Using an AT91EB40 as a Flash Programmer for AT91F40816 and AT91FR4081

AT91 ARM® Thumb® Microcontrollers

Application Note

Introduction

This Application Note describes how to use the AT91EB40 Evaluation Board with the Memory Extension Card AT91MEC01 to upload application software to an AT91 Flash-based microcontroller such as the AT91F40816 or the AT91FR4081. This method is an improvement over the existing AT91 PC Host Flash loader solution. The AT91 PC Host Flash loader is limited to a transfer rate of 115200 bit/s, whereas the AT91EB40 Flash Programmer can reach a transfer rate of 512000 bit/s.

Definitions of Terms

A *Flash-based AT91* is a member of the Atmel AT91 microcontroller family based on the ARM7TDMI processor that combines the microcontroller with a Flash memory in a single compact 120-ball BGA package. These include (but are not limited to) the AT91F40816 and AT91FR4081. The Flash-based AT91 is delivered by Atmel with resident boot software able to upload application software into its Flash memory.

The *AT91 Flash Programmer* is the code described in this Application Note that runs on the AT91EB40 Evaluation Board. It is supplied as a source code module (EB40_Flash_Programmer.zip) that can be downloaded and unzipped from Atmel's Web site.

A *host* is any item of equipment with a standard asynchronous serial communication port able to provide the Rx and Tx signals to the Flash-based AT91 and to upload application software into this Flash memory using the protocol defined by Atmel and described in this document. In the context of this Application Note, a host can be a PC or an AT91EB40 Evaluation Board.

The *target board* is either the AT91EB40 Evaluation Board or the user's application development board incorporating a Flash-based AT91. The latter is not supplied by Atmel.

The *binary code image* is the compiled or assembled application software to be uploaded into the Flash-based AT91.

Warranty

All source code modules supplied with this Application Note are free of charge and can be copied or modified without authorization. The software is delivered "AS IS" without warranty or condition of any kind, either express, implied or statutory. This includes, without limitation, any warranty or condition with respect to merchantability or fitness for any particular purpose, or against the infringements of intellectual property rights of others.

Rev. 1798A–18-Jan-02

Flash Memory Requirements

The AT91EB40 Flash Programmer is usable with the following Atmel Flash memories:

- AT49BV16x4, as incorporated into the AT91F40816
- AT49BV8011, as incorporated into the AT91FR4081

System Requirements

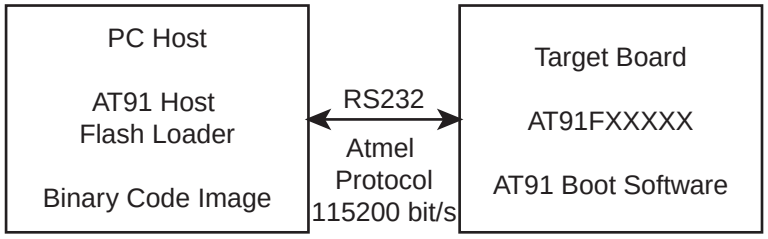
The following equipment is required:

- AT91EB40 Evaluation Board including:
  - AT91R40807 microcontroller
  - AT29LV1024 Flash memory
  - AT91EB40 Flash Programmer software (programmed into the AT29LV1024)
- AT91MEC01 Memory Extension Card including:
  - AT49BV1604 Flash memory
- PC Host including:
  - Connection to AT91EB40 through serial port A (USART0)
  - AT91 Host Flash loader software
- Target board with Flash-based AT91 including:
  - Connection to AT91EB40 through serial port B (USART1)
  - Factory-programmed resident boot software

AT91EB40 Flash Programmer

The AT91EB40 Flash Programmer is an improvement on the current AT91 PC Host Flash Loader used to upload and program an application in the Flash memory of a Flash-based AT91 microcontroller (refer to the AT91 Host Flash Loader documentation for more details). The AT91 PC Host Flash Loader is based on an RS232 communication link between the PC and the target board (Figure 1), using a protocol defined by Atmel and described later in this Application Note. The RS232 serial link between the PC and the target board is limited by the PC to a transfer rate of 115200 bit/s.

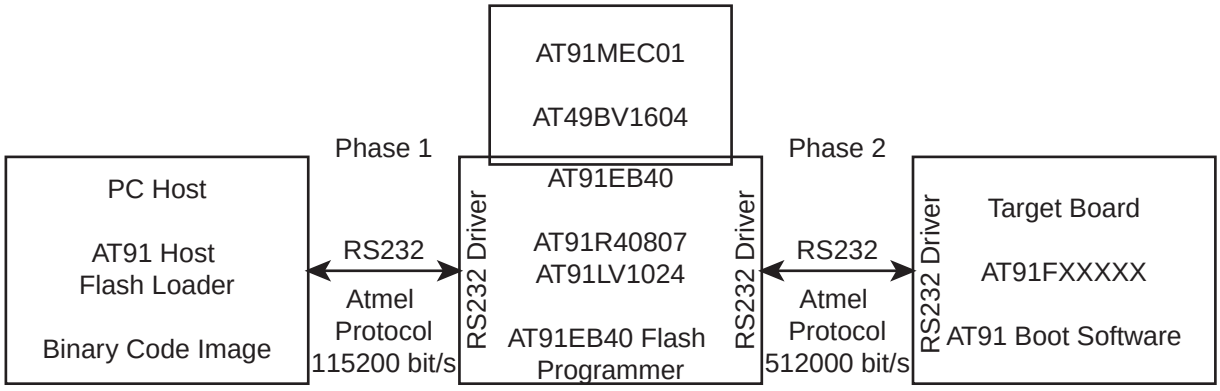
Figure 1. AT91 Host Flash Loader



As shown in Figure 2 below, the AT91EB40 Flash Programmer is based on the AT91EB40 Evaluation Board together with the AT91MEC01 Memory Extension Card (MEC). The procedure consists of downloading a binary code image into the Memory Extension Card at a transfer baud rate of 115200 bit/s and then uploading the same binary file into the target board with a transfer baud rate up to 512000 bit/s.

Communication between either the AT91EB40 Flash Programmer or the PC host and the target board is done via the RS232 serial link, using the protocol defined by Atmel.

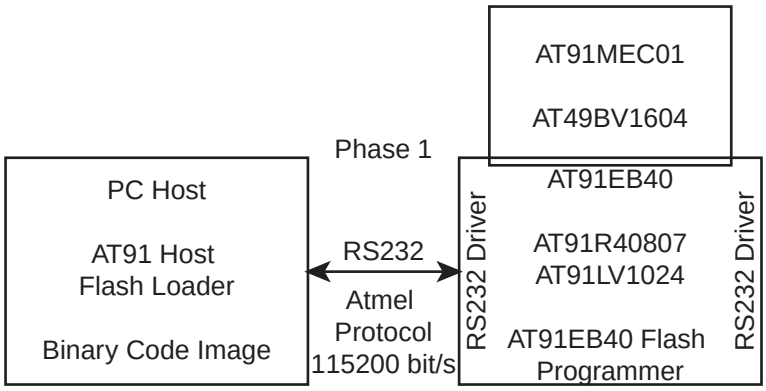
Figure 2. AT91EB40 Flash Programming Procedure



The AT91EB40 Flash Programmer operates in two phases:

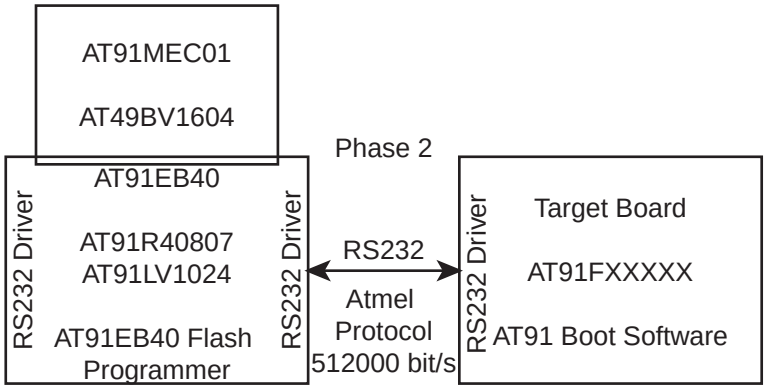
1. In the first, or download, phase, the AT91EB40 is used to download the binary code image via the AT91 PC Host Flash loader into the Memory Extension Card with a transfer baud rate limited to 115200 bit/s as shown Figure 3. In this phase, the AT91EB40 is considered to be the target board, and the PC is the host system. The download phase is done only once, then the user can disconnect the PC host and start the second phase as described below.

Figure 3. Download Phase



2. In the second, or upload, phase, the user can upload the binary code image previously downloaded from the Memory Extension Card into the Flash-based AT91 with a transfer baud rate up to 512000 bit/s as shown Figure 4. In this phase, the AT91EB40 is considered to be the host system, and the Flash-based AT91 is the target board. The binary code image can be uploaded into several target boards without repeating the download phase, thus is useful for mass production.

Figure 4. Upload Phase



AT91EB40 Flash Programmer Transfer Protocol

The AT91EB40 Flash Programmer transfer protocol is used for both the upload and the download phases described previously. It consists of two parts: the first part in the host to send data, and the second part in the target board to receive and upload the data into Flash memory.

The main features are:

- Full-duplex RS232 interface
- Automatic speed selection
- High-speed download

The following choices have been made for high-speed Flash uploads:

- Automatic transfer speed detection
- Sector erasing host implementation
- Minimum size of character transfer: 1 start bit, 8 data bits and one stop bit (no parity check)
- Reception and upload time-sharing using USART in interrupt mode with two data buffers

Protocol Requirements

The AT91EB40 Flash Programmer transfer protocol has the following requirements:

- Full-duplex communication between transmitter and receiver
- Asynchronous bi-directional full-duplex serial link, 1 start bit, 8 data bits and one stop bit (no parity check)
- No baud rate restriction (only limited by the AT91 MCKI)

Each transmitted data block contains a checksum that enables the receiver to verify the integrity of the transmitted data.

- The transmitter calculates the checksum and inserts this at the end of the block.
- The receiver calculates the checksum of the received data and compares it to the received checksum.

Specific characters are defined by the protocol, as shown in Table 1. They introduce a command block as defined in the section “Protocol Commands” on page 11.

The protocol is implemented by the following functions:

- Protocol speed selection
- Incoming command transfer: from interface
- Incoming data transfer: from interface
- Outgoing data transfer: to interface
- Application transfer and programming, page by page, from interface to chip
- Memory verification checksum.

Table 1. Protocol-specific Characters

Character	Hex Code	Meaning	Source
<SYNC>	0x80	Synchronization	Host
<ATS>	0x42 ‘B’	Answer to Synchronization	AT91
<ACK>	0x41 ‘A’	Acknowledge	AT91/Host
<NACK>	0x4E ‘N’	No Acknowledge	AT91/Host
<SPEED>	0x53 ‘S’	Speed Selection	Host

Table 1. Protocol-specific Characters (Continued)

Character	Hex Code	Meaning	Source
<ERASE>	0x45 ‘E’	Erase	Host
<WRITE>	0x57 ‘W’	Write	Host
<READ>	0x52 ‘R’	Read	Host
<DATA>	0x44 ‘D’	Data	AT91/Host
<VERIFY>	0x56 ‘V’	Verify	Host
<ERROR>	0x46 ‘F’	Error	AT91
<RESET>	0x5A ‘Z’	Software Reset	Host

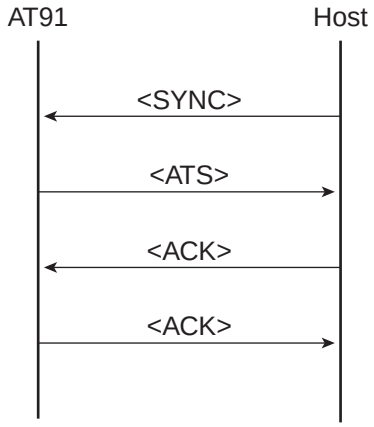
Protocol Procedures

All AT91EB40 Flash Programmer transfer protocol procedures as described below are Atmel-defined.

Transfer Speed Procedure

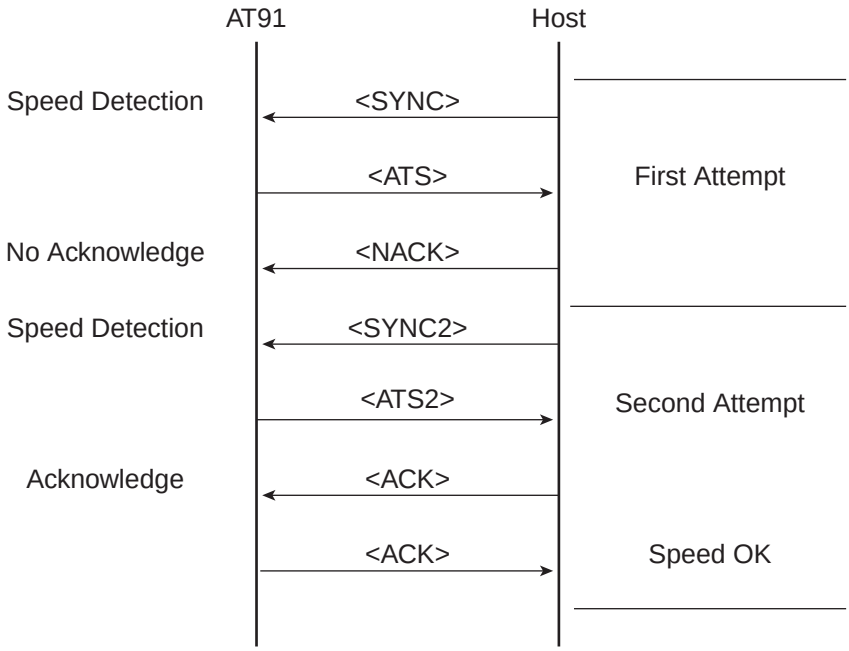
This procedure allows the AT91 in the target board to detect the serial communication speed of the host system and to send configuration data. The host system sends a single-byte <SYNC> synchronization command that enables the AT91 boot software to detect the transmission speed of the host system. The AT91 boot software waits for the falling edge on the reception of the start bit before sending an <ATS> Answer to Synchronization. The Answer to Synchronization includes the AT91 and the Flash specification, clock division factor and chip select base address.

Figure 5. Speed Detection Procedure



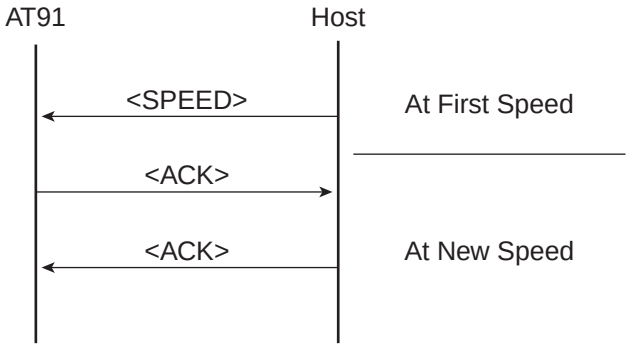
If the AT91 sets an incorrect speed, the host rejects the <ATS> command by sending a single-byte <NACK> command. The host then re-selects a new speed using the <SYNC> command to which the AT91 responds with an <ATS>. The AT91 stops the speed select procedure when it receives an <ACK> from the host and responds with an <ACK>. Otherwise, it re-enters speed selection for the next character.

Figure 6. Speed Detection Retry Procedure



At this time it is possible to change the speed for an improved transmission by the command <SPEED> Speed Selection.

Figure 7. Speed Modification Procedure

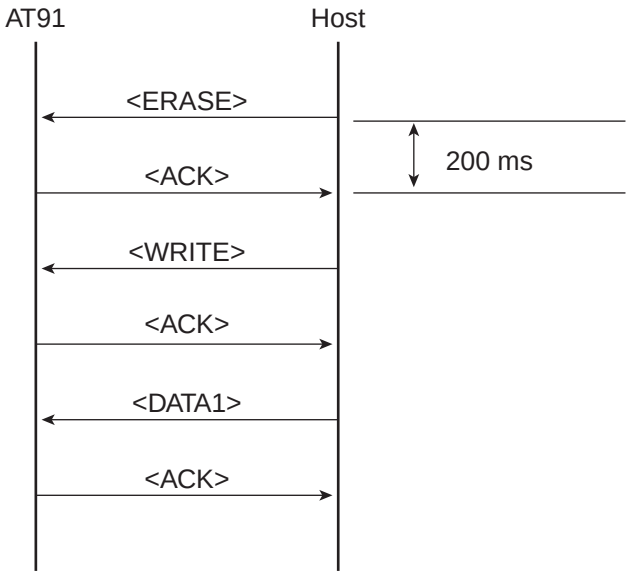


Data Programming Procedure

When the interface has been synchronized, data programming into the Flash memory of the target board can begin. This procedure requires the following steps:

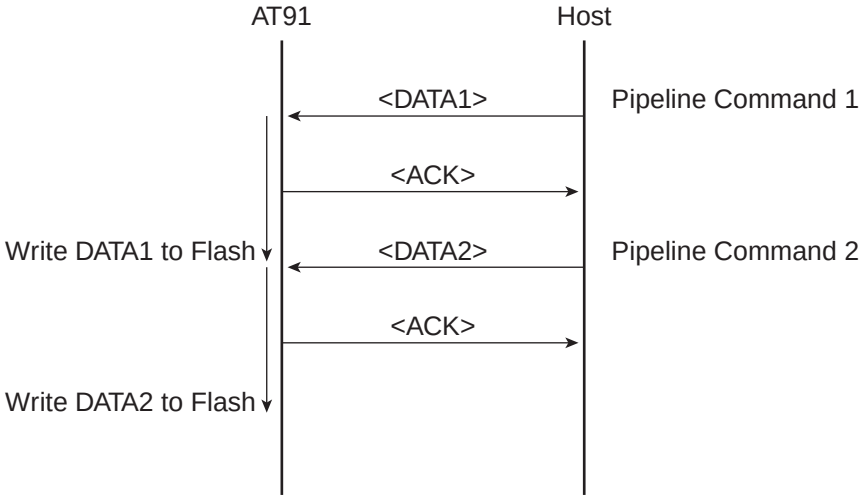
- Sector erasing
- Write command
- Data command

Figure 8. Data Programming Procedure



During these steps it is possible to use the <DATA> command in a pipelined manner. The AT91 boot software replies with an <ACK> after the block checksum verification but before data programming into the Flash. Therefore it is possible to receive a part of another data block during the Flash write time (for example, writing 128 bytes in Flash takes 128 * 20 µs = 2,56 ms).

Figure 9. Data Pipeline Procedure



Note: Warning: The AT91 boot software is stored at the beginning of the first Flash sector. It is selected by the Chip select 0 (see the EBI feature in the AT91 Datasheet). This Flash is selected at reset for the startup program. If this sector is overwritten, the boot software is erased.

Verification Procedures

There are two ways to check the contents of the Flash memory: the first is to use the <VERIFY> command, the second is to use the <READ> command. In addition, when a word is written into the Flash, the Flash programmer driver checks it.

Figure 10. Verify Command

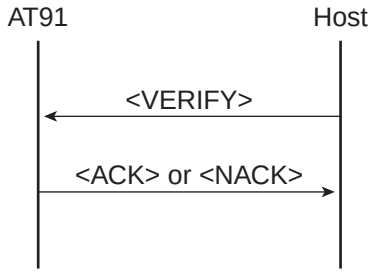
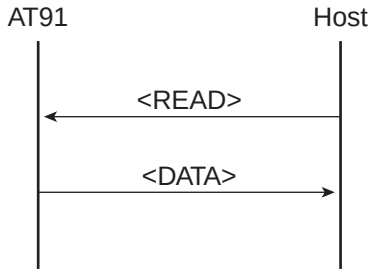


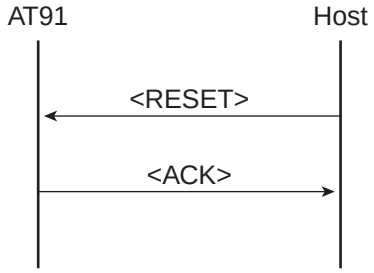
Figure 11. Verify by Read Command



Reset Procedure

The reset procedure allows the host to request an internal AT91 hardware reset in the target board via the watchdog timer. Before the watchdog reset, the target board sends an <ACK> to acknowledge this command to the host system.

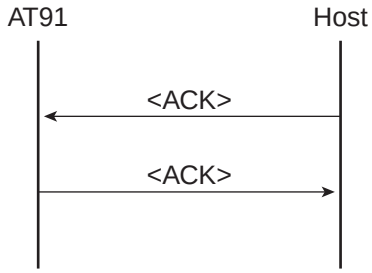
Figure 12. Reset Procedure



Acknowledgment Procedure

The acknowledgement procedure consists of an exchange of <ACK> commands between the host and the AT91 target board

Figure 13. Acknowledgement Procedure



Protocol Commands

Synchronization (SYNC)

Sent by the host after power-up to synchronize the interface at the correct baud rate.

First Byte Number (Hex)	Size (Bytes)	Name	Hex Value	Content
00	1	<SYNC>	0x80 ‘S’	Command code

Answer To Synchronization (ATS)

Sent by the AT91 after the synchronization procedure. The ATS frame consists of global boot software information.

First Byte Number (Hex)	Size (Bytes)	Name	Hex Value	Content
00	1	<ATS>	0x42 ‘B’	Command code
01	1	Chip		Type of AT91
02	2	Manufacturer	0x001F	Manufacturer Code
04	2	Flash		Flash Code
06	2	Version		Upload Version
08	2	CD		Clock Divisor
10	4	ADDR		Base Address Chip Select 0

Speed Selection (SPEED)

Sent by the host for an update of the clock divisor. This frame consists of the clock divisor value selected by the host.

First Byte Number (Hex)	Size (Bytes)	Name	Hex Value	Content
00	1	<SPEED>	0x53 ‘S’	Command code
01	2	CD		Clock divisor

Acknowledgement(ACK and NACK)

Command acknowledgement consists of transmitting the <ACK> command code when the command has been executed.

For the <DATA> command, the acknowledgement is transmitted if the integrity of the receive buffer is verified. This integrity is checked by comparing the calculated checksum with the received checksum. Otherwise, the <NACK> command is transmitted.

First Byte Number (Hex)	Size (Bytes)	Name	Hex Value	Content
00	1	<ACK>	0x41 ‘A’	Command code

First Byte Number (Hex)	Size (Bytes)	Name	Hex Value	Content
00	1	<NACK>	0x4E ‘N’	Command code

Sector Erase (ERASE)

Sector Erase command consists of erasing a sector of the Flash at the specific address.

First Byte Number (hex)	Size (Bytes)	Name	Hex Value	Content
00	1	<ERASE>	0x45 ‘E’	Command code
01	4	SECT		Address of Sector to erase

Target Reset (RESET)

Target Reset command consists of setting the watchdog timer and putting the hardware in wait mode. After reception of this command, the target sends an <ACK>

First Byte Number (Hex)	Size (Bytes)	Name	Hex Value	Content
00	1	<RESET>	0x5A ‘Z’	Command code

Data Write (WRITE)

Data Write command consists of sending the write address and the block length to the AT91 boot loader.

First Byte Number (Hex)	Size (Bytes)	Name	Hex Value	Content
00	1	<WRITE>	0x57 ‘W’	Command code
01	4	ADDR		Start address to write
05	1	LEN		Next Data Block length

Data Read (READ)

Data Read command consists of sending the read address and the block length to the AT91 boot loader.

First Byte Number	Size (Bytes)	Name	Hex Value	Content
00	1	<READ>	0x52 ‘R’	Command code
01	4	ADDR		Start address to read
05	1	LEN		Next Data Block length

Data Block (DATA)

Data Block is used to transfer data from host to AT91 and from AT91 to the host.

First Byte Number	Size (Bytes)	Name	Hex Value	Content
00	1	<DATA>	0x44 ‘D’	Command code
02	LEN value	D0..Dn		Data LEN defined in R/W

Memory Verification (VERIFY)

Memory Verification consists of receiving a start address, a size and a checksum, and comparing it to the checksum calculated in memory. The checksum is a 32-bit sum that is calculated by adding all bytes from BASE to BASE + SIZE-1 of the memory space and is used as a check. Failure of checksum verification causes <NACK>.

First Byte Number	Size (Bytes)	Name	Hex Value	Content
00	1	<VERIFY>	0x56 ‘V’	Command code
01	4	ADDR		Start address to verify
05	4	LNG		Size of code to verify
09	4	CHK		Checksum

Error Notification (ERROR)

Error Notification is sent by the AT91 to notify an error.

First Byte Number	Size (Bytes)	Name	Hex Value	Content
00	1	<ERROR>	0x46 ‘F’	Command code
01	1	Error		Error Code

The Error Codes are as follows:

Hex Value	Content
1	Flash write error
2	Flash read error
3	Flash erase sector error
4	Unrecognized command

AT91EB40 Flash Programmer Software

The AT91 Flash Programmer provided with this Application Note is intended to be run on the AT91EB40 Evaluation Board together with the AT91MEC01 Memory Extension Card. Once programmed, the AT91 Flash Programmer remains permanently in the AT29LV1024 Flash memory of the AT91EB40 Evaluation Board.

The binary code image is the application code to be programmed in the target Flash memory by the AT91 Flash Programmer . The AT91 Flash Programmer is first used to download the binary code image into the AT49BV16x4 Flash memory of the Memory Extension Card connected to the AT91EB40 Evaluation Board and then to upload the binary code image into the target Flash memory.

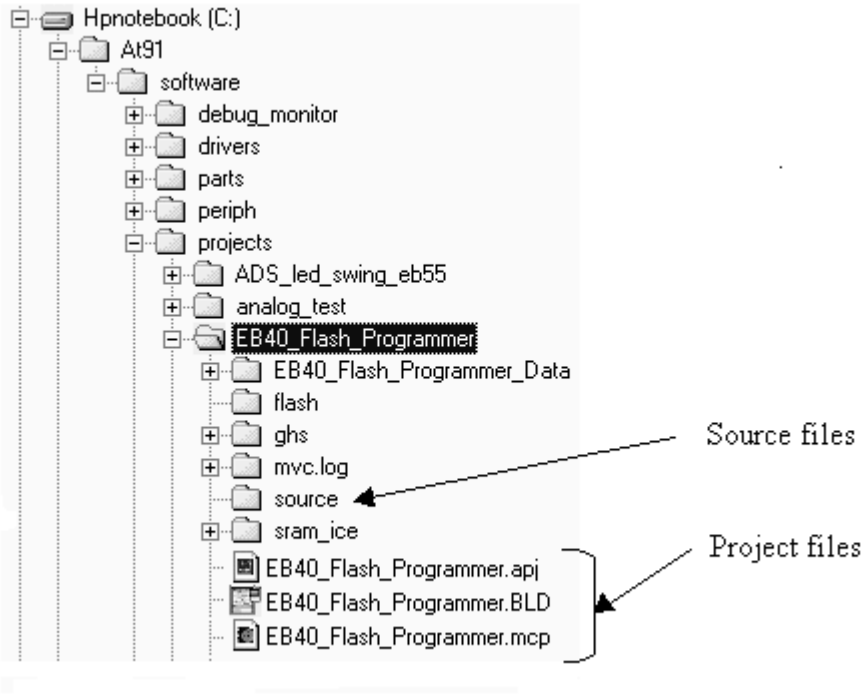
Software Implementation

It is advised to install the AT91 Flash Programmer by downloading the zip file EB40_Flash_Programmer.zip from Atmel's Web site so as to be compatible with the AT91 library. After extracting this file, the project files for the software development tools used are now available in the directory C:\AT91\software\projects\EB40_Flash_Programmer and the source files in the directory C:\AT91\software\projects\EB40_Flash_Programmer\source. See Figure 14.

Project files:

- `EB40_Flash_Programmer.apj` project file for ARM Software Development Toolkit V2.51
- `EB40_Flash_Programmer.bld` project file for Green Hills Multi 2000 V3.0
- `EB40_Flash_Programmer.mcp` project file for ARM Developer Suite V1.1

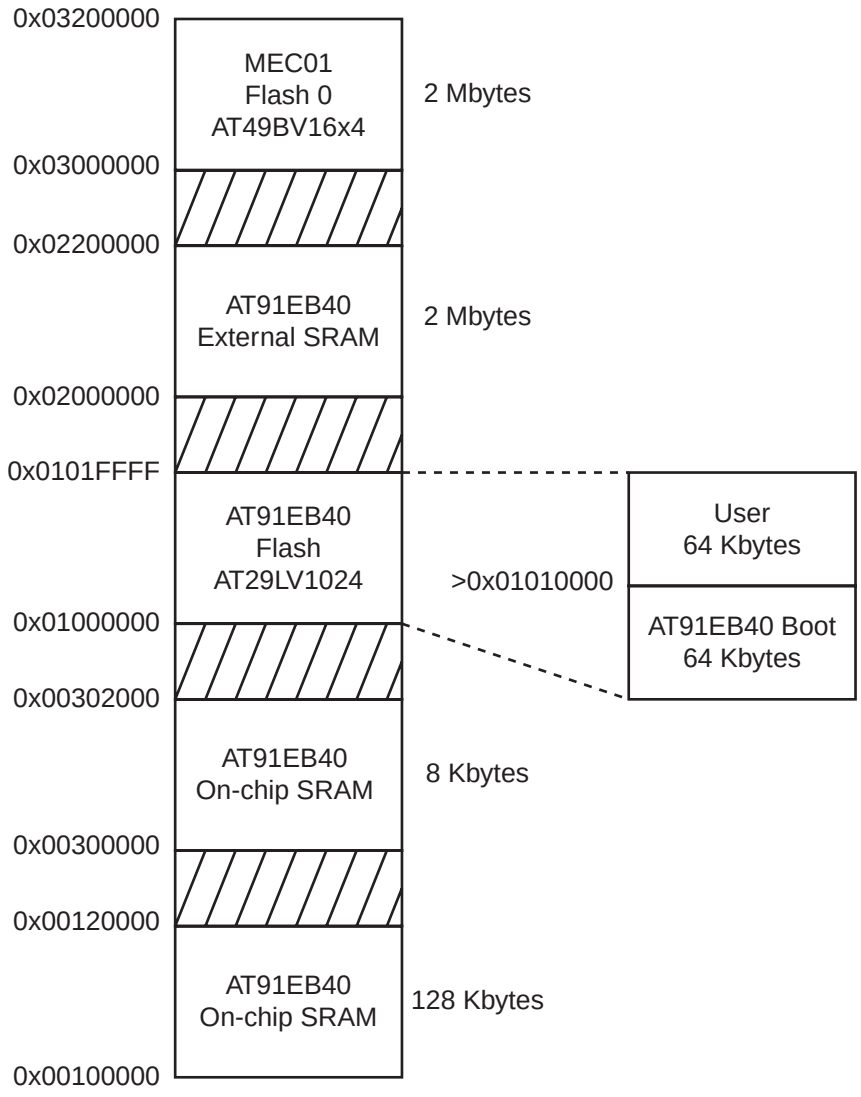
Figure 14. Project Workspace



Software Memory Mapping

After a reset, the AT91 Flash Programmer must always be executed first. To ensure this, the AT91 Flash Programmer must be located in the same Flash memory sector as the reset vector. On the AT91EB40 Evaluation Board, this is the AT29LV1024 Flash memory selected by Chip Select 0. The AT29LV1024 Flash memory is mapped at address 0x01000000 of the AT91EB40 platform memory mapping. When the AT91 Flash Programmer starts, it re-copies itself into the internal SRAM to run faster. The SRAM used by the AT91 Flash Programmer is mapped at address 0x00100000. The large 128K byte on-chip SRAM memory of the AT91R40807 is used because the AT91 Flash Programmer code size is greater than 8K bytes. See Figure 15.

Figure 15. AT91EB40 Flash Programmer Memory Mapping



The cstartup file must be modified to enable access to the large SRAM memory by setting the bit RAMWU of the Special Function Mode register as shown below:

Cstartup File

```
IF :DEF:AT91_DEBUG_NONE; {
INCLUDE ../../targets/cstartup_flash.arm
;-------------------------------------------------------------------------
; Call __low_level_init to perform initialization before initializing
; AIC and calling main.
; Diasable all peripherial clock
;-------------------------------------------------------------------------
    SF_MMR         EQU        0xFFF0000C     ;SF_MMR Address
    PtSF_MMR       DCD        0xFFF0000C     ;SF_MMR Base Address
__low_level_init
    ;values (relative)
    ldr                r1,PtSF_MMR
    mov     r0, #1
    str                r0,[R1]
    mov     pc,r14              ;Return
ENDIF    ;AT91_DEBUG_NONE }
```

The cstartup_flash file must be modified to copy Flash code to internal SRAM as shown below:

Cstartup Flash File

```
;- Copy Flash to Ram (ARM SDT or ARM ADS examples)
;- Get the Area Base and Limit
         mov      r0, #0                  ;read Flash at 0
         ldr      r1, =|Image$$RW$$Base|  ;Get pointer to bottom of data
         ldr      r2, =0x00100000         ;Internal RAM address

CopyLoop
         ldr      r10,[r0],#4
         str      r10,[r2],#4
         cmp      r2, r1
         blo      CopyLoop

;- Copy Flash to Ram (Green Hills Multi 2000 example)
;- Get the Area Base and Limit
         mov      r0, #0                  ;read Flash at 0
         ldr      r1, =__ghsbegin_data    ;Get pointer to bottom of data
         ldr      r2, =0x00100000         ;Internal RAM

CopyLoop
         ldr      r10,[r0],#4
         str      r10,[r2],#4
         cmp      r2, r1
         blo      CopyLoop
```

Detailed Implementation

Bootloader.c Module

The bootloader.c module is the main module of the AT91EB40 Flash Programmer software. It has two parts: initialization and interrupt routine. Initialization includes EBI initialization, LED initialization and the interrupt configuration. The interrupt routine involves checking the interrupt controller to verify which interrupt occurred and calling the corresponding functions; memory_download( ) or memory_upload( ) as defined in the fmu.c file. The interrupt routine is based on two external interrupt sources FIQ and IRQ0 from SW3 and SW5, respectively.

```
void main (void)
```

This is the main function of the bootloader.c module. After initialization, this function enters an endless loop while(1) and waits for an interrupt sources from external interrupts FIQ or IRQ0.

```
void Init_Led(void)
void Init_Ebi(void)
void Init_Interrupt(void)
```

These functions initialize the LEDs for the user interface, the External Bus Interface to configure the Memory Extension Card and interrupts.

```
void at91_IRQ0_handler(void)
void FIQ_handler (void)
```

These interrupt handler functions set the corresponding interrupt flags.

Fmu.c Module

The fmu.c module handles the downloading sequence between the host PC and the AT91EB40 and the uploading sequence between the AT91EB40 and the target board according to the protocol defined by Atmel and described previously.

```
u_char memory_download (void)
u_char memory_upload (void)
```

These functions are the main functions of the fmu.c module and are called by the main function depending on the interrupt source. The memory_download function enters the download phase between the host PC and the AT91EB40, allowing the AT91MEC01 Flash memory to be programmed using the AT91 PC loader, whereas the memory_upload function enters the upload phase between the AT91EB40 and the target. These functions return TRUE if the corresponding phase has been carried out without error, otherwise return FALSE.

```
void Init_Terminal(void)
```

This function initializes the USART0 and USART1 communication terminals and also the terminal buffers.

```
void pio_c_irq_handler (void)
```

This interrupt handler function handles the PIO IRQ interrupt on the USART0 to detect the SYNC command from the host.

```
u_char do_synchronization (void)
```

This function is called by the memory_download function and returns TRUE when the synchronization has been done with the host.

```
u_int speed_detection (void)
```

This function is called by the do_synchronisation function and returns the transfer baud rate detection.

```
u_char send_synchro(void)
u_char send_speed(void)
u_char send_erase(unsigned int addr)
u_char send_write(unsigned int addr, unsigned char len)
u_char send_data( unsigned char *buffer, unsigned int len)
u_char send_verify(unsigned int addr, unsigned int len, unsigned int
checksum)
```

These functions are used during the memory upload phase and send the corresponding commands to the target board. They return TRUE after receiving the acquisition command from the target board, otherwise they return FALSE.

```
void send_command (char command_id, char length)
```

This function is called by the functions defined above and configure the terminal before transmission.

```
u_char get_command_status(void)
```

This function is also called by the functions defined above. It gets the acquisition command from the target board and returns TRUE, otherwise it returns FALSE.

Term1.c Module

The term1.c module manages the USART communication interface terminal using USART interrupt to transfer and receive commands.

```
void at91_term1_c_handler ( TerminalDesc *term_desc )
```

This interrupt handler function is the main function of the term1.c module and handles the interrupts on USART0 and USART1 during the reception and transmission procedures.

```
void at91_term1_open ( TerminalDesc *term_desc )
void at91_term1_close ( TerminalDesc *term_desc )
```

These functions open and close the USART communication interface.

lib_flash_at49.c Module

This module handles Flash operations during downloading and uploading sequences.

```
static int at91_wait_flash_ready ( flash_word *address, flash_word data )
void at91_flash_identify ( flash_word *base_addr, flash_word *manuf_code,
flash_word *device_code )
int at91_erase_sector (flash_word *base_addr, flash_word *sector_addr)
int at91_write_flash ( flash_word *base_addr, flash_word *load_addr,
flash_word data )
void at91_init_flash_table ( FlashAt49BVDef *FlashTable )
unsigned int at91_check_erase_sector ( FlashAt49BVDef *FlashTable,unsigned
int sector_id)
void at91_set_erase_sector ( FlashAt49BVDef *FlashTable,unsigned int
sector_id)
flash_word at91_get_flash_sector_size ( FlashAt49BVDef *FlashTable, unsigned
int sector)
int at91_get_flash_sector ( FlashAt49BVDef *FlashTable,unsigned int
load_addr,
unsigned int base_addr)
u_char check_flash_sector(unsigned int current_add)
```

These functions are used during the memory download phase and operate on the Flash AT91MEC01 memory.

```
u_char Flash_upload(void)
```

This function is called by the memory_upload function and executes the Flash uploading routine.

```
void Init_FlashTable(void)
```

This function is called by the memory_upload function and initializes the user Flash table according to the target.

```
u_char write_flash(unsigned int *current_add,unsigned int *current_size_send,
unsigned int max_size, unsigned int *checksum, unsigned char *input)
```

This function is called by the Flash_upload function and is used to write data in target Flash.

```
unsigned int verifyChecksum(unsigned char *buffer, unsigned int size)
```

This function is called by the Flash_upload function and returns the checksum memory.

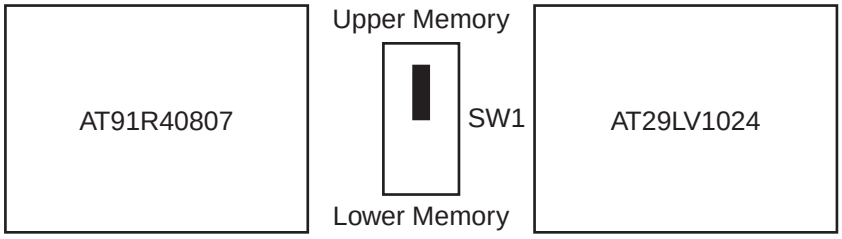
AT91EB40 Configuration

The EB40 Flash programmer uses the following features of the AT91EB40:

- AT91R40807
- AT29LV1024
- Two serial ports (Serial A and B)
- Reset button (SW2)
- Three LEDs (LED1, LED2, LED3)
- Two application-dedicated buttons (SW3, SW5)
- External Bus Interface

As described above, the AT91 Flash Programmer must be downloaded into the AT29LV1024 Flash memory that has been selected by Chip select 0 after reset. To do this, the user can use any Flash downloader solution. The AT29LV1024 Flash memory is mapped at address 0x01000000 of the AT91EB40 platform memory mapping. The switch SW1 drives the bit A15 of this Flash and allows the user to reach or not the entire Flash. For this application, it is advised to use the switch SW1 as shown in Figure 16 in "UPPER MEM" space from 0x01010000 up to 0x0101FFFF. The upper 64K of the Flash can be overwritten regardless of the position of the switch SW1. The lower 64K are write-protected, regardless of the position of the switch SW1. This is to prevent the boot and Angel software stored in the lower 64K bytes from being erased. Nevertheless, it is always possible to make this space unprotected by setting a jumper or a link on the footprint J7.

Figure 16. AT29LV1024 Flash Memory Space Selection.



The Serial A port (USART0) is used to link the AT91EB40 platform to the PC host loader during the downloading phase, whereas the Serial B port (USART1) is used to link the AT91EB40 to the target board during the upload phase. Special care must be taken as to avoid any inversion.

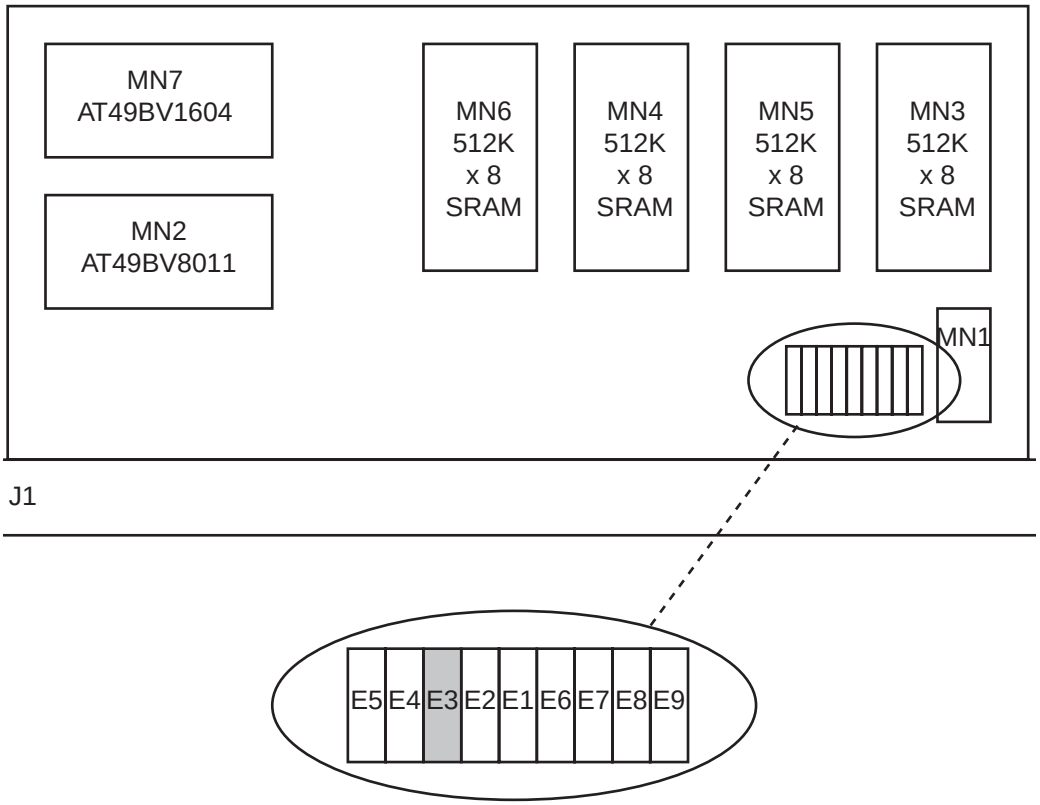
If the External Bus Interface connector has not been fitted at the factory, the user must fit any 32 x 2 connector to the Memory Extension Card.

AT91MEC01 Configuration

As described previously, the AT91 Flash Programmer uses the 2-Mbyte AT49BV1604 Flash memory of the AT91MEC01 memory extension card to download the binary file for upload in the target board. The External Bus Interface (EBI) must be configured to support the AT91MEC01 memory extension card and integrate the additional memory into the AT91EB40 platform memory mapping. The base addresses of the memory banks can be configured as required by the user.

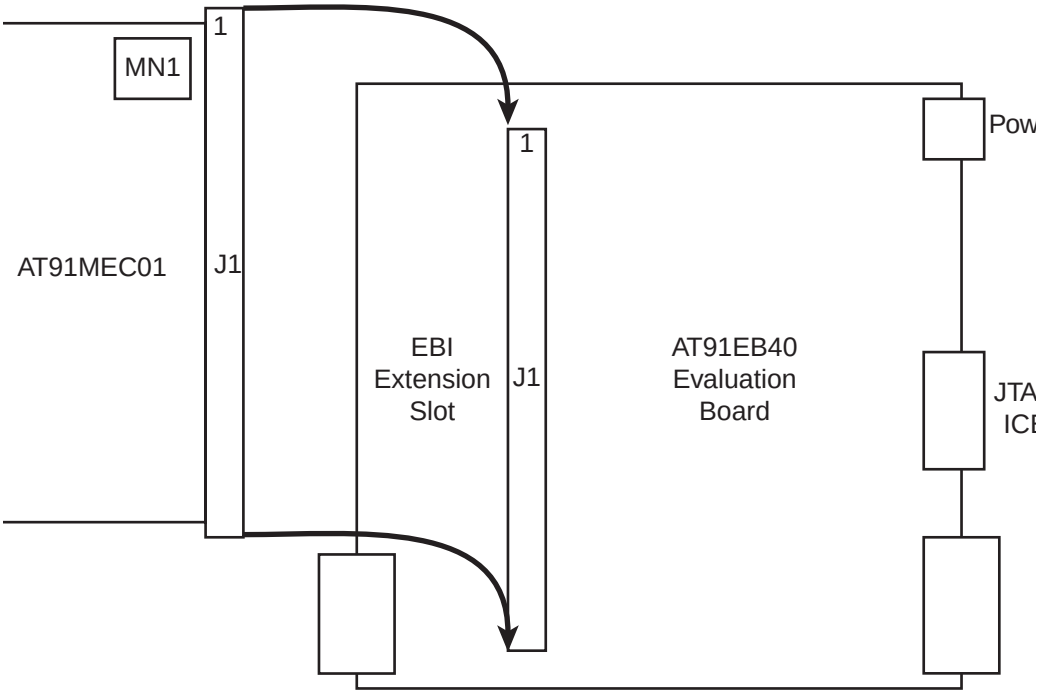
In the AT91 Flash Programmer software, the EBI is configured in the function Init_Ebi( ) of the bootloader.c module and the base address of the AT49BV1604 Flash memory called Flash 0 is 0x03000000 and selected by NCS2 when Jumper E3 is closed. So care must be taken to close the Jumper E3 on the Memory Extension Card as shown in Figure 17.

Figure 17. Layout of the AT91MEC01.



To connect the AT91MEC01 Memory Extension Card to the AT91EB40 evaluation board, solder the connector on the Evaluation Board. Plug in the Memory Extension Card slot into the connector on the Evaluation Board. Take care to match Pin 1 as shown in Figure 18.

Figure 18. Connecting the AT91MEC01 to the AT91EB40.



How to Use the AT91 Flash Programmer

AT91 Flash Programmer Customization

The purpose of this section is to help users customize the AT91 Flash Programmer according to their application needs. In most cases, only the header file version.h available in the directory C:\AT91\software\projects\EB40_Flash_Programmer\source needs to be modified by the user in order to configure the parameters as described below.

As stated at the beginning of this Application Note, the AT91EB40 Flash programmer can be used with specific Atmel Flash memory devices. Because all the Flash memory information (chip ID, organization, ...) is already defined as constants in the lib_flash_at49.c module, the user has to select the corresponding Flash device available on his target application and all the information corresponding to this device will be defined.

```
//* Select target Flash to program
#define    AT49BV8011   0
#define    AT49BV8011T  0
#define    AT49BV16X4   1
#define    AT49BV16X4T  0
```

Now, the user has to configure the communication interface between the AT91EB40 and the target board during the uploading phase to upload the binary file previously downloaded in the AT49BV1604 by setting the corresponding Clock Divisor (CD) value "EB40_CD".

According to this EB40_CD value, the user also has to adjust the corresponding "TARGET_CD" to have the same baud rate between the AT91EB40 Flash programmer (defined by "EB40_CD") and the target board (defined by "TARGET_CD").

Example

The target board is running at 16Mhz and the binary code image is to be uploaded with a transfer baud rate of 256000 bit/s. The corresponding CD values have to be chosen as shown below.

```
//* EB40 CD value to select transfer baud rate
//* EB40 MCK = 32.768 MHz
//* Transfer Baud rate = EB40_MCK/(16 x EB40_CD)
//* Baud rate = 512000 ----> CD = 4
//* Baud rate = 256000 ----> CD = 8
//* Baud rate = 128000 ----> CD = 16
#define    EB40_CD            8
//* Target CD value to reach transfer baud rate from EB40 + MEC
//* baud rate target must be equal to transfer baud rate!!
//* baud rate target = MCK_target/(16 x TARGET_CD)
#define    TARGET_CD          4
```

Next, the user has to define the base address of the Target Flash and the load address to upload the image binary file.

```
#define    BASE_ADDRESS       (0x01000000) //* Target flash base address
#define    LOAD_ADDRESS       (0x01010000) //* Target flash load address
```

The user can also define the "FILE_ADDRESS" of the AT49BV1604 of the AT91MEC01 where he wants to download the binary file. However, it is advised to keep this value fixed to 0x3000000 to prevent confusion and also to reach up to 2 Mbytes.

```
#define    FILE_ADDRESS      ((u_char *)0x03000000) //Flash MEC address
```

The last parameter is called "FILE_SIZE_DEFAULT". It defines the file size of the binary code image to upload in the target board. This value is important due to the fact there is no way to know the binary file size. If the user has previously downloaded the binary file into the AT91MEC01 Flash, the binary file size is automatically calculated by the program. Otherwise the value defined by the user is taken by default. After reset, the AT91 Flash Programmer takes this FILE_SIZE_DEFAULT value by default.

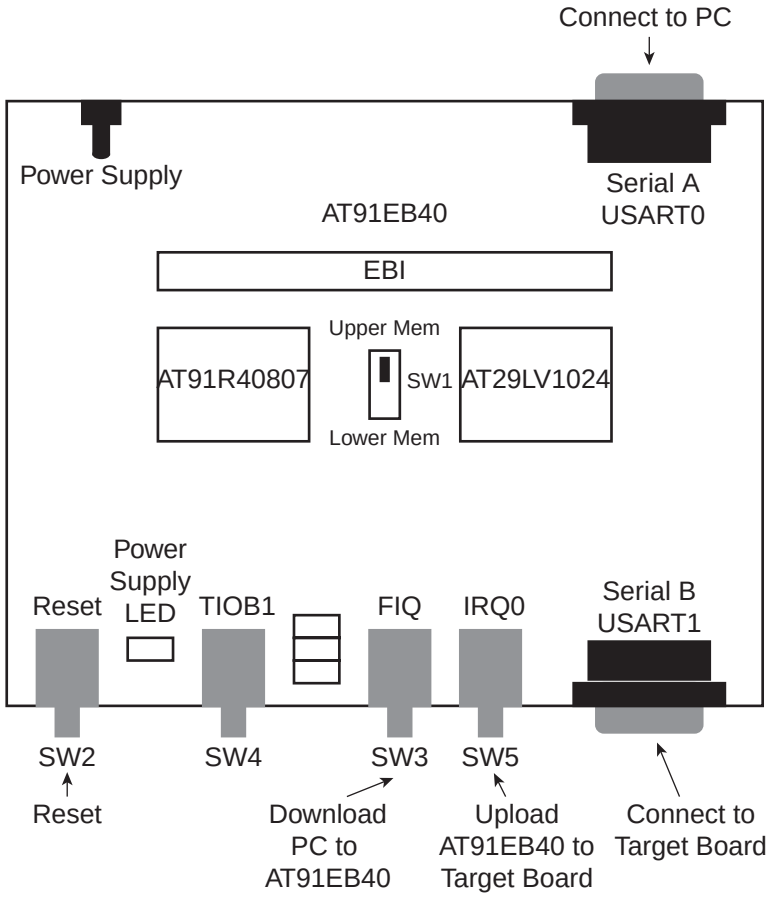
```
#define    FILE_SIZE_DEFAULT   (0x80000)//Default File size to upload
```

After modifying the version.h file according to the user application (target Flash, transfer baud rate, as described above), the entire project must be compiled and a new binary file generated to download the AT91 Flash Programmer software into the Flash AT29LV1024 (upper memory at address 0x1010000) using any Flash downloader.

Board-level Interface

The Board-level interface between the user and the AT91EB40 Flash Programmer is created by using the three LEDs (LED1, LED2 and LED3) and two application-dedicated buttons, SW3 and SW5 (FIQ and IRQ0). The three LEDs display the state of the programming sequence and the two application-oriented buttons are used to select the corresponding programming sequence as shown in Figure 19.

Figure 19. Board-level User Interface.



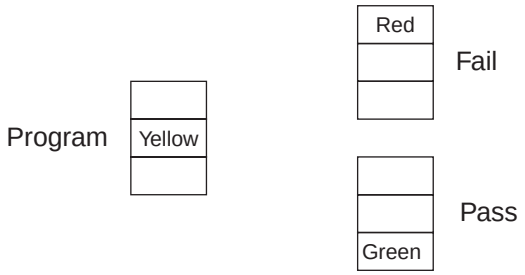
After Reset, The AT91EB40 Flash programmer software is started and the three LEDs (LED1, LED2 and LED3) are lit up. At this time the software is waiting for an interrupt from one of two different sources, SW3 (FIQ) and SW5 (IRQ0).

If the user presses SW3, the program enters download phase and can download a binary code image through Serial port A in the AT91MEC01 Flash AT49BV1604 by running the AT91 PC Host Flash loader.

If the user presses SW5, the program enters in uploading phase and can upload the binary code image through Serial B port to the target board.

During the downloading or uploading phase LED2 (Yellow) is lit up and LED1, LED3 are extinguished. When the phase is finished and verified LED3 (Green) is lit up. However if an error occurs during the programming phase LED1 (Red) is lit up. See Figure 20.

Figure 20. AT91EB40 LED Indications



Benchmark

The AT91EB40 Flash Programmer was used to program the AT49BV1614 Flash device available on the AT91EB55 Evaluation Board. This benchmark was realized at MCK = 32.768 Mhz. Compared to the AT91 PC Host Flash Loader, the programming time can be improved by 4 with the AT91EB40 Flash Programmer running at 512000 bit/s.

Table 2. Benchmark Transfer Times

Baud Rate (Bits/sec)	Clock Divisor	1 Mbyte	2 Mbytes
113700	18	100 s	200 s
256000	8	60 s	95 s
341333	6	40 s	75 s
512000	4	30 s	55 s
AT91 PC Host Flash Loader	18	115 s	215 s

References

AT91MEC01 Memory Extension Card User Guide (Literature No. 1387)

AT91EB40 Evaluation Board User Guide (Literature No. 1706)

Document Details

Title	Application Note: Using an AT91EB40 as a Flash Programmer for AT91F40816 and AT91FR4081
Literature Number	1798

Revision History

Version A	**Publication Date:** 18-Jan-02

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
Casa Postale 80
CH-1705 Fribourg
Switzerland
TEL (41) 26-426-5555
FAX (41) 26-426-5500

Asia
Atmel Asia, Ltd.
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimhatsui
East Kowloon
Hong Kong
TEL (852) 2721-9778
FAX (852) 2722-1369

Japan
Atmel Japan K.K.
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
Atmel Corporate
2325 Orchard Parkway
San Jose, CA 95131
TEL 1(408) 436-4270
FAX 1(408) 436-4314

Microcontrollers
Atmel Corporate
2325 Orchard Parkway
San Jose, CA 95131
TEL 1(408) 436-4270
FAX 1(408) 436-4314

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

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

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

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

Atmel Colorado Springs
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
Atmel Grenoble
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

© 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® and the Atmel logo are the registered trademarks of Atmel.

ARM® and Thumb® are the registered trademarks of ARM Ltd.

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

Printed on recycled paper.

1798A–ATARM–18-Jan-02 0M