

---

# AVR060: JTAG ICE Communication Protocol

## Introduction

This application note describes the communication protocol used between AVR Studio<sup>®</sup> and JTAG ICE.

- **Commands Sent from AVR Studio to JTAG ICE are Described in Detail**
- **Replies Sent from JTAG ICE to AVR Studio are Described in Detail**
- **Configurable Parameters are Described**
- **Different Memory Types are Described**
- **Special Characters and Packet Formats for Packet Synchronization and Error Control Described**
- **Break Point Handling in JTAG ICE is Described**

The purpose of this document is to enable third party vendors to design their own front-end to the JTAG ICE emulator.



---

**8-bit AVR<sup>®</sup>**  
**Microcontroller**

---

**Application**  
**Note**

Rev. 2524A-AVR-09/02



## Front-end Commands

The following section contains a description of the commands sent from AVR Studio to control the JTAG ICE emulator. All commands are independent and the sequence of commands is therefore insignificant. All commands are followed by a 2-byte synchronization word, Sync\_CRC/EOP. If the Sync\_CRC/EOP is detected, JTAG ICE acknowledges it by a Resp\_OK. If no Sync\_CRC/EOP is detected when expected JTAG ICE replies Resp\_SYNC\_ERROR. This applies to all commands. Note! For any unknown command JTAG ICE should return Resp\_SYNC\_ERROR.

The field format [BYTE] indicates a single unsigned eight bits character. The field format [WORD] indicates a single unsigned 16 bits integer.

### Check if Emulator Present

Reads a sign-on message from the emulator, to determine if the emulator is actually present.

#### Command

Cmnd\_GET\_SIGN\_ON, Sync\_CRC/EOP

#### Response

If the Sync\_CRC/EOP was read successfully:

Resp\_OK, SIGN\_ON\_MESSAGE, Resp\_OK

**Table 1.** Parameters

Parameter Name	Field Usage	Field Format
SIGN_ON_MESSAGE	Text string: "AVRNOCD". It consists of 7 characters and is not zero terminated.	[BYTE] * 7

### Write Emulator Parameter

Writes new settings to key parameters in JTAG ICE. The emulator always returns to default parameter settings on power up.

#### Command

Cmnd\_SET\_PARAMETER, parameter, setting, Sync\_CRC/EOP

#### Response

If Sync\_CRC/EOP was read successfully, and requested parameter code is recognised:

Resp\_OK, Resp\_OK

Note: When writing baud rate parameters the acknowledge is transmitted before the baud rate is changed.

If Sync\_CRC/EOP was read successfully, but requested parameter code is not recognised:

Resp\_OK, Resp\_FAILED

**Table 2.** Parameters

Parameter Name	Field Usage	Field Format
Parameter	Identification of the parameter to be written	[BYTE]
setting	New value for parameter	[BYTE]

See "Parameters" on page 10 for a description of parameters that can be written.

## Read Emulator Parameter

Reads the settings of key parameters in JTAG ICE.

### Command

Cmnd\_GET\_PARAMETER, parameter, Sync\_CRC/EOP

### Response

If Sync\_CRC/EOP was read successfully and requested parameter code is recognised:

Resp\_OK, setting, Resp\_OK

If Sync\_CRC/EOP was read successfully, but requested parameter code is not recognised:

Resp\_OK, Resp\_FAILED , Resp\_FAILED

**Table 3.** Parameters

Parameter Name	Field Usage	Field Format
Parameter	Identification of the parameter to be read	[BYTE]
Setting	Value of parameter	[BYTE]

See “Parameters” on page 10 for a description of parameters that can be read.

## Write Memory

Enables writing of registers, I/O area, SRAM, EEPROM, and Program Memory. Note that the memory can not be written to while the emulator is in Run mode.

Cmnd\_WRITE\_MEMORY works in conjunction with Cmnd\_DATA, which contains the actual data block.

### Command

Cmnd\_WRITE\_MEMORY, memory type, word count, start address, Sync\_CRC/EOP, Cmnd\_DATA, word0, word1 (...) wordn.

### Response

For each Sync\_CRC/EOP that was read successfully:

Resp\_OK

AVR Studio should detect the Resp\_OK before the data block is written to the emulator. The emulator will not expect any data following the Resp\_SYNC\_ERROR indication sent if the Sync\_CRC/EOP fails. Writing the data block might in that case cause unexpected emulator behaviour.

**Table 4.** Parameters

Parameter Name	Field Usage	Field Format
Memory Type	Memory type	[BYTE]
Word Count	Number of words in package minus 1. (word count = 0 means 1 word, 1 means 2 words, 255 means 256 words)	[BYTE]
Start Address	Starting memory address	[BYTE]*3,MSB first
Word0 - Wordn	Words written to memory	[BYTE]/[WORD] <sup>(1)</sup>

Note: 1. See “Memory Types” on page 12 for a description of the memory types.

## Read Memory

Enables reading of registers, SRAM, I/O area, EEPROM, Program Memory and Break Register. Note that the memory can not be read while the emulator is in Run mode.

### Command

Cmnd\_READ\_MEMORY, memory type, word count, start address, Sync\_CRC/EOP

### Response

If Sync\_CRC/EOP was read successfully, and all bytes have been read successfully:

Resp\_OK, word0, word1, ... , wordn, checksum, Resp\_OK

If Sync\_CRC/EOP was read successfully, but emulator failed to read data:

Resp\_OK, word0, word1, ... , wordn, checksum, Resp\_FAILED

Note: If by some reason the emulator failed to read the data, the correct number of words will still be returned, so that the system will still be in sync. AVR Studio should rely on Resp\_FAILED to detect the error.

**Table 5.** Parameters

Parameter Name	Field Usage	Field Format
Memory Type	Memory type	[BYTE]
Word Count	Number of words in package - 1. (word count = 0 means 1 word, 1 means 2 words, 255 means 256 words)	[BYTE]
Start Address	Starting memory address	[BYTE]*3,MSB first
Word0 - Wordn	Words read from memory	[BYTE]/[WORD] <sup>(1)</sup>
Checksum	No checksum check currently implemented in JTAG ICE. 0x00 is sent.	[BYTE]

Note: 1. Program Memory: One word is two bytes, MSB first. All other memory types: One byte.

See "Memory Types" on page 12 for a description of the memory types.

## Write Program Counter

Write new contents to the AVR Program Counter. Note that the Program Counter can not be written while the emulator is in Run mode.

### Command

Cmnd\_WRITE\_PC, program counter, Sync\_CRC/EOP

### Response

If Sync\_CRC/EOP was read successfully:

Resp\_OK, Resp\_OK

**Table 6.** Parameters

Parameter Name	Field Usage	Field Format
Program Counter	New Program Counter	[BYTE]*3, MSB first <sup>(1)</sup>

Note: 1. Note that the Program Counter in JTAG ICE is only 16 bits wide, therefore the high byte is ignored. It should still be transmitted.

**Read Program Counter** Read the current Program Counter. Note that the Program Counter can not be read while the emulator is in Run mode.

**Command** Cmnd\_READ\_PC, Sync\_CRC/EOP

**Response** If Sync\_CRC/EOP was read successfully, and the program counter was correctly returned:

Resp\_OK, program counter, Resp\_OK

If Sync\_CRC/EOP was read successfully, but the program counter could not be determined:

Resp\_OK, program counter, Resp\_FAILED (program counter =0XAA55AA)

PARAMETERS

**Table 7.** Parameters

Parameter Name	Field Usage	Field Format
Program Counter	Current program counter	[BYTE]*3, MSB first <sup>(1)</sup>

Note: 1. Note that the Program Counter in JTAG ICE is only 16 bits wide, therefore the high byte reads 0x00. It should still be transmitted.

**Start Program Execution** Starts program execution at current Program Counter address.

**Command** Cmnd\_GO, Sync\_CRC/EOP

**Response** If Sync\_CRC/EOP was read successfully and the execution was started:

Resp\_OK, Resp\_OK

**Table 8.** Parameters

Parameter Name	Field Usage	Field Format
No Parameters	–	–

**Single Step** Start one instruction execution at current Program Counter address

**Command** Cmnd\_SINGLE\_STEP, Sync\_CRC/EOP

**RESPONSE** If Sync\_CRC/EOP was read successfully:

Resp\_OK, Resp\_OK

**Table 9.** Paramters

Parameter Name	Field Usage	Field Format
No Parameters	–	–

## Stop Program Execution

Stop program execution

### Command

Cmnd\_FORCED\_STOP, Sync\_CRC/EOP

### Response

If Sync\_CRC/EOP was read successfully:  
Resp\_OK, checksum program counter, Resp\_OK

**Table 10.** Paramters

Parameter Name	Field Usage	Field Format
Program Counter	Current Program Counter	[BYTE]*3 (MSB first)
Checksum	See Table 5.	[BYTE]

## Reset User Program

Emulator performs all the actions needed to restart program execution

### Command

Cmnd\_RESET, Sync\_CRC/EOP

### Response

If Sync\_CRC/EOP was read successfully, and program reset completed:  
Resp\_OK, Resp\_OK

**Table 11.** Paramters

Parameter Name	Field Usage	Field Format
No Parameters	–	–

## Get Sync

Sent from AVR Studio to enable JTAG ICE to gain communication synchronization when synchronization is lost.

### Command

Cmnd\_GET\_SYNC

### Response

Resp\_OK

**Table 12.** Parameters

Parameter Name	Field Usage	Field Format
No parameters	–	–

## Read Debug Information

Returns 0x00.

### Command

Cmnd\_GET\_DEBUG\_INFO, Sync\_CRC/EOP

### Response

If Sync\_CRC/EOP was read successfully:  
Resp\_OK, checksum, Resp\_OK

**Table 13.** Parameters

Parameter Name	Field Usage	Field Format
CChecksum, see Table 5.	–	–

**Set Device Descriptor**

AVR Studio sends over AVR device specific information.

**Command**

Cmnd\_Set\_Device\_Descriptor, Structure , Sync\_CRC/EOP

**Response**

If Sync\_CRC/EOP was read successfully:  
Resp\_OK, Resp\_OK

**Table 14.** Parameters

Parameter Name	Field Usage	Field Format
Structure	Device specific information	[BYTE]*123

Note: 1. See Table 30, “Device Description,” on page 18 for actual contents of this structure.

**Erase Page SPM**

Erases a whole page in Flash memory using SPM. This command along with Read Memory and Cmnd Write Memory allows the Flash memory to be modified at run-time.

**Command**

Cmnd\_ERASEPAGE\_SPM, PageAddress, Sync\_CRC/EOP

**Response**

If Sync\_CRC/EOP was read successfully:  
Resp\_OK, Resp\_OK

**Table 15.** Parameters

Parameter Name	Field Usage	Field Format
PageAddress	Address of first byte of page to be erased	[WORD]

**Firmware Upgrade**

Forces the JTAG ICE into Upgrade mode. In this mode, AVR Prog can connect to the emulator and update the application firmware.

**Command**

CmndFirmwareUpgrade, string, Sync\_CRC/EOP

**Response**

If Sync\_CRC/EOP was read successfully:  
Resp\_OK, Resp\_OK

**Table 16.** Parameters

Parameter Name	Field Usage	Field Format
String	Upgrade string “JTAGupgr”	[BYTE]*8

## JTAG ICE Responses

The following section describes the responses sent from JTAG ICE to AVR Studio. Most responses are sent as a result from received commands. Resp\_BREAK, Resp\_INFO and Resp\_SLEEP are sent from Run mode. The response values are listed in Table 27.

**OK** Acknowledge to AVR Studio. Resp\_OK is sent after Sync\_CRC/EOP has been detected and after a valid command has been correctly executed.

**Response** Resp\_OK

**Table 17.** Parameters

Parameter Name	Field Usage	Field Format
No Parameters	–	–

**Failed** Sent if a command execution fails.

**Response** Resp\_FAILED

**Table 18.** Parameters

Parameter Name	Field Usage	Field Format
No Parameters	–	–

**Synchronization Error** Sent to AVR Studio when communication synchronization is lost. Resp\_SYNC\_ERROR is sent after Sync\_CRC/EOP was expected but not detected.

**Response** Resp\_SYNC\_ERROR

**Command** AVR Studio tries to re-establish synchronisation by sending repeatedly:  
Cmd\_GET\_SYNC

**Response** If JTAG ICE detects Cmd\_GET\_SYNC the command should be acknowledged by:  
Resp\_OK

**Table 19.** Parameters

Parameter Name	Field Usage	Field Format
No Parameters	–	–

**Break** Sent if a program is stopped by a Break Point.

**Response** Resp\_BREAK, status

**Table 20.** Parameters

Parameter Name	Field Usage	Field Format
Status	JTAG ICE status word	[WORD]

**Info** If IDR is dirty the IDR will be read and the contents reported to AVR Studio.

**Response** Resp\_INFO, IDR, Resp\_OK

**Table 21.** Parameters

Parameter Name	Field Usage	Field Format
IDR	JTAG ICE IDR BYTE	[BYTE]

**Sleep** If the AVR MCU executes a SLEEP instruction, or resumes execution after sleep, the JTAG ICE detects this and sends this response:

**Response** Resp\_SLEEP, status, Resp\_OK

**Table 22.** Parameters

Parameter Name	Field Usage	Field Format
Status	Indicates sleep status. TRUE or FALSE	[BYTE]

## Parameters

The following JTAG ICE parameters can be read/written by the Cmd\_GET\_PARAMETER and Cmd\_SET\_PARAMETER commands. See Table 28 for parameter values.

**Table 23.** Key Parameters in JTAG ICE

Parameter Name	Description	Read/Write	Field Format
Baud Rate	Determines the communication baud rate. See Table 24 for Baud Rate codes.	R/W	[BYTE]
Flash Page size	Sets the current Flash page size	W	[WORD]
EEPROM Page size	Sets the current EEPROM page size	W	[BYTE]
Hardware Version <sup>(1)</sup>	Defines the current hardware version. For JTAG ICE V1.10 this is 0x40.	R	[BYTE]
Software Version <sup>(1)</sup>	Defines the current software version. For JTAG ICE V1.10 this is 0x72, but changes with new sw versions.	R	[BYTE]
JTAG ID Byte0, 1, 2, and 3.	Indicates the JTAG ID for the emulated device. The ID is kept in a structure in AVR Studio. The ID is four bytes long.	R	[BYTE]
Timers Running	Enable Timers to run.	W	[BYTE]
Units Before	Number of units before the AVR to be communicated with. Default = 0.	W	[BYTE]
Units After	Number of units after the AVR to be communicated with. Default = 0.	W	[BYTE]
Bit Before	Number of IR-bits before AVR. Default = 0.	W	[BYTE]
Bit After	Number of IR-bits after AVR. Default = 0.	W	[BYTE]
Change of Flow	CPU is set in Stopped mode when break on flow conditions are met.	W	[BYTE]
OCD Vtarget	Measure target voltage. Answer with 8-bit resolution.	R	[BYTE]
OCD JTAG Clock	Determines the JTAG clock rate. It is ¼ of the source clock.	R/W	[BYTE]
Break Adress 1 H/L	Break Adress 1	W	[BYTE]
Break Adress 2 H/L	Break Adress 2	W	[BYTE]
Combined Break Control	Value to be written to the Break Control Register (BCR). This parameter is called after setting Break Adress 1 and 2.	R/W	[BYTE]

**Table 23.** Key Parameters in JTAG ICE

Parameter Name	Description	Read/Write	Field Format
IReg H/L	Used by AVR Studio to put an instruction in the internal scan chain. To avoid flash wear it is put back in the internal scan chain and executed instantly instead of being placed in Flash. Used when dealing with the AVR Break Instruction.	R/W	[WORD]
OCD Break Cause	Determines the value of the Break Status Register.	R	[BYTE]
External Reset	Performs an External Reset by forcing the nSRST pin low.	W	[BYTE]
MCU_mode	Used by AVR Studio to detect if part is in Run or Stopped mode.	R	[BYTE]
PSB0 H/L	Used to set up a 16-bit address in a Compare Register. PC will break on this address.	W	[BYTE]
PSB1 H/L	Used to set up a 16-bit address in a Compare Register. PC will break on this address.	W	[BYTE]

Note: 1. These parameter values can be read in AVR Studio under “Help”, “About AVR Studio”, “Info”.

**Table 24.** Baud Rate Parameters Value

Parameter Value	Baud Rate
0xFF	115200
0xFE	57600
0xFD	38400
0xFA	19200
0xF8	14400
0xF4	9600

## Memory Types

Following memory type codes are supported in JTAG ICE for use with the commands Cmnd\_WRITE\_MEMORY and Cmnd\_READ\_MEMORY.

**Table 25.** Memory Type Constants [MEM\_TYPE]

Memory Area	MEM_TYPE Value
SRAM	0x20
EEPROM	0x22
PML <sup>(2)</sup>	0xA0
BreakReg	0x90
IOShadow	0x30
FLASH_JTAG <sup>(1)</sup>	0xB0
EEPROM_JTAG <sup>(1)</sup>	0xB1
FUSE_JTAG <sup>(1)</sup>	0xB2
LOCK_JTAG <sup>(1)</sup>	0xB3
SIGN_JTAG <sup>(1)</sup>	0xB4
OSCAL_JTAG <sup>(1)</sup>	0xB5

- Notes:
1. Accessing these memory types will access the JTAG Programming Interface. The device must be in Programming mode before using these commands. FLASH\_JTAG is the memory type used when uploading the program to the target device.
  2. PML access Flash using SPM and LPM. Writing is on page basis, and a page must be erased before writing it, using the Cmnd\_ERASEPAGE\_SPM command.

If AVR Studio issues a “Read Memory” or “Write 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 JTAG ICE 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 and more that 0x005F, 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.

Memory type EEPROM\_JTAG uses page write or byte write and byte read. Memtypes FUSE\_JTAG, LOCK\_JTAG, SIGN\_JTAG and OSCAL\_JTAG use byte read and (write).

The address sent to JTAG ICE is a word address for Program Memory and byte address for all other memory types. Only the start address and the number of bytes to be read/written must 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.

## Synchronization

### Sync\_CRC/EOP

The Sync\_CRC/EOP is a 2-byte constellation placed at the end of all packets going from the PC to the JTAG ICE. CRC checking is not implemented in JTAG ICE and therefore this 2-byte constellation always reads 0x20, 0x20. When detected by the JTAG ICE it should be acknowledged by a Resp\_OK. Note! No command sent from the PC will execute unless the transmitted Sync\_CRC/EOP is acknowledged. The JTAG ICE will look for this 2-byte constellation at the end of every command sent from the PC. If the JTAG ICE receives some character other than the Sync\_CRC/EOP when expected, the JTAG ICE responds by sending Resp\_SYNC\_ERROR.

### Synchronization Recovery

If the JTAG ICE does not read a Sync\_CRC/EOP when expected, it automatically assumes it is out of sync with the PC. The JTAG ICE will terminate command execution and return a Resp\_SYNC\_ERROR to the PC. When the JTAG ICE detects a valid Cmnd\_GET\_SYNC, the JTAG ICE will acknowledge this with a Resp\_OK. If no Resp\_OK is received from the JTAG ICE, continuous Cmnd\_GET\_SYNC commands must be transmitted until acknowledged.

## Break Points

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.

PSB0 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 JTAG ICE AVR Studio uses `Cmdn_setParameter` and parameters PSB0 H/L and PSB1 H/L 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). Necessary modification of BCR is done automatically.

`CmdnSetParameter` 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 `CmdnSetParameter` and parameter `CombBreakCtrl` should be used to set BCR to the proper value.

When the JTAG ICE 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.

## JTAG ICE Communication Protocol Summary

The following section summarizes the JTAG ICE Communication Protocol. Data sent from the JTAG ICE is shown in bold/italic.

**Table 26.** Commands

Hex	Command	SEQUENCE
0x20	Get Synch	<b>[Resp_OK]</b>
0x31	Single Step	[Sync_CRC/EOP] <b>[Resp_OK]</b>
0x32	Read PC	[Sync_CRC/EOP] <b>[Resp_OK]</b> [program counter] <b>[Resp_OK]</b>
0x33	Write PC	[program counter] [Sync_CRC/EOP] <b>[Resp_OK]</b> <b>[Resp_OK]</b>
0xA2	Firmware Upgrade	[upgrade string] [Sync_CRC/EOP] <b>[Resp_OK]</b> <b>[Resp_OK]</b>
0xA0	Set Device Descriptor	[device info] [Sync_CRC/EOP] <b>[Resp_OK]</b> <b>[Resp_OK]</b>
0x42	Set Parameter	[parameter] [setting] [Sync_CRC/EOP] <b>[Resp_OK]</b> <b>[Resp_OK]</b>
0x46	Forced Stop	[Sync_CRC/EOP] <b>[Resp_OK]</b> <b>[checksum]</b> <b>[program counter]</b> <b>[Resp_OK]</b>
0x47	Go	[Sync_CRC/EOP] <b>[Resp_OK]</b>
0x52	Read Memory	[memory type] [word count] [start address] [Sync_CRC/EOP] <b>[Resp_OK]</b> <b>[word 0]</b> ... <b>[word n]</b> <b>[checksum]</b> <b>[Resp_OK]</b>
0x53	Get Sign On	[Sync_CRC/EOP] <b>[Resp_OK]</b> ["AVRNOCD"] <b>[Resp_OK]</b>
0xA1	Erase Page spm	[address] [Sync_CRC/EOP] <b>[Resp_OK]</b> <b>[Resp_OK]</b>
0x57	Write Memory	[memory type] [word count] [start address] [Sync_CRC/EOP] <b>[Resp_OK]</b> [Cmd_DATA] [word 0] ... [word n]
0x64	Get Debug Info	[Sync_CRC/EOP] <b>[Resp_OK]</b> <b>[0x00]</b> <b>[Resp_OK]</b>
0x71	Get Parameter	[parameter] [Sync_CRC/EOP] <b>[Resp_OK]</b> <b>[setting]</b> <b>[Resp_OK]</b>
0x78	Reset	[Sync_CRC/EOP] <b>[Resp_OK]</b> <b>[Resp_OK]</b>
0xA3	Enter Progmode	[Sync_CRC/EOP] <b>[Resp_OK]</b> <b>[Resp_OK]</b>
0xA4	Leave Progmode	[Sync_CRC/EOP] <b>[Resp_OK]</b> <b>[Resp_OK]</b>
0xA5	Chip Erase	[Sync_CRC/EOP] <b>[Resp_OK]</b> <b>[Resp_OK]</b>

**Table 27. Responses**

Hex	ASCII	Response	SEQUENCE
0x41	A	OK	
0x42	B	Break	[Resp_Break] [break status register H] [break status register L]
0x47	G	Info	[IDR dirty] [Resp_INFO] [IDR] [Resp_OK]
0x46	F	Failed	
0x45	E	Sync Error	getchar() != Sync_EOP
0x48	H	Sleep	
0x49	I	Power	-

**Table 28. Parameter Names**

Hex	ASCII	Parameter	JTAG ICE Value
0x7A		Hardware Version	0xC0
0x81		Ireg High	
0x82		Ireg Low	
0x62	b	Baudrate	
0x7B	{	SwVersion	0x68
0x84		OCD Vtarget	
0x86		OCD JTAG Clock	¼ of part frequency
0x87		OCD Break cause	
0xA0		Timers Running	
0xA1		Change of Flow	
0xA2		Break Addr1H	
0xA3		Break Addr1L	
0xA4		Break Addr2H	
0xA5		Break Addr2L	
0xA6		CombBreakCtrl	
0xA7		JTAGIDByte0	Device specific
0xA8		JTAGIDByte1	Device specific
0xA9		JTAGIDByte2	Device specific
0xAA		JTAGIDByte3	Device specific
0xAB		Units Before	
0xAC		Units After	
0xAD		Bit Before	
0xAE		Bit After	
0x8B		External Reset	

**Table 28.** Parameter Names (Continued)

Hex	ASCII	Parameter	JTAG ICE Value
0x88		Flash PageSizeL	Device specific
0x89		Flash PageSizeH	Device specific
0x8A		EEPROM PageSize	Device specific
0xB3		MCU_mode	
0xAF		PSB0L	
0xB0		PSB0H	
0xB1		PSB1L	
0xB2		PSB1H	

**Table 29.** Memory Types

Memory	Address
IO Shadow	0x30
Sram	0x20
Eeprom	0x22
Event L	0x60
PML	0xA0
FLASH_JTAG	0xB0
Break Reg	0x90
EEPROM_JTAG	0XB1
FUSE_JTAG	0XB2
LOCK_JTAG	0XB3
SIGN_JTAG	0XB4
OSCCAL_JTAG	0XB5

**Table 30.** Device Description

Device	Structure Values
ATmega16	0xCF,0xAF,0xFF,0xFF,0xFE,0xFF,0xFF,0xFF, 0x87,0x26,0xFF,0xEF,0xFE,0xFF,0x3F,0xFA, 0x00,0x00,0x00,0x00,0x00,0x2F,0x00,0x00, 0x00,0x00,0x00,0x00,0x00,0x2F,0x00,0x00, 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x31, 0x57, 0x00, 128, 0, 0, 0x80, 0x1F, 0x00, 0x00, 0
ATmega162	0xF7,0x6F,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF, 0xF3,0x66,0xFF,0xFF,0xFF,0xFF,0xFF,0xFA, 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, 0x02,0x18,0x00,0x30,0xF3,0x0F,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x02,0x18,0x00,0x20,0xF3,0x0F,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x04, 0x57, 0x00, 128, 0, 4, 0x80, 0x1F, 0x00, 0x00, 0x8B
ATmega169	0xFF,0xFF,0xFF,0xF0,0xDF,0x3C,0xBB,0xE0, 0xB6,0x6D,0x1B,0xE0,0xDF,0x3C,0xBA,0xE0, 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, 0x43,0xDA,0x00,0xFF,0xF7,0x0F,0x00,0x00,0x00,0x00,0x4D,0x07,0x37,0x00,0x00,0x00, 0xF0,0xF0,0xDE,0x7B, 0x43,0xDA,0x00,0xFF,0xF7,0x0F,0x00,0x00,0x00,0x00,0x4D,0x05,0x36,0x00,0x00,0x00, 0xE0,0xF0,0xDE,0x7B, 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x31, 0x57, 0x00, 128, 0, 4, 0x80, 0x1F, 0x00, 0x00, 0xFE

Table 30. Device Description (Continued)

Device	Structure Values
ATmega323	0xCF,0xAF,0xFF,0xFF,0xFE,0xFF,0xFF,0xFF, 0x87,0x26,0xFF,0xEF,0xFE,0xFF,0x3F,0xFA, 0x00,0x00,0x00,0x00,0x00,0x2F,0x00,0x00, 0x00,0x00,0x00,0x00,0x00,0x2F,0x00,0x00, 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x31, 0x57, 0x00, 128, 0, 0, 0x00, 0x3F, 0x00, 0x00, 0
ATmega32	0xFF,0x6F,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF, 0xFF,0x66,0xFF,0xFF,0xFF,0xFF,0xBF,0xFA, 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x31, 0x57, 0x00, 128, 0, 4, 0x00, 0x3F, 0x00, 0x00, 0
ATmega128	0xCF,0x2F,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF, 0xCF,0x27,0xFF,0xFF,0xFF,0xFF,0xFF,0xFE, 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, 0x3E,0xB5,0x1F,0x37,0xFF,0x1F,0x21,0x2F,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x3E,0xB5,0x0F,0x27,0xFF,0x1F,0x21,0x27,0x00,0x00,0x00,0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x22, 0x68, 0x3B, 0, 1, 8, 0x00, 0xFE, 0x00, 0x00, 0x9D





## Atmel Headquarters

### *Corporate Headquarters*

2325 Orchard Parkway  
San Jose, CA 95131  
TEL 1(408) 441-0311  
FAX 1(408) 487-2600

### *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 Tsimhatsui  
East Kowloon  
Hong Kong  
TEL (852) 2721-9778  
FAX (852) 2722-1369

### *Japan*

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

## Atmel Operations

### *Memory*

2325 Orchard Parkway  
San Jose, CA 95131  
TEL 1(408) 441-0311  
FAX 1(408) 436-4314

### *Microcontrollers*

2325 Orchard Parkway  
San Jose, CA 95131  
TEL 1(408) 441-0311  
FAX 1(408) 436-4314

La Chantrerie  
BP 70602  
44306 Nantes Cedex 3, France  
TEL (33) 2-40-18-18-18  
FAX (33) 2-40-18-19-60

### *ASIC/ASSP/Smart Cards*

Zone Industrielle  
13106 Rousset Cedex, France  
TEL (33) 4-42-53-60-00  
FAX (33) 4-42-53-60-01

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

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

### *RF/Automotive*

Theresienstrasse 2  
Postfach 3535  
74025 Heilbronn, Germany  
TEL (49) 71-31-67-0  
FAX (49) 71-31-67-2340

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

### *Biometrics/Imaging/Hi-Rel MPU/ High Speed Converters/RF Datacom*

Avenue de Rochepleine  
BP 123  
38521 Saint-Egreve Cedex, France  
TEL (33) 4-76-58-30-00  
FAX (33) 4-76-58-34-80

---

### *e-mail*

[literature@atmel.com](mailto:literature@atmel.com)

### *Web Site*

<http://www.atmel.com>

### © Atmel Corporation 2002.

Atmel Corporation makes no warranty for the use of its products, other than those expressly contained in the Company's standard warranty which is detailed in Atmel's Terms and Conditions located on the Company's web site. The Company assumes no responsibility for any errors which may appear in this document, reserves the right to change devices or specifications detailed herein at any time without notice, and does not make any commitment to update the information contained herein. No licenses to patents or other intellectual property of Atmel are granted by the Company in connection with the sale of Atmel products, expressly or by implication. Atmel's products are not authorized for use as critical components in life support devices or systems.

ATMEL®, AVR®, and AVR Studio® are the registered trademarks of Atmel.

Other terms and product names may be the trademarks of others.



Printed on recycled paper.