StrongARM™ SA-1100 Development Board Firmware Kit

User's Guide

November 1998

Order Number: 278238-001

Contents

Figures

Tables

Introduction 1

This document describes the contents of the firmware kit available for the SA-1100 development board (DE-1S110-OA). The chapters of this document are arranged as follows:

Chapter 2, “Firmware Kit Contents”, describes the SA-1100 development board-specific software, the uHAL, and the StrongARM™ applications.

Chapter 3, “Upgrading to the Latest Angel* Debug Monitor”, describes how to determine which version of the Angel Debug Monitor is on your system and how to upgrade to the latest version.

Chapter 4, “The Flash Management Utility”, describes where the flash components are located and lists the FMU commands.

Chapter 5, “Booting an Image From Flash”, provides an example of how to connect to the Angel Debug Monitor, and how to upload and run images.

Firmware Kit Contents 2

The firmware kit for the StrongARM™ SA-1100 development board (DE-1S110-OA) is comprised of the following software:

Table 2-1. SA-1100 Development Board Firmware Kit Software

Software:	Description	Available From:
Angel* board-specific software, including the Flash management utility (FMU.axf).	Remote debug monitor software, which allows images to be uploaded and debugged, that has been customized for the SA-1100 development board. The Flash management utility allows the user to program any binary format file into any flash location.	The Intel website for developers at: http://developer.intel.com
uHAL	Application layer that runs on several StrongARM designs.	
SA-1100 development board-specific applications and a README.TXT file.	Applications that will only run on the SA-1100 development board. The README.TXT file provides configuration information for SA-1100 development boards.	
Angel sources	Remote debug monitor source files to upload images onto the board and debug them.	The ARM™ website at: http://www.arm.com

2.1 SA-1100 Development Board-Specific Software

The software images and source files provided in this kit are board-specific software for SA-1100 development boards. For more information about the contents of this firmware kit and how to use it, see the README.TXT file supplied with the kit.

2.1.1 Angel*

Angel is a remote debug monitor which communicates with a debug host running the ARM Software Development Toolkit (SDT) via the Angel Debug Protocol (ADP). Angel, which is stored in components U35 and U45, allows the developer to upload images onto the board and to debug them.

Note: If you are not using version 1.05 or later of Angel, you **must** upgrade. For information about upgrading, see Chapter 3.

The firmware image itself contains both a bootloader and the Angel firmware. The bootloader will run Angel, but images held in the U34 and U44 flash parts can also be run. The bootloader uses S21 and S22 (GPIO pins 1 and 0 respectively) to determine which image to run.

To program images into these flash parts, use the Flash Management Utility described in Chapter 4.

2.1.2 Demon*

Demon is ARM's earlier version of the remote debug monitor and is still supported by version 2.11 of the ARM SDT for upgrades.

Note: If you are using any version of Demon, you **must** upgrade. For information about upgrading, see Chapter 3.

2.1.3 Flash Management Utility (FMU)

The SA-1100 development board contains programmable flash parts that can store stand-alone images that can be run at boot time.

The Flash Management Utility allows the developer to manage those images in flash. See Section 4 for a full description of the Flash Management Utility.

2.2 uHAL

The uHAL is an application layer that runs on several StrongARM evaluation boards. Version 1.0 and later versions fully support the SA-1100 development board. The uHAL allows the developer to write applications (for example, benchmarks) that can be loaded and run via the Angel Debug Monitor or be placed into flash and run at boot time if selected. In addition, several operating systems use uHAL, making it easier to port applications built using those operating systems between StrongARM evaluation boards.

For more information on building and using uHAL, see the documentation supplied with the uHAL software.

2.3 StrongARM™ Applications

The StrongARM applications are built using the uHAL libraries. For more information about StrongARM applications, see the documentation supplied with the StrongARM applications software.

Upgrading to the Latest Angel* Debug Monitor 3

To determine which version of Angel* you are running, connect a serial line to J23 of the SA-1100 development board and to a terminal emulator. Configure the terminal emulator as described in the README.TXT file.

Reset or power on the microprocessor evaluation board and observe the output on the terminal emulator. If the output indicates an Angel version earlier than version 1.05, you must upgrade. The following output indicates that version 1.05 of the Angel Debug Monitor is running on your board:

```
Angel Debug Monitor for Brutus (FIQ), MMU on, Caches enabled, Clock Switching on (serial)
1.05 (Advanced RISC Machines SDT 2.11a) rebuilt on Aug 28 1998 at 12:01.42
Serial Rate 11520
```

3.1 Upgrading From an Earlier Version of Angel*

The Angel and bootloader image runs from flash parts U35 and U45, while stand-alone images (for example a uHAL stand-alone image) runs from flash parts U34 and U44.

Parts U35 and U45 are not programmable in place and are known as ROM parts. Parts U34 and U44 can be programmed via the Flash Management Utility (FMU). The FMU is a program that is loaded and run via the Angel Debug Monitor. The FMU verifies that the image being programmed has a valid image number and that it is being programmed at an address it can run from. This mode allows the image to boot from the bootloader at reset time.

To use the FMU for programming flash parts U34 and U44 with an image that will be run from U35 and U45, the ROM mode must be selected. In this special mode, the FMU ignores U35 and U45 and treats U34 and U44 as if they were the ROM parts. This mode is used when upgrading to a new Angel and bootloader image.

Note: The list option may not function correctly if parts in the flash sockets have been programmed in ROM mode. Also, deleting blocks will require the FMU to be in the same mode as when the programming of the parts took place.

3.1.1 Upgrading From an Earlier Version of Angel*

Use the following procedure to upgrade from an earlier version of Angel to a new Angel image.

1. Build an Angel/bootloader image targeted for address 0x40000c0, or use a new Angel image (bootld.axf) supplied in a kit.
2. Boot the current Angel image and connect to it using an appropriate version of the ARM™ SDT (Version 2.11a or later).
3. Load and run the Flash Management Utility (fmu.axf).

4. Type 'listblock'. This should show the usage of the eight blocks. If there are images in the upper four blocks, these images may need to be erased to make room for the image. If the images were programmed in ROM mode, then the images will need to be erased in ROM mode.
5. Type 'rom' to enter ROM mode.
6. Verify that the listblock displays only blocks 0-3, which are physically the same as blocks 4-7 in normal mode.
7. Program the image into flash using the following command:
   `'program 0 ImageName Filename 0'`
   (Program into flash the image in file 'FileName', label it as 'ImageName', and execute it when the switch setting indicates image '0' (the first zero) and then put it into block 0.)
8. When asked about overriding the bootloader to relocate the image, answer 'y' for yes. Relocating the image allows it to run directly from flash. The entry point will be correct when the flash parts have been swapped over.
9. Verify that the list indicates that the program is in block 0.
10. Power off and swap the parts U34 and 44 into U35 and U45. If the GPIO switches are set to image 0, then the board should boot from the image programmed.

3.1.2 An Example of Upgrading to a Newer Version Of Angel*

The following text is a screen capture from starting the ARM™ source-level debugger:

```
$armsd -remote -adp -port 2 -line 115200
.
.
.
armsd: load fmu.axf
armsd: go
Brutus Flash Management Utility [v0.1] (Angel)
Entering SVC mode...Done
Disabling caches...Done
Searching for flash device
Flash found at 0x08000000 (8 blocks of size 0x10000)
Scanning Flash blocks for usage
FMU> listblock
 0: (Unused)   0xe1a00000 0xe1a00000 0xe1a00000 0xe1a00000 0xe59f42dc
 1: (Unused)   0x00000c1e 0x00000c1e 0x00000c1e 0x00000c1e 0x00000c1e
 2: (Unused)   0xe1a0c00d 0xe10f0000 0xe200001f 0xe3500010 0x0a00000a
 3: (Unused)   0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
 4: (Unused)   0xffffffff 0xffffffff 0xffffffff 0xffffffff 0xffffffff
 5: (Unused)   0xffffffff 0xffffffff 0xffffffff 0xffffffff 0xffffffff
 6: (Unused)   0xffffffff 0xffffffff 0xffffffff 0xffffffff 0xffffffff
 7: (Unused)   0xffffffff 0xffffffff 0xffffffff 0xffffffff 0xffffffff
FMU> rom
Scanning Flash blocks for usage
FMU> listblock
 0: (Unused)   0xffffffff 0xffffffff 0xffffffff 0xffffffff 0xffffffff
 1: (Unused)   0xffffffff 0xffffffff 0xffffffff 0xffffffff 0xffffffff
 2: (Unused)   0xffffffff 0xffffffff 0xffffffff 0xffffffff 0xffffffff
 3: (Unused)   0xffffffff 0xffffffff 0xffffffff 0xffffffff 0xffffffff
FMU> program 0 angel bootld.axf 0
```

```
Caching the file in memory
Attempting to get aif header
Writing bootld.axf into flash block 0
Deleting blocks ready to program:
Deleting block 0
Delete flash block at offset 0x0 words
Attempting Flash Sync
Sync'd flash.
Calculating checksum
Writing Flash header at 0x0
Writing flash image header
Image is non-executable AIF file
The bootloader will copy this image to 40000c0 before executing it Eprom is from
    0x8000000 to 0x803ffff
Do you want to override this (Y/n)? y
padding is 0 bytes
Writing image file
################################################################################
Attempting Flash Sync
Sync'd flash.
Scanning Flash blocks for usage
FMU> list
Listing images in Flash
Image 0 "angel" Length 50296 bytes, Map 0x00000001
FMU> quit
```

3.2 Upgrading from Demon*

This section provides step by step instructions for upgrading from Demon* to Angel. Before upgrading you will need:

- The ARM debugger—an application to connect to the Demon Debug Monitor.
- The upgrader.axf image—an application used to program the flash parts.
- The latest Angel/bootloader image—an image contained in the bootld.bin binary image file.

Use the following procedure for upgrading to Angel:

1. Insert the flash parts to be programmed with Angel into sockets U34 and U44 on the SA-1100 development board.
2. Power on the SA-1100 development board and connect to the Demon Debug Monitor using an appropriate version of the ARM SDT, for example version 2.11a or later.
3. Load and run the upgraded image (upgrader.axf). A menu will be displayed.
4. Select option 1, to load an image into the buffer, and supply the path to the bootld.bin file. Load the file at offset zero (you will be prompted to supply this offset after the return key is pressed).
5. Wait while the file downloads—this may take a few minutes. Once the download is complete the image may be copied to flash by selecting option 2 in the menu.
6. After successfully downloading the image, power off the SA-1100 development board and swap the components in sockets U34 and U44 with the components in sockets U35 and U45 respectively. When the SA-1100 development board is powered up, it will run the supplied Angel image, provided that GPIO 0 and 1 are both in the dot (off) position.

3.2.1 An Example of Upgrading From Demon* to Angel*

This section provides an example of how to use the upgrade utility for the SA-1100 development board.

```
Main Menu
1 - Load an image into the buffer
2 - Program the buffer into the flash
5 - Exit
Enter: 1
Enter the name of the file: bootld.bin
Enter the offset: 0
File ’bootld.bin’ has been loaded into the buffer at offset 0x00000.
Main Menu
1 - Load an image into the buffer
2 - Program the buffer into the flash
5 - Exit
Enter: 2
The buffer has been successfully programmed into the flash.
Main Menu
1 - Load an image into the buffer
2 - Program the buffer into the flash
5 - Exit
Enter:
```

The Flash Management Utility 4

4.1 Overview

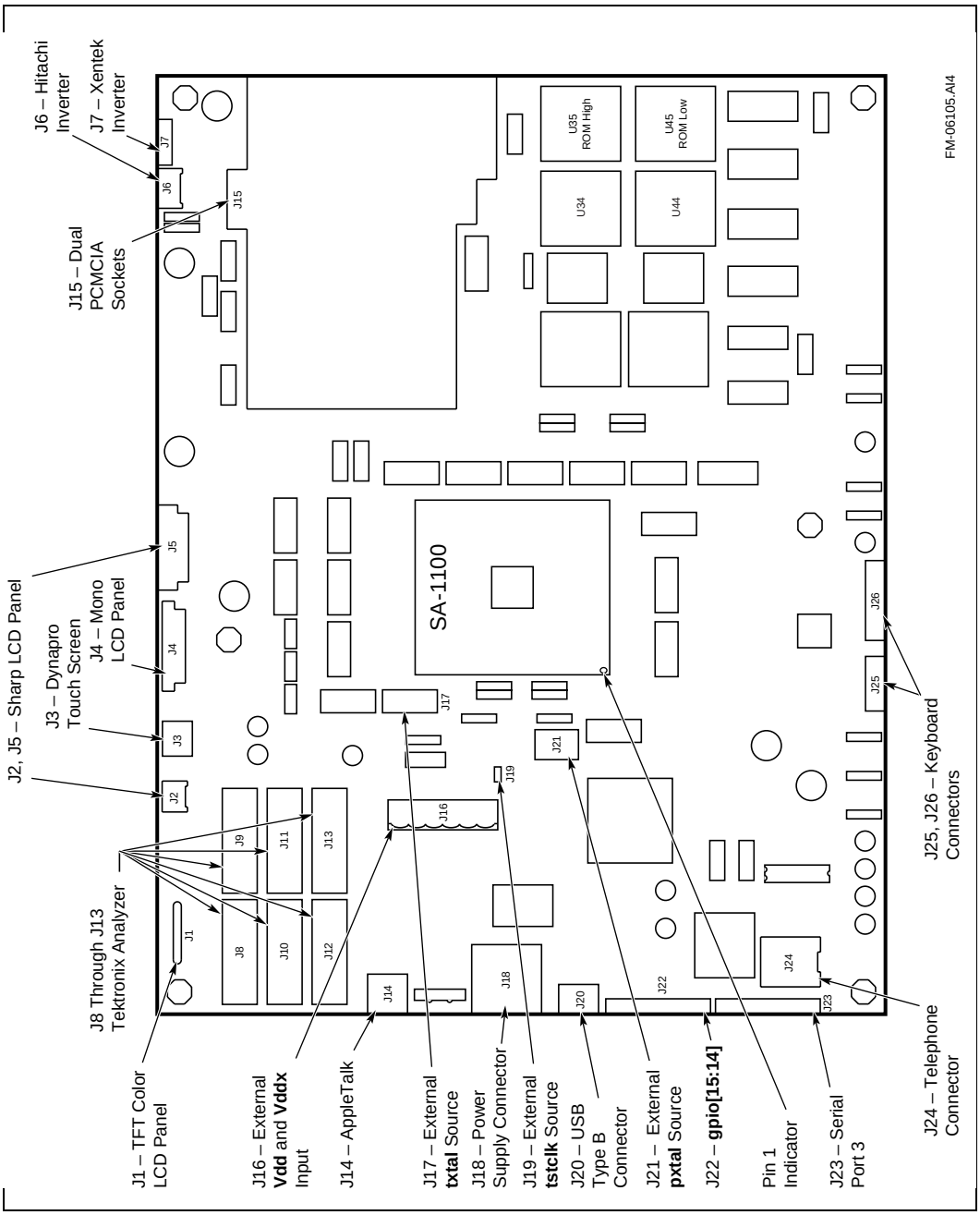
The SA-1100 development board has provisions for flash parts on the board labeled U34, U44, U35 and U45. These sockets are populated with Atmel A29LV1024 parts that are organized as 64Kx16-bits each. The parts are banked in pairs, each of the pair providing half of the 16-bits on the 32-bit wide data bus. Parts U44 and U45 are mapped into data bus lines 0-15 while parts U34 and U35 are mapped onto data lines 16-31.

The non-writable bank of flash, called ROM, is mapped into virtual address 0x1000000 in some early versions of Angel* and at 0x4000000 for versions 1.05 and later. It is mapped to physical address 0x00 when powered on. ROM consists of parts U35 and U45 (see Figure 4-1). The write-able bank, called flash, is mapped at address 0x8000000. Blocks 0-3 are not writable, these are mapped to the ROM, while banks 4-7 are mapped to the flash.

The Flash Management Utility (FMU) runs from the ARM™ Debugger. It keeps a RAM copy of the flash bank to manipulate and keeps track of which blocks are changed, then updates the flash accordingly.

Image files that are programmed into the flash are required to be in ARM Image Format (AIF) if they need to be executed using the GPIO switch selection. This version of the FMU allows the user to program any binary format file into any flash location. The AIF format is described in the *ARM Toolkit Reference Manual*.

Figure 4-1. Flash Component Layout



4.2 FMU Commands

The Flash Management Utility supports the following commands:

Command	Description
List	no parameters List all of the images currently programmed into the flash and ROM
Listblocks	no parameters Lists the contents of the first 4 words of each flash block and the image, if any, associated with that block.
Testblock	<block number> Writes a test pattern into the given flash block in order to verify its functionality. Note: This destroys the block contents
Delete	<image number> Delete the image from the flash.
Deleteblock	<block number> Delete the block. This writes 0xFF to the entire flash block. It will not allow deletion of a block that is part of an image.
Deleteall	<no parameters> Deletes all of the flash blocks.
Program	<Image No> <Name> <Filename> <Block no> [<Noboot>] Programs an image into flash. Image No is the tag that FMU will use to label the image. 'Name' is the label that will appear in the list function. 'Filename' is the full pathname to the AIF file to put into flash. The user may optionally supply a block number into which the image will go, otherwise FMU will find the first empty block and put it there. The image may span multiple blocks. The 'Noboot' option allows the user to specify that image will not execute but will be loaded.
Read	<image number> <filename> Read an image from the flash and save it into a file.
Copyblock	<block from> <block to> Copy from one block to another.
Copyflash	No parameters. Copy the flash part in socket U34 to the flash part in U44
Readflash	<address> <filename> [<length>] Read from flash into a file. The address is a byte offset from the ROM base. That is, addresses 0 to 262140 (0-256K) are in ROM space while 262144-524288 (256-512K) are in programmable flash area. 'length' bytes are read, unless not supplied. In this case the rest of the flash will be read.
Writeflash	<address> <filename> [<length>] Write to flash from a file. That is, addresses 0 to 262140 (0-256K) are in ROM space and hence invalid, while 262144-524288 (256-512K) are the programmable flash area. 'length' bytes will be written or the whole file if it is not supplied.
Rom	Set special configuration mode. When this mode is set, FMU behaves as if the flash parts are the ROM parts - this allows the user to program flash parts destined for the ROM sockets. See notes below.
Flash	Return to normal configuration, used to return to normal after using the ROM command. Intel recommends that you exit and reset the system rather than use this—the FMU will be confused if there are multiple images with the same image number.
Quit	This calls the exit routine which is in fact a SWI call that uses semi hosting to communicate to the host that the program has terminated.

Booting an Image From Flash 5

The uHAL contains example images that can either be loaded and run via the Angel* Debug Monitor (Angel images) or run at boot time (stand-alone images). Please refer to the uHAL documentation and sources for more information on how to build a stand-alone image for the SA-1100 development board. The uHAL contains code that sets up the virtual memory system appropriately. For information about memory, see the *Memory Organization on the SA-1100 Evaluation Platform Application Note* that is supplied with the SA-1100 development board specific software.

Once you have a standalone image, for example ping.axf, a uC/OS application, you will need to program it into flash using the FMU and then select it for running.

The following procedure loads and boots an image from flash:

1. Establish a connection with the Angel Debug Monitor; upload fmu.axf and run it.

```
$ armsd -remote -port 2 -adp -line 115200
.
.
.
armsd: load fmu.axf
armsd: go
Brutus Flash Management Utility [v0.1] (Angel)
Entering SVC mode...Done
Disabling caches...Done
Searching for flash device
Flash found at 0x08000000 (8 blocks of size 0x10000)
Scanning Flash blocks for usage
FMU>
```

2. Program ping.axf into flash as image 2 in flash block 4.

```
FMU> program 2 ping standalone/ping.axf 4
Caching the file in memory
Attempting to get aif header
Writing standalone/ping.axf into flash block 4
Deleting blocks ready to program:
Deleting block 4
Delete flash block at offset 0x0 words
Attempting Flash Sync
Sync'd flash.
Calculating checksum
Writing Flash header at 0x40000
Writing flash image header
Image is non-executable AIF file
Image will execute from Flash
padding is 0 bytes
Writing image file
############################################################
Attempting Flash Sync
Sync'd flash.
Scanning Flash blocks for usage
FMU>
```

3. Select image 2 to run at boot time by setting GPIO 1 to 1 (S21 no dot) and GPIO 0 to 0 (S22 dot).
4. Quit FMU and the Angel Debug Monitor.
5. Reset the SA-1100 development board.

   You will then see the following output on the second serial port (J22) and on the 8 bit LCD panel:

```
TargetInit() called
Setting up the soft vectors
Timer init
TargetInit() complete
Ping.c, calling OSStart()
IRQInstall() called 0x1C
1+[1-2] 1+[1-2] 1+[1-2] 1+[1-2] 1+[1-2]
```

6. Building Angel images

   The SA-1100 development board includes the system specific sources to the Angel Debug Monitor. You will need to combine these sources with those in the ARM™ SDT. The Angel source tree's Angel subdirectory of the ARM SDT contains the system independent Angel sources (for example angel/startrom.s contains the first code executed when Angel is run). Within this directory there are two directories for each system that have had Angel ported to it. For the SA-1100 development board, they are brutus and brutus.b. The brutus directory contains the SA-1100 development board-specific sources and the brutus.b directory contains files used to build (hence the .b suffix) Angel images.

   Starting with a clean Angel source tree in version 2.11a or later of the ARM SDT, you must copy over the two directories brutus/angel/sources/brutus and brutus/angel/sources/brutus.b. You must also replace any non-system specific Angel source files with those from brutus/angel/sources (cdefs.h and startrom.s). If you already have an Angel source tree, for example, to build Angel images for the EBSA-285 StrongARM™ Evaluation Board, you must be careful that the versions of these common files are the same.

   Once you have done that, you can build new Angel images using the GNU make file in angel/brutus.b/gccsunos/makefile or the ARM project file in angel/brutus.b/apm/angelsa.apj. Both of these build the same images:

   angel.axf, angel.srec—These are Angel images that will execute from flash parts U34 and U44. You must use the Flash Management Utility to program the angel.axf into flash. angel.srec can be programmed using an external flash programmer.

   bootld.axf, bootld.srec— These are Angel images that will execute from flash parts U35 and U45.

Support, Products, and Documentation

If you need technical support, a *Product Catalog*, or help deciding which documentation best meets your needs, visit the Intel World Wide Web Internet site:

http://www.intel.com

Copies of documents that have an ordering number and are referenced in this document, or other Intel literature may be obtained by calling **1-800-332-2717** or by visiting Intel's website for developers at:

http://developer.intel.com

You can also contact the Intel Massachusetts Information Line or the Intel Massachusetts Customer Technology Center. Please use the following information lines for support:

For documentation and general information:	
Intel Massachusetts Information Line	
United States:	1–800–332–2717
Outside United States:	1–303-675-2148
Electronic mail address:	techdoc@intel.com

For technical support:	
Intel Massachusetts Customer Technology Center	
Phone (U.S. and international):	1–978–568–7474
Fax:	1–978–568–6698
Electronic mail address:	techsup@intel.com