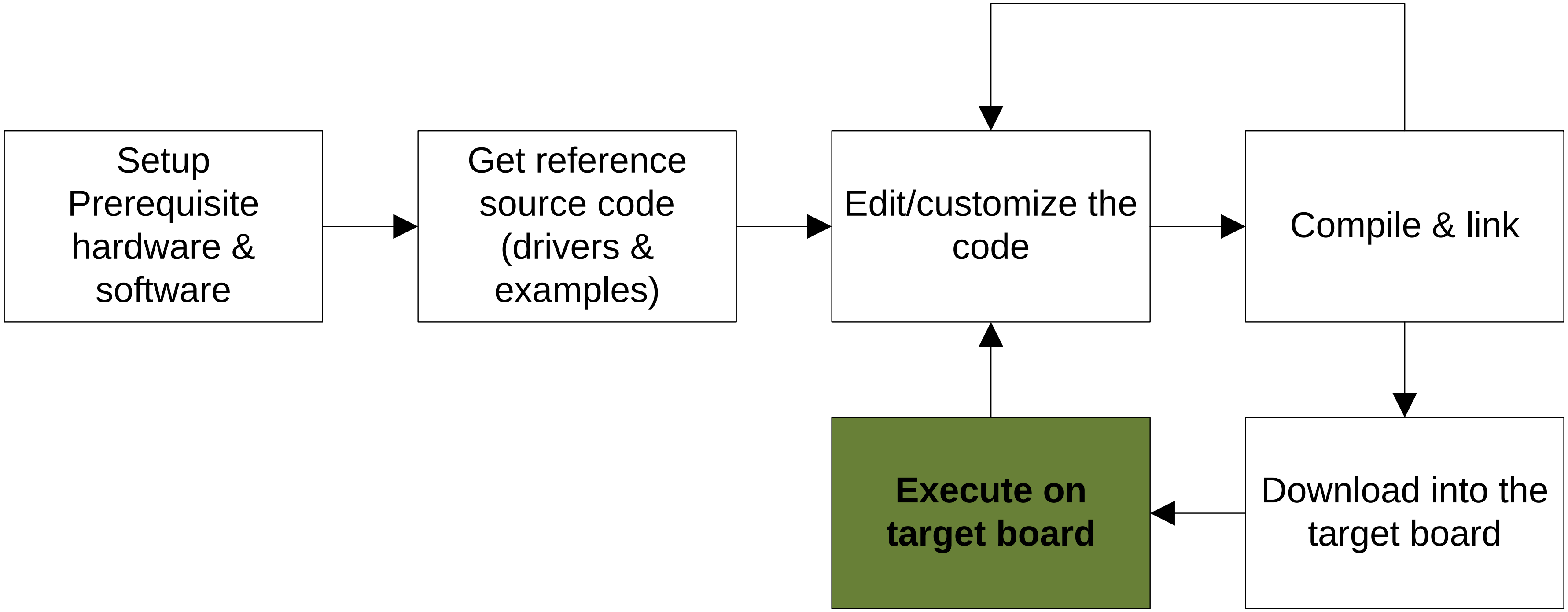
Board Bring up Considerations

MPU Team

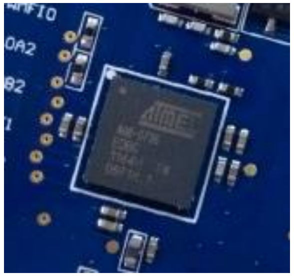
June 2016

Board bring-up - the big picture


Setup Prerequisite hardware & software
Get reference source code (drivers & examples)
Edit/customize the code
Compile & link
Download into the target board
Execute on target board

Board bring-up - prerequisites

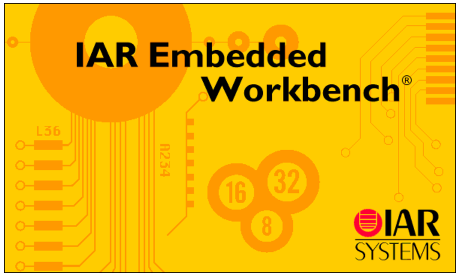
1- a debug interface (= USB ⇔ JTAG translator)



EDBG (or JLINK-OB)

SAM-ICE

2- a SW development environment a.k.a. “IDE” running on a host computer
IDE = source code editor + compiler + linker + debugger


IAR Embedded Workbench®

ARM

DS-5™

Development Studio 5

Loading Workbench

Copyright © 2013 ARM Ltd. All rights reserved.
This product is protected by US and international laws.

Board bring-up – heads-up

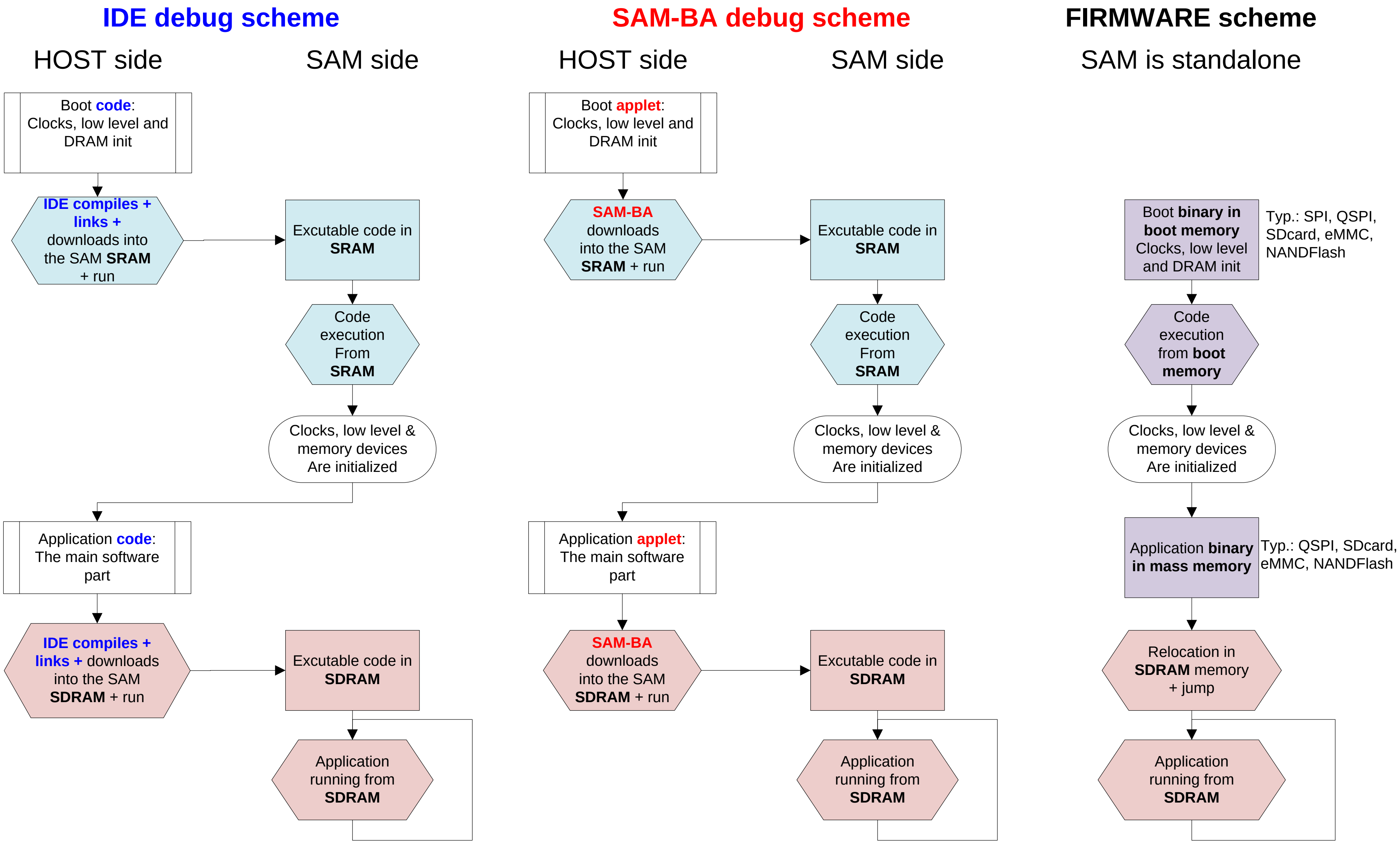
- *“Waking up the darn thing”* is the first challenge our customers have to overcome when 1st receiving their prototype board – and that’s a crucial one!

- Hence that’s the #1 topic FAEs and support forces are confronted to, which takes a mighty 50% of all support bandwidth!

Besides software considerations described in further slides, let’s point at a few hardware things that may mess-up with the SAM startup or the board boot:

- Too weak power supply block (insufficient current source)
- Not all power domains are powered (careless design / power tree bug)
- Design error around Xtal oscillator (crystal does not oscillate)
- Design error around nRESET signal (keeps stuck low)
- Bad PCB design around signal integrity (corrupted DRAM busses)
- Major design issue (e.g. wrong connection, signal contention, etc.)
- SAMA5D2 wrong fuses configuration (can ban some boot devices)

Board bring-up – SW overview



IDE debug scheme

HOST side

SAM side

Boot **code**:
Clocks, low level and DRAM init

IDE compiles + links + downloads into the SAM **SRAM** + run

Excutable code in **SRAM**

Code execution From **SRAM**

Clocks, low level & memory devices Are initialized

Application **code**:
The main software part

IDE compiles + links + downloads into the SAM **SDRAM** + run

Excutable code in **SDRAM**

Application running from **SDRAM**

SAM-BA debug scheme

HOST side

SAM side

Boot **applet**:
Clocks, low level and DRAM init

SAM-BA downloads into the SAM **SRAM** + run

Excutable code in **SRAM**

Code execution From **SRAM**

Clocks, low level & memory devices Are initialized

Application **applet**:
The main software part

SAM-BA downloads into the SAM **SDRAM** + run

Excutable code in **SDRAM**

Application running from **SDRAM**

FIRMWARE scheme

SAM is standalone

Boot **binary in boot memory**
Clocks, low level and DRAM init

Typ.: SPI, QSPI, SDcard, eMMC, NANDFlash

Code execution from **boot memory**

Clocks, low level & memory devices Are initialized

Application **binary in mass memory**

Typ.: QSPI, SDcard, eMMC, NANDFlash

Relocation in **SDRAM** memory + jump

Application running from **SDRAM**

Board bring-up SW flow - IDE

- This is the most common scheme when **developing/debugging** software through an IDE.

- Key aspects are:

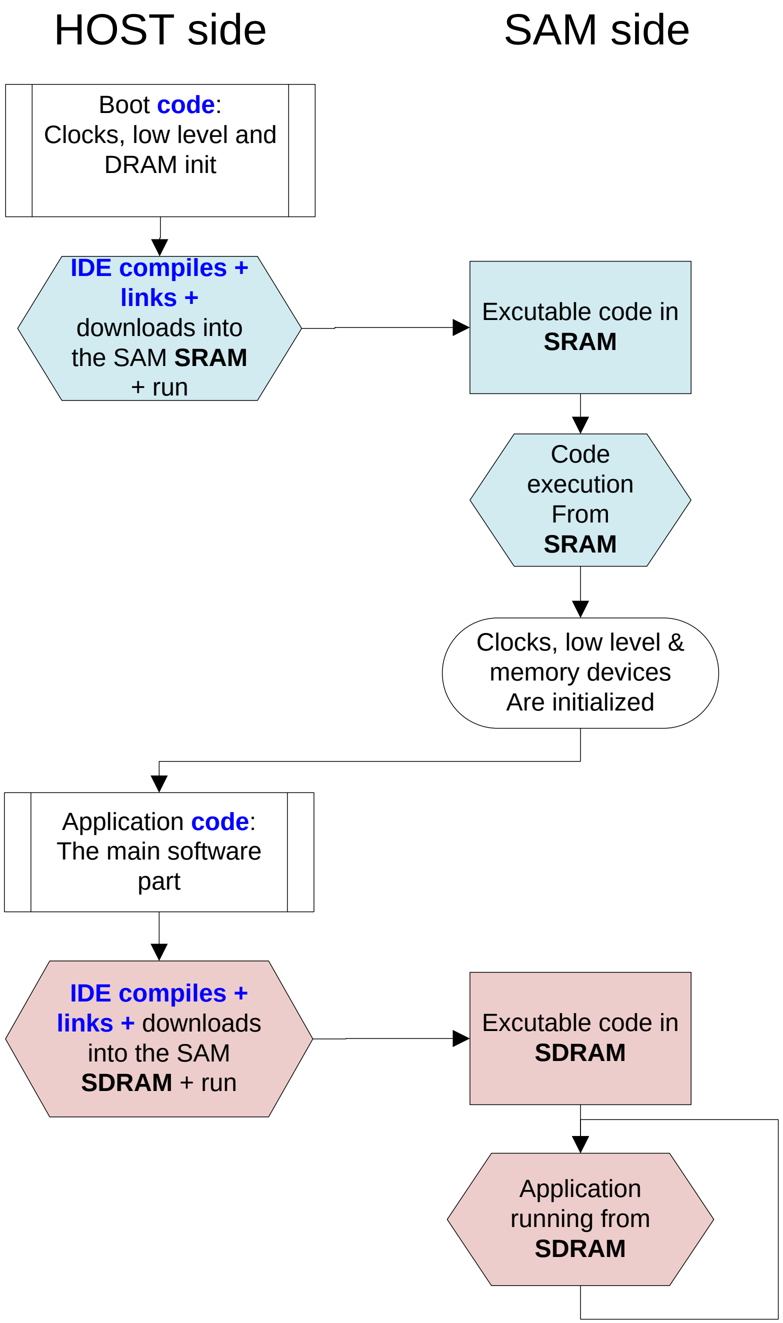
1. It’s the IDE that takes care of uploading + running the different parts of code in the different memory areas

2. The sequence has to be respected:

    1. Download the init code in SRAM and execute it **first**

    2. Then only can you download the main application code into SDRAM and run it there.

IDE debug scheme



HOST side
SAM side
Boot code: Clocks, low level and DRAM init
IDE compiles + links + downloads into the SAM SRAM + run
Excutable code in SRAM
Code execution From SRAM
Clocks, low level & memory devices Are initialized
Application code: The main software part
IDE compiles + links + downloads into the SAM SDRAM + run
Excutable code in SDRAM
Application running from SDRAM

Board bring-up SW flow – SAM-BA

- This is a typical scheme when implementing some usable program as **temporary** setup. E.g.: doing customer demos, automating tests during boards production, etc.

- Key aspects are:

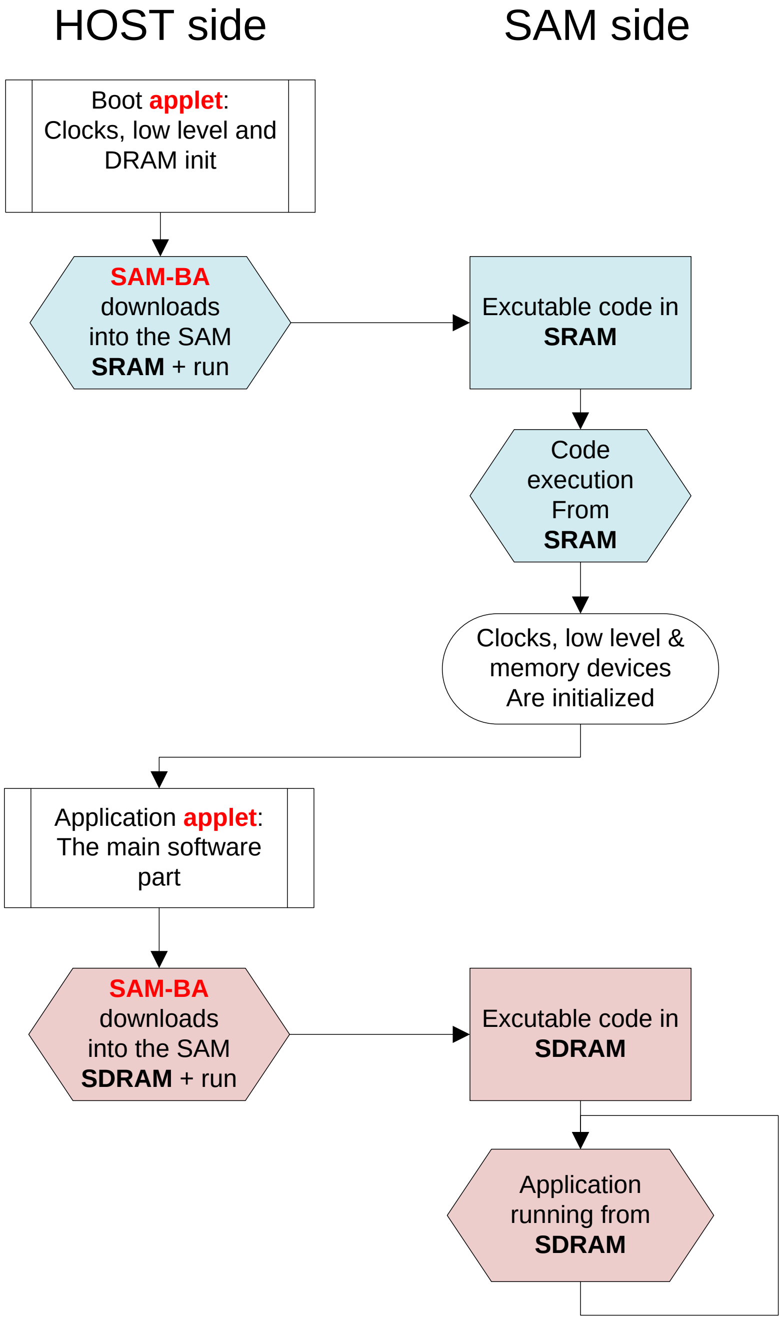
1. It’s SAM-BA that takes care of uploading + running the different parts of code in the different memory areas,

2. That can be scripted/automated,

3. It’s **temporary** (power-off = memory lost),

4. Same requirement as before about sequencing (init first, then can use SDRAM).

SAM-BA debug scheme


HOST side
SAM side
Boot applet: Clocks, low level and DRAM init
SAM-BA downloads into the SAM SRAM + run
Excutable code in SRAM
Code execution From SRAM
Clocks, low level & memory devices Are initialized
Application applet: The main software part
SAM-BA downloads into the SAM SDRAM + run
Excutable code in SDRAM
Application running from SDRAM

Board bring-up FW flow

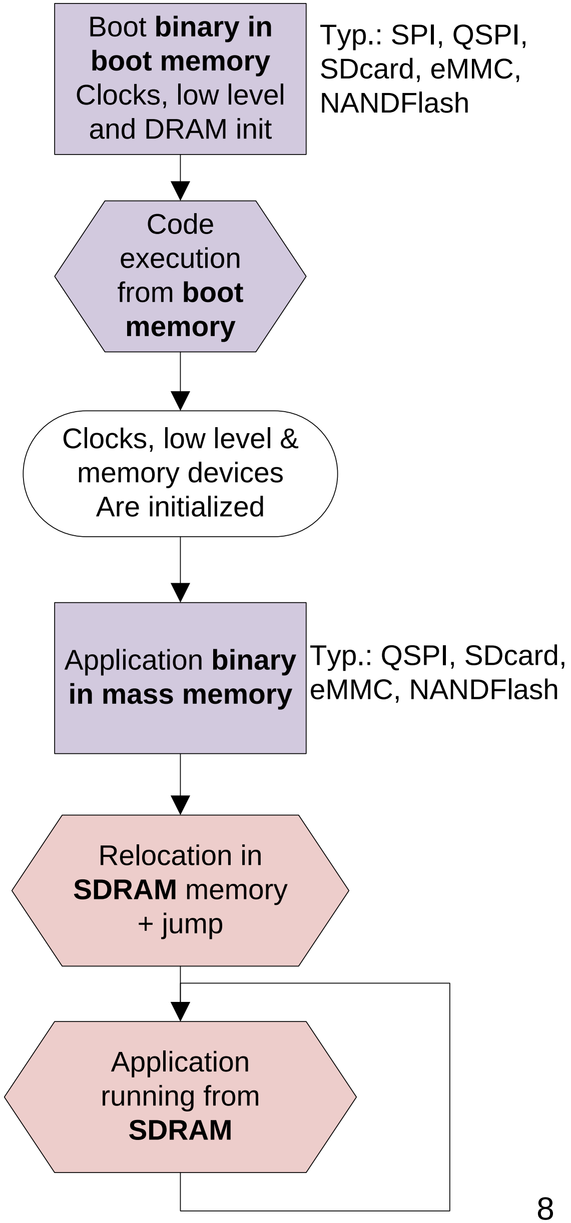
- This is the typical scheme of a **final product**, the code (now called “firmware”) is stored in **permanent memory**.

- Key aspects are:

1. The SAM boots and execute by itself the correct setup + application sequence,
2. It’s **permanent** (power-off = memory preserved),
3. Same requirement as before about sequencing (init first, then can use SDRAM).

FIRMWARE scheme

SAM is standalone

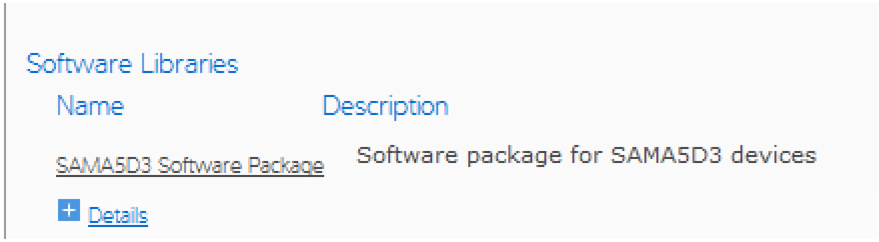

Boot **binary in boot memory** Clocks, low level and DRAM init
Typ.: SPI, QSPI, SDcard, eMMC, NANDFlash
Code execution from **boot memory**
Clocks, low level & memory devices Are initialized
Application **binary in mass memory**
Typ.: QSPI, SDcard, eMMC, NANDFlash
Relocation in **SDRAM** memory + jump
Application running from **SDRAM**

Board bring-up – reference code

NO NEED TO REINVENT THE WHEEL!! ➔ Get bunches of source code from atmel.com : the so-called "Software Package"

Beware that's board-dependent ➔development on a **custom/customer** board = Software Package sources **customization**.

- Go to the concerned SAM **tool** page e.g.
  http://www.atmel.com/devices/ATSAMA5D36.aspx?tab=tools
- Click on the Software Package link i.e.



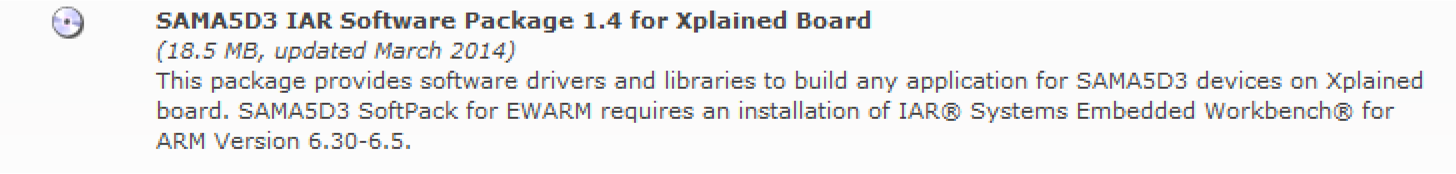
Software Libraries

Name	Description
SAMA5D3 Software Package	Software package for SAMA5D3 devices

⊞ Details

http://www.atmel.com/tools/SAMA5D3SOFTWAREPACKAGE.aspx

- Download the one that suits your SW development environment



SAMA5D3 IAR Software Package 1.4 for Xplained Board
(18.5 MB, updated March 2014)
This package provides software drivers and libraries to build any application for SAMA5D3 devices on Xplained board. SAMA5D3 SoftPack for EWARM requires an installation of IAR® Systems Embedded Workbench® for ARM Version 6.30-6.5.

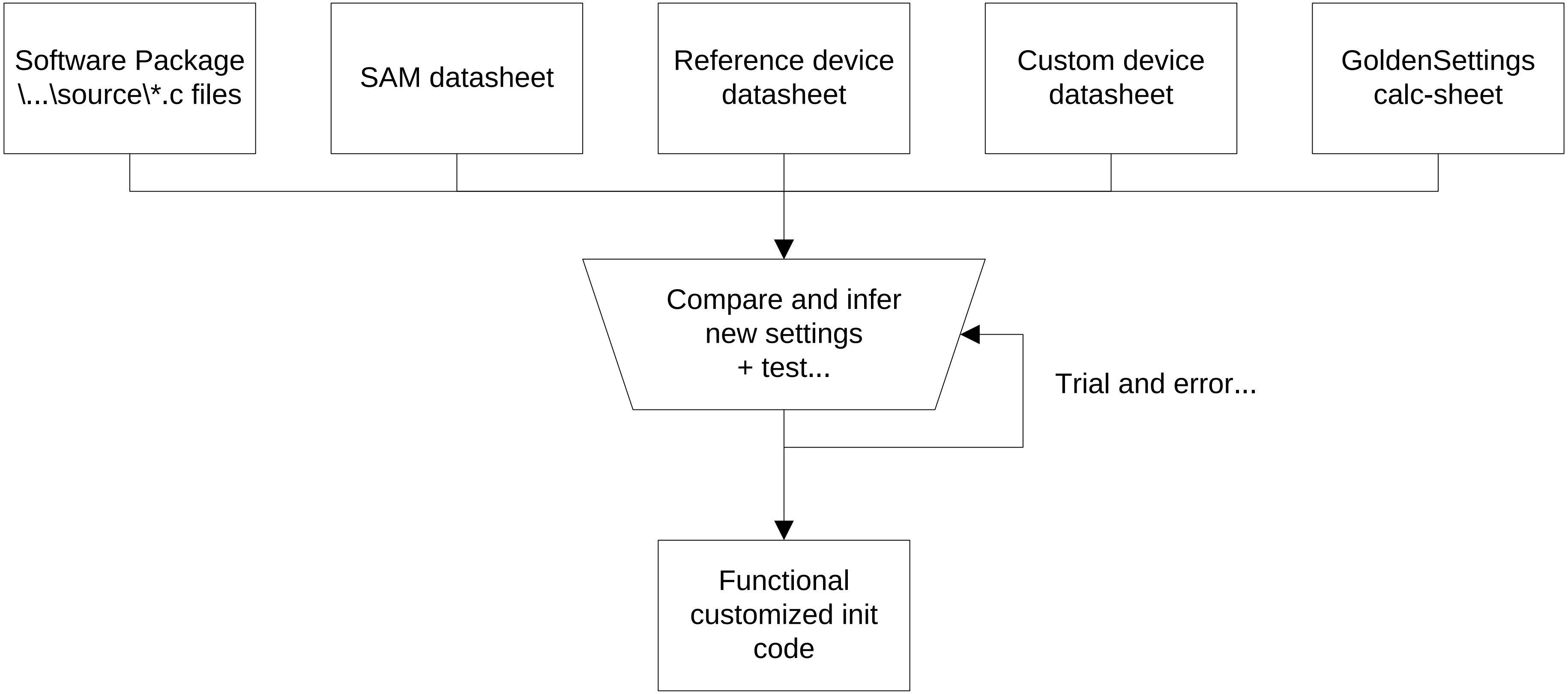
Board bring-up – init code customization for custom devices

- Essential place to go edit is ***\…\libboard_<boardname>\source*** directory
  In particular:
  - ***board_lowlevel.c*** which contains the low level initialization (clocks, interrupts)
  - ***board_memories.c*** which sets up the board memories

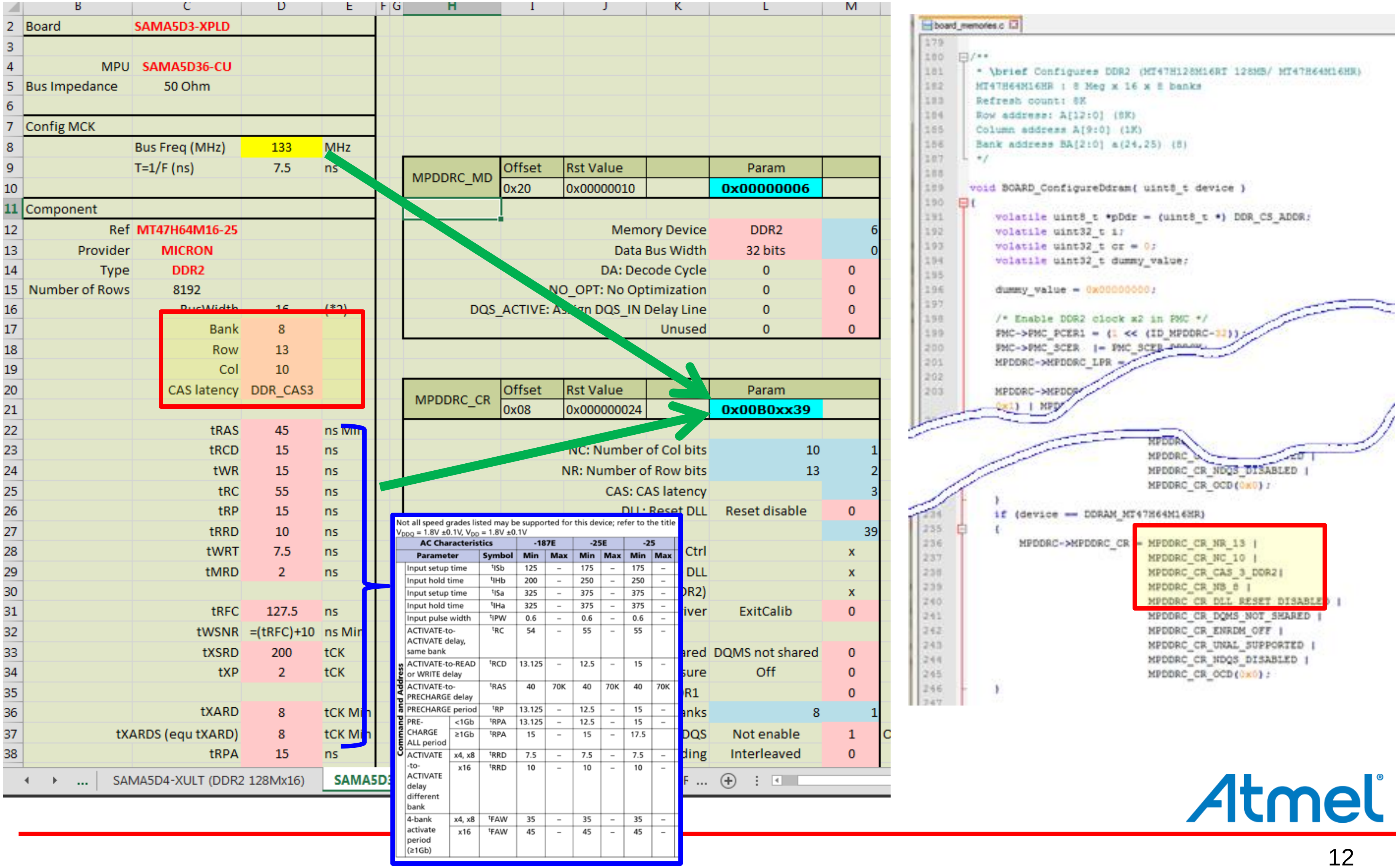
- The parameters customization process supposes to:
  - **Understand** what the current **Software Package** code does…
  - …versus the registers usage advised by the SAM **datasheet**…
  - …versus the current parameters for device access advised by its **datasheet**,
  - Then tune the code to adapt to <u>new</u> parameters of <u>new</u> devices.

- A calculation sheet named "Golden Settings" (available upon request) is here to ease DDR parameters determination (but it does not dispense from reading + understanding the datasheets ;)).

Board bring-up – init code customization for custom devices


Software Package \...\source*.c files
SAM datasheet
Reference device datasheet
Custom device datasheet
GoldenSettings calc-sheet
Compare and infer new settings + test...
Trial and error...
Functional customized init code

Golden Settings sheet vs. board-memories.c vs. DDR2 datasheet



	B	C	D	E
2	Board	SAMA5D3-XPLD		
3				
4	MPU	SAMA5D36-CU		
5	Bus Impedance	50 Ohm		
6				
7	Config MCK			
8		Bus Freq (MHz)	133	MHz
9		T=1/F (ns)	7.5	ns
10				
11	Component			
12	Ref	MT47H64M16-25		
13	Provider	MICRON		
14	Type	DDR2		
15	Number of Rows	8192		
16		BusWidth	16	(*2)
17		Bank	8	
18		Row	13	
19		Col	10	
20		CAS latency	DDR_CAS3	
21				
22		tRAS	45	ns Min
23		tRCD	15	ns
24		tWR	15	ns
25		tRC	55	ns
26		tRP	15	ns
27		tRRD	10	ns
28		tWRT	7.5	ns
29		tMRD	2	ns
30				
31		tRFC	127.5	ns
32		tWSNR	=(tRFC)+10	ns Min
33		tXSRD	200	tCK
34		tXP	2	tCK
35				
36		tXARD	8	tCK Min
37		tXARDS (equ tXARD)	8	tCK Min
38		tRPA	15	ns

MPDDRC_MD	Offset	Rst Value		Param	
	0x20	0x00000010		0x00000006	
			Memory Device	DDR2	6
			Data Bus Width	32 bits	0
			DA: Decode Cycle	0	0
			NO_OPT: No Optimization	0	0
			DQS_ACTIVE: Assign DQS_IN Delay Line	0	0
			Unused	0	0

MPDDRC_CR	Offset	Rst Value		Param	
	0x08	0x000000024		0x00B0xx39	
			NC: Number of Col bits	10	1
			NR: Number of Row bits	13	2
			CAS: CAS latency		3
			DLL: Reset DLL	Reset disable	0
					39
			Ctrl		x
			DLL		x
			DR2)		x
			iver	ExitCalib	0
			ared	DQMS not shared	0
			sure	Off	0
			R1		0
			anks	8	1
			DQS	Not enable	1
			ding	Interleaved	0

SAMA5D4-XULT (DDR2 128Mx16) | SAMA5D3

Not all speed grades listed may be supported for this device; refer to the title
V_{DDQ} = 1.8V ±0.1V, V_{DD} = 1.8V ±0.1V

AC Characteristics				-187E		-25E		-25	
	Parameter		Symbol	Min	Max	Min	Max	Min	Max
Command and Address	Input setup time		tISb	125	–	175	–	175	–
	Input hold time		tIHb	200	–	250	–	250	–
	Input setup time		tISa	325	–	375	–	375	–
	Input hold time		tIHa	325	–	375	–	375	–
	Input pulse width		tIPW	0.6	–	0.6	–	0.6	–
	ACTIVATE-to-ACTIVATE delay, same bank		tRC	54	–	55	–	55	–
	ACTIVATE-to-READ or WRITE delay		tRCD	13.125	–	12.5	–	15	–
	ACTIVATE-to-PRECHARGE delay		tRAS	40	70K	40	70K	40	70K
	PRECHARGE period		tRP	13.125	–	12.5	–	15	–
	PRE-CHARGE ALL period	<1Gb	tRPA	13.125	–	12.5	–	15	–
		≥1Gb	tRPA	15	–	15	–	17.5	
	ACTIVATE-to-ACTIVATE delay different bank	x4, x8	tRRD	7.5	–	7.5	–	7.5	–
		x16	tRRD	10	–	10	–	10	–
	4-bank activate period (≥1Gb)	x4, x8	tFAW	35	–	35	–	35	–
		x16	tFAW	45	–	45	–	45	–

board_memories.c

```
/**
 * \brief Configures DDR2 (MT47H128M16RT 128MB/ MT47H64M16HR)
 MT47H64M16HR : 8 Meg x 16 x 8 banks
 Refresh count: 8K
 Row address: A[12:0] (8K)
 Column address A[9:0] (1K)
 Bank address BA[2:0] a(24,25) (8)
 */

void BOARD_ConfigureDdram( uint8_t device )
{
    volatile uint8_t *pDdr = (uint8_t *) DDR_CS_ADDR;
    volatile uint32_t i;
    volatile uint32_t cr = 0;
    volatile uint32_t dummy_value;

    dummy_value = 0x00000000;

    /* Enable DDR2 clock x2 in PMC */
    PMC->PMC_PCER1 = (1 << (ID_MPDDRC-32));
    PMC->PMC_SCER  |= PMC_SCER_DDRCK;
    MPDDRC->MPDDRC_LPR =

    MPDDRC->MPDDR
    0x1) | MPD

                                   MPDDR
                                   MPDDRC_C                ED |
                                   MPDDRC_CR_NDQS_DISABLED |
                                   MPDDRC_CR_OCD(0x0);
    }
    if (device == DDRAM_MT47H64M16HR)
    {
        MPDDRC->MPDDRC_CR = MPDDRC_CR_NR_13 |
                            MPDDRC_CR_NC_10 |
                            MPDDRC_CR_CAS_3_DDR2|
                            MPDDRC_CR_NB_8 |
                            MPDDRC_CR_DLL_RESET_DISABLED |
                            MPDDRC_CR_DQMS_NOT_SHARED |
                            MPDDRC_CR_ENRDM_OFF |
                            MPDDRC_CR_UNAL_SUPPORTED |
                            MPDDRC_CR_NDQS_DISABLED |
                            MPDDRC_CR_OCD(0x0);
    }
```

Usual SW implementation pitfalls

- **System parameters settings** is the compulsory 1st step to execute
  - PLL & clock settings
  - Enabling of the clocks of to be used peripherals

- 2nd step is **configuring the external SDRAM** memories if needed to store and run the application code

- Registers contents matter, but **settings sequence <u>may</u> matter too**
  - => e.g. a specific configuration order is mandatory for DDR settings

- System **clocks** (core and bus) **speed** of course have an **impact on settings**
  - => especially true for DDR memory settings
  - => the GoldenSettings calc-sheet is precisely there to help resolving the *setting=f(CLK)* equation

- Beware of boot devices sequence: some **higher priority device** (than the one you intend to use) **may contain bootable code already**, which sure won't behave as you wanted!

Summary

- Board startup is a twofold operation:

  1. Boot & configure system resources
  2. (relocate and) jump to main application

- Be aware of booting device sequence vs. priority vs. inherited contents.
- The Software Package is your one-stop source of reference settings (source code), it contains ready to use initialization code and driver examples.
- Customizing parameters is a tough job, understanding the datasheets is indispensable.

MPU Software Packages links collection

- Search link -> here

- http://www.atmel.com/tools/SAMA5D3SOFTWAREPACKAGE.aspx
- http://www.atmel.com/tools/sama5d4softwarepackage.aspx
- http://www.atmel.com/tools/sama5d2-software-package.aspx
- http://www.atmel.com/tools/at91samsoftwarepackage.aspx
- http://www.atmel.com/tools/sam9g15softwarepackage.aspx
- http://www.atmel.com/tools/sam9m10-g45softwarepackage.aspx
- Etc etc etc…

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