



SAM4S Boot Strategies & Programming solutions



SAM4S Boot Strategies

Boot Strategies

SAM4S provides 2 booting solutions:

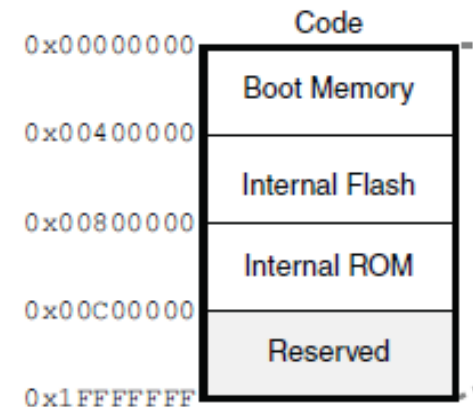
- Boot from the embedded ROM (**SAM-BA Boot**)
- Boot from the embedded Flash (**User application**)

Boot mode selection is made by setting/clearing GPNVM bit 1

- GPNVM bit 1 = 0: boot from ROM
- GPNVM bit 1 = 1: boot from Flash

Embedded ROM contains:

- SAM-BA Boot
- In Application Programming function (IAP)
- Fast Flash Programming Interface (FFPI)



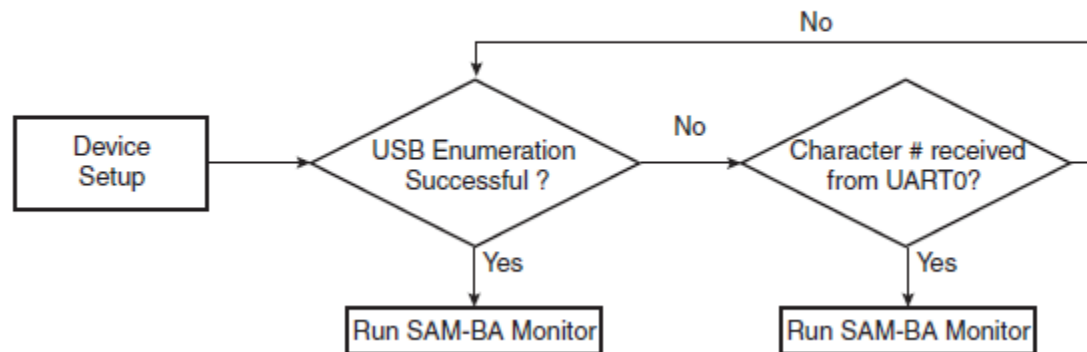
SAM4S memory mapping

SAM-BA Boot

GPNVM bit 1 = 0

- SAM-BA Boot is a default Boot Program which provides an easy way to program in-situ the on-chip Flash memory.
- SAM-BA Boot supports serial communication via the UART and USB.
- SAM-BA Boot provides an interface with SAM-BA Graphic User Interface (GUI).

Figure 22-1. Boot Program Algorithm Flow Diagram



SAM-BA Boot: Hardware & Software constraints

- SAM-BA Boot uses the first 2048 bytes of the SRAM for variables and stacks. The remaining available size can be used for user's code.
- USB requirements: external crystal or external clock with frequency of:
 - 11,289 MHz
 - 12,000 MHz
 - 16,000 MHz
 - 18,432 MHz
- UART0 requirements: None
 - URXD0 (PA9) and UTXD0 (PA10) are driven during execution
- External clock must be a 1.2V square wave signal

IAP - In Application Programming Function

- IAP feature is a function located in ROM
 - Can be called at any time
- When called, this function sends the desired FLASH command to the EEFC and waits for the Flash to be ready
- Allows flash programming while the code is running out of flash
 - No need to have programming routines in SRAM
- This function takes one argument in parameter: the command to be sent to the EEFC
- The IAP function entry point is retrieved by reading the NMI vector in ROM (0x00800008).

IAP - In Application Programming Function

```
(unsigned int) (*IAP_Function)(unsigned long);

void main (void){
unsigned long FlashSectorNum = 200; //
unsigned long flash_cmd = 0;
unsigned long flash_status = 0;
unsigned long EFCIndex = 0; // 0:EEFC0, 1: EEFC1

/* Initialize the function pointer (retrieve function address from NMI
vector) */
IAP_Function = ((unsigned long) (*)(unsigned long)) 0x00800008;

/* Send your data to the sector here */
/* build the command to send to EEFC */
flash_cmd = (0x5A << 24) | (FlashSectorNum << 8) | AT91C_MC_FCMD_EWP;

/* Call the IAP function with appropriate command */
flash_status = IAP_Function (EFCIndex, flash_cmd);
}
```

SAM4S Programming Solutions

Programing solutions

- Development Tools such as IAR, Keil integrate their own flash loaders utility to flash the application during debug phase.



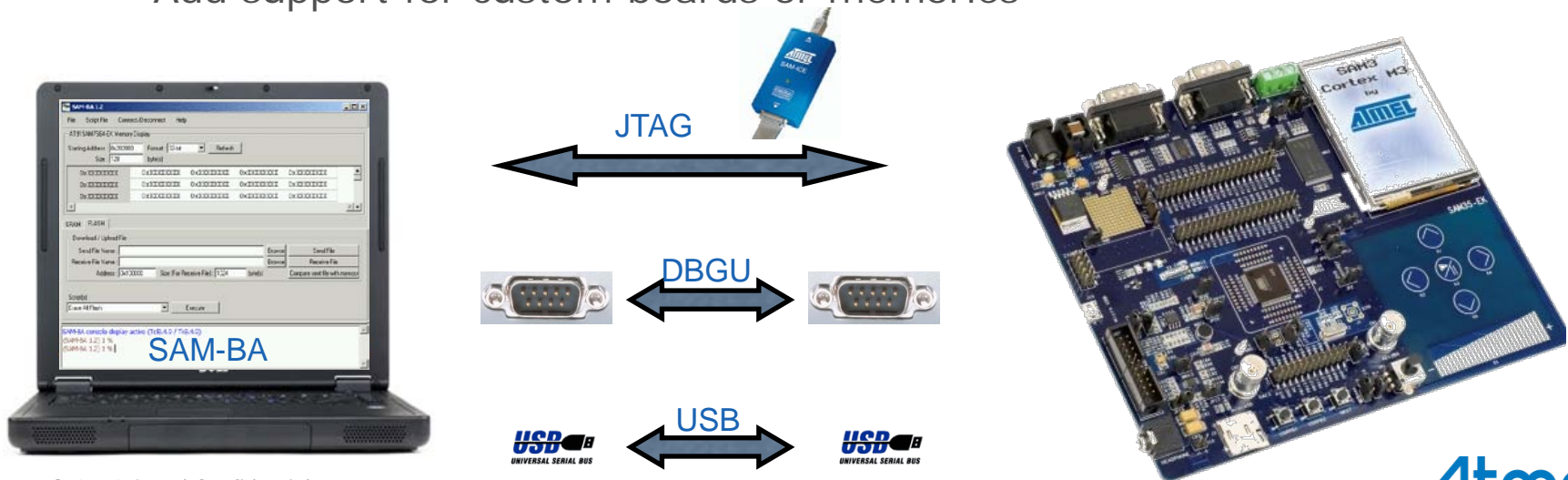
- J-Flash from Segger can be used to program the on-chip flash memory



- SAMBA_DLL.dll: [Atmel's Free](#) solution for customers to create their own GUI Interfaces
- Gang Programmers: support for our whole flash-based microcontroller thanks to our [FFPI](#).

SAM-BA GUI

- SAM-BA UI provides In-System Programming solutions for on-chip and/or on-board memories
- Support many communication channels
 - USB, Serial (DBGU/UART) with SAM-BA Boot running out of the target
 - JTAG ICE Port (no need for SAM-BA boot)
- Customizable
 - Other ISP solution can use SAMBA_DLL (Custom GUI)
 - Add support for custom boards or memories



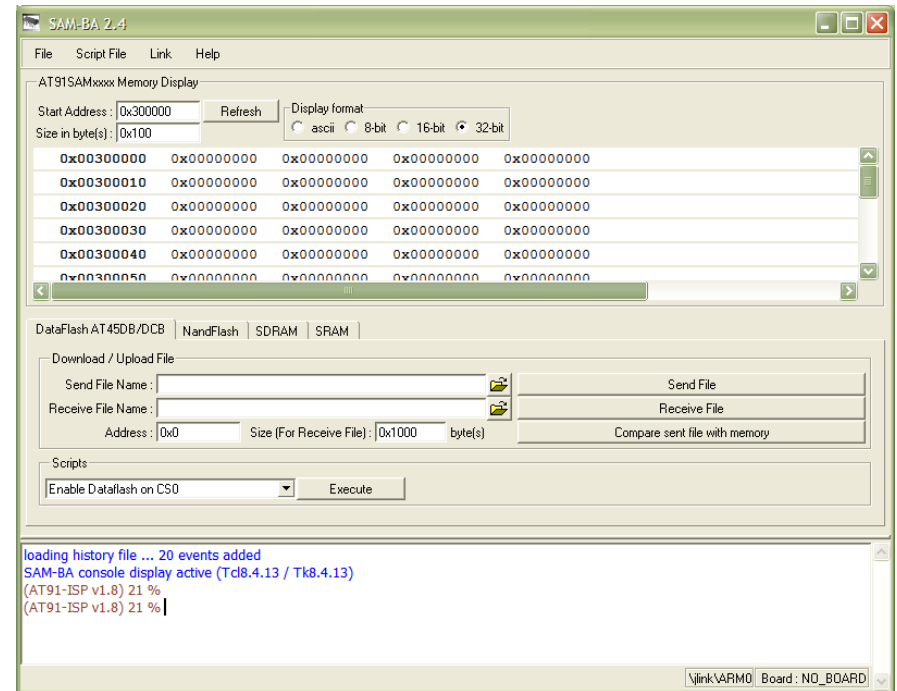
SAM-BA GUI

- Customizing SAM-BA is possible by adding or modifying TCL scripts files

Create your own board

Add memory modules

Modify Memory Algorithms



- Command Line Mode: allows memory programming without any GUI interaction



Enabling Unlimited Possibilities®

© 2012 Atmel Corporation. All rights reserved.

Atmel®, Atmel logo and combinations thereof, Enabling Unlimited Possibilities®, and others are registered trademarks or trademarks of Atmel Corporation or its subsidiaries. Other terms and product names may be trademarks of others.

Disclaimer: The information in this document is provided in connection with Atmel products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Atmel products. EXCEPT AS SET FORTH IN THE ATMEL TERMS AND CONDITIONS OF SALES LOCATED ON THE ATMEL WEBSITE, ATMEL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS AND PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ATMEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Atmel makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and products descriptions at any time without notice. Atmel does not make any commitment to update the information contained herein. Unless specifically provided otherwise, Atmel products are not suitable for, and shall not be used in, automotive applications. Atmel products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.