nohau
CORPORATION

EMUL-ARM

Board Support: Atmel EB55

February 25, 2004

Contents

About This Guide

EMUL–ARM is a PC-based hardware debugger for the ARM™ Core (currently ARM7 and ARM9 cores). Seehau is the name of the user interface of EMUL-ARM – Seehau and EMUL-ARM is often used interchangeably.

This User's Guide helps you to understand how to use the Atmel EB55 evaluation board with:

- EMUL-ARM
- HI-TECH ARM-C and HI-TIDE
- IAR EWARM C/C++

Note that there might exist two versions of this document (and other BSP/Target Board Related documents):

- If it is located in the "SeehauARM\Documents" install directory then it relates to currently installed Seehau.
- If it is located in the "C:\Nohau\BSP_ARM\Doc" directory then it relates to the currently installed BSP.

Please note that it is required to install BSPs to "C:\Nohau\BSP_ARM" because compiler project files are often dependent on file location.

1 INTRODUCTION

Important Notes

The target board comes with a manual – please read it. This document is intended to complement the board documentation – not replace it.

Atmel has some FAQs on their web site with information – currently at:

- http://www.atmel.com/dyn/products/app_notes.asp?family_id=605&part_id=1991

It is assumed and required that the BSP is installed to c:\Nohau\BSP_ARM.

Memory Configuration – EB55 (AT91R55807)

Internal memory:

- SRAM: 8 KB. Address: 30_0000 - 30_1FFF **Before remap**
- SRAM: 8 KB. Address: 0 - 1FFF **After remap, EBI_RCR register**

External memory:

- SRAM: 128 KB. Address: 20_0000 - 11_FFFF
- Flash: 128 KB. Address: 0 – 1_FFFF **Before remap**
- Flash: 128 KB. Address: 100_0000 – 101_FFFF **After remap, chip selects**

Chip selects to remap external Flash:

- EBI_CSR1 = 01002535
- EBI_CSR2 = 02002121

Remap the internal RAM

The ARM vectors (Reset, Interrupt etc) are mapped from address 0x0 to address 0x20. In order to allow these vectors to be redefined dynamically by the software, the AT91R55 Microcontroller use a remap command that enables switching between the boot memory and the internal primary SRAM bank addresses. The remap command is accessible through the EBI User Interface, by writing one in RCB of EBI_RCR (Remap Control Register). Performing a remap command is mandatory if access to the other external devices (connected to chip selects 1 to 7) is required. The remap operation can only be changed back by an internal reset or an NRST assertion.

Seehau supports this remapping (enabled by default). Change by menu:

- Config | Emulator – then use the Misc Tab.

Flash Programming with EMUL-ARM

Seehau has built in support for flash programming for Atmel EB55. However it is disabled by default. When flash programming is enabled, normal "load" causes flash to erased and written as needed. To enable, use menu:

- Config | Emulator – then use the Misc Tab, and uncheck "Flash Written as RAM".

Seehau uses a "target definition file" to define which the flash memory is, and where it is located. It is automatically setup if you configured for Atmel EB55 during the initial configuration. It can be configured later by selecting a file on the Misc tab:

- Atmel_EB55.tdf – assumes flash at address 0x1000000.
- Atmel_EB55_0.tdf – assumes flash at address 0.

There are two alternatives for flash programming – flash at address 0x0 or address 0x1000000. Each has its own target definition file:

- Address 0x1000000 (default) – Atmel_EB55.tdf
- Address 0 – Atmel_EB55_0.tdf

Flash programming is disabled by default, enable by using menu "Config | Emulator" – Misc Tab:

- Un-check "Flash Written as RAM".

Also, "Monitor Load" is required, enable by using menu "Config | Emulator" – Misc Tab:

- Check "Monitor Load".

Clock Speed

The EB55 starts at low frequency – 32KHz. Unless the flash is erased after it came from Atmel, it will be brought up to full speed by the boot-strap in the flash. If not, you have to take care of it yourselves, or use the supplied macros.

The TimerFlash example has startup code that does the speed up.

AIC Protect Mode

The AIC (Advanced Interrupt Controller) has a Protect Mode – from the manual:

- The Protect Mode permits reading of the Interrupt Vector Register without performing the associated automatic operations. This is necessary when working with a debug system.

Please note that the debugger does nothing to enable Protect Mode automatically. See the MCU Manual.

Board Configuration

JP1 should be “User Position”.

Other Documents

Please also see following documents:

- **ARM_BSP.pdf** – overview of general source code that is supplied with EMUL-ARM for Atmel EB55A – see ARM_BSP.pdf. This document discusses Timer, different Target Console applications and uC/OS-II.
- **ARM_Compilers.pdf** – how to set up different compilers to be used with EMUL-ARM.
- **Nohau_Monitor .pdf –** using the Nohau Monitor to get printf() from target application in Seehau etc.

Target Board CD

The Atmel EB55 Target Board comes with a CD from Atmel.

2 DEVELOPING SOFTWARE

Include Files

The compiler will generally have #include files for MCU registers. If yours doesn't have it, you can use following file (where the initial N is used to denote Nohau and avoiding name conflicts):

- c:\Nohau\BSP_ARM\Src\Inc\ **NAT91M55800A.h**

All examples in Nohau BSP ARM are based on a couple of .h files. These are:

- **NConfig.h** (located in the application directory)
- c:\Nohau\BSP_ARM\Src\Inc**NDefs.h**
- c:\Nohau\BSP_ARM\Src\Inc**NArm.h** (there is an NArm.c file aswell)

The Nohau Monitor uses an additional .h file in the application directory, and so does uC/OS-II:

- **NMon_Cfg.h** – Nohau Monitor.

Startup Code

Startup code for EB55 is located in:

- c:\Nohau\BSP_ARM\Boards\Atmel\EB55\Startup:

IAR

For IAR C/C++, there are two files with startup code in directory

- **cstartup_IAR.s –** assumes program memory (flash or ram) at address zero.
- **cstartup_Remap_IAR.s –** assumes that the application shall execute at address 0x1000000, but start at 0x0 after reset, for which reason it performs the remap.

HI-TECH

For HI-TECH ARM-C, there are three files with startup code in directory

- **stacks_HTC.c** – sets up stacks for selected modes.
- **vectors_HTC.as** – sets up the interrupt vectors.
- **powerup_Remap_HTC.as** – startup code that speeds up CPU, sets up memory, copies vectors and performs remap.

Memory Map / Linking

HI-TECH ARM-C does not use link control files. Memory ranges are defined within the HI-TIDE project.

For the IAR compiler, link control files are provided in directory:

- c:\Nohau\BSP_ARM\Boards\Atmel\EB40\Link\

Following files are available:

- **RAM_IAR.xcl** – code and data in RAM at address 0x0.
- **FLASH_IAR.xcl** – code in Flash at address 0x0, RAM at address 0x30000.
- **FLASH_REMAP_IAR.xcl** – code in Flash at address 0x1000000, RAM at address 0x0.

3 EXAMPLE APPLICATIONS

General Examples

These are applications that are shared between most supported boards. See ARM_BSP.pdf for additional information.

The application categories are available:

- **TargetConsole** - Shows the capabilities of the EMUL-ARM debugger-target communication.
- **Interrupt** – Shows how to do basic interrupts with and without the debugger target-communication.
- **uCOSII** – Shows the uC/OS-II RTOS.

Notice the Interrupt/TimerFlash example, which shows how to create a program that "remaps" and speed up the hardware.

4 MACROS

The AT91R55807 CPU will start running at clock frequency 32KHz by default. This causes problems for the debugger – JTAG debuggers are sensitive to **low** clock speed. The macros below can be used to initialize the board, including programming the clock speed.

Following macros are available in directory: C:\Nohau\BSP_ARM\Boards\Atmel\EB55\Macros:

- **Speedup.bas** – sets MCU to run at full speed.
- **Remap.bas** – remaps memories to default locations.
- **EnableExSRAM.bas** – enables the external RAM.
- **Boardinit.bas** – Initializes the board – including remap and MCU speedup.