AVR1327: Two Wire Interface (TWI) Slave Bootloader for Atmel AVR XMEGA

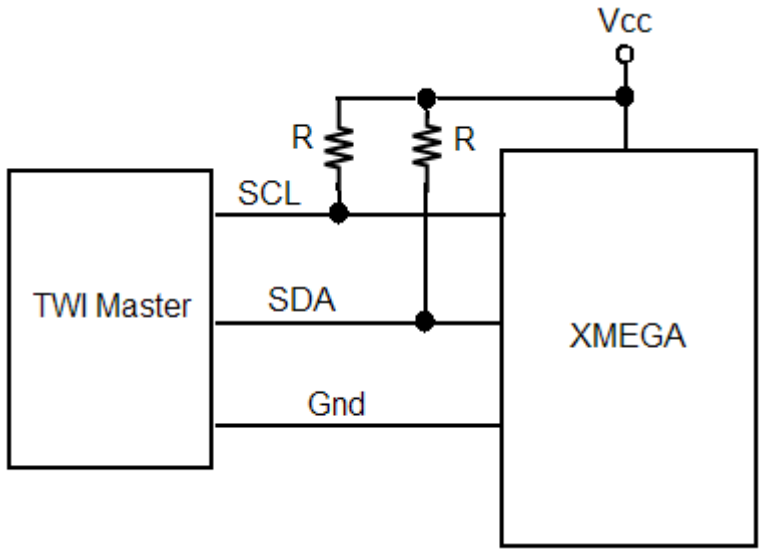
Features

- **Atmel® AVR® XMEGA® family devices**
- **Bootloader protocol for any ATxmega over two wire interface**
- **Host can read ATXmega parameters such as flash page size and number of pages for easy driver development**
- **Host can write new XMEGA code to XMEGA flash program and EEPROM memory**

1. Introduction

This document discusses a general purpose ATxmega Bootloader that is first programmed into the ATxmega device. Next, the ATxmega Bootloader code is set in a mode to communicate with the TWI Master via Two Wire Interface shown in **Figure 1-1.** The Host Computer can issue TWI commands to the ATxmega and write to the ATxmega flash program memory. Finally, the ATxmega will execute this newly written code from its flash memory.

Figure 1-1. TWI Master-to-ATxmega block diagram.



Resistor values of R may be 1K-10K depending on selected TWI bus speed.

8-bit Atmel Microcontrollers

Application Note

Rev. 8435A-AVR-09/11

2. Prerequisites

The Bootloader application discussed in this document requires basic familiarity with following:

- C programming language for embedded systems
- For the software project included with this app note, compiling C projects with IAR™ C/C++ Compiler for AVR 5.51 compiler.
- General familiarity with Bootloader as related to embedded systems and microcontrollers.
- A method to debug and test the compiled application, or download the application hex files into the targeted ATxmega device, such as the Atmel JTAGICE mkII or JTACICE3.
- A general familiarity with TWI protocol and electrical connection requirements

3. Limitations

- For security reasons, specifically to protect IP code from unauthorized copying, the Bootloader code does not implement a method to read any program or other type of the ATxmega memory. The Bootloader can only write to these memories. However, the code could be expanded and modified to execute memory reads.
- A CRC command is implemented to allow detection of errors in application code after download into the ATxmega application flash memory.
- The software solution with this application note was compiled with the IAR C compiler.

4. Memory Protection

While the Bootloader code has no mechanism to read ATxmega memories, the ATxmega memories and associated IP can still be read by Atmel tools or third party programmers. To insure that this does not occur, the ATxmega has four levels related to Bootloader code protection: Nolock, Writelock, Readlock and Read and Writelock. This protection is set via fuses ***or*** the **Nonvolatile Memory Lock Bit Register.** For more information, refer to the ATxmega data sheet **Boot Lock Bits for the Boot Loader Section.**

Additionally, the flash program memory allocated to the Application has identical levels of protection Nolock, Writelock, Readlock and Read and Writelock. This protection is also set via fuses ***or*** the **Nonvolatile Memory Lock Bit Register. For more information on protection related to the Application Section**, refer to the ATxmega data sheet **Boot Lock Bit Application Section.**

5. Number System

The Bootloader uses little endian byte ordering. As an example 0x1234 will be sent as 0x34 – 0x12 on the TWI.

6. Abbreviations

- **<SLA+W>:** Used to indicate slave address and that the following transaction is a TWI write operation. The slave address has been specified in the conf-twi.h file as follows:

```
#define TWI_ADDRESS(0x30)//User may select a different address
```

- **<SLA+R>:** Used to indicate slave address and that the following transaction is a TWI read operation.
- **<s>:** Stop condition.
- **<rs>:** Repeated start condition, usage is optional and can be omitted here.
- **N:** Number of bytes in one flash page, defined below

7. ATXmega target device resource requirements

Table 7-1 and **7-2** describe typical Peripheral requirements and Memory requirements.

Other versions of the ATmega have variations on the peripherals available. An example is the ATxmega32A4 which has two TWI modules: TWIC and TWIE.

Table 7-1. Peripheral requirements.

Peripheral	Pin(s)	Configurable?
An ATXmega TWI module,	In this example, TWIC and ,PORTC pins 0 and 1	TWIC or TWIE In conf-twi.h file: #define TWI_BASE (TWIC.SLAVE)
No interrupts, no additional peripherals		In conf-twi.h file: #define TWI_ADDRESS (0x30) User may select a different slave address.

Table 7-2. Memory requirements [(1)].

Memory	Typical size	Maximum size
Program memory	1382 Bytes if optimized for size	1746 Bytes with zero optimization
Data memory	352 Bytes	352 Bytes
Internal EEPROM memory	EEPROM location 0x00 only. See Note 2.	

Note: 1. Exact memory requirements depend on a variety of factors, such as compiler version, optimization levels, and addition or removal of configurable functionality.

2. A single EEPROM is used as follows: If 0xAA is written to this location, the Bootloader code will begin execution after Reset. The Bootloader will write a 0x00 to this location which prevents re-entry, and thereby allows the execution of the application code.

In main.c,

```
#define BOOTLOADER_WILDCARD     (0xAA)
#define WILDCARD_ADDRESS        (0x00)
```

8. How to build and run the software

1. Create a new directory on your PC. From the website www.atmel.com, locate this app note, avr1327 from this web page, locate and download the software to your local directory.
2. Extract the files to your new directory.
3. Download and install the IAR EWAVR C compiler. To locate this C Compiler, refer to this document, Section **13. References**.
4. In this new directory, double click on the avr1327-bootloader-xmega.eww file (the workspace file). The workspace will open, and the project's files can be examined.

9. Modifying the TWI Slave Bootloader for other ATXmega Devices

9.1 How to establish the Bootloader starting address

9.1.1 ***Very Important:*** Since this is a Bootloader application, it must be compiled and linked to run in the ATxmega's Bootloader Section of flash memory. Examples:

ATxmega16A1 or 16D4 devices: Bootloader starts at word address 0x2000.
ATxmega32A1 or 32D4 devices: Bootloader starts at word address 0x4000.

For more information on Flash Memory addresses, check the ATxmega A or ATxmega D User's Guide, search for Flash Program Memory

Different ATxmegas have their Bootloader memory space at different addresses. The compiler will determine the starting address of the Bootloader from the .xcl files below.

Note: The following is the IAR Compiler-generated default .xcl file, which links the code to start in low flash memory space, after the interrupt vector table.

$TOOLKIT_DIR$\src\template\cfgxm32d4.xcl

Figure 9.1.1 Compiler supplied .xcl file

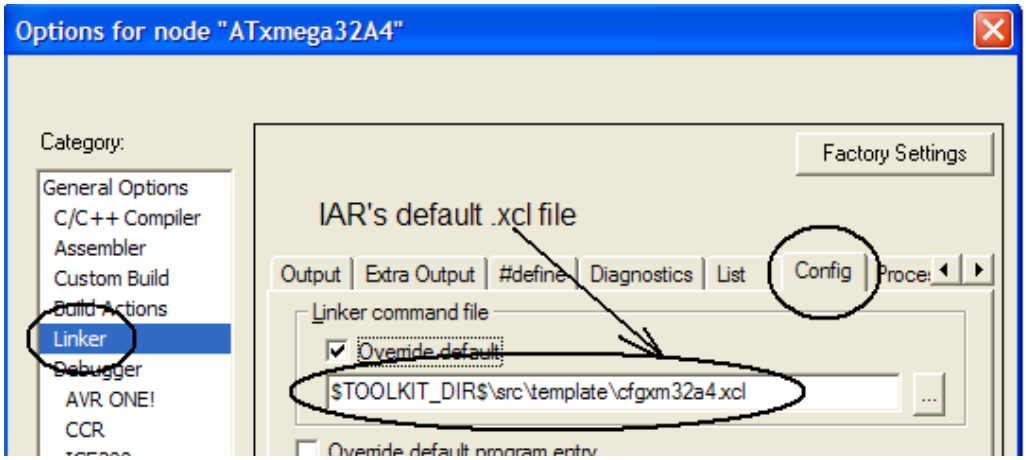
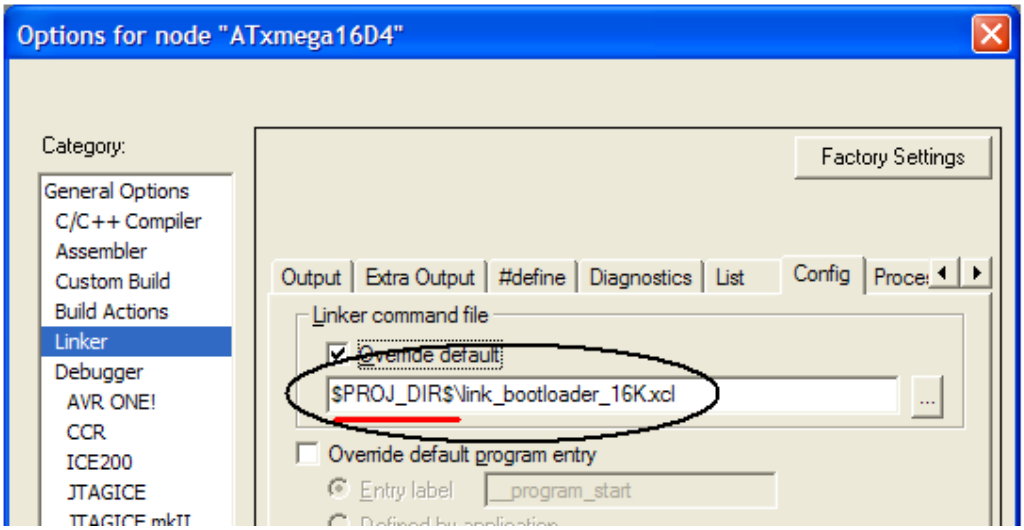


Figure 9.1.2 How to use a custom .xcl file



9.1.2 **Figure 9.1.1** shows the .xcl linker file that is created by the IAR Linker as the project is built. **Figure 9.1.2** shows the format for using a custom .xcl file. Notice the $PROJ_DIR$\ prefix to the .xcl file name.

9.1.3 In the Workspace directory, a custom **link_bootloader32K.xcl** is included and may be modified with a different application start address. Search for the following text and modify, if desired.

```
-D_..X_APPLICATION_SECTION_START= 2000 //Word address
```

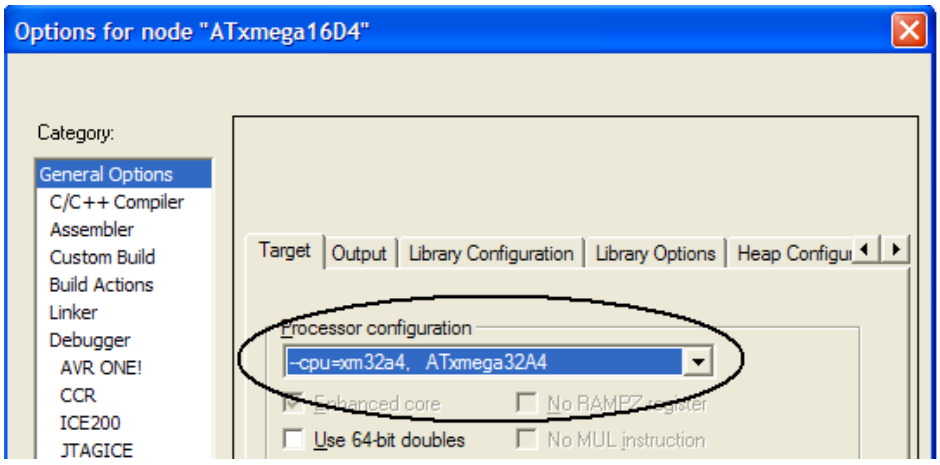
9.1.4 Based on your choice of ATxmega, place the number 2000 or 4000 in the file at the above instruction's location, where 2000 is shown above.

9.1.5 Compile the project.

9.2 Selecting a different ATXmega

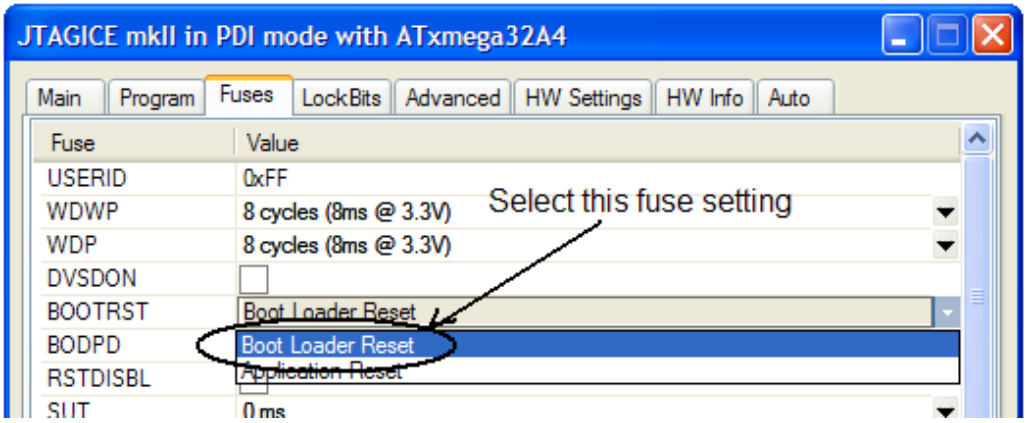
9.2.1 You may change to a different ATxmega if desired. Under IAR project General Options, select your ATxmega Refer to Figure 9.2:

Figure 9.2. Options for ATxmega device selection



9.3 ATXmega Fuse settings necessary for Bootloader operation

The ATXmega has a jump-to-Bootloader fuse that must be enabled, so that the AVR will begin execution at the beginning of the Bootloader code. As described above, this will be 0x2000, 0x4000 (word address). The ATxmega **BOOTRST** fuse should be set to **Boot Loader Reset**:



9.4 As code execution begins, two ways to enter the Bootloader code

Even after the Bootloader is correctly compiled and programmed into the ATxmega, the Bootloader code will only be executed if one or both of the following conditions are met:

1. By asserting a low on a user defined IO port input pin, followed a reset release. In this example this is PORTB pin 3. This pin is defined in main.c as follows:
   #define BOOT_PORT (PORTB)
2. By writing a wildcard to a defined EEPROM address. In this example the EEPROM address is 0x00 and the wildcard is 0xAA. If this EEPROM location contains 0x00, the ATxmega will begin execution of the Bootloader code, not the application code.

Important: If a low level is asserted as in 1. above, that low level must be removed or the Bootloader will be re-entered after every ATxmega Reset.

9.5 Debugging the code

You have the option of debugging this project using IAR, or Atmel AVR Studio® 4 or Studio 5

5. Verify that your code address starts approximately 0x2000 or 0x4000 word addresses, as described earlier in this app note.

6.

10. Master Commands to Bootloader

10.1 Bootloader Register and Memory Map

The registers that control and allow data to be sent to/from the Bootloader are located in the following memory map. Notice how the registers are sequential, followed by the Page Buffer, N bytes long. Following the N-byte-long page buffer is the Command Register.

Figure 10.1 describes the number of bytes per page divided by 256.

Table 10.1 defines **N.**

Figure 10.1 Bootloader Memory Map

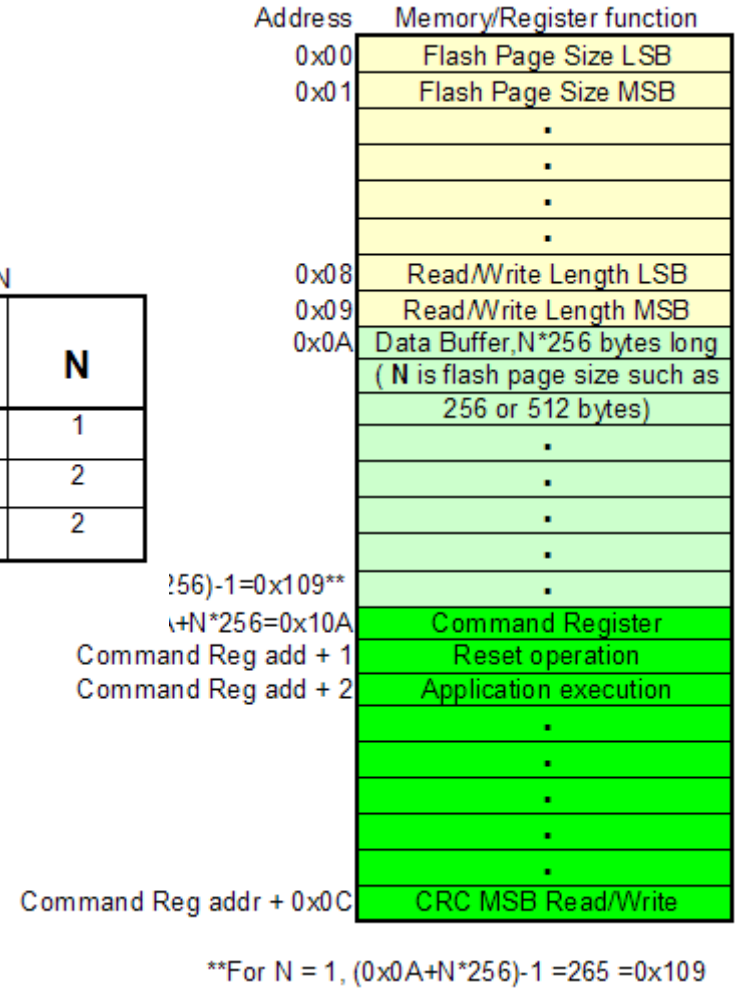


Table 10.1 Number of Flash pages and N

ATxmega device	Number of bytes per page	Number of pages	N
ATxmega64	256	256	1
ATxmega128	512	256	2
ATxmega256	512	512	2

N = (no. of bytes per page)/256

11. Operation

The TWI Slave Bootloader control and data registers allow the TWI master to perform actions such as Read Page Size, Erase Flash etc. In order for the TWI slave Bootloader to perform an action, the TWI master must execute a TWI write cycle to a specific register. Depending on the register address, the TWI master may need to write additional values. Upon receiving this information, the TWI slave Bootloader performs corresponding action. The TWI master can read the result of the operation by performing a TWI read.

11.1 Bootloader Commands

This section describes the commands and data flow from/to Master and ATXmega Slave.

Addresses of Bootloader control and data registers are described in **Table 11.1**. These registers have the format:

- SLA+R, a TWI Master reads the Bootloader register, SLA+R, followed by two 8-bit addresses. The 4th operation is reading the data.
- The registers are sequential. The Master may read or write the next register with another Read command, SLA+R or ,SLA+W>, respectively.
- SLA+W, a TWI Master writes to the Bootloader register or SLA+W, followed by two 8-bit addresses. Notice that some registers are read-only.

11.2 Auto-Increment of ATxmega Slave address

After the <SLA+W> or <SLA+R> operation is sent from the Master to the slave, the next operation defines the destination register or buffer address. Additional operations (other than <SLA+W> or <SLA+R>) will *auto-increment* the Slave register or buffer address. See the section below titled **Program Flash (0x04).**

Table 11.1 Bootloader Control Registers

	TWI Operation # Slave acknowledges each operation)					Comments
	1	2	3	4	5	
		16-bit Address				
		Lwr 8 bits	Upr 8 bits			
8-bit register Name						
Page Size LSB	SLA+R	0x00	0x00	flash pg size Bit[7:0]		
Page Size MSB	SLA+R	0x01	0x00	flash pg size Bit[15:8]		
Number of Pages	SLA+R	0x02	0x00	Bit[7:0] of no. of flash pgs		
Number of Pages	SLA+R	0x03	0x00	Bit[15:8] of no. of flash pgs		Used if over 128 flash pgs
ID0 and ID1	SLA+W	0x04	0x00	write ID0	write ID1	Sequential register writes
ID0 and ID1	SLA+R	0x04	0x00	read ID1	read ID0	Sequential register reads
ID0	SLA+W	0x04	0x00	write ID0		Single register write
ID0	SLA+R	0x04	0x00	read ID1		Single register read
ID1	SLA+W	0x05	0x00	write ID1		Single register write
ID1	SLA+R	0x05	0x00	read ID0		Single register read
Address	SLA+W	0x06	0x00	Address Bit[7:0]		
Address	SLA+R	0x06	0x00	Address Bit[7:0]		
Address	SLA+W	0x07	0x00	Address Bit[15:8]		
Address	SLA+R	0x07	0x00	Address Bit[15:8]		
Write Length LSB	SLA+W	0x08	0x00	Length Bit[7:0]		
Read Length LSB	SLA+R	0x08	0x00	Length Bit[7:0]		
Write Length MSB	SLA+W	0x09	0x00	Length Bit[15:8]		
Read Length MSB	SLA+R	0x09	0x00	Length Bit[15:8]		
N byte Data Buffer	SLA+W	0x0A	0x00	0x00	Continuous TWI writes to memory, N*256 times	
Reset	SLA+W	0x0A	0x01*	0x01		Reset command
Exit bootloader, begin application execution	SLA+W	0x0A	0x00	0x02		Exit Bootloader and run application
Flash Erase	SLA+W	0x0A	0x01*	0x03		Erase Flash
Program Flash	SLA+W	0x0A	0x01*	0x04		Program flash
Program EEPROM	SLA+W	0x0A	0x01*	0x05		Program EEPROM
Calc Flash CRC	SLA+W	0x0A	0x01*	0x06		Calc Flash CRC:TWI Write followed by TWI Read
Read Flash CRC	SLA+R	CRC[7:0]	CRC[15:8]			
Calc EEPROM CRC	SLA+W	0x0A	0x01*	0x07		Calc EEPROM CRCTWI Write followed by TWI Read
Read EEPROM CRC	SLA+R	CRC[7:0]	CRC[15:8]			
EEPROM Length LSB	SLA+W	0x0A	0x01*	0x08	Length C	
EEPROM Length LSB	SLA+R	0x0A	0x01*	0x08	Length Bit[7:0]	
EEPROM Length MSB	SLA+W	0x0A	0x01*	0x09	Length Bit[15:8]	
EEPROM Length MSB	SLA+R	0x0A	0x01*	0x09	Length Bit[15:8]	
CRC LSB	SLA+W	0x0A	0x01*	0x0B	CRC Bit[7:0]	
CRC LSB	SLA+R	0x0A	0x01*	0x0B	CRC Bit[7:0]	
CRC MSB	SLA+W	0x0A	0x01*	0x0C	CRC Bit[15:8]	
CRC MSB	SLA+R	0x0A	0x01*	0x0C	CRC Bit[15:8]	
			* means 0x01 for N = 1 * means 0x02 for N = 2			

11.3 **The Command Register**

In the above **Table 11.1**, notice the Command Register. This register is located in the ATxmega SRAM memory map, just following the Data Buffer (which is typically 256 or 512 bytes in length).

The address of this Command Register is 0x09 + 256*N, where N is defined in Table 10.1. 0x09 is necessary since it is the number of registers at the beginning of the memory map.

For this above table, Table 11.1, N = 1, and N*256 = 256 = 0x100. . So, the Command register address is formed as follows: 0x0A+0x100 = 0x10A.

- The lower 8 address bits are 0x09.

- The upper 8 address bits are 0x01.

11.3.1 Bootloader identification (test if Bootloader is in target ATxmega)

The Master may want to confirm that this particular ATxmega contains a functioning Bootloader. The Bootloader is identified by the Master, by setting the Command Index to four, then write two random bytes, followed by setting the address pointer to four and read two bytes. The two bytes that were written should be read back in reverse order. This operation will confirm that the Master is indeed communicating with TWI Slave Bootloader,

<SLA+W>0x04 0x00 0xAA 0x55<s>

<SLA+R>0x04 0x00<rs>0x55 0xAA<s>

11.3.2 Reset (0x01)

To perform a reset operation, write **0x01** to the command register. In the below example the address is formed as follows:

N = 1, and N*256 = 256 = 0x100. . So, the Command register address is formed as follows: 0x09+0x100 +1= 0x10A

- The lower 8 address bits are 0x0A
- The upper 8 address bits are 0x01.

<SLA+W>0x0A **0x01** 0x01<s>

The reset operation will immediately perform a reset, followed by code execution from the beginning of the Bootloader code, if the entry conditions are met as described in Section 9.5.

11.3.3 Application Execution (0x02)

If the input pin described in Section 9.5 is not shorted to ground, then this command will

1. Write 0x00 into the EEPROM location 0x00, which prevents re-entry into the he Bootloader.
2. Jump to the beginning of the application code.

   <SLA+W>0x0A 0x01 0x02<s> Note that the 0x01 is actually the value of **N**, which will change if the number of bytes per page changes.

11.3.4 Erase Flash (0x03)

This command will erase the whole flash except for the Bootloader. Again this example uses N = 1, as described in the above examples.

<SLA+W>0x0A 0x01 0x03<s>

11.3.5 Program Flash (0x04)

In the following example, notice that the auto-increment feature is utilized, this allows Slave registers to be accessed sequentially.

This command will take the flash page, previously loaded into the data buffer, and program it into the ATxmega application memory which is specified in the address register. In the example below, at ATxmega flash address 0x0001,

The command consists of the following TWI operations:

1. <SLA+W>0x06 0x00 — TWI master sends Address command
2. 0x01 0x00 — Write Flash 16-bit address: LSB, then MSB
3. 0x01 0x00 MSB — Write Flash Write length, 16-bit addr: LSB then
4. <SLA+W> 0x0A 0x04<s> — TWI sends the Program Flash command

The TWI sequence is as follows:

<SLA+W>0x06 0x00 [*Offset*] 0x01 0x00 [*Address*] 0x00 0x01 [*Length*] <SLA+W> 0x0A 0x01 0x04 [*Program Flash* Command]<s>

The TWI Master may also send the above operations individually:

1. <SLA+W>0x06 0x00<s> — TWI master sends Address register command
2. <SLA+W>0x01 0x00<s> — Write Flash 16-bit address: LSB, then MSB
3. <SLA+W>0x01 0x00<s> — Write Flash length, 16-bit as a 16-bit value: LSB then MSB
4. <SLA+W> 0x0A 0x01 0x04<s> — TWI sends the Program Flash command

11.3.6 Program EEPROM (0x05)

This command is very similar to the Program Flash command, above. Program EEPROM will take the EEPROM data page, previously loaded into the data buffer, and program it into the ATxmega EEPROM memory at the location given in the address register and the number of bytes given in the length register.

In the example that follows, three bytes are programmed three to EEPROM address 0x0005.

The command consists of the following TWI operations:

1. <SLA+W>0x06 0x00 — TWI master sends Address register command
2. 0x05 0x00 — Write EEPROM as a 16-bit address: LSB, then MSB
3. 0x03 0x00 — Write EEPROM Write length, as 16-bit value: LSB then MSB
4. <SLA+W> 0x0A 0c01 0x05<s> — TWI sends the Program EEPROM command

The TWI sequence is as follows:

<SLA+W>0x06 0x00 [*Offset*] 0x05 0x00 [*Address*] 0x03 0x00 [*Length*] [s>

<SLA+W>0x0A 0x01 0x04<s>

The TWI Master may also send the above operations individually:

1. <SLA+W>0x06 0x00<s> TWI master sends Address register command
2. <SLA+W>0x05 0x00<s> Write EEPROM address
3. <SLA+W>0x03 0x00<s> Write EEPROM length
4. <SLA+W> 0x0A 0x01 0x04<s> TWI sends the Program EEPROM command

11.3.7 Calculate CRC for Flash (0x06)

This command will cause the cyclic redundancy check to be calculated for the flash up to and including the last page written.

In the below example it is assumed that the flash page is 256 bytes.

<SLA+W>0x0A 0x01 0x06<s>

<SLA+R>[2 Bytes read]<s>

Below is the routine implemented in C for calculating the CRC.

```
static uint16_t crc_ccitt_update(uint16_t crc, uint8_t data)
  {
      data ^= crc & 0xFF;
      data ^= data << 4;

    return ((((uint16_t)data << 8) | ((crc & 0xFF00) >> 8)) ^ \
            (uint8_t)(data >> 4) ^ \
            ((uint16_t)data << 3));
  }
```

11.3.8 Calculate CRC for EEPROM (0x07)

This command will cause the cyclic redundancy check to be calculated for the EEPROM from the byte address in the address register and then over the next N bytes – where N is set in the length register. In this example, it is assumed that the flash page is 256 bytes.

<SLA+W>0x0A 0x01 0x07<s>

<SLA+R>[2 Bytes read]<s>

The routine for calculating the CRC is the same as the one outlined for the flash memory.

12. Recommended Master-to-Slave transactions to Write ATxmega Flash

Figure 12.1 describes the sequence for the Master to issue a complete sequence for writing a page of bytes to the ATxmega's slave application code space. Notice in step 12 that this command writes a 0x00 into EEPROM. When the ATxmega is reset, exeution will commence from the Reset Vector at 0x0000, not the Bootloader starting address, which could be 0x2000, 0x4000 or other value.

Figure 12.1 Master-to-Salve transactions for a flash page write

Step #	Transaction	Description	Comment
1	Determine if Bootloader is present	<SLA+W>0x04 0x00 0xAA 0x55<s> <SLA+R>0x04 0x00<rs>0x55 0xAA<s>	Bytes read are reverse order of bytes writtten
2a	Determine the page size	<SLA+R> 0x00 0x00<s>	Page size command
2b1		<SLA+R>[2 Bytes read]<s>	a 16 bit value Typically 256 or 512
3a	Determine the number of pages	<SLA+R> 0x02 0x00<s>	No of Pages command
3b		<SLA+R>[2 Bytes read]<s>	a 16 bit value
4	Erase the Application flash	<SLA+W>0x0A **n** 0x03<s>	Erase the **n**th page (a hex value) 1 <= **n** <= **N**
5a	Send destination address in XMEGA flash	<SLA+W> 0x06 0x00	Address command
5b		Bits[7..0], Bits[15..8]	2 bytes sent to Slave
6	Send write length in XMEGA flash	<SLA+W> 0X08 0X00	Write length command
		Bits[7..0], Bits[15..8]	2 bytes sent to Slave
7	Fill the page buffer with application code	<SLA+W> 0X00 0X00	up to 256 continuous byte writes
		writes fill page_buffer auto increment[0..255]	
8	Execute Program Flash command	<SLA+W> 0x0A **N** 0x04<s>	Writes 256 bytes to XMEGA flash
9	Repeat Step 4-8 above until all flash pages are programmed	See above commands	Erase each page before programming
10	Execute Calculate Flash CRC command	<SLA+W>0x0A **N** 0x06<s>	XMEGA Slave calculates CRC
11	Execute Read Flash CRC command and read value	<SLA+R>0x0A **N** 0x06<s>	Read CRC Command
		CRC[7..0] CRC[15..8]	2 bytes used by Host
12	Jump to Application	<SLA+W>0x0A **N** 0x02<s>	Writes a 0x00 into EEPROM location 0x00 and begins execution of application code

Note: **N** is from Table 10.1

13. Conclusion

The Atmel AVR XMEGA family devices contain a Bootloader Section, either 4Kbytes or 8 Kbytes. This Application Note refers to an IAR C-based project which is an ATxmega Bootloader. This code may be modified by a user, or compiled as is and programmed into an ATxmega device via an Atmel or 3rd party programmer

Under control of the TWI Master, the Bootloader is able to erase and program that flash memory. EEPROM can also be erased and programmed.

As an IP protection feature, no method has been included in this code to read out the contents of flash or EEPROM memories

14. References

1. IAR EVAVR C compiler, version 5.51, the 4K or Eval versions.

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

Atmel Asia Limited
Unit 01-5 & 16, 19F
BEA Tower, Milennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
HONG KONG
Tel: (+852) 2245-6100
Fax: (+852) 2722-1369

Atmel Munich GmbH
Business Campus
Parkring 4
D-85748 Garching b. Munich
GERMANY
Tel: (+49) 89-31970-0
Fax: (+49) 89-3194621

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

8435A-AVR-09/11