AVR067: JTAGICE mkII Communication Protocol

8-bit AVR® Microcontrollers

Application Note

Features

- **Commands sent from AVR Studio to JTAGICE mkII**
- **Replies sent from JTAGICE mkII to AVR Studio**
- **Configurable Parameters**
- **Different Memory Types**
- **Special Characters and Packet Formats for Packet Synchronization and Error Control**
- **Break Point Handling in JTAGICE mkII**

1 Introduction

This document describes the communication protocol used between AVR Studio and JTAGICE mkII. The purpose of this document is to enable third party vendors to design their own front-end to the JTAGICE mkII.

Rev. 2587C-AVR-03/06

2 Theory of operation

The communication protocol operates in a master – slave environment, AVR Studio being the master and JTAGICE mkII being the slave.

The JTAGICE mkII slave is always operating in one of three states (with enumeration):

1. STOPPED
2. RUNNING
3. PROGRAMMING

Some commands can only be executed while the slave is in STOPPED mode, others may be allowed in all modes. The details are described for each different command.

Events are messages sent from the slave with no prior commands given from the master.

Serial communication is asynchronous by nature, so is the JTAGICE mkII protocol. All commands and responses have tokens and CRC checks included enabling the receiver to verify the validity of the received message.

This transport protocol guarantees that every packet that is received is verified correctly, but it does not guarantee that every packet is received. Retransmission and lost packets are to be handled by the application layer, not the transmission layer. When USB is used, the hardware layer handles retransmission automatically. This is not the case with RS232.

The protocol is “Little Endean”, LSB is always transferred first in memory addresses.

2.1 USB and RS-232 Connection

The JTAGICE mkII has both a USB and an RS-232 interface for connecting to a host computer. They share the same message format (see section 3).

USB is preferred due to speed.

2.1.1 RS-232 Connection

The RS-232 connection’s power-on settings are 19,200 bps, 8N1, no handshake. However, using the Set Parameter command and the Baud rate parameter, the baud rate can be changed.

2.1.2 USB Connection

The JTAGICE mkII has a USB port capable of USB full-speed connection (12Mbps).

In addition to data transfer, the USB cable can also supply power to the emulator, so no external power is required. After enumeration, the emulator can draw up to 500mA from the USB hub.

2.1.3 USB Configuration

The JTAGICE mkII has one configuration with one interface containing two bulk transfer endpoints. The USB descriptors can be found in section 7.

The JTAGICE mkII messages are transferred through the bulk endpoints. Maximum packet size is shown in the endpoint descriptors.

In addition to the framing mechanisms supplied by the JTAGICE mkII frame format, the emulator uses short packets (both IN and OUT endpoints) to indicate which USB packet is the last one in a JTAGEICE mkII frame. The host driver must also support short packets.

3 Message Format

All commands, responses and events share a common message format:

<MESSAGE_START, SEQUENCE_NUMBER, MESSAGE_SIZE, TOKEN, MESSAGE_BODY, CRC>

Table 3-1. Parameters of messages.

Parameter Name	Usage	Format
MESSAGE_START	ASCII 27 (ESC character)	[BYTE]
SEQUENCE_NUMBER	Incremented by one for each message sent. Wraps to zero after 0xFFFE. 0xFFFF is reserved for (asynchronous) EVENTS.	[BYTE]*2, LSB first.
MESSAGE_SIZE	Size of the message body in bytes	[BYTE]*4, LSB first
TOKEN	ASCII 14, making the protocol more robust.	[BYTE]
MESSAGE_BODY	Message body	[BYTE]*MESSAGE_SIZE
CRC	Uses all characters in message including MESSAGE_START and MESSAGE_BODY.	[BYTE]*2

First byte of the message body is always the MESSAGE_ID. The size of the MESSAGE_BODY is at least 1 byte.

- Messages use MESSAGE_ID in the range from 0x01 to 0x3F
- Internal commands use MESSAGE_ID in the range from 0x40 to 0x7F
- Success responses use MESSAGE_ID in the range from 0x80 to 0x9F
- Failure responses use MESSAGE_ID in the range from 0xA0 to 0xBF
- Events use MESSAGE_ID in the range from 0xE0 to 0xFF
- Message start is ASCII 27 (0x1B), not to be changed.
- Token is ASCII 14 (0x0E), not to be changed.

4 Message parsing

4.1 Introduction

Messages either way should be read as a byte stream by a state machine. Every time a MESSAGE_START character occurs, the state machine starts to decode the message, either completing the message or falling out of the byte stream if a message rule is violated.

Messages can come in any order and there is no specific timing regarding required timeframes between commands. The emulator implements a message queue, when

the state machine has parsed and verified a complete message it is serialized an put on a message queue. The emulator executes the messages in the sequence found on the queue. Responses to the messages are given when they are processed by the emulator.

The same state machine is used for both RS232 and USB transmission. An interrupt routine on the communication control processor in the JTAGICE mkII reads the appropriate UART and puts the incoming data in a FIFO implemented in SRAM. The state machine reads and parses the data from the FIFO.

4.2 Common State Machine Implementation

Figure 4-1. State Diagram

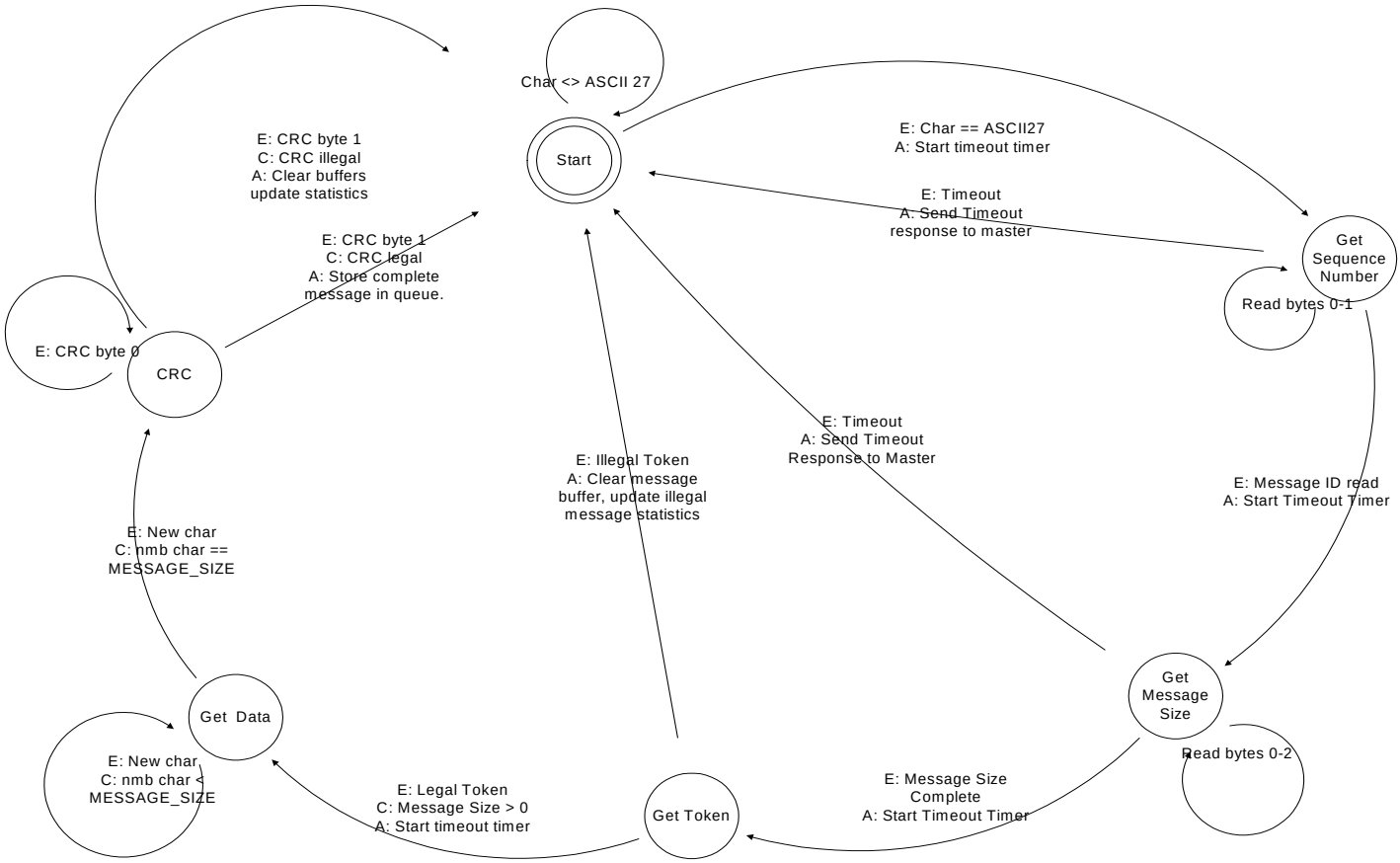

Table 4-1. State table. Gray fields represent “message failed”, non grayed fields are “message valid”.

Current State	Event	Condition	Action	Next State
Start	Read character from inbuf	Char == ASCII 27	Start timeout timer	Get Sequence Number
	Read character from inbuf	Char <> ASCII 27	None	Start
Get Sequence Number	Read character from inbuf	Sequence Number Byte Counter == 2	Start timeout timer	Get Message Size
	Read character from inbuf	Sequence Number Byte Counter < 2	Start timeout timer	Get Sequence Number
	Timeout	None	Update message statistics	Start
Get Message Size	Read character from inbuf	Message Size Byte Counter < 4	Start timeout timer	Get Message Size
	Read character from inbuf	Message Size Byte Counter ==4	Start timeout timer Calculate MESSAGE_SIZE	Get Token
	Timeout	None	Update message statistics	Start
Get Token	Read character from inbuf	Char == ASCII 14	Start timeout timer	Get Data
	Read character from inbuf	Char <> ASCII 14	Update message statistics Stop timeout timer	Start
	Timeout	None	Update message statistics	Start
Get Data	Read character from inbuf	Nmb char read == MESSAGE_SIZE	Start timeout timer	Get CRC
	Read character from inbuf	Nmb char read < MESSAGE_SIZE	Start timeout timer	Get Data
	Timeout	None	Update message statistics	Start
Get CRC	Read character from inbuf	CRC Byte Count == 2 CRC OK	Calculate CRC Update message statistics Execute command Stop timeout timer	Start
	Read character from inbuf	CRC Byte Count == 2 CRC NOT OK	Calculate CRC Update message statistics Stop timeout timer	Start
	Read character from inbuf	CRC Byte Count < 2	Start timeout timer	Get CRC
	Timeout	None	Update message statistics	Start

4.2.1 Message Statistics

Both ends of the protocol shall maintain the following set of parameters:

- Number of messages verified OK
- Number of messages failed.

If emulator DEBUG mode is enabled (through Set Parameter), a DEBUG event will be sent from emulator to host every time a message parses wrong. See the description of the debug event.

4.2.2 Timeout Timer

The time-out timer is a one-shot timer that is reset every time it is started. The Timeout Timer is used as a mechanism to set maximum inter-character delay – if the delay is larger it is considered illegal and parsing of the command stops.

5 Commands and responses

5.1 Master Commands

5.1.1 Sign off (CMND_SIGN_OFF: 0x00)

The master issues this command when it terminates a debugging session.

Slave state: All states.

Table 5-1. Message Format

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	Incremented for each packet sent.
MESSAGE_SIZE	0x01
TOKEN	ASCII 14
MESSAGE_BODY	None
CRC	2-byte CRC.

The MESSAGE_BODY consists of:

<MESSAGE_ID>

Table 5-2. Parameters

Parameter Name	Usage	Format
MESSAGE_ID	0x00 (CMND_SIGN_OFF)	[BYTE]

Table 5-3. Response

Parameter Name	Description
RSP_OK	The command was executed.
RSP_FAILED	The JTAGICE mkII is there, but it did not understand the command.

5.1.1.1 Internal Handling

M_MCU responds and may then enter idle state / power down mode.

5.1.2 Check if Emulator is present (CMND_GET_SIGN_ON: 0x01)

The master uses this command to see if an emulator is attached and alive.

Slave state: All states.

Table 5-4. Message Format.

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	Incremented for each packet sent.
MESSAGE_SIZE	0x01
TOKEN	ASCII 14
MESSAGE_BODY	None
CRC	2-byte CRC.

The MESSAGE_BODY consists of:

<MESSAGE_ID>

Table 5-5. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	0x01 (CMND_GET_SIGN_ON)	[BYTE]

Table 5-6. Response.

Parameter Name	Description
RSP_SIGN_ON	Contains sign on string for JTAGICE mkII. Compatibility issues with other emulators must be resolved. Sign on (RSP_SIGN_ON : 0x86)
RSP_FAILED	The JTAGICE mkII is there, but it did not understand the command.
TIMEOUT	No slave response within the defined timeout time. The slave is either not there, or it is not turned on.
Garbage	If serial communication is used, HW error UART configuration error (wrong baud rate, parity bit setting etc) If USB communication is used, HW error.

5.1.2.1 Internal Handling

M_MCU responds with sign on string. Sign-on to the S_MCU should already have been performed by the M_MCU upon power-up.

5.1.3 Write Emulator Parameter (CMND_SET_PARAMETER: 0x02)

The emulator hosts a number of set-up parameters. This command is used to write all parameters. The parameter ID identifies the parameter written.

Slave state: Depending on the parameter.

Table 5-7. Message Format.

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	Incremented for each packet sent.
MESSAGE_SIZE	The size of the MESSAGE_BODY in bytes.
TOKEN	ASCII 14
MESSAGE_BODY	See below
CRC	2-byte CRC.

The message body consists of:

<MESSAGE_ID, PARAMETER_ID, PARAMETER_VALUE>

Table 5-8. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	0x02 (CMND_SET_PARAMETER)	[BYTE]
PARAMETER_ID	Parameter ID	[BYTE]
PARAMETER_VALUE	Parameter value, can be of any size	[BYTE] * N, LSB first if applicable.

Table 5-9. Response.

Parameter Name	Description
RSP_OK	The parameter is written OK.
RSP_FAILED	The command was not understood by the JTAGICE mkII
RSP_ILLEGAL_PARAMETER	The JTAGICE mkII does not support the selected parameter (AVR Studio 4 – JTAGICE mkII FW incompatibility).
RSP_ILLEGAL_VALUE	The given value was invalid or out or range
RSP_ILLEGAL_EMULATOR_MODE	The operation cannot be performed in this emulator mode. (JTAG / DebugWire).
Timeout	No response from the slave within the timeout time.

The parameter description is found in a later section in this document.

5.1.3.1 Internal handling

M_MCU either responds itself, or forwards the request to the S_MCU, depending upon the PARAMETER_ID.

Table 5-10. USART Command Field.

Data	Usage
cmdID	CMND_SET_PARAMETER
ucPar0 (unsigned char)	PARAMETER_ID
ulPar1 (unsigned long)	PARAMETER_VALUE
ulPar2 (unsigned long)	<not used>

Table 5-11. USART Response.

Data	Usage
RSP_OK	Parameter written OK
RSP_ILLEGAL_PARAMETER	Parameter ID not known
RSP_ILLEGAL_EMULATOR_MODE	The parameter cannot be written in this emulator mode. (JTAG / DebugWire).

Table 5-12. FIFO Data (Command)

Data	Usage
<not used>	<not used>

Table 5-13. FIFO Data (Response).

Data	Usage
<not used>	<not used>

5.1.4 Read Emulator Parameter (CMND_GET_PARAMETER: 0x03)

Read emulator set-up parameters back to AVR Studio. Parameter descriptions found in section Write Emulator Parameter.

Slave state: Depending on the parameter.

Table 5-14. Message Format.

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	Incremented for each packet sent.
MESSAGE_SIZE	0x02
TOKEN	ASCII 14
MESSAGE_BODY	Defined below
CRC	2-byte CRC.

The MESSAGE_BODY consists of:

<MESSAGE_ID, PARAMETER_ID>

Table 5-15. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	0x03 (CMND_GET_PARAMETER)	[BYTE]
PARAMETER_ID	The requested emulator parameter	[BYTE]

Table 5-16. Response.

Parameter Name	Description
RSP_PARAMETER	Returns the requested parameter. Parameter return (RSP_PARAMETER : 0x81)
RSP_FAILED	The command was not understood by the JTAGICE mkII
RSP_ILLEGAL_PARAMETER	The JTAGICE mkII does not support the selected parameter (AVR Studio 4 – JTAGICE mkII FW incompatibility).
RSP_ILLEGAL_EMULATOR_MODE	The operation cannot be performed in this emulator mode. (JTAG / DebugWire).
Timeout	No response from the slave within the timeout time.

5.1.4.1 Internal handling

M_MCU either responds itself, or forwards the request to the S_MCU, depending upon the PARAMETER_ID.

Table 5-17. USART Command Field.

Data	Usage
CmdID	CMND_GET_PARAMETER
ucPar0 (unsigned char)	PARAMETER_ID
ulPar1 (unsigned long)	<not used>
ulPar2 (unsigned long)	<not used>

Table 5-18. USART Response.

Data	Usage
RSP_OK	Parameter read OK
RSP_FAILED	Parameter could not be read
RSP_ILLEGAL_PARAMETER	Parameter ID not known
RSP_ILLEGAL_EMULATOR_MODE	The parameter cannot be read in this emulator mode. (JTAG / DebugWire).

Table 5-19. FIFO Data (Command).

Data	Usage
<not used>	<not used>

Table 5-20. FIFO Data (Response).

Data	Usage
FIFO_SOF	Start-of-frame
unsigned long value (LSB first) **OR** *unsigned int* value (LSB first) **OR** *unsigned char* value	(M_MCU reads the number of bytes it expects for the that PARAMETER_ID)
FIFO_EOF	End-of-frame

5.1.5 Write Memory (CMND_WRITE_MEMORY: 0x04)

Write a memory block to any address in any memory area.

Slave state: STOPPED.

Table 5-21. Message Format.

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	Incremented for each packet sent.
MESSAGE_SIZE	The size of the MESSAGE_BODY in bytes.
TOKEN	ASCII 14
MESSAGE_BODY	See below
CRC	2-byte CRC.

The MESSAGE_BODY consists of:

<MESSAGE_ID, MEMORY_TYPE, BYTE_COUNT, START_ADDRESS, DATA>

Table 5-22. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	0x04 (CMND_WRITE_MEMORY)	[BYTE]
MEMORY_TYPE	Memory type (Flash, SRAM, EEPROM...)	[BYTE]
BYTE_COUNT	Number of bytes to be written	[BYTE]*4 LSB first
START_ADDRESS	Start memory address	[BYTE]*4 LSB first
DATA	The data	[BYTE] * BYTE_COUNT

Definitions and parameter values of the different memory types are found in the Memory Types section.

Table 5-23. Response.

Parameter Name	Description
RSP_OK	The memory is written OK.
RSP_FAILED	The command was not understood by the JTAGICE mkII
RSP_ILLEGAL_MEMORY_TYPE	The JTAGICE mkII does not support the selected memory.
RSP_ILLEGAL_MEMORY_RANGE	The memory write was outside the bounds of the selected memory area.
RSP_ILLEGAL_EMULATOR_MODE	The operation cannot be performed in this emulator mode. (JTAG / DebugWire).
RSP_ILLEGAL_MCU_STATE	The operation cannot be performed with the target MCU in its current state.
Timeout	No response from the slave within the timeout time.

Note: The existing JTAGICE FW will not allow the user to write an ODD number of bytes to any memory area. The new FW is BYTE oriented and will allow handling of an ODD number of bytes for all memory areas.

5.1.5.1 Internal handling

The request is passed to the S_MCU. If MEMORY_TYPE is EVENT_MEMORY, this is handled by the M_MCU.

Table 5-24. USART Command Field.

Data	Usage
cmdID	CMND_WRITE_MEMORY
ucPar0 (unsigned char)	MEMORY_TYPE
UlPar1 (unsigned long)	BYTE_COUNT
UlPar2 (unsigned long)	START_ADDRESS

Table 5-25. USART Response.

Data	Usage
RSP_OK	Memory written OK
RSP_FAILED	Memory could not be written
RSP_ILLEGAL_MEMORY_TYPE	Memory type invalid
RSP_ILLEGAL_MEMORY_RANGE	Memory range invalid
RSP_SOF_FAILED	FIFO Start frame invalid
RSP_EOF_FAILED	FIFO End frame invalid

Table 5-26. FIFO Data (Command).

Data	Usage
FIFO_SOF	Start-of-frame
BYTE_COUNT bytes of raw data to be written	
FIFO_EOF	End-of-frame

Table 5-27. FIFO Data (Response).

Data	Usage
<not used>	<not used>

5.1.6 Read Memory (CMND_READ_MEMORY: 0x05)

Read a memory block from any address in any memory area.

Slave state: STOPPED

Table 5-28. Message Format.

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	Incremented for each packet sent.
MESSAGE_SIZE	10 bytes.
TOKEN	ASCII 14
MESSAGE_BODY	See below
CRC	2-byte CRC.

The MESSAGE_BODY consists of:

<MEMORY_TYPE, BYTE_COUNT, START_ADDRESS>

Table 5-29. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	0x05 (CMND_READ_MEMORY)	[BYTE]
MEMORY_TYPE	Memory type (Flash, SRAM, EEPROM...)	[BYTE]
BYTE_COUNT	Number of bytes to be read	[BYTE]*4 LSB first
START_ADDRESS	Start memory address	[BYTE]*4 LSB first

Definitions and parameter values of the different memory types are found in the Memory Types section.

Table 5-30. Response.

Parameter Name	Description
RSP_MEMORY	The memory is read OK. Memory read (RSP_MEMORY : 0x82)
RSP_FAILED	The command was not understood by the JTAGICE mkII
RSP_ILLEGAL_MEMORY_TYPE	The JTAGICE mkII does not support the selected memory.
RSP_ILLEGAL_MEMORY_RANGE	The memory write was outside the bounds of the selected memory area.
RSP_ILLEGAL_EMULATOR_MODE	The operation cannot be performed in this emulator mode. (JTAG / DebugWire).
RSP_ILLEGAL_MCU_STATE	The operation cannot be performed with the target MCU in its current state.
Timeout	No response from the slave within the timeout time.

5.1.6.1 Internal handling

The request is passed to the S_MCU. If MEMORY_TYPE is EVENT_MEMORY, this is handled by the M_MCU.

Table 5-31. USART Command Field.

Data	Usage
CmdID	CMND_READ_MEMORY
ucPar0 (unsigned char)	MEMORY_TYPE
ulPar1 (unsigned long)	BYTE_COUNT
ulPar2 (unsigned long)	START_ADDRESS

Table 5-32. USART Response.

Data	Usage
RSP_MEMORY	Memory read OK
RSP_FAILED	Memory could not be read
RSP_ILLEGAL_MEMORY_TYPE	Memory type invalid
RSP_ILLEGAL_MEMORY_RANGE	Memory range invalid

Table 5-33. FIFO Data (Command).

Data	Usage
<not used>	<not used>

Table 5-34. FIFO Data (Response).

Data	Usage
FIFO_SOF	Start-of-frame
BYTE_COUNT bytes of raw data read	
FIFO_EOF	End-of-frame

5.1.7 Write Program Counter (CMND_WRITE_PC: 0x06)

Writes the AVR Program Counter.

Slave state: STOPPED.

Table 5-35. Message Format.

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	Incremented for each packet sent.
MESSAGE_SIZE	5
TOKEN	ASCII 14
MESSAGE_BODY	See below.
CRC	2-byte CRC.

The MESSAGE_BODY consists of:

<MESSAGE_ID, PROGRAM_COUNTER>

Table 5-36. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	0x06 (CMND_WRITE_PC)	[BYTE]
PROGRAM_COUNTER	Program Counter	[BYTE]*4 LSB first.

Table 5-37. Response.

Parameter Name	Description
RSP_OK	The program counter was written OK.
RSP_FAILED	The command was not understood by the JTAGICE mkII
RSP_ILLEGAL_EMULATOR_MODE	The operation cannot be performed in this emulator mode. (JTAG / DebugWire).
RSP_ILLEGAL_MCU_STATE	The operation cannot be performed with the target MCU in its current state.
Timeout	No response from the slave within the timeout time.

5.1.7.1 Internal handling

The request is passed to the S_MCU.

Table 5-38. USART Command Field.

Data	Usage
CmdID	CMND_WRITE_PC
ucPar0 (unsigned char)	<not used>
ulPar1 (unsigned long)	P.C. value LSB first
ulPar2 (unsigned long)	<not used>

Table 5-39. USART Response.

Data	Usage
RSP_OK	PC written OK
RSP_FAILED	PC could not be written
RSP_ILLEGAL_EMULATOR_MODE	PC cannot be written in this emulator mode. (JTAG / DebugWire).
RSP_ILLEGAL_MCU_STATE	PC cannot be written with the target MCU in its current state.

Table 5-40. FIFO Data (Command).

Data	Usage
<not used>	<not used>

Table 5-41. FIFO Data (Response)

Data	Usage
<not used>	<not used>

5.1.8 Read Program Counter (CMND_READ_PC: 0x07)

Reads the AVR Program Counter.

Slave state: STOPPED.

Table 5-42. Message Format.

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	Incremented for each packet sent.
MESSAGE_SIZE	1
TOKEN	ASCII 14
MESSAGE_BODY	See below.
CRC	2-byte CRC.

The MESSAGE_BODY consists of:

< MESSAGE_ID>

Table 5-43. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	0x07 (CMND_READ_PC)	[BYTE]

Table 5-44. Response.

Parameter Name	Description
RSP_PC	The program counter is returned. Program Counter Read (RSP_PC : 0x84)
RSP_FAILED	The command was not understood by the JTAGICE mkII
RSP_ILLEGAL_EMULATOR_MODE	The operation cannot be performed in this emulator mode. (JTAG / DebugWire).
RSP_ILLEGAL_MCU_STATE	The operation cannot be performed with the target MCU in its current state.
Timeout	No response from the slave within the timeout time.

5.1.8.1 Internal handling

The request is passed to the S_MCU

Table 5-45. USART Command Field.

Data	Usage
CmdID	CMND_READ_PC
ucPar0 (unsigned char)	<not used>
ulPar1 (unsigned long)	<not used>
ulPar2 (unsigned long)	<not used>

Table 5-46. USART Response.

Data	Usage
RSP_PC	PC read OK
RSP_FAILED	PC could not be read
RSP_ILLEGAL_EMULATOR_MODE	PC cannot be read in this emulator mode. (JTAG / DebugWire).
RSP_ILLEGAL_MCU_STATE	PC cannot be read with the target MCU in its current state.

Table 5-47. FIFO Data (Command).

Data	Usage
<not used>	<not used>

Table 5-48. FIFO Data (Response).

Data	Usage
FIFO_SOF	Start-of-frame
4 bytes PC, LSB first	
FIFO_EOF	End-of-frame

5.1.9 Start Program Execution (CMND_GO: 0x08)

Starts program execution at current Program Counter address.

Slave state: STOPPED or RUNNING. When the command is executed the state either changes to or remains in RUNNING.

Table 5-49. Message Format.

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	Incremented for each packet sent.
MESSAGE_SIZE	1
TOKEN	ASCII 14
MESSAGE_BODY	None
CRC	2-byte CRC.

The MESSAGE_BODY consists of:

< MESSAGE_ID >

Table 5-50. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	CMND_GO : 0x08	[BYTE]

Table 5-51. Response.

Parameter Name	Description
RSP_OK	The command was executed.
RSP_FAILED	The command was not understood by the JTAGICE mkII
RSP_ILLEGAL_EMULATOR_MODE	The operation cannot be performed in this emulator mode. (JTAG / DebugWire).
Timeout	No response from the slave within the timeout time.

5.1.9.1 Internal handling

The request is passed to the S_MCU.

Table 5-52. USART Command Field.

Data	Usage
CmdID	CMND_GO
ucPar0 (unsigned char)	<not used>
ulPar1 (unsigned long)	INST H | INST L | INST FLAG | XX
ulPar2 (unsigned long)	<not used>

Table 5-53. USART Response.

Data	Usage
RSP_OK	Command executed OK
RSP_FAILED	Command failed
RSP_ILLEGAL_EMULATOR_MODE	The operation cannot be performed in this emulator mode. (JTAG / DebugWire).
RSP_ILLEGAL_MCU_STATE	The operation cannot be performed with the target MCU in its current state.

Table 5-54. FIFO Data (Command).

Data	Usage
<not used>	<not used>

Table 5-55. FIFO Data (Response).

Data	Usage
<not used>	<not used>

5.1.10 Single Step (CMND_SINGLE_STEP: 0x09)

Starts one-step execution at current Program Counter address.

Slave state: STOPPED. After execution the slave is in RUNNING state until it hits the next breakpoint. Then a Break Event is issued, and the slave returns to STOPPED state.

Table 5-56. Message Format.

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	Incremented for each packet sent.
MESSAGE_SIZE	3
TOKEN	ASCII 14
MESSAGE_BODY	High level or low level flag, see below.
CRC	2-byte CRC.

The MESSAGE_BODY consists of:

<MESSAGE_ID, FLAG>

Table 5-57. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	0x09 (CMND_SINGLE_STEP)	[BYTE]
FLAG	0x01 : Low level 0x02 : High level.	[BYTE]
STEP_MODE	0x00: STEP_OVER 0x01: STEP_INTO 0x02: STEP_OUT	[BYTE]

Low level or High level debugging (DebugWire, but also next implementation of the JTAGICE).

Table 5-58. Response.

Parameter Name	Description
RSP_OK	The command is executed.
RSP_FAILED	The command was not understood by the JTAGICE mkII
RSP_ILLEGAL_EMULATOR_MODE	The operation cannot be performed in this emulator mode. (JTAG / DebugWire).
RSP_ILLEGAL_MCU_STATE	The operation cannot be performed with the target MCU in its current state.
Timeout	No response from the slave within the timeout time.

5.1.10.1 Internal handling

The request is passed to the S_MCU.

Table 5-59. USART Command Field.

Data	Usage
CmdID	CMND_SINGLE_STEP
ucPar0 (unsigned char)	FLAG
ulPar1 (unsigned long)	INST H | INST L | CTRL_FLAGS | STEP_MODE
ulPar2 (unsigned long)	P.C. for B.P.

Table 5-60. CTRL_FLAGS (can be set independently).

define	value	Description
MASK_OVERRIDE	0x01	0 = normal mode, 1 = MONINST override mode

Table 5-61. USART Response.

Data	Usage
RSP_OK	Command executed OK
RSP_FAILED	Command failed
RSP_ILLEGAL_EMULATOR_MODE	The operation cannot be performed in this emulator mode. (JTAG / DebugWire).
RSP_ILLEGAL_MCU_STATE	The operation cannot be performed with the target MCU in its current state.

Table 5-62. FIFO Data (Command).

Data	Usage
<not used>	<not used>

Table 5-63. FIFO Data (Response).

Data	Usage
<not used>	<not used>

5.1.11 Stop Program Execution (CMND_FORCED_STOP: 0x0A)

Stops program execution.

Slave state: STOPPED or RUNNING. After execution of the command the slave is in STOPPED state.

Table 5-64. Message Format.

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	Incremented for each packet sent.
MESSAGE_SIZE	2 Bytes
TOKEN	ASCII 14
MESSAGE_BODY	See below
CRC	2-byte CRC.

The MESSAGE_BODY consists of:

<MESSAGE_ID, MODE>

Table 5-65. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	0x0A (CMND_FORCED_STOP)	[BYTE]
MODE	0x01: LOW LEVEL 0x02: HIGH LEVEL (stops at next high level statement)	[BYTE]

Table 5-66. Response.

Parameter Name	Description
RSP_OK	The command is executed.
RSP_FAILED	The command was not understood by the JTAGICE mkII
RSP_ILLEGAL_EMULATOR_MODE	The operation cannot be performed in this emulator mode. (JTAG / DebugWire).
Timeout	No response from the slave within the timeout time.

5.1.11.1 Internal handling

The request is passed to the S_MCU.

Table 5-67. USART Command Field.

Data	Usage
cmdID	CMND_FORCED_STOP
ucPar0 (unsigned char)	<not used>
ulPar1 (unsigned long)	<not used>
ulPar2 (unsigned long)	<not used>

Table 5-68. USART Response.

Data	Usage
RSP_OK	Command executed OK
RSP_FAILED	Command failed
RSP_ILLEGAL_EMULATOR_MODE	The operation cannot be performed in this emulator mode. (JTAG / DebugWire).

Table 5-69. FIFO Data (Command).

Data	Usage
<not used>	<not used>

Table 5-70. FIFO Data (Response).

Data	Usage
FIFO_SOF	Start-of-frame
4 bytes PC, LSB first	
FIFO_EOF	End-of-frame

5.1.12 Reset User Program (CMND_RESET: 0x0B)

Emulator performs all actions to restart program execution

Slave state: Any. If the slave is in RUNNING mode prior to this operation, the slave will change to STOPPED state prior to the execution of this command

Table 5-71. Message Format.

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	Incremented for each packet sent.
MESSAGE_SIZE	Size of message body
TOKEN	ASCII 14
MESSAGE_BODY	See below High level or low level flag. High level typically returns to main. Give a procedure. Give low level reset with “run afterwards”.
CRC	2-byte CRC.

The MESSAGE_BODY consists of:

<MESSAGE_ID, FLAG>

Table 5-72. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	0x0B (CMND_RESET)	[BYTE]
FLAG	0x01: Low level 0x02: High level 0x04: Reset w/ MonCom disable. High level typically returns to start and executes to main or to any other function given (breakpoint set). Gives a low level reset with a run command afterwards.	[BYTE]

Table 5-73. Response.

Parameter Name	Description
RSP_OK	The command is executed.
RSP_FAILED	The command was not understood by the JTAGICE mkII
RSP_ILLEGAL_EMULATOR_MODE	The operation cannot be performed in this emulator mode. (JTAG / DebugWire).
Timeout	No response from the slave within the timeout time.

5.1.12.1 Internal handling

The request is passed to the S_MCU.

Table 5-74. USART Command Field.

Data	Usage
cmdID	CMND_RESET
ucPar0 (unsigned char)	FLAG
ulPar1 (unsigned long)	<not used>
ulPar2 (unsigned long)	<not used>

Table 5-75. USART Response.

Data	Usage
RSP_OK	Command executed OK
RSP_FAILED	Command failed
RSP_ILLEGAL_EMULATOR_MODE	The operation cannot be performed in this emulator mode. (JTAG / DebugWire).

Table 5-76. FIFO Data (Command).

Data	Usage
<not used>	<not used>

Table 5-77. FIFO Data (Response).

Data	Usage
<not used>	<not used>

5.1.13 Set Device Descriptor (CMND_SET_DEVICE_DESCRIPTOR: 0x0C)

Transmits all parameters relevant for a specific device set-up

Slave state: STOPPED.

Table 5-78. Message Format.

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	Incremented for each packet sent.
MESSAGE_SIZE	Size of message body
TOKEN	ASCII 14
MESSAGE_BODY	All parameters for a specific device set-up.
CRC	2-byte CRC.

The MESSAGE_BODY consists of:

<MESSAGE_ID, PARAMETERS>

Table 5-79. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	0x0C (CMND_SET_DEVICE_DESCRIPTOR)	[BYTE]
PARAMETERS	All parameters required to set up a device in raw binary form, identical to the current JTAGICE descriptor.	[BYTE] * 298

Table 5-80. Response.

Parameter Name	Description
RSP_OK	The command is executed.
RSP_FAILED	The command was not understood by the JTAGICE mkII
RSP_ILLEGAL_EMULATOR_MODE	The operation cannot be performed in this emulator mode. (JTAG / DebugWire).
RSP_ILLEGAL_MCU_STATE	The operation cannot be performed with the target MCU in its current state.
Timeout	No response from the slave within the timeout time.

5.1.13.1 Internal handling

The request is passed to the S_MCU.

Table 5-81. USART Command Field.

Data	Usage
cmdID	CMND_SET_DEVICE_DESCRIPTOR
ucPar0 (unsigned char)	<not used>
ulPar1 (unsigned long)	<not used>
ulPar2 (unsigned long)	<not used>

Table 5-82. USART Response.

Data	Usage
RSP_OK	Command executed OK
RSP_FAILED	Command failed
RSP_ILLEGAL_EMULATOR_MODE	The operation cannot be performed in this emulator mode. (JTAG / DebugWire).
RSP_ILLEGAL_MCU_STATE	The operation cannot be performed with the target MCU in its current state.

Table 5-83. FIFO Data (Command).

Data	Usage
FIFO_SOF	Start-of-frame
292 bytes descriptor structure in binary format, identical to current JTAGICE descriptor	
FIFO_EOF	End-of-frame

Table 5-84. FIFO Data (Response).

Data	Usage
<not used>	<not used>

5.1.14 Erase Page SPM (CMND_ERASEPAGE_SPM: 0x0D)

Erases a whole page in Flash memory

Slave state: STOPPED

Table 5-85. Message Format.

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
MESSAGE_ID	0x0D (CMND_ERASEPAGE_SPM)
SEQUENCE_NUMBER	Incremented for each packet sent.
MESSAGE_SIZE	5
TOKEN	ASCII 14
MESSAGE_BODY	See below
CRC	2-byte CRC.

The MESSAGE_BODY consists of:

<MESSAGE_ID, PAGE_ADDRESS>

Table 5-86. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	0x0D (CMND_ERASEPAGE_SPM)	[BYTE]
PAGE_ADDRESS	Address to the page to be erased	[BYTE]*4, LSB first

Table 5-87. Response.

Parameter Name	Description
RSP_OK	The command is executed.
RSP_FAILED	The command was not understood by the JTAGICE mkII
RSP_ILLEGAL_EMULATOR_MODE	The operation cannot be performed in this emulator mode. (JTAG / DebugWire).
RSP_ILLEGAL_MCU_STATE	The operation cannot be performed with the target MCU in its current state.
Timeout	No response from the slave within the timeout time.

5.1.14.1 Internal handling

The request is passed to the S_MCU.

Table 5-88. USART Command Field.

Data	Usage
cmdID	CMND_ERASEPAGE_SPM
ucPar0 (unsigned char)	<not used>
ulPar1 (unsigned long)	Page address LSB first
ulPar2 (unsigned long)	<not used>

Table 5-89. USART Response.

Data	Usage
RSP_OK	Command executed OK
RSP_FAILED	Command failed
RSP_ILLEGAL_EMULATOR_MODE	The operation cannot be performed in this emulator mode. (JTAG / DebugWire).
RSP_ILLEGAL_MCU_STATE	The operation cannot be performed with the target MCU in its current state.

Table 5-90. FIFO Data (Command).

Data	Usage
<not used>	<not used>

Table 5-91. FIFO Data (Response).

Data	Usage
<not used>	<not used>

5.1.15 Get Sync (CMND_GET_SYNC 0x0F)

Sent from AVR Studio in order to regain synchronization with JTAGICE mkII, if lost

Slave state: any. If slave responds to the command the slave will be in state STOPPED after command execution.

Table 5-92. Message Format.

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	Incremented for each packet sent.
MESSAGE_SIZE	1
TOKEN	ASCII 14
MESSAGE_BODY	See below
CRC	2-byte CRC.

The MESSAGE_BODY consists of:

<MESSAGE_ID>

Table 5-93. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	0x0F (CMND_GET_SYNC)	[BYTE]

Table 5-94. Response.

Parameter Name	Description
RSP_OK	The command is executed.
RSP_FAILED	The command was not understood by the JTAGICE mkII.
Timeout	No response from the slave within the timeout time.

5.1.15.1 Internal handling

The request is handled by the M_MCU.

5.1.16 Self test (CMND_SELFTEST: 0x10)

Makes the JTAGICE mkII perform self test and report back to the master.

Slave state: STOPPED

Table 5-95. Message Format.

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character).
SEQUENCE_NUMBER	Incremented for each packet sent.
MESSAGE_SIZE	Size of message body.
TOKEN	ASCII 14
MESSAGE_BODY	Parameters for self test (choose what to test).
CRC	2-byte CRC.

The MESSAGE_BODY consists of:

<MESSAGE_ID, FLAGS>

Table 5-96. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	0x0F (CMND_SELFTEST)	[BYTE]
FLAGS	FLAGS identifying which tests to be performed. bit 7: internal tests (SRAM, FIFO…) bit 6: <not used> bit 5: <not used> bit 4: <not used> bit 3: STK500 RESET JUMPER detector bit 2: JTAG PUSH PULL bit 1: DebugWire Capacitance bit 0: DebugWire PUSH PULL	[BYTE]

Table 5-97. Response.

Parameter Name	Description
RSP_SELFTEST	The command is executed. Selftest (RSP_SELFTEST : 0x85)
RSP_FAILED	The command was not understood by the JTAGICE mkII.
RSP_ILLEGAL_EMULATOR_MODE	The operation cannot be performed in this emulator mode. (JTAG / DebugWire).
RSP_ILLEGAL_MCU_STATE	The operation cannot be performed with the target MCU in its current state.
Timeout	No response from the slave within the timeout time.

5.1.16.1 Internal handling

The request is handled by the M_MCU, making use of internal commands to request the slave to perform tests (documented later).

5.1.17 Set Breakpoint (CMND_SET_BREAK: 0x11)

Set a breakpoint.

Slave state: STOPPED

Table 5-98. Message Format.

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	Incremented for each packet sent.
MESSAGE_SIZE	Size of message body
TOKEN	ASCII 14
MESSAGE_BODY	Breakpoint setting (see format below).
CRC	2-byte CRC.

The MESSAGE_BODY consists of:

<MESSAGE_ID, TYPE, NUMBER, ADDRESS, MODE>

Table 5-99. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	0x11 (CMND_SET_BREAK)	[BYTE]
TYPE	0x01: program memory breakpoint 0x02: data breakpoint 0x03:mask for data breakpoint	[BYTE]
NUMBER	Breakpoint number, 0x01 to 0x04 (up to 4 breakpoints are supported 0x00 for SW BP	[BYTE]
ADDRESS	Break address of mask value	[BYTE] * 4, LSB first
MODE	0x00: Break on memory read 0x01: Break on memory write 0x02: Break on memory read or write. 0x03: Program breakpoint	[BYTE]

Table 5-100. Response.

Parameter Name	Description
RSP_OK	The command is executed.
RSP_FAILED	The command was not understood by the JTAGICE mkII.
RSP_ILLEGAL_BREAKPOINT	The breakpoint number does not exist. Illegal breakpoint (RSP_ILLEGAL_BREAKPOINT: 0xA8)
RSP_ILLEGAL_EMULATOR_MODE	The operation cannot be performed in this emulator mode. (JTAG / DebugWire).
RSP_ILLEGAL_MCU_STATE	The operation cannot be performed with the target MCU in its current state.
RSP_ILLEGAL_BREAKPOINT	The breakpoint number is not supported (in this mode)
Timeout	No response from the slave within the timeout time.

5.1.17.1 Internal handling

The request is passed to the S_MCU.

Table 5-101. USART Command Field.

Data	Usage
cmdID	CMND_SET_BREAK
ucPar0 (unsigned char)	TYPE (data / program)
ulPar1 (unsigned long)	ADDRESS 4 bytes LSB first
ulPar2 (unsigned long)	XX | XX | MODE | NUMBER

Table 5-102. USART Response.

Data	Usage
RSP_OK	Command executed OK
RSP_FAILED	Command failed
RSP_ILLEGAL_EMULATOR_MODE	The operation cannot be performed in this emulator mode. (JTAG / DebugWire).
RSP_ILLEGAL_MCU_STATE	The operation cannot be performed with the target MCU in its current state.
RSP_ILLEGAL_BREAKPOINT	The breakpoint number is not supported (in this mode)

Table 5-103. FIFO Data (Command).

Data	Usage
<not used>	<not used>

Table 5-104. FIFO Data (Response).

Data	Usage
<not used>	<not used>

5.1.18 Get Breakpoint (CMND_GET_BREAK: 0x12)

Used by the host to read back the current breakpoint setting in the JTAGICE mkII.

Slave state: STOPPED

Table 5-105. Message Format.

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	Incremented for each packet sent.
MESSAGE_SIZE	2
TOKEN	ASCII 14
MESSAGE_BODY	See below
CRC	2-byte CRC.

The MESSAGE_BODY consists of:

<MESSAGE_ID, NUMBER>

Table 5-106. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	0x12 (CMND_GET_BREAK)	[BYTE]
NUMBER	Breakpoint number, 0x01 to 0x04 (up to 4 breakpoints are supported	[BYTE]

Table 5-107. Response.

Parameter Name	Description
RSP_GET_BREAK	The command is executed, breakpoint returned. Breakpoint read (RSP_GET_BREAK: 0x83)
RSP_FAILED	The command was not understood by the JTAGICE mkII
RSP_ILLEGAL_EMULATOR_MODE	The operation cannot be performed in this emulator mode. (JTAG / DebugWire).
RSP_ILLEGAL_MCU_STATE	The operation cannot be performed with the target MCU in its current state.
RSP_ILLEGAL_BREAKPOINT	The breakpoint number is not supported (in this mode)
Timeout	No response from the slave within the timeout time.

5.1.18.1 Internal handling

The request is passed to the S_MCU.

Table 5-108. USART Command Field.

Data	Usage
cmdID	CMND_GET_BREAK
ucPar0 (unsigned char)	NUMBER
ulPar1 (unsigned long)	<not used>
ulPar2 (unsigned long)	<not used>

Table 5-109. USART Response.

Data	Usage
RSP_GET_BREAK	Command executed OK Breakpoint read (RSP_GET_BREAK : 0x83)
RSP_FAILED	Command failed
RSP_ILLEGAL_EMULATOR_MODE	The operation cannot be performed in this emulator mode. (JTAG / DebugWire).
RSP_ILLEGAL_MCU_STATE	The operation cannot be performed with the target MCU in its current state.
RSP_ILLEGAL_BREAKPOINT	The breakpoint number is not supported (in this mode)

Table 5-110. FIFO Data (Command).

Data	Usage
<not used>	<not used>

Table 5-111. FIFO Data (Response).

Data	Usage
FIFO_SOF	Start-of-frame
TYPE 0x01 : program memory breakpoint 0x02 : data breakpoint	[BYTE]
Breakpoint address	[BYTE] * 4 LSB first
MODE: 0x00 : (Program breakpoint – not used) 0x01 : Break on memory read 0x02 : Break on memory write 0x03 : Break on memory read or write.	[BYTE]
FIFO_EOF	End-of-frame

5.1.19 Chip erase (CMD_CHIP_ERASE: 0x13)

Used in order to do a complete chip erase. Note that, this command in not in use for DebugWire. (Flash pages are automatically being erased.)

Table 5-112. Message Format.

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	Incremented for each packet sent.
MESSAGE_SIZE	1
TOKEN	ASCII 14
MESSAGE_BODY	See below
CRC	2-byte CRC.

The MESSAGE_BODY consists of:

<MESSAGE_ID>

Table 5-113. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	0x13 (CMD_CHIP_ERASE)	[BYTE]

Table 5-114. Response.

Parameter Name	Description
RSP_OK	The command is executed.
RSP_FAILED	The command was not understood by the JTAGICE mkII
RSP_ILLEGAL_EMULATOR_MODE	The operation cannot be performed in this emulator mode. (JTAG / DebugWire).
RSP_ILLEGAL_MCU_STATE	The operation cannot be performed with the target MCU in its current state.
Timeout	No response from the slave within the timeout time.

5.1.19.1 Internal handling

The request is passed to the S_MCU.

Table 5-115. USART Command Field.

Data	Usage
cmdID	CMND_CHIP_ERASE
ucPar0 (unsigned char)	<not used>
ulPar1 (unsigned long)	<not used>
ulPar2 (unsigned long)	<not used>

Table 5-116. USART Response.

Data	Usage
RSP_OK	Command executed OK
RSP_FAILED	Command failed
RSP_ILLEGAL_EMULATOR_MODE	The operation cannot be performed in this emulator mode. (JTAG / DebugWire).
RSP_ILLEGAL_MCU_STATE	The operation cannot be performed with the target MCU in its current state.

Table 5-117. FIFO Data (Command).

Data	Usage
<not used>	<not used>

Table 5-118. FIFO Data (Response).

Data	Usage
<not used>	<not used>

5.1.20 Enter programming mode (CMND_ENTER_PROGMODE: 0x14)

Enter programming mode.

Table 5-119. Message Format.

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	Incremented for each packet sent.
MESSAGE_SIZE	1
TOKEN	ASCII 14
MESSAGE_BODY	See below
CRC	2-byte CRC.

The MESSAGE_BODY consists of:

<MESSAGE_ID>

Table 5-120. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	0x14 (CMND_ENTER_PROGMODE)	[BYTE]

Table 5-121. Response.

Parameter Name	Description
RSP_OK	The command is executed.
RSP_FAILED	The command was not understood by the JTAGICE mkII
RSP_ILLEGAL_JTAG_ID	The JTAG ID does not match the target device's. Illegal JTAG ID (RSP_ILLEGAL_JTAG_ID : 0xA9)
RSP_ILLEGAL_EMULATOR_MODE	The operation cannot be performed in this emulator mode. (JTAG / DebugWire).
RSP_ILLEGAL_MCU_STATE	The operation cannot be performed with the target MCU in its current state.
Timeout	No response from the slave within the timeout time.

5.1.20.1 Internal handling

The request is passed to the S_MCU.

Table 5-122. USART Command Field.

Data	Usage
cmdID	CMND_ENTER_PROGMODE
ucPar0 (unsigned char)	<not used>
ulPar1 (unsigned long)	<not used>
ulPar2 (unsigned long)	<not used>

Table 5-123. USART Response.

Data	Usage
RSP_OK	Command executed OK
RSP_FAILED	Command failed
RSP_ILLEGAL_EMULATOR_MODE	The operation cannot be performed in this emulator mode. (JTAG / DebugWire).
RSP_ILLEGAL_MCU_STATE	The operation cannot be performed with the target MCU in its current state.
RSP_ILLEGAL_JTAG_ID	The operation cannot be performed since the JTAG ID is wrong.

Table 5-124. FIFO Data (Command).

Data	Usage
<not used>	<not used>

Table 5-125. FIFO Data (Response)

Data	Usage
<not used>	<not used>

5.1.21 Leave programming mode (CMND_LEAVE_PROGMODE: 0x15)

Leave programming mode.

Table 5-126. Message Format.

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	Incremented for each packet sent.
MESSAGE_SIZE	1
TOKEN	ASCII 14
MESSAGE_BODY	See below
CRC	2-byte CRC.

The MESSAGE_BODY consists of:

<MESSAGE_ID>

Table 5-127. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	0x15 (CMND_LEAVE_PROGMODE)	[BYTE]

Table 5-128. Response.

Parameter Name	Description
RSP_OK	The command is executed.
RSP_FAILED	The command was not understood by the JTAGICE mkII
RSP_ILLEGAL_EMULATOR_MODE	The operation cannot be performed in this emulator mode. (JTAG / DebugWire).
RSP_ILLEGAL_MCU_STATE	The operation cannot be performed with the target MCU in its current state.
Timeout	No response from the slave within the timeout time.

5.1.21.1 Internal handling

The request is passed to the S_MCU.

Table 5-129. USART Command Field.

Data	Usage
cmdID	CMND_LEAVE_PROGMODE
ucPar0 (unsigned char)	<not used>
ulPar1 (unsigned long)	<not used>
ulPar2 (unsigned long)	<not used>

Table 5-130. USART Response.

Data	Usage
RSP_OK	Command executed OK
RSP_FAILED	Command failed
RSP_ILLEGAL_EMULATOR_MODE	The operation cannot be performed in this emulator mode. (JTAG / DebugWire).
RSP_ILLEGAL_MCU_STATE	The operation cannot be performed with the target MCU in its current state.

Table 5-131. FIFO Data (Command).

Data	Usage
<not used>	<not used>

Table 5-132. FIFO Data (Response).

Data	Usage
<not used>	<not used>

5.1.22 Write Emulator Parameters (CMND_SET_N_PARAMETERS: 0x16)

Note: Not currently implemented! This is an intended optimization for future use whereby several parameters can be set in one single call.

The emulator hosts a number of set-up parameters. This command is used to write several parameters in one call. The parameter ID identifies the parameter written.

Slave state: Depending on the parameter.

Table 5-133. Message Format.

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	Incremented for each packet sent.
MESSAGE_SIZE	The size of the MESSAGE_BODY in bytes.
TOKEN	ASCII 14
MESSAGE_BODY	See below
CRC	2-byte CRC.

The MESSAGE_BODY consists of:

<MESSAGE_ID, NUMBER, (1…N) {PARAMETER_ID, PARAMETER_VALUE}>

Table 5-134. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	0x16 (CMND_SET_N_PARAMETERS)	[BYTE]
NUMBER	The number of parameters being set	[BYTE]
1...N {		
PARAMETER_ID_N	Parameter ID	[BYTE]
PARAMETER_VALUE_N	Parameter value, can be of any size	[BYTE] * 4

All parameters are transferred as LONG values, which are mapped to the smaller memory types by the JTAGICE mkII. Parameters larger than 4 bytes are not supported.

Table 5-135. Response.

Parameter Name	Description
RSP_OK	The parameter is written OK.
RSP_FAILED	The command was not understood by the JTAGICE mkII
RSP_SET_N_PARAMETERS	One or more errors occurred whilst setting parameters. Response Set Parameters (RSP_SET_N_PARAMETERS: 0xA7)
RSP_ILLEGAL_EMULATOR_MODE	The operation cannot be performed in this emulator mode. (JTAG / DebugWire).
Timeout	No response from the slave within the timeout time.

The parameter description is found in a later section in this document.

5.1.22.1 Internal handling

M_MCU either responds itself, or forwards the request to the S_MCU, depending upon the PARAMETER_ID.

Table 5-136. USART Command Field.

Data	Usage
cmdID	CMND_SET_N_PARAMETERS
ucPar0 (unsigned char)	NUMBER
ulPar1 (unsigned long)	<not used>
ulPar2 (unsigned long)	<not used>

Table 5-137. USART Response.

Data	Usage
RSP_OK	Parameter written OK
RSP_SET_N_PARAMETERS	One or more errors occurred
RSP_ILLEGAL_EMULATOR_MODE	The parameter cannot be written in this emulator mode. (JTAG / DebugWire).

Table 5-138. FIFO Data (Command).

Data	Usage
FIFO_SOF	Start-of-frame

1..NUMBER {
PARAMETER_ID,
PARAMETER_VALUE
}
	
	[BYTE]
	[BYTE] * 4 LSB first

FIFO_EOF	End-of-frame

Table 5-139. FIFO Data (Response).

Data	Usage
FIFO_SOF	Start-of-frame

1..NUMBER {
STATUS_RESPONSE,
}
	[BYTE]

FIFO_EOF	End-of-frame

5.1.23 Clear Breakpoint (CMND_CLR_BREAK: 0x1A)

Set a breakpoint.

Slave state: STOPPED

Table 5-140. Message Format.

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	Incremented for each packet sent.
MESSAGE_SIZE	Size of message body
TOKEN	ASCII 14
MESSAGE_BODY	Breakpoint setting (see format below).
CRC	2-byte CRC.

The MESSAGE_BODY consists of:

<MESSAGE_ID, NUMBER, ADDRESS >

Table 5-141. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	0x1A (CMND_CLR_BREAK)	[BYTE]
NUMBER	Breakpoint number, 0x01 to 0x04 (up to 4 breakpoints are supported	[BYTE]
ADDRESS	Break address (for SW BP)	[BYTE] * 4, LSB first

Table 5-142. Response.

Parameter Name	Description
RSP_OK	The command is executed.
RSP_FAILED	The command was not understood by the JTAGICE mkII
RSP_ILLEGAL_BREAKPOINT	The breakpoint number does not exist. Illegal breakpoint (RSP_ILLEGAL_BREAKPOINT : 0xA8)
RSP_ILLEGAL_EMULATOR_MODE	The operation cannot be performed in this emulator mode. (JTAG / DebugWire).
RSP_ILLEGAL_MCU_STATE	The operation cannot be performed with the target MCU in its current state.
RSP_ILLEGAL_BREAKPOINT	The breakpoint number is not supported (in this mode)
Timeout	No response from the slave within the timeout time.

5.1.23.1 Internal handling

The request is passed to the S_MCU.

Table 5-143. USART Command Field.

Data	Usage
cmdID	CMND_CLR_BREAK
ucPar0 (unsigned char)	<not used>
ulPar1 (unsigned long)	ADDRESS 4 bytes LSB first
ulPar2 (unsigned long)	XX | XX | XX | NUMBER

Table 5-144. USART Response.

Data	Usage
RSP_OK	Command executed OK
RSP_FAILED	Command failed
RSP_ILLEGAL_EMULATOR_MODE	The operation cannot be performed in this emulator mode. (JTAG / DebugWire).
RSP_ILLEGAL_MCU_STATE	The operation cannot be performed with the target MCU in its current state.
RSP_ILLEGAL_BREAKPOINT	The breakpoint number is not supported (in this mode)

Table 5-145. FIFO Data (Command).

Data	Usage
<not used>	<not used>

Table 5-146. FIFO Data (Response).

Data	Usage
<not used>	<not used>

5.1.24 Run to Address (CMND_RUN_TO_ADDR: 0x1C)

Starts program execution at current Program Counter address and runs to the given Program Counter Address.

Slave state: STOPPED. When the command is executed the state changes to RUNNING.

Table 5-147. Message Format.

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	Incremented for each packet sent.
MESSAGE_SIZE	5
TOKEN	ASCII 14
MESSAGE_BODY	None
CRC	2-byte CRC.

The MESSAGE_BODY consists of:

< MESSAGE_ID, P.C. Address >

Table 5-148. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	CMND_RUN_TO_ADDR: 0x1C	[BYTE]
Program Counter	Address to stop at	[BYTE] * 4, LSB first

Table 5-149. Response.

Parameter Name	Description
RSP_OK	The command was executed.
RSP_FAILED	The command was not understood by the JTAGICE mkII
RSP_ILLEGAL_EMULATOR_MODE	The operation cannot be performed in this emulator mode. (JTAG / DebugWire).
Timeout	No response from the slave within the timeout time.

5.1.24.1 Internal handling

The request is passed to the S_MCU.

Table 5-150. USART Command Field.

Data	Usage
cmdID	CMND_RUN_TO_ADDR
ucPar0 (unsigned char)	<not used>
ulPar1 (unsigned long)	INST H | INST L | INST FLAG | XX
ulPar2 (unsigned long)	Program Counter value

Table 5-151. USART Response.

Data	Usage
RSP_OK	Command executed OK
RSP_FAILED	Command failed
RSP_ILLEGAL_EMULATOR_MODE	The operation cannot be performed in this emulator mode. (JTAG / DebugWire).
RSP_ILLEGAL_MCU_STATE	The operation cannot be performed with the target MCU in its current state.

Table 5-152. FIFO Data (Command).

Data	Usage
<not used>	<not used>

Table 5-153. FIFO Data (Response).

Data	Usage
<not used>	<not used>

5.1.25 Universal SPI command (CMND_SPI_CMD: 0x1D)

Runs a universal SPI command.

Slave state: STOPPED.

Table 5-154. Message Format.

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	Incremented for each packet sent.
MESSAGE_SIZE	5
TOKEN	ASCII 14
MESSAGE_BODY	None
CRC	2-byte CRC.

The MESSAGE_BODY consists of:

< MESSAGE_ID, COMMAND [4]>

Table 5-155. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	CMND_SPI_ENABLE_DW_FUSE : 0x1D	[BYTE]
COMMAND	4 byte SPI command	[BYTE] * 4

Table 5-156. Response.

Parameter Name	Description
RSP_SPI_DATA	The command was executed.
RSP_FAILED	The command was not understood by the JTAGICE mkII
RSP_ILLEGAL_EMULATOR_MODE	The operation cannot be performed in this emulator mode. (Must be in SPI mode).
Timeout	No response from the slave within the timeout time.

5.1.25.1 Internal handling

The request is passed to the S_MCU.

Table 5-157. USART Command Field.

Data	Usage
cmdID	CMND_SPI_CMD
ucPar0 (unsigned char)	<not used>
ulPar1 (unsigned long)	Command [BYTE] * 4
ulPar2 (unsigned long)	<not used>

Table 5-158. USART Response.

Data	Usage
RSP_SPI_DATA SPI data returned (RSP_SPI_DATA : 0x88)	Command executed OK
RSP_FAILED	Command failed
RSP_ILLEGAL_EMULATOR_MODE	The operation cannot be performed in this emulator mode. (Must be in SPI mode).
RSP_ILLEGAL_MCU_STATE	The operation cannot be performed with the target MCU in its current state.

Table 5-159. FIFO Data (Command).

Data	Usage
<not used>	<not used>

Table 5-160. FIFO Data (Response).

Data	Usage
FIFO_SOF	Start-of-frame
DATA	Data clocked out during SPI command
FIFO_EOF	End-of-frame

5.1.26 Clear event memory (CMND_CLEAR_EVENTS: 0x22)

Instructs the ICE to clear its event memory (for SW breakpoints). Must be called AFTER setting the device descriptor to ensure that only the required amount of event memory is cleared.

Table 5-161. Message Format.

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	Incremented for each packet sent.
MESSAGE_SIZE	1
TOKEN	ASCII 14
MESSAGE_BODY	See below
CRC	2-byte CRC.

The MESSAGE_BODY consists of:

<MESSAGE_ID >

Table 5-162. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	0x22 (CMND_CLEAR_EVENTS)	[BYTE]

Table 5-163. Response.

Parameter Name	Description
RSP_OK	The command is executed.
RSP_FAILED	The command was not understood by the JTAGICE mkII
RSP_ILLEGAL_MCU_STATE	The operation cannot be performed with the target MCU in its current state.
Timeout	No response from the slave within the timeout time.

5.1.26.1 Internal handling

The request is handled entirely on the M_MCU.

5.1.27 Restore target (CMND_RESTORE_TARGET: 0x23)

Instructs the ICE to restore the target device into a state where it can run freely. This is to be done upon terminating a debug session, before signing off and before disabling the OCD fuse (for JTAG). The command does the following:

- Stop target
- Replaces SW breakpoints with original instructions
- Reset target
- Run target

Table 5-164. Message Format.

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	Incremented for each packet sent.
MESSAGE_SIZE	1
TOKEN	ASCII 14
MESSAGE_BODY	See below
CRC	2-byte CRC.

The MESSAGE_BODY consists of:

<MESSAGE_ID >

Table 5-165. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	0x23 (CMND_RESTORE_TARGET)	[BYTE]

Table 5-166. Response.

Parameter Name	Description
RSP_OK	The command is executed.
RSP_FAILED	The command was not understood by the JTAGICE mkII
RSP_ILLEGAL_MCU_STATE	The operation cannot be performed with the target MCU in its current state.
Timeout	No response from the slave within the timeout time.

5.1.27.1 Internal handling

The request is handled entirely on the M_MCU.

5.1.28 Encapsulated ISP command (CMND_ISP_PACKET: 0x2F)

Used to send any ISP programming commands to the JTAGICE mkII. An ISP packet compatible with the AVRISP mkII is encapsulated in a JTAGICE mkII packet. The JTAGICE mkII then operates on the ISP packet in the same way as the AVRISP mkII.

Table 5-170 shows the ISP commands, which are supported on the JTAGICE mkII

For further information regarding the ISP commands, consult document AVR069: AVRISP mkII Communication Protocol.

Table 5-167. Message Format.

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	Incremented for each packet sent.
MESSAGE_SIZE	Size of message body

Parameter Name	Description
TOKEN	ASCII 14
MESSAGE_BODY	See below
CRC	2-byte CRC.

The MESSAGE_BODY consists of:

<MESSAGE_ID >, <ISP_PACKET>

Table 5-168. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	0x2F (CMND_ISP_PACKET)	[BYTE]
ISP_PACKET	Encapsulated AVRISP data	[BYTE] * n

Table 5-169. Response.

Parameter Name	Description
RSP_OK	The command is executed.
RSP_FAILED	The command was not understood by the JTAGICE mkII
RSP_ILLEGAL_MCU_STATE	The operation cannot be performed with the target MCU in its current state.
Timeout	No response from the slave within the timeout time.

Table 5-170. Supported AVRISP mkII commands on the JTAGICE mkII

Command	Usage
CMD_SET_PARAMETER	Sets a parameter
CMD_GET_PARAMETER	Reads back a parameter value
CMD_OSCCAL	Performs OSCCAL calibration sequence
CMD_LOAD_ADDRESS	Loads an address for programming functions
CMD_ENTER_PROGMODE_ISP	Puts the target device into programming mode
CMD_LEAVE_PROGMODE_ISP	Target device leaves programming mode
CMD_CHIP_ERASE_ISP	Perform a chip erase on the target device
CMD_PROGRAM_FLASH_ISP	Programs the flash memory on the target device
CMD_READ_FLASH_ISP	Reads the flash memory on the target device
CMD_PROGRAM_EEPROM_ISP	Programs the EEPROM on the target device
CMD_READ_EEPROM_ISP	Reads the EEPROM on the target device
CMD_PROGRAM_FUSE_ISP	Programs fuses on the target device
CMD_READ_FUSE_ISP	Reads fuses on the target device
CMD_PROGRAM_LOCK_ISP	Programs lock bits on the target device
CMD_READ_LOCK_ISP	Reads lock bits on the target device
CMD_READ_SIGNATURE_ISP	Reads the signature bytes on the target device
CMD_READ_OSCCAL_ISP	Reads the OSCCAL byte on the target device

Command	Usage
CMD_SPI_MULTI	Generic command used to execute ISP commands

5.2 Slave Responses

5.2.1 OK (RSP_OK : 0x80)

Acknowledge: The slave understood the command and everything is OK.

Table 5-170. Message Format.

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	Identical to the associated request
MESSAGE_SIZE	1
TOKEN	ASCII 14
MESSAGE_BODY	See below
CRC	2-byte CRC.

The MESSAGE_BODY consists of:

<MESSAGE_ID>

Table 5-171. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	0x80 (RSP_OK)	[BYTE]

5.2.2 Failed (RSP_FAILED: 0xA0)

The slave did not understand the command and did nothing. The message body can be extended to encompass detailed information on WHAT actually failed, this specification may be updated at this point.

Table 5-172. Message Format.

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	Identical to the associated request
MESSAGE_SIZE	1
TOKEN	ASCII 14
MESSAGE_BODY	See below
CRC	2-byte CRC.

The MESSAGE_BODY consists of:

<MESSAGE_ID>

Table 5-173. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	0xA0 (RSP_FAILED)	[BYTE]

5.2.3 Illegal Parameter (RSP_ILLEGAL_PARAMETER: 0xA1)

The master has tried to write or read an emulator parameter that does not exist.

Table 5-174. Message Format.

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	Identical to the associated request
MESSAGE_SIZE	1
TOKEN	ASCII 14
MESSAGE_BODY	See below
CRC	2-byte CRC.

The MESSAGE_BODY consists of:

<MESSAGE_ID>

Table 5-175. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	0xA1 (RSP_ILLEGAL_PARAMETER)	[BYTE]

5.2.4 Parameter return (RSP_PARAMETER: 0x81)

The emulator returns the requested parameter.

Table 5-176. Message Format.

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	Identical to the associated request
MESSAGE_SIZE	See below
TOKEN	ASCII 14.
MESSAGE_BODY	See below
CRC	2-byte CRC.

The MESSAGE_BODY consists of:

<MESSAGE_ID, PARAMETER_VALUE>

Table 5-177. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	0x81 (RSP_PARAMETER)	[BYTE]
PARAMETER_VALUE	The parameter value	[BYTE]*N

5.2.5 Illegal Memory Access (RSP_ILLEGAL_MEMORY_TYPE: 0xA2)

The master has tried to write or read emulator memory type that does not exist.

Table 5-178. Message Format.

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	Identical to the associated request
MESSAGE_SIZE	1
TOKEN	ASCII 14
MESSAGE_BODY	See below
CRC	2-byte CRC.

The MESSAGE_BODY consists of:

<MESSAGE_ID>

Table 5-179. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	0xA2 (RSP_ILLEGAL_MEMORY_TYPE)	[BYTE]

5.2.6 Illegal Memory Access (RSP_ILLEGAL_MEMORY_RANGE: 0xA3)

The master has tried to write or read an unsupported memory size.

Table 5-180. Message Format.

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	Identical to the associated request
MESSAGE_SIZE	1
TOKEN	ASCII 14
MESSAGE_BODY	See below
CRC	2-byte CRC.

The MESSAGE_BODY consists of:

<MESSAGE_ID>

Table 5-181. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	0xA3 (RSP_ILLEGAL_MEMORY_RANGE)	[BYTE]

5.2.7 Memory read (RSP_MEMORY: 0x82)

The emulator returns the requested memory block

Table 5-182. Message Format.

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	Identical to the associated request
MESSAGE_SIZE	The size of MESSAGE_BODY in bytes.
TOKEN	ASCII 14
MESSAGE_BODY	See below
CRC	2-byte CRC.

The MESSAGE_BODY consists of:

<MESSAGE_ID, DATA>

Table 5-183. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	0x82 (RSP_MEMORY)	[BYTE]
DATA	Memory read!	[BYTE] * N

5.2.8 Breakpoint read (RSP_GET_BREAK: 0x83)

The emulator returns only the requested breakpoint:

Table 5-184. Message Format.

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	Identical to the associated request
MESSAGE_SIZE	The size of MESSAGE_BODY in bytes.
TOKEN	ASCII 14
MESSAGE_BODY	See below
CRC	2-byte CRC.

The MESSAGE_BODY consists of:

<MESSAGE_ID, TYPE, ADDRESS, MODE>

Table 5-185. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	0x83 (RSP_GET_BREAK)	[BYTE]
TYPE	0x01: program breakpoint 0x02: data breakpoint	[BYTE]
ADDRESS	Address	[BYTE] * 4, LSB first
MODE	0x00: (Program breakpoint – not used) 0x01: Break on memory read 0x02: Break on memory write 0x03: Break on memory read or write.	[BYTE]

5.2.9 Operation cannot be performed (RSP_ILLEGAL_EMULATOR_MODE: 0xA4)

The master has requested an operation that cannot be performed in the current emulator mode:

Table 5-186. Message Format.

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	Identical to the associated request
MESSAGE_SIZE	2
TOKEN	ASCII 14
MESSAGE_BODY	See below
CRC	2-byte CRC.

The MESSAGE_BODY consists of:

<MESSAGE_ID, CURRENT_MODE>

Table 5-187. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	0xA4 (RSP_ILLEGAL_EMULATOR_MODE)	[BYTE]
CURRENT_MODE	0x00: EMULATOR_MODE_DEBUGWIRE 0x01: EMULATOR_MODE_JTAG 0x02: EMULATOR_MODE_UNKNOWN	[BYTE]

5.2.10 Operation cannot be performed (RSP_ILLEGAL_MCU_STATE: 0xA5)

The master has requested an operation that cannot be performed in the current operating state of the target MCU:

Table 5-188. Message Format

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	Identical to the associated request
MESSAGE_SIZE	2
TOKEN	ASCII 14
MESSAGE_BODY	See below
CRC	2-byte CRC.

The MESSAGE_BODY consists of:

<MESSAGE_ID, CURRENT_MODE>

Table 5-189. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	0xA5 (RSP_ILLEGAL_MCU_STATE)	[BYTE]
CURRENT_MODE	0x00: STOPPED 0x01: RUNNING 0x02: PROGRAMMING	[BYTE]

5.2.11 Program Counter Read (RSP_PC: 0x84)

Returns the current program counter to the master

Table 5-190. Message Format.

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	Identical to the associated request
MESSAGE_SIZE	5
TOKEN	ASCII 14
MESSAGE_BODY	See below
CRC	2-byte CRC.

The MESSAGE_BODY consists of:

<MESSAGE_ID, PROGRAM_COUNTER>

Table 5-191. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	0x84 (RSP_PC)	[BYTE]
MESSAGE_BODY	Program counter	[BYTE] * 4, LSB first

5.2.12 Selftest (RSP_SELFTEST: 0x85)

Returns the result of a self-test. Content of self-test is described later. The self-test serves two purposes:

1. Verifying the integrity of the unit itself.
2. Verifying that the actual debug set-up is working (device interface is ok, no stuck lines etc)

Table 5-192. Message Format.

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	Identical to the associated request
MESSAGE_SIZE	The size of MESSAGE_BODY in bytes.
TOKEN	ASCII 14
MESSAGE_BODY	See below
CRC	2-byte CRC.

The MESSAGE_BODY consists of 9 parts:

< MESSAGE_ID, SELFTEST_0, …, SELFTEST_7>

Table 5-193. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	0x85 (RSP_SELFTEST)	[BYTE]
SELFTEST_0	TEST_SPECIFIC_RESULT	[BYTE]
- ... -	- ... -	- ... -
SELFTEST_7	TEST_SPECIFIC_RESULT	[BYTE]

Up to 8 individual self-test functions can be selected by the CMND_SELFTEST command. Each test returns one byte to the response, regardless of whether it is executed or not. The individual results are SELFTEST values, which can be customized further for each specific test.

Table 5-194. Response.

Self-test response	Value	Meaning
SELFTEST_SKIPPED	0x00	This self-test was not executed
SELFTEST_OK	0x01	Self-test passed
SELFTEST_FAILED	0x80	Self-test failed
ST_USART_FAILURE	0x81	Internal USART test failed
ST_FIFO_M_FAILURE	0x82	Master side FIFO reading failed
ST_FIFO_S_FAILURE	0x83	Slave side FIFO reading failed
ST_FIFO_M_EMPTY_FAILURE	0x84	Master side EMPTY bit reading failed
ST_FIFO_S_EMPTY_FAILURE	0x85	Slave side EMPTY bit reading failed
ST_FIFO_M_FULL_FAILURE	0x86	Master side FULL bit reading failed
ST_FIFO_S_FULL_FAILURE	0x87	Slave side FULL bit reading failed
ST_FIFO_M_NINE_FAILURE	0x88	Master side NINTH bit reading failed
ST_FIFO_S_NINE_FAILURE	0x89	Slave side NINTH bit reading failed
ST_SRAM_FAILURE	0x8A	Internal SRAM read-write test failed
ST_JTAG_TMS_STUCK_HIGH	0x8B	JTAG TMS line cannot be driven low!
ST_JTAG_TCK_STUCK_HIGH	0x8C	JTAG TCK line cannot be driven low!

Self-test response	Value	Meaning
ST_JTAG_TDI_STUCK_HIGH	0x8D	JTAG TDI line cannot be driven low!
ST_JTAG_TMS_STUCK_LOW	0x8E	JTAG TMS line cannot be driven high!
ST_JTAG_TCK_STUCK_TMS	0x8F	JTAG lines TCK and TMS are possibly tied together.
ST_JTAG_TDI_STUCK_TMS	0x90	JTAG lines TDI and TMS are possibly tied together.
ST_JTAG_TCK_STUCK_LOW	0x91	JTAG TCK line cannot be driven high!
ST_JTAG_TMS_STUCK_TCK	0x92	JTAG lines TMS and TCK are possibly tied together.
ST_JTAG_TDI_STUCK_TCK	0x93	JTAG lines TDI and TCK are possibly tied together.
ST_JTAG_TDI_STUCK_LOW	0x94	JTAG TDI line cannot be driven high!
ST_JTAG_TMS_STUCK_TDI	0x95	JTAG lines TMS and TDI are possibly tied together.
ST_JTAG_TCK_STUCK_TDI	0x96	JTAG lines TCK and TDI are possibly tied together.

- JTAG self-test messages should be in the form:

"JTAG connection failed: %message. Please check the physical connection between the ICE and the target."

- If the *DebugWire Capacitance* test fails, the message should read:

"The JTAGICE mkII has detected a high capacitance on the RESET line, which may be affecting the DebugWire communication. Please remove any capacitors from the reset line in your target circuit."

- If the *DebugWire PUSH PULL* test fails, the message should read:

"The JTAGICE mkII is unable to pull the reset line low in your target application. Using a too strong pull-up resistor (to low resistance) can cause this. The recommended value is in the order of 10k."

5.2.13 SPI data returned (RSP_SPI_DATA: 0x88)

Returns the data clocked out from an SPI command.

Table 5-195. Message Format.

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	Identical to the associated request
MESSAGE_SIZE	Size of message body in bytes
TOKEN	ASCII 14
MESSAGE_BODY	See below
CRC	2-byte CRC.

The MESSAGE_BODY consists of:

<MESSAGE_ID>

Table 5-196. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	0x88 (RSP_SPI_DATA)	[BYTE]
DATA	Data clocked out from SPI command	[BYTE]

5.2.14 Illegal Command (RSP_ILLEGAL_COMMAND: 0xAA)

The master has tried to access an illegal emulator command.

Table 5-197. Message Format.

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	Identical to the associated request
MESSAGE_SIZE	1
TOKEN	ASCII 14
MESSAGE_BODY	See below
CRC	2-byte CRC.

The MESSAGE_BODY consists of:

<MESSAGE_ID>

Table 5-198. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	0xAA (RSP_ILLEGAL_COMMAND)	[BYTE]

5.2.15 Illegal Value (RSP_ILLEGAL_VALUE: 0xA6)

The master has tried to write an illegal value to an emulator parameter.

Table 5-199. Message Format.

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	Identical to the associated request
MESSAGE_SIZE	1
TOKEN	ASCII 14
MESSAGE_BODY	See below
CRC	2-byte CRC.

The MESSAGE_BODY consists of:

<MESSAGE_ID>

Table 5-200. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	0xA6 (RSP_ILLEGAL_VALUE)	[BYTE]

5.2.16 Sign on (RSP_SIGN_ON: 0x86)

Response to the Master sign on command. The purpose of the response is to tell AVR Studio which device is attached and which HW and FW versions are in use.

Table 5-201. Message Format

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	Identical to the associated request
MESSAGE_SIZE	Size of message body in bytes
TOKEN	ASCII 14
MESSAGE_BODY	See below
CRC	2-byte CRC.

The MESSAGE_BODY consists of:

<MESSAGE_ID, COMM_ID, M_MCU_BLDR, M_MCU_FW_MIN, M_MCU_FW_MAJ, M_MCU_HW, S_MCU_BLDR, S_MCU_FW_MIN, S_MCU_FW_MAJ, S_MCU_HW, SERIAL_NUMBER, DEVICE_ID_STR>

Table 5-202. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	0x86 (RSP_SELFTEST)	[BYTE]
COMM_ID	Communications protocol version	[BYTE]
M_MCU_BLDR	M_MCU boot-loader FW version	[BYTE]
M_MCU_FW_MIN	M_MCU firmware version (minor)	[BYTE]
M_MCU_FW_MAJ	M_MCU firmware version (major)	[BYTE]
M_MCU_HW	M_MCU hardware version	[BYTE]
S_MCU_BLDR	S_MCU boot-loader FW version	[BYTE]
S_MCU_FW_MIN	S_MCU firmware version (minor)	[BYTE]
S_MCU_FW_MAJ	S_MCU firmware version (major)	[BYTE]
S_MCU_HW	S_MCU hardware version	[BYTE]
SERIAL_NUMBER	(USB) EEPROM stored s/n	[BYTE] * 6, LSB FIRST
DEVICE_ID_STR	“JTAGICE mkII\0” Null terminated ASCII string identifying the device	[BYTE] * N

5.2.17 Illegal breakpoint (RSP_ILLEGAL_BREAKPOINT: 0xA8)

The master has attempted to set or get a breakpoint, which does not exist.

Table 5-203. Message Format.

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	Identical to the associated request
MESSAGE_SIZE	1
TOKEN	ASCII 14
MESSAGE_BODY	See below
CRC	2-byte CRC.

The MESSAGE_BODY consists of:

<MESSAGE_ID>

Table 5-204. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	0xA8 (RSP_ILLEGAL_BREAKPOINT)	[BYTE]

5.2.18 Illegal JTAG ID (RSP_ILLEGAL_JTAG_ID: 0xA9)

The master has attempted to enter programming mode but the JTAG ID does not match the target device.

Table 5-205. Message Format.

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	Identical to the associated request
MESSAGE_SIZE	1
TOKEN	ASCII 14
MESSAGE_BODY	See below
CRC	2-byte CRC.

The MESSAGE_BODY consists of:

<MESSAGE_ID>

Table 5-206. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	0xA9 (RSP_ILLEGAL_JTAG_ID)	[BYTE]

5.2.19 Illegal Command (RSP_ILLEGAL_COMMAND: 0xAA)

The master has attempted to execute an unknown command.

Table 5-207. Message Format.

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	Identical to the associated request
MESSAGE_SIZE	1
TOKEN	ASCII 14
MESSAGE_BODY	See below
CRC	2-byte CRC.

The MESSAGE_BODY consists of:

<MESSAGE_ID>

Table 5-208. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	0xAA (RSP_ILLEGAL_COMMAND)	[BYTE]

5.2.20 Illegal Target Power State (RSP_NO_TARGET_POWER: 0xAB)

The master has attempted to execute a command but the target device is switched off or disconnected.

Table 5-209. Message Format.

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	Identical to the associated request
MESSAGE_SIZE	1
TOKEN	ASCII 14
MESSAGE_BODY	See below
CRC	2-byte CRC.

The MESSAGE_BODY consists of:

<MESSAGE_ID>

Table 5-210. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	0xAB (RSP_NO_TARGET_POWER)	[BYTE]

5.2.21 DebugWire sync failed (RSP_DEBUGWIRE_SYNC_FAILED: 0xAC)

The master has attempted to enter DebugWire mode, but the target did not respond to a reset pulse. DebugWire synchronization was not achieved.

Can only be returned by a set_parameter emulator mode to DebugWire, or possibly other DebugWire mode commands.

Table 5-211. Message Format.

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	Identical to the associated request
MESSAGE_SIZE	1
TOKEN	ASCII 14
MESSAGE_BODY	See below
CRC	2-byte CRC.

The MESSAGE_BODY consists of:

<MESSAGE_ID>

Table 5-212. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	0xAC (RSP_DEBUGWIRE_SYNC_FAILED)	[BYTE]

5.2.22 ICE has not enough power to run (RSP_ILLEGAL_POWER_STATE: 0xAD)

This response is sent to any command when the ICE is running off USB power but has only enumerated for 100mA operation. The ICE is in power save mode and will not respond normally to any commands.

Table 5-213. Message Format.

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	Identical to the associated request
MESSAGE_SIZE	1
TOKEN	ASCII 14
MESSAGE_BODY	See below
CRC	2-byte CRC.

The MESSAGE_BODY consists of:

<MESSAGE_ID>

Table 5-214. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	0xAD (RSP_ILLEGAL_POWER_STATE)	[BYTE]

6 Events

6.1 Introduction

Events are messages sent from the slave (JTAGICE mkII) to the master (AVR Studio) with no prior command going the other way. A typical example of an event is a BREAK, the emulator can go on running forever, but suddenly a break condition is true, and the emulator breaks. The emulator issues a BREAK event to the master, who takes the appropriate actions. This section lists all events. In order to ensure that sync is never lost between master and slave, AVR Studio should intermittently (every second?) poll the JTAGICE mkII for its operating state (RUNNING or STOPPED). All other events are for non-critical information.

6.2 Events (Message ID range 0xE0 – 0xFF)

Note regarding internal event handling: Internally the JTAGICE mkII handles events sourcing from both M_MCU and S_MCU. M_MCU triggered events include for example loss of target power. S_MCU triggered events include for example the BREAK event. The M_MCU implements a lightweight event queue in the USART receive buffer – message responses are separated from events as they arrive, allowing events to be handled after response retransmission, thus simplifying the synchronous command-response mechanism.

The internal event queue only allows single byte events to be sent from the S_MCU to the M_MCU. In the event of more data being available, the M_MCU must explicitly query this from the S_MCU – for example the PC is queried after a BREAK event.

Lastly, events should be '**debounced'** before transmission to avoid flooding the queue with several 'detections' of the same event.

6.2.1 Event Break (EVT_BREAK: 0xE0)

A break condition is met; the emulator goes from RUNNING to STOPPED state.

Table 6-1. Message Format.

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	0xFFFF (reserved for EVENTS)
MESSAGE_SIZE	6
TOKEN	ASCII 14
MESSAGE_BODY	See below
CRC	2-byte CRC.

The MESSAGE_BODY consists of:

<MESSAGE_ID, PROGRAM_COUNTER, BREAK_CAUSE>

Table 6-2. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	0xE0 (EVT_BREAK)	[BYTE]
PROGRAM_COUNTER	Program counter...	[BYTE]*4, LSB first
BREAK_CAUSE	0x00 = unspecified 0x01 = program break 0x02 = data break PDSB 0x03 = data break PDMSB	[BYTE]

6.2.1.1 Internal handling

Upon receipt of this event, the M_MCU should query the PC from the S_MCU and dispatch the event (including PC) to the host.

6.2.2 Event Run (EVT_RUN: 0xE1)

The emulator has started running by it's own decision.

Table 6-3. Message Format.

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	0xFFFF (reserved for EVENTS)
MESSAGE_SIZE	2
TOKEN	ASCII 14
MESSAGE_BODY	See below
CRC	2-byte CRC.

The MESSAGE_BODY consists of:

<MESSAGE_ID, RUN_CAUSE>

Table 6-4. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	0xE1 (EVT_RUN)	[BYTE]
RUN_CAUSE	TBD	[BYTE]

6.2.3 Reserved - 0xE2

Reserved for future use.

6.2.4 Reserved - 0xE3

Reserved for future use.

6.2.5 Event Target Power On (EVT_TARGET_POWER_ON: 0xE4)

The emulator sends this event if the target power is off and then is turned on. AVR Studio must act accordingly.

Table 6-5. Message Format.

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	0xFFFF (reserved for EVENTS)
MESSAGE_SIZE	1
TOKEN	ASCII 14
MESSAGE_BODY	See below
CRC	2-byte CRC.

The MESSAGE_BODY consists of:

<MESSAGE_ID>

Table 6-6. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	0xE4 (EVT_TARGET_POWER_ON)	[BYTE]

6.2.6 Event Target Power Off (EVT_TARGET_POWER_OFF: 0xE5)

The emulator sends this event when target power is on and then is turned off. AVR Studio must act accordingly.

Table 6-7. Message Format.

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	0xFFFF (reserved for EVENTS)
MESSAGE_SIZE	1
TOKEN	ASCII 14
MESSAGE_BODY	See below
CRC	2-byte CRC.

The MESSAGE_BODY consists of:

<MESSAGE_ID>

Table 6-8. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	0xE5 (EVT_TARGET_POWER_OFF)	[BYTE]

6.2.7 Event Debug (EVT_DEBUG: 0xE6)

This event is set if the user has enabled the feature with the Set parameter command. The event is sent every time a debug criterion is met. The event can be used in numerous debug situations.

Table 6-9. Message Format.

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	0xFFFF (reserved for EVENTS)
MESSAGE_SIZE	Size of message body in bytes
TOKEN	ASCII 14
MESSAGE_BODY	See below
CRC	2-byte CRC.

Host communication debugging

DEBUG_EVENT_ID = 0x01 (DEBUG_HOST_COMM)

The MESSAGE_BODY consists of:

<MESSAGE_ID, DEBUG_EVENT_ID, COMM_STATE, COMM_ERROR>

Table 6-10. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	0xE6 (EVT_DEBUG)	[BYTE]
DEBUG_EVENT_ID	Identifies the debug event	[BYTE]
COMM_STATE	0x00: Start 0x01: Get Seq 0x02: Get Size 0x03: Get Token 0x04: Get Data 0x05: Get CRC	[BYTE]
COMM_ERROR	0x01: Error in byte value 0x02: Timeout	[BYTE]

6.2.8 Event Target External Reset (EVT_EXT_RESET: 0xE7)

The target is reset externally (is still running).

Table 6-11. Message Format.

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	0xFFFF (reserved for EVENTS)
MESSAGE_SIZE	1
TOKEN	ASCII 14
MESSAGE_BODY	See below
CRC	2-byte CRC.

The MESSAGE_BODY consists of:

<MESSAGE_ID>

Table 6-12. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	0xE7 (EVT_EXT_RESET)	[BYTE]

6.2.9 Event Target Enter Sleep (EVT_TARGET_SLEEP: 0xE8)

The target is running, but has entered sleep mode.

Table 6-13. Message Format.

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	0xFFFF (reserved for EVENTS)
MESSAGE_SIZE	1
TOKEN	ASCII 14
MESSAGE_BODY	See below
CRC	2-byte CRC.

The MESSAGE_BODY consists of:

<MESSAGE_ID>

Table 6-14. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	0xE8 (EVT_TARGET_SLEEP)	[BYTE]

6.2.10 Event Target Wakeup (EVT_TARGET_WAKEUP: 0xE9)

The target has woken up from sleep.

Table 6-15. Message Format.

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	0xFFFF (reserved for EVENTS)
MESSAGE_SIZE	1
TOKEN	ASCII 14
MESSAGE_BODY	See below
CRC	2-byte CRC.

The MESSAGE_BODY consists of:

<MESSAGE_ID>

Table 6-16. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	0xE9 (EVT_TARGET_WAKEUP)	[BYTE]

6.2.11 Event ICE illegal power state (EVT_ICE_POWER_ERROR_STATE: 0xEA)

The ICE has entered an illegal power state (e.g.: tries to get power from an un-powered USB hub) and cannot continue debugging.

Table 6-17. Message Format.

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	0xFFFF (reserved for EVENTS)
MESSAGE_SIZE	1
TOKEN	ASCII 14
MESSAGE_BODY	See below
CRC	2-byte CRC.

The MESSAGE_BODY consists of:

<MESSAGE_ID>

Table 6-18. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	0xEA (EVT_ICE_POWER_ERROR_STATE)	[BYTE]

6.2.12 Event ICE power OK (EVT_ICE_POWER_OK: 0xEB)

The ICE power source is OK for full operation.

Table 6-19. Message Format.

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	0xFFFF (reserved for EVENTS)
MESSAGE_SIZE	1
TOKEN	ASCII 14
MESSAGE_BODY	See below
CRC	2-byte CRC.

The MESSAGE_BODY consists of:

<MESSAGE_ID>

Table 6-20. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	0xEB (EVT_ICE_POWER_OK)	[BYTE]

6.2.13 Event IDR dirty (EVT_IDR_DIRTY: 0xEC)

The target device has written the IDR register.

Table 6-21. Message Format.

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	0xFFFF (reserved for EVENTS)
MESSAGE_SIZE	1
TOKEN	ASCII 14
MESSAGE_BODY	See below
CRC	2-byte CRC.

The MESSAGE_BODY consists of:

<MESSAGE_ID, IDR value>

Table 6-22. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	0xEC (EVT_IDR_DIRTY)	[BYTE]
IDR	Value written to IDR by the target app	[BYTE]

Note: IDR value is only included in the message form the M_MCU to the host PC. M_MCU must use getParameter (IDR_VALUE) to retrieve value from the S_MCU.

6.2.14 Event None (EVT_NONE: 0xEF)

Used internally as a dummy event.

6.2.15 Internal Event (EVT_PROGRAM_BREAK: 0xF1)

This event is sent each time the emulator stops at a program breakpoint. The program breakpoint can be caused by either hardware or software.

6.2.16 Internal Event (EVT_PDSB_BREAK: 0xF2)

Program or Data Single Breakpoint.

6.2.17 Internal Event (EVT_PDSMB_BREAK: 0xF3)

Program, Data or Masked Single Breakpoint.

6.2.18 DebugWire Error Events (EVT_ERROR_PHY_X)

Table 6-23. Error Events.

Event Name	Value
EVT_ERROR_PHY_FORCE_BREAK_TIMEOUT	0xE2
EVT_ERROR_PHY_RELEASE_BREAK_TIMEOUT	0xE3
EVT_ERROR_PHY_MAX_BIT_LENGTH_DIFF	0xED
EVT_ERROR_PHY_SYNC_TIMEOUT	0xF0
EVT_ERROR_PHY_SYNC_TIMEOUT_BAUD	0xF4
EVT_ERROR_PHY_SYNC_OUT_OF_RANGE	0xF5
EVT_ERROR_PHY_SYNC_WAIT_TIMEOUT	0xF6
EVT_ERROR_PHY_RECEIVE_TIMEOUT	0xF7
EVT_ERROR_PHY_RECEIVED_BREAK	0xF8
EVT_ERROR_PHY_OPT_RECEIVE_TIMEOUT	0xF9
EVT_ERROR_PHY_OPT_RECEIVED_BREAK	0xFA
EVT_RESULT_PHY_NO_ACTIVITY	0xFB

The DebugWire error events should only be visible when the debug option is checked in Studio.

For normal operation when an error condition occurs just display DebugWire access failed. The user doesn't have to bother with the low-level error messages.

Table 6-24. Message Format.

Parameter Name	Description
MESSAGE_START	ASCII 27 (ESC character)
SEQUENCE_NUMBER	0xFFFF (reserved for EVENTS)
MESSAGE_SIZE	1
TOKEN	ASCII 14
MESSAGE_BODY	See below
CRC	2-byte CRC.

The MESSAGE_BODY consists of:

<MESSAGE_ID>

Table 6-25. Parameters.

Parameter Name	Usage	Format
MESSAGE_ID	(EVT_ERROR_PHY_X)	[BYTE]

7 USB Descriptors

This section list the descriptors used in relation to the JTAGICE mkII USB communication.

7.1 DEVICE Descriptor

Table 7-1. Device descriptors.

Offset	Field	Value	Description
0	bLength	0x12	The size of this descriptor is 18 bytes
1	bDescriptorType	0x01	DEVICE Descriptor Type
2	bcdUSB	0x0110	Device compliant to the USB specification version 1.10
4	bDeviceClass	0xFF	The device class is vendor-specific
5	bDeviceSubClass	0x00	Each interface specifies its own subclass information
6	bDeviceProtocol	0x00	No protocols on the device basis
7	bMaxPacketSize0	0x10	Maximum packet size for endpoint zero is 16
8	idVendor	0x03EB	Vendor ID is 1003: Atmel Corporation
10	idProduct	0x2103	The Product ID is 0x2103
12	bcdDevice	0x0200	The device release number is 2.00
14	iManufacturer	0x01	The index of the string descriptor describing the manufacturer is 1
15	iProduct	0x02	The index of the string descriptor describing the product is 2
16	iSerialNumber	0x03	The index of the string descriptor describing the serial number is 3
17	bNumConfigurations	0x01	The device has 1 possible configurations

7.2 CONFIGURATION Descriptor

Table 7-2. Configuration descriptors.

Offset	Field	Value	Description
0	bLength	0x09	The size of this descriptor is 9 bytes
1	bDescriptorType	0x02	CONFIGURATION Descriptor Type
2	wTotalLength	0x0020	The total length of data for this configuration is 32. This includes the combined length of all the descriptors returned
4	bNumInterfaces	0x01	This configuration supports 1 interfaces
5	bConfigurationValue	0x01	The value 1 should be used to select this configuration
6	iConfiguration	0x00	The device doesn't have the string descriptor describing this configuration
7	bmAttributes	0x80	Configuration characteristics:Bit 7: Reserved
8	MaxPower	0xFA	Maximum power consumption of the device in this configuration is 500 mA

7.3 INTERFACE Descriptor

Table 7-3. Interface descriptors.

Offset	Field	Value	Description
0	bLength	0x09	The size of this descriptor is 9 bytes
1	bDescriptorType	0x04	INTERFACE Descriptor Type
2	bInterfaceNumber	0x00	The number of this interface is 0
3	bAlternateSetting	0x00	The value used to select alternate setting for this interface is 0
4	bNumEndpoints	0x02	The number of endpoints used by this interface is 2(excluding endpoint zero)
5	bInterfaceClass	0xFF	The interface class is vendor-specific
6	bInterfaceSubClass	0x00	The subclass code is 0x00
7	bInterfaceProtocol	0x00	The interface doesn't use any class-specific protocols
8	iInterface	0x00	The device doesn't have the string descriptor describing this interface

7.4 IN ENDPOINT Descriptor

Table 7-4. In end-point descriptors

Offset	Field	Value	Description
0	bLength	0x07	The size of this descriptor is 7 bytes
1	bDescriptorType	0x05	ENDPOINT Descriptor Type
2	bEndpointAddress	0x82	This is an IN endpoint with address (endpoint number) 2
3	bmAttributes	0x02	Types –Transfer:BULKSync:No SyncUsage:Data EP
4	wMaxPacketSize	0x0040	Maximum packet size value for this endpoint is 0x40(Bits 12-11: Addtl. Transactions/frame)
6	bInterval	0x0A	bInterval:10. The polling interval value is bInterval or 2**(bInterval-1)

7.5 OUT ENDPOINT Descriptor

Table 7-5. Out end-point descriptors.

Offset	Field	Value	Description
0	bLength	0x07	The size of this descriptor is 7 bytes
1	bDescriptorType	0x05	ENDPOINT Descriptor Type
2	bEndpointAddress	0x02	This is an OUT endpoint with address (endpoint number) 2
3	bmAttributes	0x02	Types –Transfer:BULKSync:No SyncUsage:Data EP
4	wMaxPacketSize	0x0040	Maximum packet size value for this endpoint is 0x40(Bits 12-11: Addtl. Transactions/frame)
6	bInterval	0x0A	bInterval:10. The polling interval value is bInterval or 2**(bInterval-1)

8 Parameters

The following table describes all JTAGICE mkII parameters maintained through the Set and Get Parameter commands.

Table 8-1. Parameter Description

ID	Description	Format	Access	Value
0x01	Hardware Version	[BYTE] M_MCU ver [BYTE] S_MCU ver	R	Hardvare version string
0x02	FW version	[BYTE] M_MCU_min [BYTE] M_MCU_maj [BYTE] S_MCU_min [BYTE] S_MCU_maj	R	Firmware version string
0x03	Emulator MODE (DebugWire or JTAG)	[BYTE]	R/W	0x00: DebugWire 0x01: JTAG 0x02: unknown (default) 0x03: SPI
0x04	Ireg	[BYTE] * 2, LSB first	R/W	(not used)
0x05	Baud rate (the default value is 4 : 19200) The RS232 setup is: Baud rate, No Parity, 8 data bit, 1 stop bit.	[BYTE]	R/W	0x01 : 2400 0x02 : 4800 0x03 : 9600 0x04 : 19200 default 0x05 : 38400 0x06 : 57600 0x07 : 115200 0x08 : 14400
0x06	OCD Vtarget	[BYTE] * 2, LSB first	R	Read from target
0x07	OCD JTAG Clock	[BYTE]	R/W	Delay between setting and clearing the JTAG clock. 0x00 for no delay (target at 4MHz+)
0x08	OCD Break Cause	[BYTE]	R	Read from target upon break
0x09	Timers Running	[BYTE]	R/W	0x00 : Timers stopped (default) 0x01 : Timers Running
0x0A	Break on Change of Flow	[BYTE]	R/W	0x00

ID	Description	Format	Access	Value
0x0B	Break Addr1	[BYTE] * 2, LSB first	R/W	(not used)
0x0C	Break Addr2	[BYTE] * 2, LSB first	R/W	(not used)
0x0D	CombBreakCtrl	[BYTE]	R/W	(not used)
0x0E	JTAGID string	[BYTE] * 4	R	Device specific
0x0F	Units Before	[BYTE]	R/W	(not used)
0x10	Units After	[BYTE]	R/W	(not used)
0x11	Bit Before	[BYTE]	R/W	(not used)
0x12	Bit After	[BYTE]	R/W	(not used)
0x13	External Reset	[BYTE]	R/W	0x00 : No 0x01 : Yes (default)
0x14	Flash Page Size	[BYTE] * 2, LSB first	R/W	Device spesific
0x15	EEPROM Page Size	[BYTE]	R/W	Device spesific
0x16				(not used)
0x17	PSB0	[BYTE] * 2, LSB first	R/W	(not used)
0x18	PSB1	[BYTE] * 2, LSB first	R/W	(not used)
0x19	Protocol Debug Event	[BYTE]	R/W	0x00 : Event OFF (default) 0x01 : Event ON
0x1A	Target MCU STATE	[BYTE]	R	0x00 : STOPPED 0x01 : RUNNING 0x02 : PROGRAMMING
0x1B	Daisy chain info	[BYTE] units before [BYTE] units after [BYTE] bits before [BYTE] bits after	R/W	
0x1C	Boot address	[BYTE] * 4, LSB first	R/W	
0x1D	Target Signature	[BYTE] * 2, LSB first	R	Read from target
0x1E	DebugWire baudrate	[BYTE] * 4, LSB first	R/W	(not used)
0x1F	Program entry point	[BYTE] * 4, LSB first	W	
0x40	Packet parsing errors	[BYTE] * 4, LSB first	R	
0x41	Valid packets received	[BYTE] * 4, LSB first	R	
0x42	Intercommunication TX failures	[BYTE] * 4, LSB first	R	
0x43	Intercommunication RX failures	[BYTE] * 4, LSB first	R	
0x44	CRC errors	[BYTE] * 4, LSB first	R	
0x45	Power source	[BYTE]	R	0x00 : External 0x01 : USB
0x22	CAN_FLAG	[BYTE]	R/W	0x00: Don’t read CAN-mailbox 0x01: Read CAN-mailbox
0x23	PAR_ENABLE_IDR_IN_RUN_MODE	[BYTE]	W	0x00: Access OSCCAL 0x01: Access IDR
0x24	PAR_ALLOW_PAGEPROGRAMMING_IN_SCANCHAIN	[BYTE]	W	0x00: Not allowed 0x01: Allowed

9 Device Descriptor fields

Figure 9-1. Defined parameters used by the “Set Device Descriptor” command.

```
unsigned char ucReadIO[8];           //LSB = IOloc  0, MSB = IOloc63
unsigned char ucReadIOShadow[8];     //LSB = IOloc  0, MSB = IOloc63
unsigned char ucWriteIO[8];          //LSB = IOloc  0, MSB = IOloc63
unsigned char ucWriteIOShadow[8];    //LSB = IOloc  0, MSB = IOloc63
unsigned char ucReadExtIO[52];       //LSB = IOloc  96, MSB = IOloc511
unsigned char ucReadIOExtShadow[52]; //LSB = IOloc  96, MSB = IOloc511
unsigned char ucWriteExtIO[52];      //LSB = IOloc  96, MSB = IOloc511
unsigned char ucWriteIOExtShadow[52];//LSB = IOloc  96, MSB = IOloc511
unsigned char ucIDRAddress;          //IDR address
unsigned char ucSPMCRAddress; //SPMCR Register address and dW BasePC
unsigned long ulBootAddress;  //Device Boot Loader Start Address
unsigned char ucRAMPZAddress; //RAMPZ Register address in SRAM I/O
                             //space
unsigned int  uiFlashPageSize;   //Device Flash Page Size, Size =
                                 //2 exp ucFlashPageSize
unsigned char ucEepromPageSize;  //Device Eeprom Page Size in bytes
unsigned int uiUpperExtIOLoc;    //Topmost (last) extended I/O
                                 //location, 0 if no external I/O
unsigned long ulFlashSize;       //Device Flash Size
unsigned char ucEepromInst[20];  //Instructions for W/R EEPROM
unsigned char ucFlashInst[3];    //Instructions for W/R FLASH
unsigned char ucSPHaddr;         // Stack pointer high
unsigned char ucSPLaddr;         // Stack pointer low
unsigned int  uiFlashpages;      // number of pages in flash
unsigned char ucDWDRAddress;     // DWDR register address
unsigned char ucDWBasePC;        // Base/mask value of the PC
unsigned char ucAllowFullPageBitstream; // FALSE on ALL new
                                        //parts
unsigned int uiStartSmallestBootLoaderSection //
unsigned char EnablePageProgramming; // For JTAG parts only,
                                     // default TRUE
unsigned char ucCacheType;    // CacheType_Normal 0x00,
                              // CacheType_CAN 0x01,
                              // CacheType_HEIMDALL 0x02
unsigned int uiSramStartAddr     // Start of SRAM
unsigned char ucResetType; // Selects reset type. ResetNormal = 0x00
                           // ResetAT76CXXX = 0x01
unsigned char ucPCMaskExtended;  // For parts with extended PC
unsigned char ucPCMaskHigh;      // PC high mask
unsigned char ucEindAddress;     // Selects reset type.
unsigned int EECRAddress;        // EECR IO address
```

10 Memory Types

Table 10-1. Memory Types.

Memory area	MEM_TYPE value
IO_SHADOW	0x30
SRAM	0x20
EEPROM	0x22
EVENT	0x60
SPM	0xA0
FLASH_PAGE	0xB0
EEPROM_PAGE	0xB1
FUSE_BITS	0xB2
LOCK_BITS	0xB3
SIGN_JTAG	0xB4
OSCCAL_BYTE	0xB5
CAN	0xB6

If AVR Studio issues a ‘Read Memory’ with memory type SRAM three different actions must be taken depending on the address range. If the address is in the range 0x0000 to 0x001F JTAGICE mkII must fetch the data from the general purpose working register file. If the address is in the range 0x0020 to 0x005F data must be fetched from the IO register file. If the AVR supports Extended IO registers and the address is in the range less than 0x00FF data must be fetched from the External IO register file.

If the address is beyond 0x005F data is fetched from internal/external SRAM when no external IO register file is used. If external IO register file is used the address must be beyond 0x0FF to fetch data from internal/external SRAM.

Read memory type SPM reads a single program word while FLASH_PAGE reads a whole page. Write memory uses FLASH_PAGE to program the whole flash. SPM is used only for programming a single page.

Memory type EEPROM_PAGE uses page write and byte read. Memtypes FUSE_BITS, LOCK_BITS and OSCCAL_BYTE use byte read and (write).

The address sent to JTAGICE mkII is byte address for all memory types. Only the start address needs to be supplied.

The Emulator program reports Read Memory errors as Resp_FAILED. Unknown Read Memory types are ignored. Write Memory unknown types are also ignored.

Memory Type CAN is handled by reading 300 bytes each time the memory type is accessed. The access of the CAN memory type is controlled by PARAM_CAN_FLAG. If PARAM_CAN_FLAG = TRUE then read all the 300 bytes if not (FALSE) don’t read.

Writing to the CAN-mailbox is also done by using MemType_CAN. For each byte edited/written there should be a Cmnd_Write_Memory call for this memory type along with the address for the byte.

11 Breakpoints

The OCD system uses Break Point Comparators to set Break Points. The Break Point control unit contains two single Program Memory Break Points, and two combined Break Points. Together, the four Break Points can be configured as either: (One Break Point is always used for single step.)

- 4 single Program Memory Break Points.
- 3 Single Program Memory Break Point + 1 single Data Memory Break Point
- 2 Single Program Memory Break Point + 2 single Data Memory Break Point
- 2 Single Program Memory Break Point + 1 Program Memory Break Point with mask (“range Break Point”)
- 2 Single Program Memory Break Point + 1 Data Memory Break Point with mask (“range Break Point”)

The Data Memory Break Point can be set to one out of three modes; Data Memory Read, Data Memory Write, or Data Memory Read or Write. A Data Memory break sets the AVR CPU in the Stopped mode after finishing the instruction causing the break condition.

Break on data content is not supported.

The OCD system contains different registers in the Break Point control unit.

PSBO and PSB1 – Program Break on single address – are 16-bit compare registers for the Program Counter from the CPU.

PDMSB – Program/Data Mask or Single Break – is the register used for setting a single program Break Point on either a Program Memory or a Data Memory address. Alternatively, PDMSB can act as a mask on the address to the PDSB comparator, thereby implementing a “range-break”.

PDSB – Program/Data Single Break – is used for setting a single Break Point on either a Program Memory or a Data Memory address. Alternatively, PDMSB can mask the address to the PDSB Comparator, thereby implementing a "range-break".

BCR – Break Control Register – is among other things used to control the settings of the four different Break Registers mentioned above.

To set Break Points in the JTAGICE AVR Studio uses CMND_SET_PARAMETER to set the PSB0 or PSB1 Registers. In this case, the address of the Set Parameter command is the address where the Break Points will be located, and the value indicates if the PSB0 or PSB1 Break Points should be set to this location (0 = PSB0, 1 = PSB1). Neccessary modification of BCR is done automatically.

CmndSetParameter and Parameter BreakAddr set the PDMSB and PDSB Registers. BCR is not automatically modified in this case. To activate the PDMSB and PDSB Break Points the command CmndSetParameter and parameter CombBreakCtrl should be used to set BCR to the proper value. When the JTAGICE breaks all Break Points are cleared therefore, Break Points must be set prior to each run. There is no need for Clear Break Points commands.

12 CRC16 Calculations

The following source code performs CRC16 calculation and verification for the JTAGICE mkII Communication Protocol. Use the VerifyChecksum function to verify the integrity of a received command. Returns TRUE if the message integrity is maintained, returns FALSE otherwise.

Use the AppendChecksum function to append the 2 CRC16 bytes to a message prior to sending the message.

Figure 12-1. File: crc16.h.

```
#ifndef CRC16_H
#define CRC16_H
class Crc16
{
public:
    // Calculate the checksum of a message.
    static unsigned short Checksum(  const unsigned char* message,
                                     unsigned long length,
                                     unsigned short crc = 0xffff);
    // Verify that the last two bytes is a (LSB first)
     // valid CRC of the message.
    static bool VerifyChecksum(      const unsigned char* message,
                                     unsigned long length);
    // Append a two byte CRC (LSB first) to message.
     // length is size of message excluding crc.
    // Space for the CRC bytes must be allocated in advance!
    static void AppendChecksum(      unsigned char* message,
                                     unsigned long length);
};
#endif
```

Figure 12-2. File: crc16.cpp

```
#include "Crc16.h"
// Code taken from DataTransportLayer crc16.h
// CRC16 Definitions
const unsigned short crc_table[256] =
{   0x0000, 0x1189, 0x2312, 0x329b, 0x4624, 0x57ad, 0x6536, 0x74bf,
    0x8c48, 0x9dc1, 0xaf5a, 0xbed3, 0xca6c, 0xdbe5, 0xe97e, 0xf8f7,
    0x1081, 0x0108, 0x3393, 0x221a, 0x56a5, 0x472c, 0x75b7, 0x643e,
    0x9cc9, 0x8d40, 0xbfdb, 0xae52, 0xdaed, 0xcb64, 0xf9ff, 0xe876,
    0x2102, 0x308b, 0x0210, 0x1399, 0x6726, 0x76af, 0x4434, 0x55bd,
    0xad4a, 0xbcc3, 0x8e58, 0x9fd1, 0xeb6e, 0xfae7, 0xc87c, 0xd9f5,
    0x3183, 0x200a, 0x1291, 0x0318, 0x77a7, 0x662e, 0x54b5, 0x453c,
    0xbdcb, 0xac42, 0x9ed9, 0x8f50, 0xfbef, 0xea66, 0xd8fd, 0xc974,
    0x4204, 0x538d, 0x6116, 0x709f, 0x0420, 0x15a9, 0x2732, 0x36bb,
    0xce4c, 0xdfc5, 0xed5e, 0xfcd7, 0x8868, 0x99e1, 0xab7a, 0xbaf3,
    0x5285, 0x430c, 0x7197, 0x601e, 0x14a1, 0x0528, 0x37b3, 0x263a,
    0xdecd, 0xcf44, 0xfddf, 0xec56, 0x98e9, 0x8960, 0xbbfb, 0xaa72,
    0x6306, 0x728f, 0x4014, 0x519d, 0x2522, 0x34ab, 0x0630, 0x17b9,
    0xef4e, 0xfec7, 0xcc5c, 0xddd5, 0xa96a, 0xb8e3, 0x8a78, 0x9bf1,
    0x7387, 0x620e, 0x5095, 0x411c, 0x35a3, 0x242a, 0x16b1, 0x0738,
    0xffcf, 0xee46, 0xdcdd, 0xcd54, 0xb9eb, 0xa862, 0x9af9, 0x8b70,
    0x8408, 0x9581, 0xa71a, 0xb693, 0xc22c, 0xd3a5, 0xe13e, 0xf0b7,
    0x0840, 0x19c9, 0x2b52, 0x3adb, 0x4e64, 0x5fed, 0x6d76, 0x7cff,
    0x9489, 0x8500, 0xb79b, 0xa612, 0xd2ad, 0xc324, 0xf1bf, 0xe036,
    0x18c1, 0x0948, 0x3bd3, 0x2a5a, 0x5ee5, 0x4f6c, 0x7df7, 0x6c7e,
    0xa50a, 0xb483, 0x8618, 0x9791, 0xe32e, 0xf2a7, 0xc03c, 0xd1b5,
    0x2942, 0x38cb, 0x0a50, 0x1bd9, 0x6f66, 0x7eef, 0x4c74, 0x5dfd,
    0xb58b, 0xa402, 0x9699, 0x8710, 0xf3af, 0xe226, 0xd0bd, 0xc134,
    0x39c3, 0x284a, 0x1ad1, 0x0b58, 0x7fe7, 0x6e6e, 0x5cf5, 0x4d7c,
    0xc60c, 0xd785, 0xe51e, 0xf497, 0x8028, 0x91a1, 0xa33a, 0xb2b3,
    0x4a44, 0x5bcd, 0x6956, 0x78df, 0x0c60, 0x1de9, 0x2f72, 0x3efb,
    0xd68d, 0xc704, 0xf59f, 0xe416, 0x90a9, 0x8120, 0xb3bb, 0xa232,
    0x5ac5, 0x4b4c, 0x79d7, 0x685e, 0x1ce1, 0x0d68, 0x3ff3, 0x2e7a,
    0xe70e, 0xf687, 0xc41c, 0xd595, 0xa12a, 0xb0a3, 0x8238, 0x93b1,
    0x6b46, 0x7acf, 0x4854, 0x59dd, 0x2d62, 0x3ceb, 0x0e70, 0x1ff9,
    0xf78f, 0xe606, 0xd49d, 0xc514, 0xb1ab, 0xa022, 0x92b9, 0x8330,
    0x7bc7, 0x6a4e, 0x58d5, 0x495c, 0x3de3, 0x2c6a, 0x1ef1, 0x0f78
};
// CRC calculation macros
#define CRC_INIT 0xFFFF
#define CRC(crcval,newchar) crcval = (crcval >> 8) ^
                        crc_table[(crcval ^ newchar) & 0x00ff]
unsigned short Crc16::Checksum(      const unsigned char* message,
                                unsigned long length,
                                unsigned short crc)
```

```
{   for(unsigned long i = 0; i < length; i++)
    {     CRC(crc, message[i]);
    }
    return crc;
}
bool Crc16::VerifyChecksum(const unsigned char* message, unsigned
long length)
{   // Returns true if the last two bytes in a message is the crc
    // of the preceding bytes.
    unsigned short expected = Checksum(message, length - 2);
    return (expected & 0xff) == message[length - 2] &&
           ((expected >> 8) & 0xff) == message[length - 1];
}
void Crc16::AppendChecksum(unsigned char* message, unsigned long
length)
{   unsigned long crc = Checksum(message, length);
    message[length] = (unsigned char)(crc & 0xff);
    message[length+1] = (unsigned char)((crc >> 8) & 0xff);
}
```

13 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

2587C-AVR-03/06