

Features

- Protocol
 - UART Used as Physical Layer
 - Based on the Intel Hex-type Records
 - Autobaud
- In-System Programming
 - Read/Write Flash and EEPROM Memories
 - Read Device ID
 - Full-chip Erase
 - Read/Write Configuration Bytes
 - Security Setting From ISP Command
 - Remote Application Start Command
- In-Application Programming/Self Programming
 - Read/Write Flash and EEPROM Memories
 - Read Device ID
 - Block Erase
 - Read/Write Configuration Bytes
 - Bootloader Start

Description

This document describes the UART bootloader functionalities as well as the serial protocol to efficiently perform operations on the on chip Flash (EEPROM) memories. Additional information on the AT89C51AC3 product can be found in the AT89C51AC3 datasheet and the AT89C51AC3 errata sheet available on the Atmel web site.

The bootloader software package (source code and binary) currently used for production is available from the Atmel web site.

| Bootloader Revision | Purpose of Modifications | Date |
|---------------------|--------------------------|------------|
| Revision 1.0.1 | First release | 01/08/2003 |



**80C51
Microcontrollers**

**AT89C51AC3
UART
Bootloader**

Rev. 4386A-8051-10/04



Functional Description

The AT89C51AC3 Bootloader facilitates In-System Programming and In-Application Programming.

In-System Programming Capability

In-System Programming allows the user to program or reprogram a microcontroller on-chip Flash memory without removing it from the system and without the need of a pre-programmed application.

The UART bootloader can manage a communication with a host through the serial network. It can also access and perform requested operations on the on-chip Flash Memory.

In-Application Programming or Self-Programming Capability

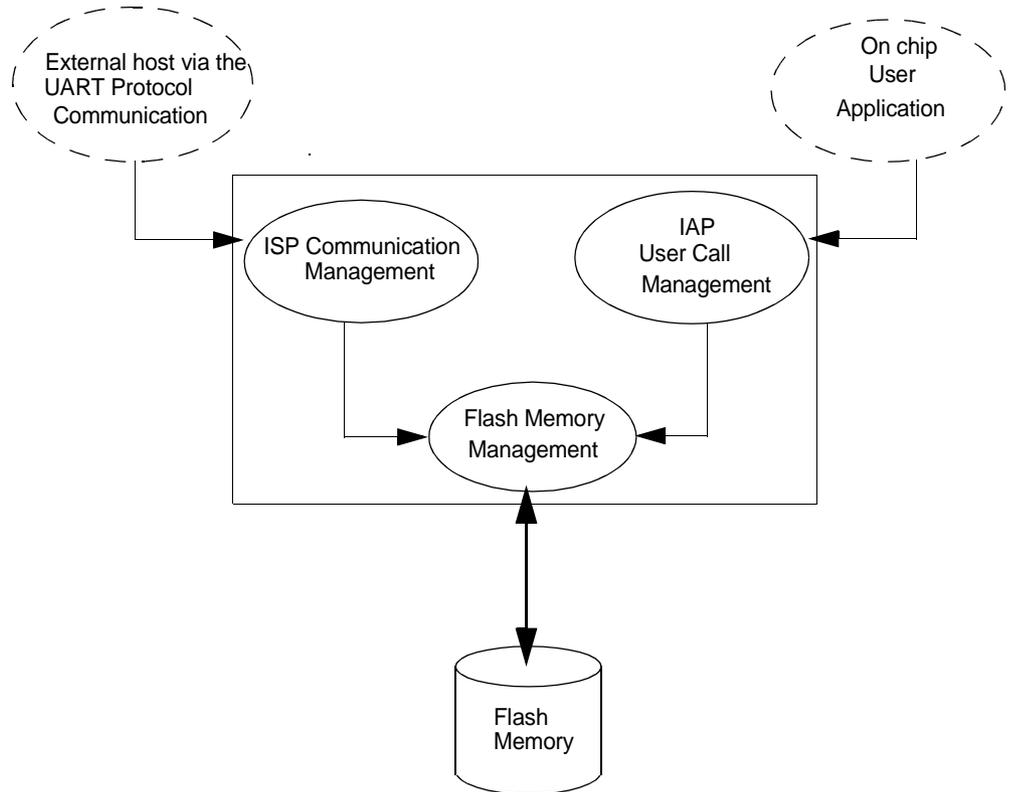
In-Application Programming (IAP) allows the reprogramming of a microcontroller on-chip Flash memory without removing it from the system and while the embedded application is running.

The UART bootloader contains some Application Programming Interface routines named API routines allowing IAP by using the user's firmware.

Block Diagram

This section describes the different parts of the bootloader. The figure below shows the on-chip bootloader and IAP processes.

Figure 1. Bootloader Process Description



ISP Communication Management

The purpose of this process is to manage the communication and its protocol between the on-chip bootloader and an external device (host). The on-chip bootloader implements a Serial protocol (see Section "Protocol"). This process translates serial communication frames (UART) into Flash memory accesses (read, write, erase...).

User Call Management

Several Application Program Interface (API) calls are available to the application program to selectively erase and program Flash pages. All calls are made through a common interface (API calls) included in the bootloader. The purpose of this process is to translate the application request into internal Flash Memory operations.

Flash Memory Management

This process manages low level accesses to the Flash memory (performs read and write accesses).

Bootloader Configuration

Configuration and Manufacturer Information

The table below lists Configuration and Manufacturer byte information used by the bootloader. This information can be accessed through a set of API or ISP commands.

| Mnemonic | Description | Default value |
|-----------------------|------------------------|---------------|
| BSB | Boot Status Byte | FFh |
| SBV | Software Boot Vector | FCh |
| SSB | Software Security Byte | FFh |
| EB | Extra Byte | FFh |
| Manufacturer | | 58h |
| Id1: Family code | | D7h |
| Id2: Product Name | | FFh |
| Id3: Product Revision | | FEh |

Mapping and Default Value of Hardware Security Byte

The 4 MSB of the Hardware Byte can be read/written by software (this area is called Fuse bits). The 4 LSB can only be read by software and written by hardware in parallel mode (with parallel programmer devices).

| Bit Position | Mnemonic | Default Value | Description |
|--------------|----------|---------------|---|
| 7 | X2B | U | To start in x1 mode |
| 6 | BLJB | P | To map the boot area in code area between F800h-FFFFh |
| 5 | reserved | U | |
| 4 | reserved | U | |
| 3 | reserved | U | |
| 2 | LB2 | P | To lock the chip (see datasheet) |
| 1 | LB1 | U | |
| 0 | LB0 | U | |

Note: U: Unprogram = 1
P: Program = 0

Security

The bootloader has Software Security Byte (SSB) to protect itself from user access or ISP access.

The Software Security Byte (SSB) protects from ISP accesses. The command "Program Software Security Bit" can only write a higher priority level. There are three levels of security:

- level 0: **NO_SECURITY** (FFh)
This is the default level.
From level 0, one can write level 1 or level 2.
- level 1: **WRITE_SECURITY** (FEh)
In this level it is impossible to write in the Flash memory, BSB and SBV.
The Bootloader returns an error message.
From level 1, one can write only level 2.
- level 2: **RD_WR_SECURITY** (FCh)
Level 2 forbids all read and write accesses to/from the Flash memory.
The Bootloader returns an error message.

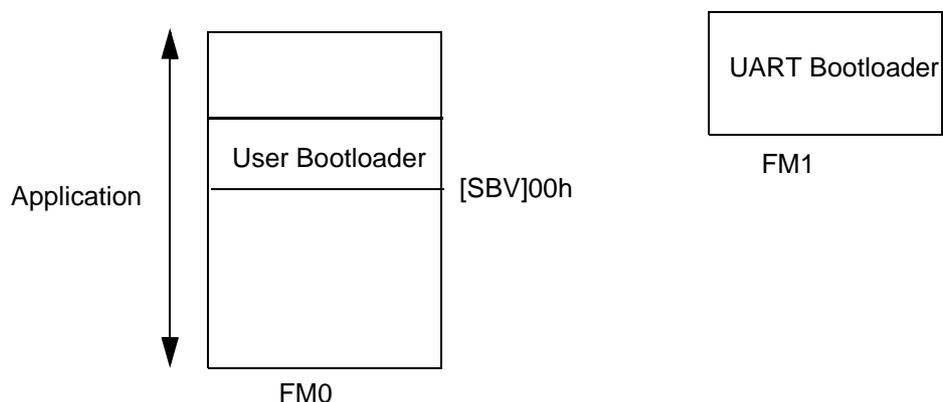
Only a full chip erase command can reset the software security bits.

| | Level 0 | Level 1 | Level 2 |
|-------------------|--------------------------|--------------------------|--------------------------|
| Flash/EEPROM | Any access allowed | Read only access allowed | All access not allowed |
| Fuse bit | Any access allowed | Read only access allowed | All access not allowed |
| BSB & SBV & EB | Any access allowed | Read only access allowed | All access not allowed |
| SSB | Any access allowed | Write level2 allowed | Read only access allowed |
| Manufacturer info | Read only access allowed | Read only access allowed | Read only access allowed |
| Bootloader info | Read only access allowed | Read only access allowed | Read only access allowed |
| Erase block | Allowed | Not allowed | Not allowed |
| Full chip erase | Allowed | Allowed | Allowed |
| Blank Check | Allowed | Allowed | Allowed |

Software Boot Vector

The Software Boot Vector (SBV) forces the execution of a user bootloader starting at address [SBV]00h in the application area (FM0).

The way to start this user bootloader is described in the section "Boot Process".



FLIP Software Program

FLIP is a PC software program running under Windows 9x / NT / 2K / XP and LINUX that supports all Atmel C51 Flash microcontrollers.

This free software program is available on the Atmel web site.

In-System Programming

The ISP allows the user to program or reprogram a microcontroller's on-chip Flash memory through the serial line without removing it from the system and without the need of a pre-programmed application.

This section describes how to start the UART bootloader and the higher level protocol over the serial line.

Boot Process

The bootloader can be activated in two ways:

- Hardware conditions
- Regular boot process

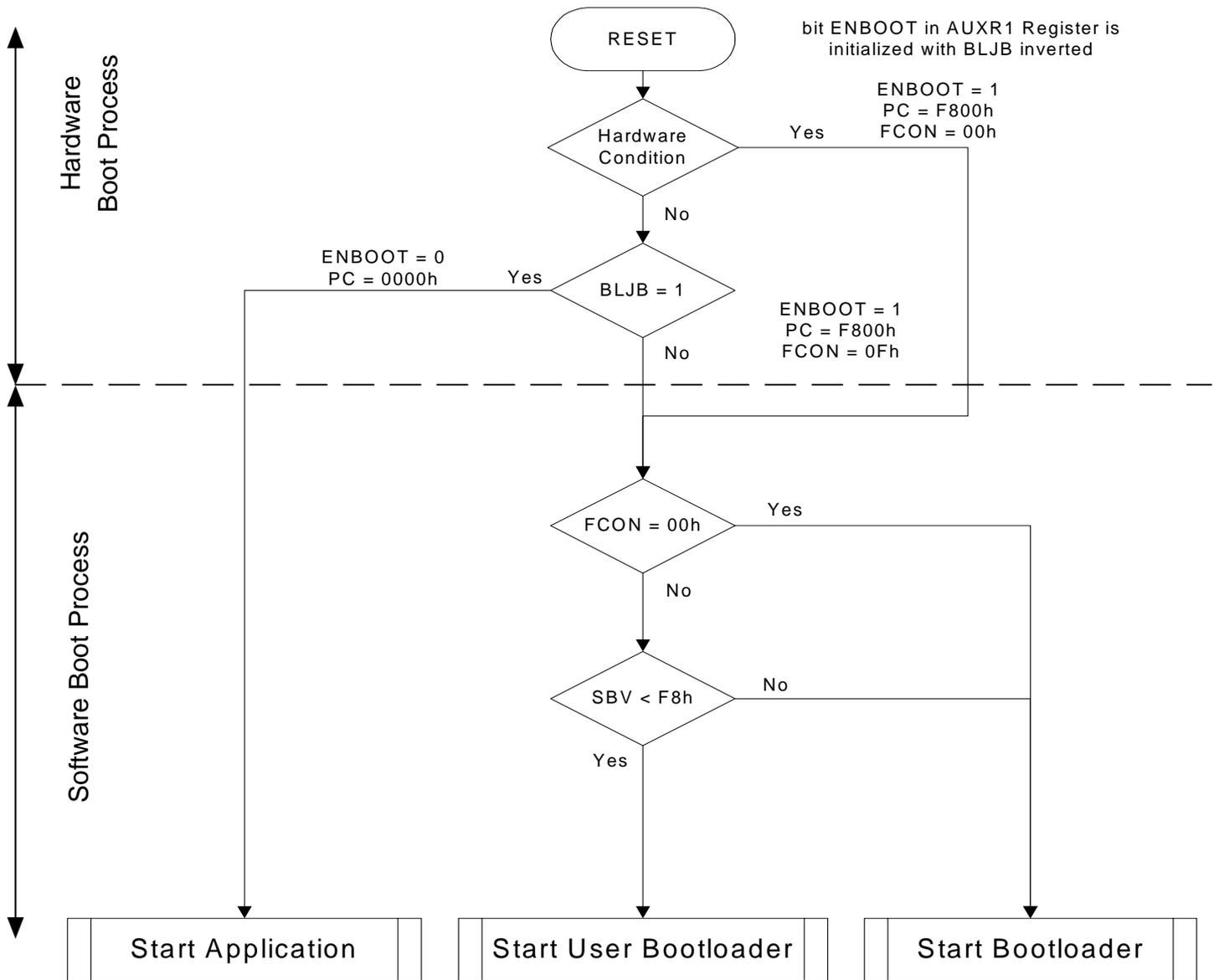
Hardware Condition

The Hardware conditions (EA = 1, PSEN = 0) during the RESET# falling edge force the on-chip bootloader execution. In this way the bootloader can be carried out whatever the user Flash memory content.

As PSEN is an output port in normal operating mode (running user application or bootloader code) after reset, it is recommended to release PSEN after falling edge of reset signal. The hardware conditions are sampled at reset signal falling edge, thus they can be released at any time when reset input is low.

To ensure correct microcontroller startup. The PSEN pin should not be tied to ground during power-on.

Regular Boot Process



Physical Layer

The UART used to transmit information has the following configuration:

- Character: 8-bit data
- Parity: none
- Stop: 2 bits
- Flow control: none
- Baud rate: autobaud is performed by the bootloader to compute the baud rate chosen by the host.

Frame Description

The Serial Protocol is based on the Intel Hex-type records.

Intel Hex records consist of ASCII characters used to represent hexadecimal values and are summarized below.

Table 1. Intel Hex Type Frame

| Record Mark ':' | Record length | Load Offset | Record Type | Data or Info | Checksum |
|-----------------|---------------|-------------|-------------|--------------|----------|
| 1 byte | 1 byte | 2 bytes | 1 bytes | n byte | 1 byte |

- Record Mark:
 - Record Mark is the start of frame. This field must contain ':'.
- Record length:
 - Record length specifies the number of Bytes of information or data which follows the Record Type field of the record.
- Load Offset:
 - Load Offset specifies the 16-bit starting load offset of the data Bytes, therefore this field is used only for
 - Data Program Record.
- Record Type:
 - Record Type specifies the command type. This field is used to interpret the remaining information within the frame.
- Data/Info:
 - Data/Info is a variable length field. It consists of zero or more Bytes encoded as pairs of hexadecimal digits. The meaning of data depends on the Record Type.
- Checksum:
 - The two's complement of the 8-bit Bytes that result from converting each pair of ASCII hexadecimal digits to one Byte of binary, and including the Record Length field to and including the last Byte of the Data/Info field. Therefore, the sum of all the ASCII pairs in a record after converting to binary, from the Record Length field to and including the Checksum field, is zero.

Protocol

Overview

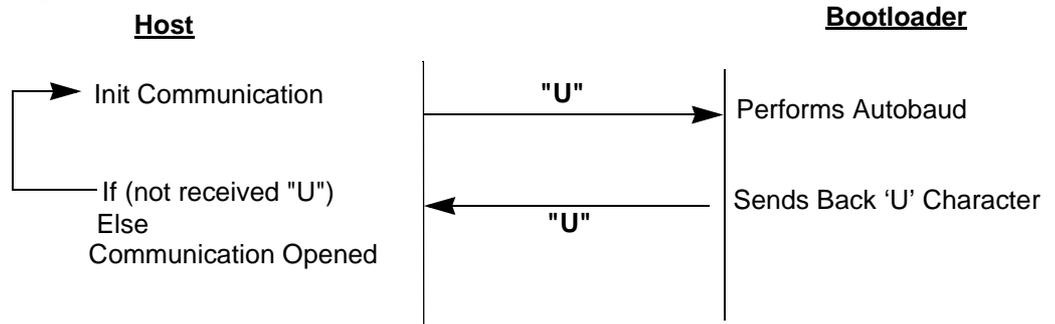
An initialization step must be performed after each Reset. After microcontroller reset, the bootloader waits for an autobaud sequence (see Section “Autobaud Performances”).

When the communication is initialized the protocol depends on the record type issued by the host.

Communication Initialization

The host initiates the communication by sending a 'U' character to help the bootloader to compute the baud rate (autobaud).

Figure 2. Initialization



Autobaud Performances

The bootloader supports a wide range of baud rates. It is also adaptable to a wide range of oscillator frequencies. This is accomplished by measuring the bit-time of a single bit in a received character. This information is then used to program the baud rate in terms of timer counts based on the oscillator frequency. Table 2 shows the autobaud capabilities.

Table 2. Autobaud Performances

| Frequency (MHz) Baudrate (kHz) | 1.8432 | 2 | 2.4576 | 3 | 3.6864 | 4 | 5 | 6 | 7.3728 |
|-----------------------------------|--------|----|--------|----|--------|----|----|----|--------|
| 2400 | OK | OK | OK | OK | OK | OK | OK | OK | OK |
| 4800 | OK | - | OK | OK | OK | OK | OK | OK | OK |
| 9600 | OK | - | OK | OK | OK | OK | OK | OK | OK |
| 19200 | OK | - | OK | OK | OK | - | - | OK | OK |
| 38400 | - | - | OK | | OK | - | OK | OK | OK |
| 57600 | - | - | - | - | OK | - | - | - | OK |
| 115200 | - | - | - | - | - | - | - | - | OK |

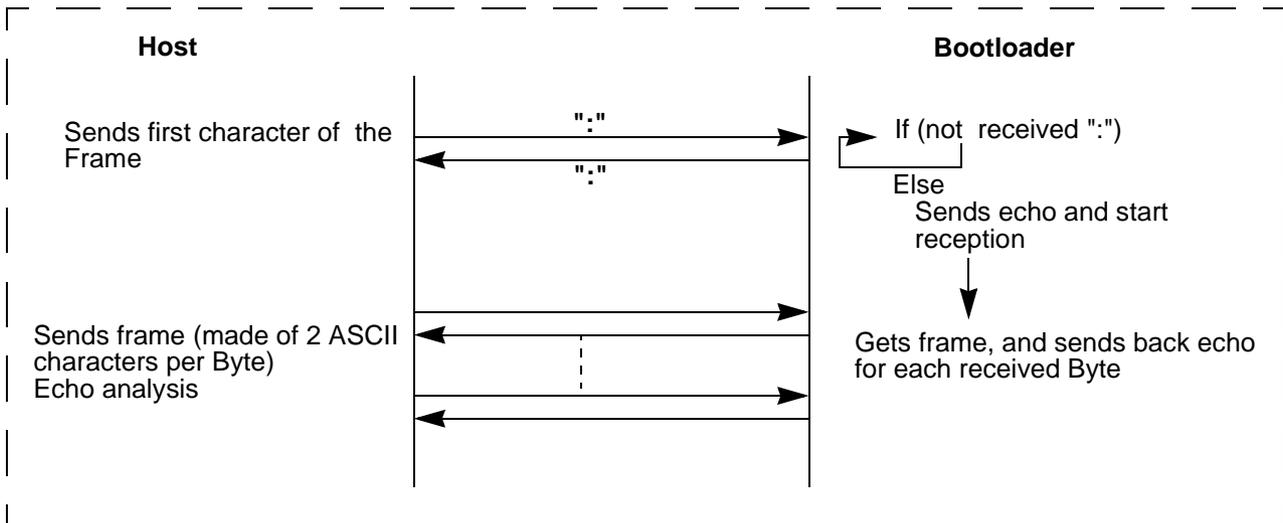
| Frequency (MHz) Baudrate (kHz) | 8 | 10 | 11.0592 | 12 | 14.746 | 16 | 20 | 24 | 26.6 |
|-----------------------------------|----|----|---------|----|--------|----|----|----|------|
| 2400 | OK | OK | OK | OK | OK | OK | OK | OK | OK |
| 4800 | OK | OK | OK | OK | OK | OK | OK | OK | OK |
| 9600 | OK | OK | OK | OK | OK | OK | OK | OK | OK |

| Frequency (MHz) Baudrate (kHz) | 8 | 10 | 11.0592 | 12 | 14.746 | 16 | 20 | 24 | 26.6 |
|-----------------------------------|----|----|---------|----|--------|----|----|----|------|
| 19200 | OK | OK | OK | OK | OK | OK | OK | OK | OK |
| 38400 | - | - | OK | OK | OK | OK | OK | OK | OK |
| 57600 | - | - | OK | - | OK | OK | OK | OK | OK |
| 115200 | - | - | OK | - | OK | - | - | - | - |

Command Data Stream Protocol

All commands are sent using the same flow. Each frame sent by the host is echoed by the bootloader.

Figure 3. Command Flow



Programming the Flash or EEPROM data

The flow described below shows how to program data in the Flash memory or in the EEPROM data memory.

The bootloader programs on a page of 128 bytes basis when it is possible.

The host must take care that:

- The data to program transmitted within a frame are in the same page.

Requests from Host

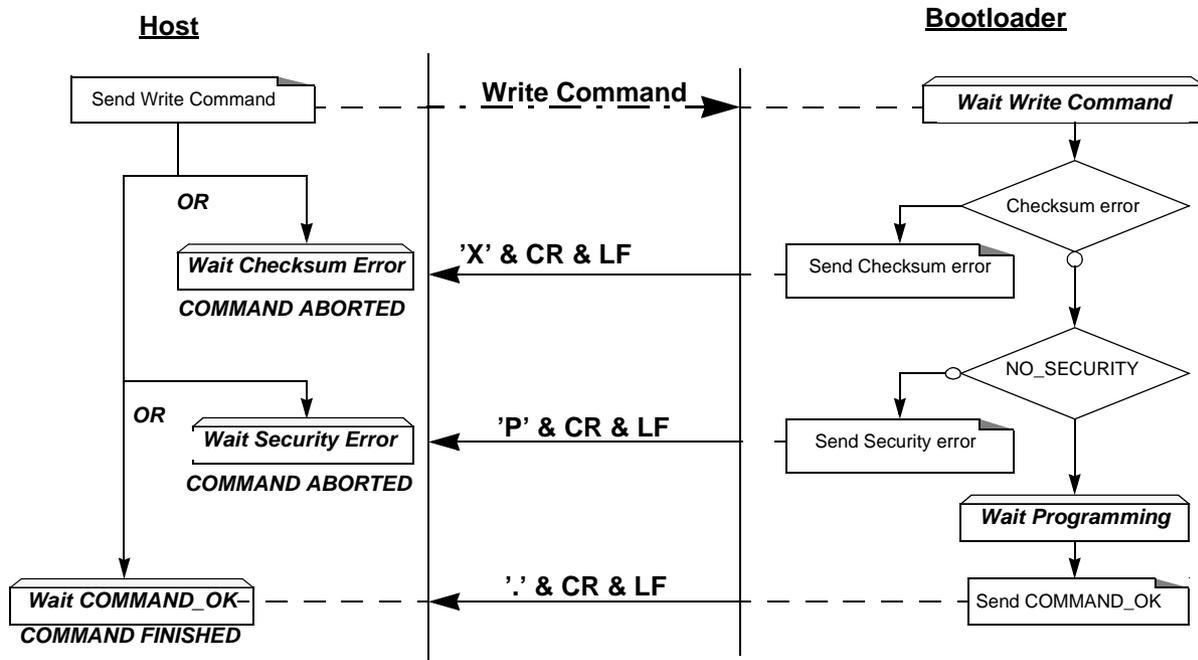
| Command Name | Record Type | Load Offset | Record Length | Data[0] | ... | Data[127] |
|---------------------|-------------|---------------|---------------|---------|-----|-----------|
| Program Flash | 00h | start address | nb of Data | x | ... | x |
| Program EEPROM Data | 07h | start address | nb of Data | x | ... | x |

Answers from Bootloader

The bootloader answers with:

- '.' & 'CR' & 'LF' when the data are programmed
- 'X' & 'CR' & 'LF' if the checksum is wrong
- 'P' & 'CR' & 'LF' if the Security is set

Flow Description



Example

Programming Data (write 55h at address 0010h in the Flash)

```

HOST          : 01 0010 00 55 9A
BOOTLOADER    : 01 0010 00 55 9A . CR LF
    
```

Read the Flash or EEPROM Data

The flow described below allows the user to read data in the Flash memory or in the EEPROM data memory. A blank check command is possible with this flow.

The device splits into blocks of 16 bytes the data to transfer to the Host if the number of data to display is greater than 16 data bytes.

Requests from Host

| Command Name | Record Type | Load Offset | Record Length | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] |
|----------------------|-------------|-------------|---------------|---------------|---------|---------|---------|---------|
| Read Flash | 04h | x | 05h | start address | | | | 00h |
| Blank check on Flash | | | | | | | | 01h |
| Read EEPROM Data | | | | | | | | 02h |

Note: The field "Load offset" is not used.

Answers from Bootloader

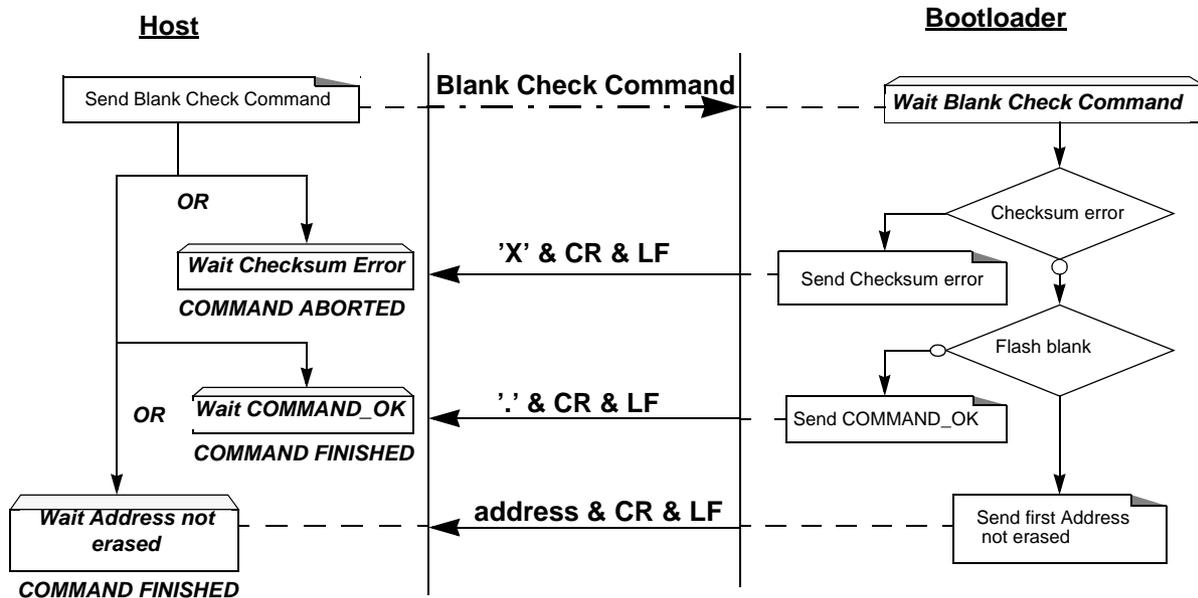
The bootloader answers to a read Flash or EEPROM Data memory command:

- 'Address = data ' & 'CR' & 'LF' up to 16 data by line.
- 'X' & 'CR' & 'LF' if the checksum is wrong
- 'L' & 'CR' & 'LF' if the Security is set

The bootloader answers to blank check command:

- '.' & 'CR' & 'LF' when the blank check is ok
- 'First Address wrong' & 'CR' & 'LF' when the blank check is fail
- 'X' & 'CR' & 'LF' if the checksum is wrong
- 'P' & 'CR' & 'LF' if the Security is set

Flow Description: Blank Check Command



Example

Blank Check ok

```

HOST      : 05 0000 04 0000 7FFF 01 78
BOOTLOADER : 05 0000 04 0000 7FFF 01 78 . CR LF
    
```

Blank Check ko at address xxxx

```

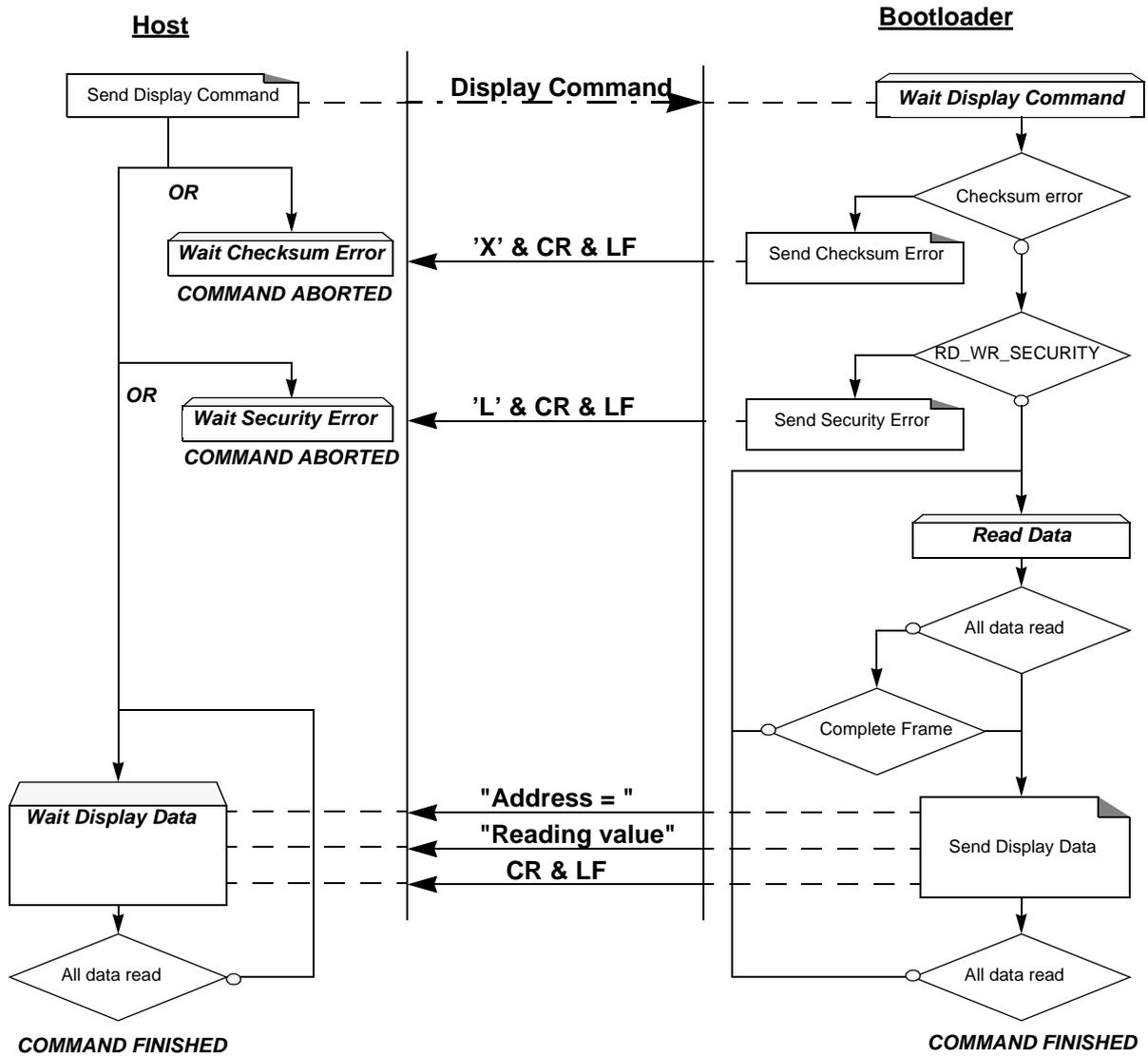
HOST      : 05 0000 04 0000 7FFF 01 78
BOOTLOADER : 05 0000 04 0000 7FFF 01 78 xxxx CR LF
    
```

Blank Check with checksum error

```

HOST      : 05 0000 04 0000 7FFF 01 70
BOOTLOADER : 05 0000 04 0000 7FFF 01 70 X CR LF CR LF
    
```

Flow Description: Read Command



Note: The maximum size of block is 400h. To read more than 400h Bytes, the Host must send a new command.

Example

Display data from address 0000h to 0020h

```

HOST          : 05 0000 04 0000 0020 00 D7
BOOTLOADER    : 05 0000 04 0000 0020 00 D7
BOOTLOADER    0000=-----data----- CR LF   (16 data)
BOOTLOADER    0010=-----data----- CR LF   (16 data)
BOOTLOADER    0020=data CR LF                   ( 1 data)
    
```

Program Configuration Information

The flow described below allows the user to program Configuration Information regarding the bootloader functionality.

The Boot Process Configuration:

BSB

SBV

Fuse bits (BLJB and X2 bits) (see Section “Mapping and Default Value of Hardware Security Byte”)

SSB

EB

Requests from Host

| Command Name | Record Type | Load Offset | Record Length | Data[0] | Data[1] | Data[2] |
|--------------------|-------------|-------------|---------------|---------|---------|-----------|
| Erase SBV & BSB | 03h | x | 02h | 04h | 00h | - |
| Program SSB level1 | | | 02h | 05h | 00h | - |
| Program SSB level2 | | | | | 01h | - |
| Program BSB | | | 03h | 06h | 00h | value |
| Program SBV | | | | | 01h | |
| Program EB | | | | | 06h | |
| Program bit BLJB | | | 03h | 0Ah | 04h | bit value |
| Program bit X2 | | | | | 08h | |

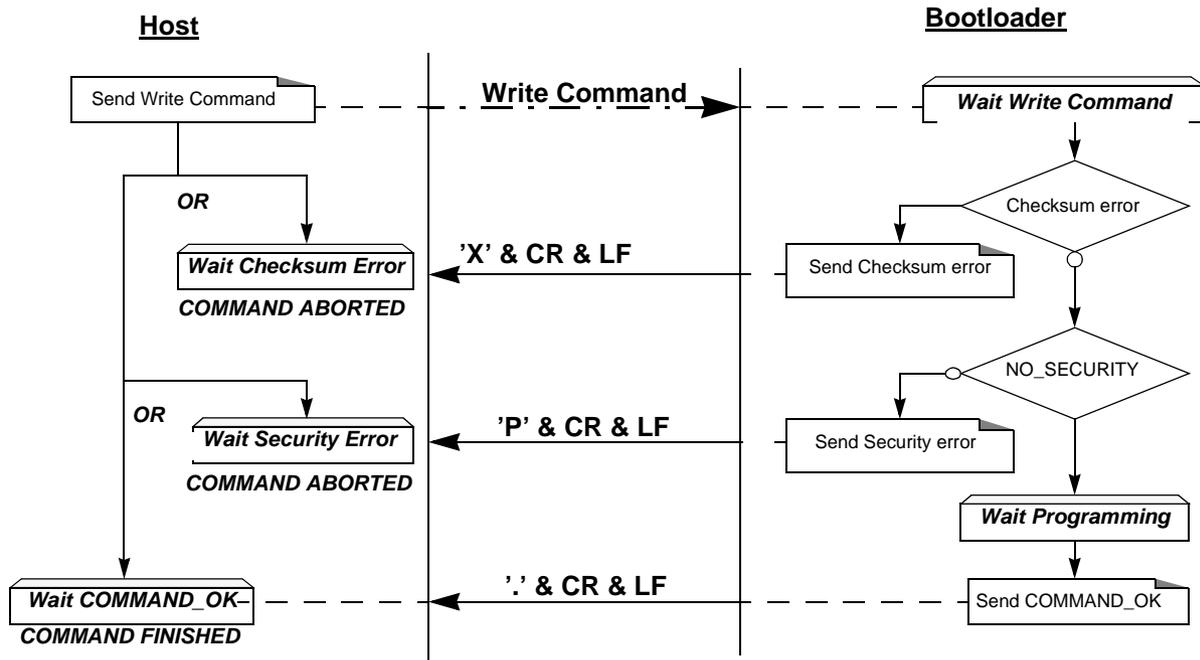
- Note:
1. The field “Load Offset” is not used
 2. To program the BLJB and X2 bit the “bit value” is 00h or 01h.

Answers from Bootloader

The bootloader answers with:

- ‘.’ & ‘CR’ & ‘LF’ when the value is programmed
- ‘X’ & ‘CR’ & ‘LF’ if the checksum is wrong
- ‘P’ & ‘CR’ & ‘LF’ if the Security is set

Flow Description



Example

Programming Atmel function (write SSB to level 2)

```

HOST          : 02 0000 03 05 01 F5
BOOTLOADER    : 02 0000 03 05 01 F5 . CR LF
    
```

Writing Frame (write BSB to 55h)

```

HOST          : 03 0000 03 06 00 55 9F
BOOTLOADER    : 03 0000 03 06 00 55 9F . CR LF
    
```

Read Configuration Information or Manufacturer Information

The flow described below allows the user to read the configuration or manufacturer information.

Requests from Host

| Command Name | Record Type | Load Offset | Record Length | Data[0] | Data[1] |
|-------------------------|-------------|-------------|---------------|---------|---------|
| Read Manufacturer Code | 05h | x | 02h | 00h | 00h |
| Read Family Code | | | | | 01h |
| Read Product Name | | | | | 02h |
| Read Product Revision | | | | | 03h |
| Read SSB | | | | 07h | 00h |
| Read BSB | | | | | 01h |
| Read SBV | | | | | 02h |
| Read EB | | | | | 06h |
| Read HSB (Fuse bit) | | | | 0Bh | 00h |
| Read Device ID1 | | | | 0Eh | 00h |
| Read Device ID2 | | | | | 01h |
| Read Bootloader version | | | | 0Fh | 00h |

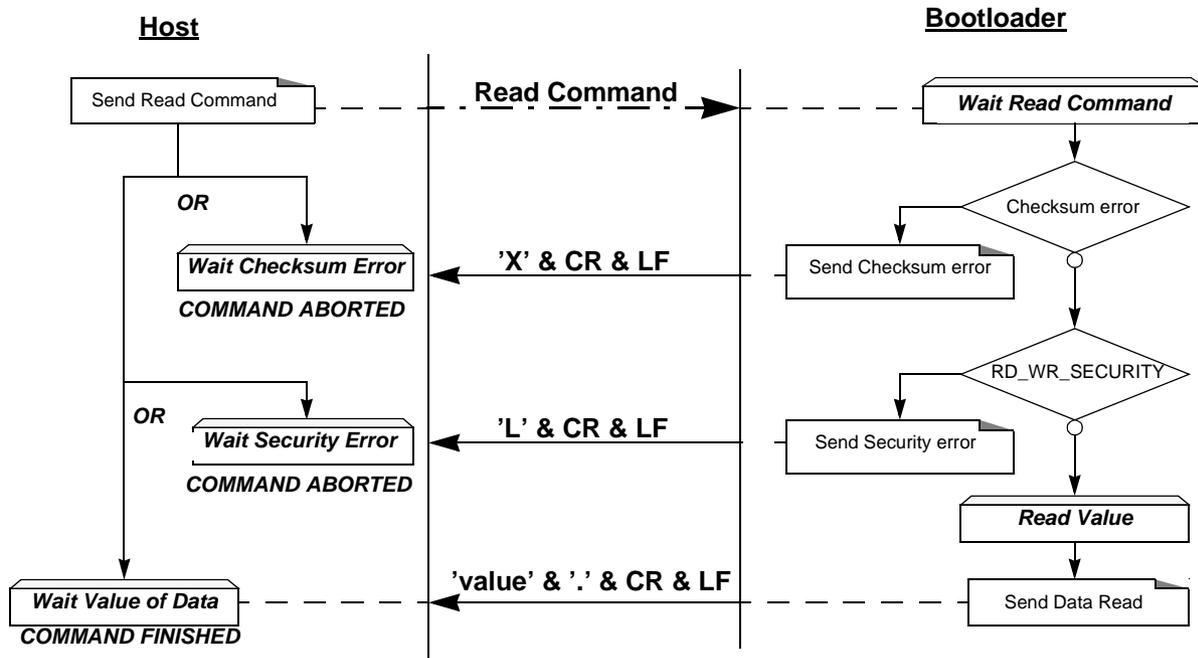
Note: The field "Load Offset" is not used

Answers from Bootloader

The bootloader answers with:

- 'value' & '.' & 'CR' & 'LF' when the value is programmed
- 'X' & 'CR' & 'LF' if the checksum is wrong
- 'P' & 'CR' & 'LF' if the Security is set

Flow Description



Example

Read function (read SBV)

```
HOST          : 02 0000 05 07 02 F0
BOOTLOADER    : 02 0000 05 07 02 F0 Value . CR LF
```

Atmel Read function (read Bootloader version)

```
HOST          : 02 0000 01 02 00 FB
BOOTLOADER    : 02 0000 01 02 00 FB Value . CR LF
```

Erase the Flash

The flow described below allows the user to erase the Flash memory.

Two modes of Flash erasing are possible:

- Full Chip erase
- Block erase

The Full Chip erase command erases the whole Flash (64 Kbytes) and sets some Configuration Bytes at their default values:

- BSB = FFh
- SBV = FCh
- SSB = FFh (NO_SECURITY)

The full chip erase is always executed whatever the Software Security Byte value is.

The Block erase command erases only a part of the Flash.

Three Blocks are defined in the AT89C51AC3:

- block0 (From 0000h to 1FFFh)
- block1 (From 2000h to 3FFFh)
- block2 (From 4000h to 7FFFh)
- block3 (From 8000h to BFFFh)
- block4 (From C000h to FFFFh)

Requests from Host

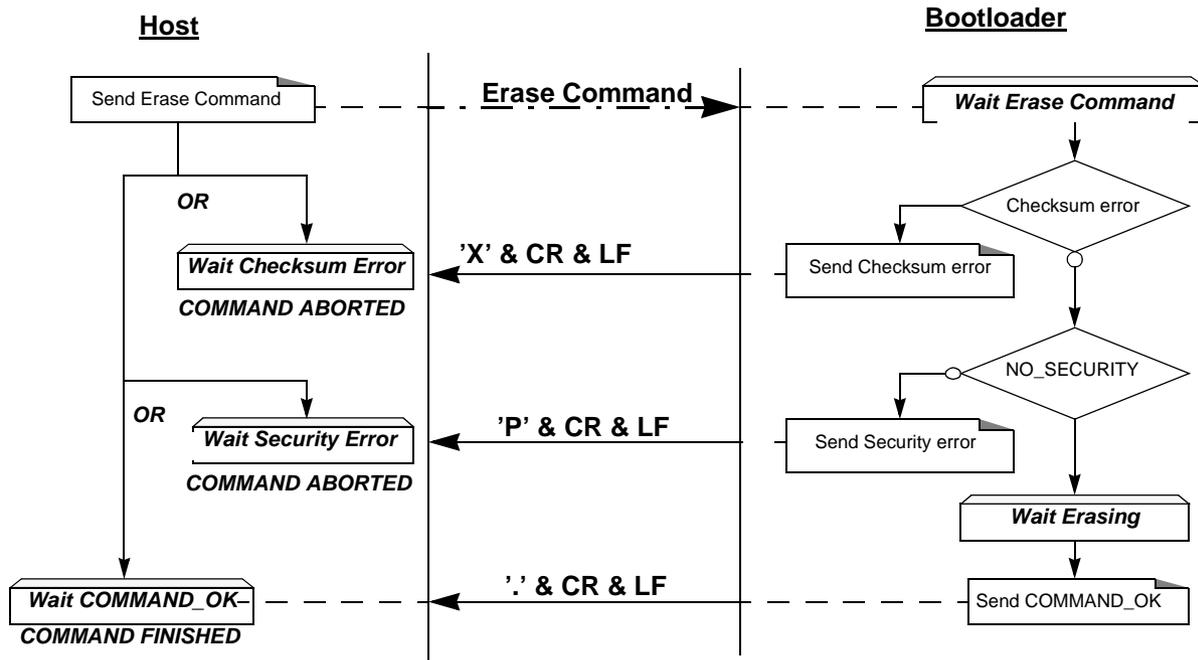
| Command Name | Record Type | Load Offset | Record Length | Data[0] | Data[1] |
|---------------------------|-------------|-------------|---------------|---------|---------|
| Erase block0 (0k to 8k) | 03h | x | 02h | 01h | 00h |
| Erase block1 (8k to 16k) | | | | | 20h |
| Erase block2 (16k to 32k) | | | | | 40h |
| Erase block3 (32k to 48k) | | | | | 80h |
| Erase block4 (48k to 64k) | | | | | C0h |
| Full chip erase | | | 01h | 07h | - |

Answers from Bootloader

As the Program Configuration Information flows, the erase block command has three possible answers:

- '.' & 'CR' & 'LF' when the data are programmed
- 'X' & 'CR' & 'LF' if the checksum is wrong
- 'P' & 'CR' & 'LF' if the Security is set

Flow Description



Example

Full Chip Erase

```

HOST          : 01 0000 03 07 F5
BOOTLOADER    : 01 0000 03 07 F5 . CR LF
    
```

Erase Block1(8k to 16k)

```

HOST          : 02 0000 03 01 20 DA
BOOTLOADER    : 02 0000 03 01 20 DA . CR LF
    
```

Start the Application

The flow described below allows to start the application directly from the bootloader upon a specific command reception.

Two options are possible:

- Start the application with a reset pulse generation (using watchdog).
When the device receives this command the watchdog is enabled and the bootloader enters a waiting loop until the watchdog resets the device.
Take care that if an external reset chip is used the reset pulse in output may be wrong and in this case the reset sequence is not correctly executed.
- Start the application without reset
A jump at the address 0000h is used to start the application without reset.

Requests from Host

| Command Name | Record type | Load Offset | Record Length | Data[0] | Data[1] | Data[2] | Data[3] |
|---|-------------|-------------|---------------|---------|---------|---------|---------|
| Start application with a reset pulse generation | 03h | x | 02h | 03h | 00h | - | - |
| Start application with a jump at "address" | | | 04h | | 01h | Address | |

Answer from Bootloader

No answer is returned by the device.

Example

Start Application with reset pulse

HOST : 02 0000 03 03 00 F8

BOOTLOADER : 02 0000 03 03 00 F8

Start Application without reset at address 0000h

HOST : 04 0000 03 03 01 00 00 F5

BOOTLOADER : 04 0000 03 03 01 00 00 F5

In-Application Programming/Self-Programming

The IAP allows to reprogram a microcontroller on-chip Flash memory without removing it from the system and while the embedded application is running.

The user application can call some Application Programming Interface (API) routines allowing IAP. These API are executed by the bootloader.

To call the corresponding API, the user must use a set of Flash_api routines which can be linked with the application.

Example of Flash_api routines are available on the Atmel web site on the software package:

C Flash Drivers for the AT89C51AC3

The Flash_api routines on the package work only with the UART bootloader.

The Flash_api routines are listed in Appendix-2.

API Call

Process

The application selects an API by setting R1, ACC, DPTR0 and DPTR1 registers.

All calls are made through a common interface "USER_CALL" at the address FFF0h.

The jump at the USER_CALL must be done by LCALL instruction to be able to come-back in the application.

Before jump at the USER_CALL, the bit ENBOOT in AUXR1 register must be set.

Constraints

The interrupts are not disabled by the bootloader.

Interrupts must be disabled by user prior to jump to the USER_CALL, then re-enabled when returning.

Interrupts must also be disabled before accessing EEPROM Data then re-enabled after.

The user must take care of hardware watchdog before launching a Flash operation.

For more information regarding the Flash writing time see the AT89C51AC3 datasheet.

API Commands

Several types of APIs are available:

- Read/Program Flash and EEPROM Data memory
- Read Configuration and Manufacturer Information
- Program Configuration Information
- Erase Flash
- Start bootloader

Read/Program Flash and EEPROM Data Memory

All routines to access EEPROM Data are managed directly from the application without using bootloader resources.

To read the Flash memory the bootloader is not involved.

For more details on these routines see the AT89C51AC3 datasheet sections “Program/Code Memory” and “EEPROM Data Memory”

Two routines are available to program the Flash:

- __api_wr_code_byte
- __api_wr_code_page

- The application program load the column latches of the Flash then call the __api_wr_code_byte or __api_wr_code_page see datasheet in section “Program/Code Memory”.
- Parameter settings

| API_name | R1 | DPTR0 | DPTR1 | Acc |
|--------------------|-----|--|--|---------------------------|
| __api_wr_code_byte | 02h | Address in Flash memory to write | - | Value to write |
| __api_wr_code_page | 09h | Address of the first Byte to program in the Flash memory | Address in XRAM of the first data to program | Number of Byte to program |

- instruction: LCALL FFF0h.

Note: No special resources are used by the bootloader during this operation

Read Configuration and Manufacturer Information

- Parameter settings

| API_name | R1 | DPTR0 | DPTR1 | Acc |
|-----------------------|-----|-------|-------|------------------------|
| __api_rd_HSB | 0Bh | 0000h | x | return HSB |
| __api_rd_BSB | 07h | 0001h | x | return BSB |
| __api_rd_SBV | 07h | 0002h | x | return SBV |
| __api_rd_SSB | 07h | 0000h | x | return SSB |
| __api_rd_EB | 07h | 0006h | x | return EB |
| __api_rd_manufacturer | 00h | 0000h | x | return manufacturer id |
| __api_rd_device_id1 | 00h | 0001h | x | return id1 |

| API_name | R1 | DPTR0 | DPTR1 | Acc |
|-----------------------------|-----|-------|-------|--------------|
| __api_rd_device_id2 | 00h | 0002h | x | return id2 |
| __api_rd_device_id3 | 00h | 0003h | x | return id3 |
| __api_rd_bootloader_version | 0Fh | 0000h | x | return value |

- Instruction: LCALL FFF0h.
- At the complete API execution by the bootloader, the value to read is in the api_value variable.

Note: No special resources are used by the bootloader during this operation

Program Configuration Information

- Parameter settings

| API Name | R1 | DPTR0 | DPTR1 | Acc |
|---------------------|-----|-------|-------|----------------|
| __api_set_X2 | 0Ah | 0008h | x | 00h |
| __api_clr_X2 | 0Ah | 0008h | x | 01h |
| __api_set_BLJB | 0Ah | 0004h | x | 00h |
| __api_clr_BLJB | 0Ah | 0004h | x | 01h |
| __api_wr_BSB | 06h | 0000h | x | value to write |
| __api_wr_SBV | 06h | 0001h | x | value to write |
| __api_wr_EB | 06h | 0006h | x | value to write |
| __api_wr_SSB_LEVEL0 | 05h | FFh | x | x |
| __api_wr_SSB_LEVEL1 | 05h | FEh | x | x |
| __api_wr_SSB_LEVEL2 | 05h | FC | x | x |

- instruction: LCALL FFF0h.

Note: 1. See in the AT89C51AC3 datasheet the time that a write operation takes.
2. No special resources are used by the bootloader during these operations

Erase Flash

The AT89C51AC3 flash memory is divided in several blocks:

Block 0: from address 0000h to 1FFFh

Block 1: from address 2000h to 3FFFh

Block 2: from address 4000h to 7FFFh

These three blocks contain 128 pages.

- Parameter settings

| API name | R1 | DPTR0 | DPTR1 | Acc |
|--------------------|-----|-------|-------|-----|
| __api_erase_block0 | 01h | 0000h | x | x |
| __api_erase_block1 | | 2000h | x | x |
| __api_erase_block2 | | 4000h | x | x |
| __api_erase_block3 | | 8000h | x | x |
| __api_erase_block4 | | C000h | x | x |

- instruction: LCALL FFF0h.

Note:

- See the AT89C51AC3 datasheet for the time that a write operation takes and this time must multiply by the number of pages.
- No special resources are used by the bootloader during these operations

Start Bootloader

This routine allows to start at the beginning of the bootloader as after a reset. After calling this routine the regular boot process is performed and the communication must be opened before any action.

- No special parameter setting
- Set bit ENBOOT in AUXR1 register
- instruction: LJUMP or LCALL at address F800h

APPENDIX-A

Table 3. Summary of Frames from Host

| Command | Record Type | Record Length | Offset | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | | | |
|---|-------------|------------------------|---------------|---------|------------------------|---------------|-------------|---------|-----|---|---|
| Program Nb Data Byte in Flash. | 00h | nb of data (up to 80h) | start address | x | x | x | x | x | | | |
| Erase block0 (0000h-1FFFh) | 03h | 02h | x | 01h | 00h | - | - | - | | | |
| Erase block1 (2000h-3FFFh) | | | | | 20h | - | - | - | | | |
| Erase block2 (4000h-7FFFh) | | | | | 40h | - | - | - | | | |
| Erase block3 (8000h-BFFFh) | | | | | 80h | - | - | - | | | |
| Erase block4 (C000h-FFFFh) | | | | | C0h | - | - | - | | | |
| Start application with a reset pulse generation | 02h | x | 03h | 00h | - | - | - | | | | |
| Start application with a jump at "address" | 04h | x | | 01h | address | | - | | | | |
| Erase SBV & BSB | 03h | 02h | x | 04h | 00h | - | - | - | | | |
| Program SSB level 1 | | | x | 05h | 00h | - | - | - | | | |
| Program SSB level 2 | | | x | | 01h | - | - | - | | | |
| Program BSB | | 03h | x | 06h | 00h | value | - | - | | | |
| Program SBV | | | x | | 01h | value | - | - | | | |
| Program EB | | | x | | 06h | value | - | - | | | |
| Full Chip Erase | | 01h | x | 07h | - | - | - | - | | | |
| Program bit BLJB | | 03h | x | 0Ah | 04h | bit value | - | - | | | |
| Program bit X2 | | | | | 08h | bit value | - | - | | | |
| Read Flash | | 04h | 05h | x | Start Address | | End Address | | 00h | | |
| Blank Check | | | | | | | | | 01h | | |
| Read EEPROM Data | | | | | | | | | 02h | | |
| Read Manufacturer Code | | 05h | 02h | x | 00h | 00h | - | - | - | | |
| Read Family Code | | | | | | 01h | - | - | - | | |
| Read Product Name | 02h | | | | | - | - | - | | | |
| Read Product Revision | 03h | | | | | - | - | - | | | |
| Read SSB | 07h | | | | 00h | - | - | - | | | |
| Read BSB | | | | | 01h | - | - | - | | | |
| Read SBV | | | | | 02h | - | - | - | | | |
| Read EB | | | | | 06h | - | - | - | | | |
| Read Hardware Byte | 0Bh | | | | 00h | - | - | - | | | |
| Read Device Boot ID1 | 0Eh | | | | 00h | - | - | - | | | |
| Read Device Boot ID2 | | | | | 01h | - | - | - | | | |
| Read Bootloader Version | 0Fh | | | | 00h | - | - | - | | | |
| Program Nb Data byte in EEPROM | 00h | | | | nb of data (up to 80h) | start address | x | x | x | x | x |

APPENDIX-B

Table 4. API Summary

| Function Name | Bootloader Execution | R1 | DPTR0 | DPTR1 | Acc |
|-----------------------------|----------------------|-----|--|--|---------------------------|
| __api_rd_code_byte | no | | | | |
| __api_wr_code_byte | yes | 02h | Address in Flash memory to write | - | Value to write |
| __api_wr_code_page | yes | 09h | Address of the first Byte to program in the Flash memory | Address in XRAM of the first data to program | Number of Byte to program |
| __api_erase_block0 | yes | 01h | 0000h | x | x |
| __api_erase_block1 | yes | 01h | 2000h | x | x |
| __api_erase_block2 | yes | 01h | 4000h | x | x |
| __api_erase_block3 | yes | 01h | 8000h | x | x |
| __api_erase_block4 | yes | 01h | C000h | x | x |
| __api_rd_HSB | yes | 0Bh | 0000h | x | return value |
| __api_set_X2 | yes | 0Ah | 0008h | x | 00h |
| __api_clr_X2 | yes | 0Ah | 0008h | x | 01h |
| __api_set_BLJB | yes | 0Ah | 0004h | x | 00h |
| __api_clr_BLJB | yes | 0Ah | 0004h | x | 01h |
| __api_rd_BSB | yes | 07h | 0001h | x | return value |
| __api_wr_BSB | yes | 06h | 0000h | x | value |
| __api_rd_SBV | yes | 07h | 0002h | x | return value |
| __api_wr_SBV | yes | 06h | 0001h | x | value |
| __api_erase_SBV | yes | 06h | 0001h | x | FCh |
| __api_rd_SSB | yes | 07h | 0000h | x | return value |
| __api_wr_SSB_level0 | yes | 05h | 00FFh | x | x |
| __api_wr_SSB_level1 | yes | 05h | 00FEh | x | x |
| __api_wr_SSB_level2 | yes | 05h | 00FCh | x | x |
| __api_rd_EB | yes | 07h | 0006h | x | return value |
| __api_wr_EB | yes | 06h | 0006h | x | value |
| __api_rd_manufacturer | yes | 00h | 0000h | x | return value |
| __api_rd_device_id1 | yes | 00h | 0001h | x | return value |
| __api_rd_device_id2 | yes | 00h | 0002h | x | return value |
| __api_rd_device_id3 | yes | 00h | 0003h | x | return value |
| __api_rd_bootloader_version | yes | 0Fh | 0000h | x | return value |
| __api_eeprom_busy | no | | | | |

Table 4. API Summary (Continued)

| Function Name | Bootloader Execution | R1 | DPTR0 | DPTR1 | Acc |
|------------------------|----------------------|----|-------|-------|-----|
| __api_rd_eeprom_byte | no | | | | |
| __api_wr_eeprom_byte | no | | | | |
| __api_start_bootloader | no | | | | |



Atmel Corporation

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

Regional Headquarters

Europe

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

Asia

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

Japan

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

Atmel Operations

Memory

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

Microcontrollers

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

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

ASIC/ASSP/Smart Cards

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

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

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

RF/Automotive

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

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

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

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

e-mail

literature@atmel.com

Web Site

<http://www.atmel.com>

Disclaimer: 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 Corporation 2004. All rights reserved. Atmel® and combinations thereof are the registered trademarks of Atmel Corporation or its subsidiaries. Other terms and product names may be the trademarks of others.



Printed on recycled paper.