\
```

Figure 4-1. XML file example: ATmega2561.xml



Open the XML file in an XML editor/viewer (e.g XML Notepad or Internet Explorer). All device specific values for AVRISP mkII are located under STK500_2 node. For parameters for e.g. the CMD_ENTER_PROGMODE_ISP command, look at the node

```
/AVRPART/ICE_SETTINGS_STK500_2/IspEnterProgMode
```

5 Command Sequence Example

This chapter contains examples of how to connect to the AVRISP mkII from the PC Frontend and how to read signature from a device.

See chapter 3 for a description of the commands and parameters.

5.1 Connect

The sequence of commands and parameters sent from AVR Studio to the AVRISP mkII 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

5.2 Read Signature

The sequence of commands and parameters sent from AVR Studio to the AVRISP mkII in order to read the device signature through ISP is listed below. Note that one already has to be connected to do this.

- CMD_SET_PARAMETER, PARAM_RESET_POLARITY
- CMD_GET_PARAMETER, PARAM_STATUS_TGT_CONN
- CMD_ENTER_PROGMODE_ISP
- CMD_READ_SIGNATURE_ISP
- CMD_READ_SIGNATURE_ISP
- CMD_READ_SIGNATURE_ISP
- CMD_LEAVE_PROGMODE_ISP

6 Appendix

6.1 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


// *** [ 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


// *** [ 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


// *** [ Parameter constants ] ***
#define PARAM_BUILD_NUMBER_LOW              0x80
#define PARAM_BUILD_NUMBER_HIGH             0x81
#define PARAM_HW_VER                        0x90
#define PARAM_SW_MAJOR                      0x91
#define PARAM_SW_MINOR                      0x92
#define PARAM_VTARGET                       0x94
#define PARAM_SCK_DURATION                  0x98
#define PARAM_RESET_POLARITY                0x9E
#define PARAM_STATUS_TGT_CONN               0xA1
#define PARAM_DISCHARGEDELAY                0xA4

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

6.2 USB Descriptors

Table 6-1. Device Descriptor

Name	Value	Hex
bLength	Valid	0x12
bDescriptorType	DEVICE	0x01
bcdUSB	1.1	0x0110
bDeviceClass	Vendor-specific	0xFF
bDeviceSubClass	Vendor-specific	0x00
bDeviceProtocol	None	0x00
bMaxPacketSize0	16	0x10
idVendor	Atmel Corporation	0x03EB
idProduct	0x2104	0x2104
bcdDevice	2.0	0x0200
iManufacturer	1	0x01
iProduct	2 "AVRISP mkII"	0x02
iSerialNumber	3	0x03
bNumConfigurations	1	0x01

Table 6-2. 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. Reserved	Zero	0x00
bmAttributes. RemoteWakeup	Not supported	0x0
bmAttributes. SelfPowered	Yes	0x1
bmAttributes. Reserved7	One	0x1
bMaxPower	200 mA	0x64

Table 6-3. 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
bInterfaceProtocol	None	0x00
iInterface	0	0x00

Table 6-4. Endpoint descriptor IN

Name	Value	Hex
bLength	Valid	0x07
bDescriptorType	ENDPOINT	0x05
bEndpointAddress	2 IN	0x82
bmAttributes. TransferType	Bulk	0x2
bmAttributes. Reserved	Zero	0x00
wMaxPacketSize	64 bytes	0x0040
bInterval	Ignored for Bulk endpoints	0x0A

Table 6-5. Endpoint descriptor OUT

Name	Value	Hex
bLength	Valid	0x07
bDescriptorType	ENDPOINT	0x05

Name	Value	Hex
bEndpointAddress	2 OUT	0x02
bmAttributes. TransferType	Bulk	0x2
bmAttributes. Reserved	Zero	0x00
wMaxPacketSize	64 bytes	0x0040
bInterval	Ignored for Bulk endpoints	0x0A

6.3 Setting SCK Frequency

The AVRISP mkII supports the SCK frequencies shown in `avrispmkIIfreqs` below. Use the `CalcSckDur( )` algorithm to find the PARAM_SCK_DURATION value for a given frequency:

```
// frequencies for AVRISP mkII ISP programming
double avrispmkIIfreqs[] = {
     8000000, 4000000, 2000000, 1000000, 500000, 250000, 125000,
     96386, 89888, 84211, 79208, 74767, 70797, 67227, 64000,
     61069, 58395, 55945, 51613, 49690, 47905, 46243, 43244,
     41885, 39409, 38278, 36200, 34335, 32654, 31129, 29740,
     28470, 27304, 25724, 24768, 23461, 22285, 21221, 20254,
     19371, 18562, 17583, 16914, 16097, 15356, 14520, 13914,
     13224, 12599, 12031, 11511, 10944, 10431, 9963, 9468,
     9081, 8612, 8239, 7851, 7498, 7137, 6809, 6478, 6178,
     5879, 5607, 5359, 5093, 4870, 4633, 4418, 4209, 4019,
     3823, 3645, 3474, 3310, 3161, 3011, 2869, 2734, 2611,
     2484, 2369, 2257, 2152, 2052, 1956, 1866, 1779, 1695,
     1615, 1539, 1468, 1398, 1333, 1271, 1212, 1155, 1101,
     1049, 1000, 953, 909, 866, 826, 787, 750, 715, 682,
     650, 619, 590, 563, 536, 511, 487, 465, 443, 422,
     402, 384, 366, 349, 332, 317, 302, 288, 274, 261,
     249, 238, 226, 216, 206, 196, 187, 178, 170, 162,
     154, 147, 140, 134, 128, 122, 116, 111, 105, 100,
     95.4, 90.9, 86.6, 82.6, 78.7, 75.0, 71.5, 68.2,
     65.0, 61.9, 59.0, 56.3, 53.6, 51.1
};

UCHAR CalcSckDur(long sckFrequency)
{
UCHAR paramSckDuration;

     // Default to slowest value
     paramSckDuration = (UCHAR)sizeof(avrispmkIIfreqs)-1;

     // Find first frequency that is less than the requested
     for (int i = 0; i < sizeof(avrispmkIIfreqs); i++)
     {
```

```
            if (avrispmkIIfreqs[i] <= sckFrequency)
            {
                paramSckDuration = i;
                break;
            }
        }

        return paramSckDuration;
    }
```

Table of Contents

Atmel Corporation

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

Regional Headquarters

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

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

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

Atmel Operations

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

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

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

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

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

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

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

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

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

Literature Requests
www.atmel.com/literature

8015B-AVR-02/06