GM8126

LINUX

User Guide

Rev.: 0.1

Issue Date: January 2011

REVISION HISTORY

Linux User Guide

Date	Rev.	From	To
Jan. 2011	0.1	-	Original

Grain Media, Inc.
5F, No. 5, Li-Hsin Road III, Hsinchu Science Park, Hsinchu City, Taiwan 300, R.O.C.

Grain Media's home page can be found at:
http://www.grain-media.com

TABLE OF CONTENTS

List of Tables

LIST OF FIGURES

Chapter 1

Introduction

This chapter contains the following sections:

1.1 General Description

The GM8126 hardware environment is a highly efficient RISC-based platform used to verify and evaluate the AMBA-based designs at the early stage of development. This sound platform consists of a main board equipped with a GM8126 chip and an embedded FA626TE CPU. This document is the Linux user guide for the GM8126 platform. Please refer to the GM8126 data sheet for further information.

Two Linux versions are involved in programming: **Linux** and **uClinux**. uClinux is a Linux version without the Memory Management Unit (MMU). The RISC FA626TE is an MMU-based CPU, while FA510 is not an MMU-based CPU. Both FA626TE and the GM peripheral IP drivers have been ported to Linux. GM provides the user guides for various IP drivers, in which the installtion and usage of the drivers are described.

For further information regarding the USB device, Watchdog Timer (WDT), Real Time Clock (RTC), security engine, USB OTG, and other latest drivers, please refer to the relevant documents.

1.2 Requirements of Host Development Environment

The required development environments of the host system for developing Linux are as below:

Hardware:

- Intel x86 compatible PC
- Standard 16550 UART

Software:

- Standard Linux distribution (Fedora core 2.6.14-FC5 or above)
- FA626TE-based Linux distribution

1.3 Requirements of Common Platform Target System

The required target system for developing the Linux kernel and device drivers is as below:

- GM8126:
  - 128 MB onboard DDR
  - 8 MB onboard Flash at least

Chapter 2

Linux Distribution Based on FA626TE

This chapter contains the following sections:

2.1 Introduction

This chapter introduces the architecture and implementations of the FA626TE-based Linux, which helps users to understand and install the FA626TE-based Linux easily and quickly.

FA626TE-based Linux implements a Linux 2.6.28 software development environment for the FA626TE processor and peripheral IPs. The information provided in this chapter will help users to promptly install the FA626TE-based Linux on the GM8126 platform to implement the applications.

The following sub-sections include the information regarding system requirements and how to install the FA626TE-Linux distribution.

2.1.1 FA626TE-Linux Distribution

The FA626TE-Linux distribution is a tar archive. An example of the file name is listed as below:
arm-linux-2.6.28.tgz

Table 2-1. Disk Space Requirements for Linux Host

Source Item	Disk Size
GCC toolchain	260 MB
Linux kernel	379 MB
User application	1 MB
Ramdisk/Rootfs-cpio sample	6 MB
Total	646 MB

2.1.2 Install FA626TE-Linux Distribution

The FA626TE-Linux can be installed by extracting the tar archive with the sequence listed below:

1. Copy the file, "**arm-linux-2.6.28.tgz**", the to /usr/src directory

   # cp arm-linux-2.6.28.tgz /usr/src

2. Extract the file, "**arm-linux-2.6.28.tgz**"

   # cd /usr/src

   # tar zxvf arm-linux-2.6.28.tgz

Installing the ARM toolchain:

1. Login as "root user"
2. Copy "arm-none-linux-gnueabi-4.4.0_ARMv5TE.tar.gz" to the root directory of the user PC Linux host system:

   "#cp arm-none-linux-gnueabi-4.4.0_ARMv5TE.tar.gz /"

3. Decompress the toolchain:

   "#tar xvfz arm-none-linux-gnueabi-4.4.0_ARMv5TE.tar.gz"

   into /opt/crosstool/arm-none-linux-gnueabi-4.4.0_ARMv5TE

4. Include the tool chain path into the user path:

   #export PATH=/opt/crosstool/arm-none-linux-gnueabi-4.4.0_ARMv5TE/bin:$PATH

5. Test-run the user toolchain:

   "# arm-none-linux-gnueabi-gcc –version"

If the installation is correct, users will see the following messages:

```
arm-none-linux-gnueabi-gcc (Faraday C/C++ Compiler Release 20100325) 4.4.0
Copyright (C) 2009 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

After completing the installation steps, users can then build the FA626TE-Linux kernel or application.

2.1.3 Directory Structure of FA626TE-Linux

Assume that the top directory is **/usr/src/arm-linux-2.6.28/** when extracting the tar archive, such as "**arm-linux-2.6.28.tgz**", a set of subdirectories will be built on **/usr/src/arm-linux-2.6.28/**.

Table 2-2 lists the source directories and the files located in it. For example, if the user application is located in the **/usr/src/arm-linux-2.6.28/user/** directory, the customized ramdisk will be located in the **/usr/src/arm-linux-2.6.28/target/rootfs-cpio** directory; and the module driver will be located in the **/usr/src/arm-linux-2.6.28/module/** directory.

Table 2-2. FA626TE Distribution Directories

Directory	Description
arm-none-linux-gnueabi-4.4.0_ARMv5TE.tar.gz	Directory for the repository of the arm-Linux toolchain
linux-2.6.28-fa/	Directory for the repository of the Linux kernel source
user/	Directory for the repository of the user application
target/rootfs-cpio	Directory for the repository of the ramdisk images
module/	Directory for the Linux 2.6.28 module
u-boot-2008.10	Directory for the arm boot
nsboot	Directory for the spi/nand boot loader

Note: The toolchain includes GCC4.4.0, GLibc-2.9, and Binutil 2.19

2.2 Building and Developing FA626TE-Linux

This section introduces how to configure and build an FA626TE-Linux kernel for the embedded system.

2.2.1 Kernel Tree

The Linux kernel is located in the following directory:

/usr/src/arm-linux-2.6.28/linux-2.6.28-fa/

The structure of the FA626TE-Linux subdirectory is identical to that of the standard Linux kernel, version 2.6.28. Table 2-3 lists and describes the Linux kernel subdirectory. **<TOPDIR>** stands for the **/usr/src/arm-linux-2.6.28/linux-2.6.28-fa** directory.

Table 2-3. Kernel Subdirectory

Kernel Subdirectory	Description
<TOPDIR>/arch/arm	Architecture-dependent code for the ARM processors
<TOPDIR>/Documentation	Documentations for the Linux kernel
<TOPDIR>/drivers	Device drivers, which are divided into various subdirectories.
<TOPDIR>/fs	Various file systems supported by the Linux kernel
<TOPDIR>/include	Kernel head files
<TOPDIR>/init	Kernel startup functions
<TOPDIR>/ipc	Sources of system V IPC
<TOPDIR>/kernel	Kernel core sources
<TOPDIR>/lib	Standard C library sources
<TOPDIR>/mm	Kernel memory management
<TOPDIR>/net	Implementations of various network protocols

2.2.2 Building Kernel

This section introduces how to build the kernel image for the FA626TE architecture.

2.2.2.1 Configuring Kernel

The first step to build the Linux kernel is to configure the kernel. The configuration file is located in **<TOPDIR>/.config** for the standard Linux.

In general, the purposes of reconfiguring FA626TE-Linux are summarized as below:

- Customize the processor and board functionalities: Modify the UART clock, system clock, and so on.
- Customize the hardware devices: Add or remove a particular device
- Customize the kernel functionalities: Add or remove a kernel feature, such as the network support

Users can copy “linux-2.6.28-fa/arch/arm/configs/GM8126_defconfig” to update “.config” and disable the items that are not needed. This file includes all the necessary items. Users should decide the required function, such as the user partition size in MTD.

There are two ways for Linux kernel to configure the options mentioned above:

- Change to the <TOPDIR> directory
  1. Use the menu display to select the options for configuration:

     # make menuconfig
  2. Use the GUI display to select the options for configuration:

     # make xconfig

Figure 2-1 shows how to use the console to configure the Linux kernel options.

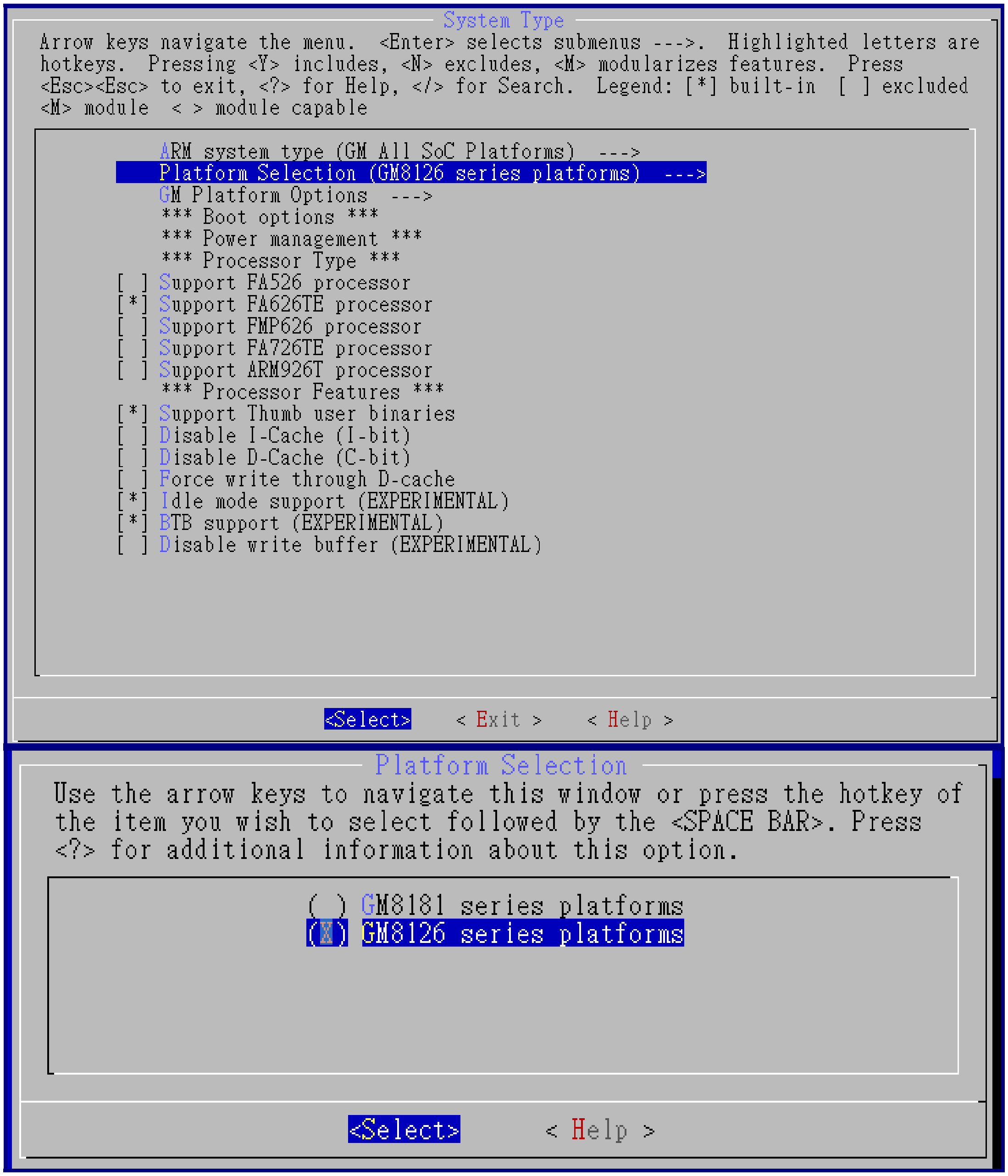


Figure 2-1. Linux Kernel Option Configuration by Using Console

Figure 2-2 shows how to use the menu display to select the configuration options of Linux kernel.

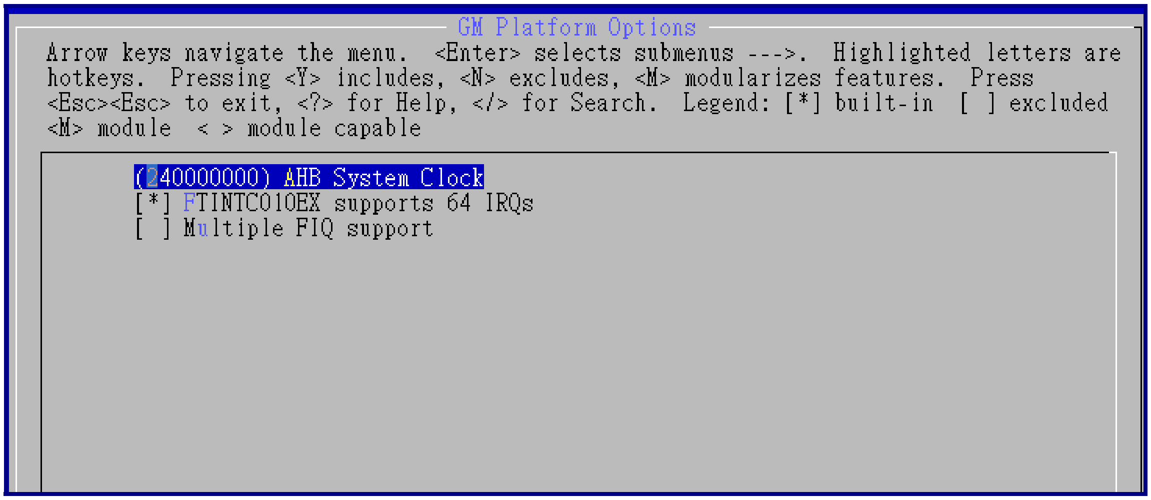
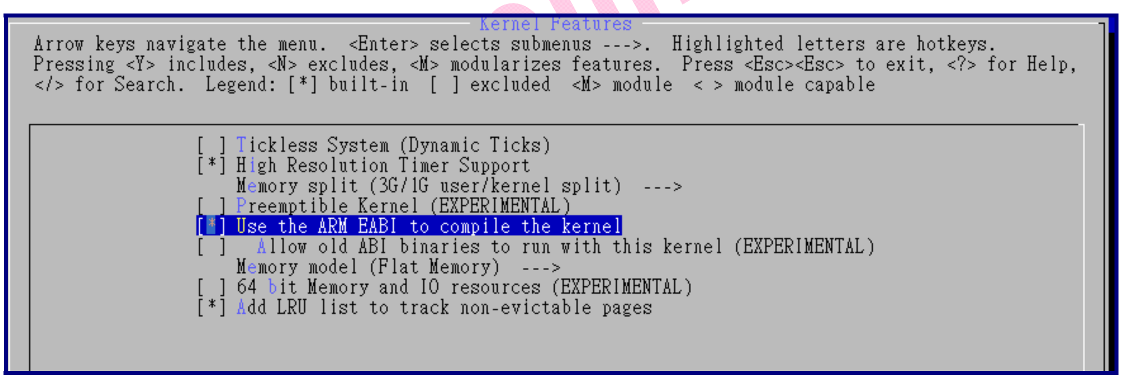


Figure 2-2. Linux Kernel Option Configuration by Using Menu Display

Before compiling the kernel, users need to select the EABI options in the kernel configuration menu ifthe gcc-4.4.0 tool chain is used. The configuration path is "Kernel configuration Menu->Kernel Features->ARM EABI", as shown below.



The main options of the FA626TE system are as follows:

- ARM system type

  Select the CPE board to support CPE

- **[*]GM All Soc Platform**

  Set the FA626TE series processor option to support the FA626TE CPU

- **Platform environment**

  Choose the settings from the following platform

  GM8126: GM8126 platform

- **(240000000) System clock**

  The default system clock is 240 MHz for the GM8126 platform. This is the only default settingwhen the system boots. It will read the configuration from PMU to know the real clock of the AHB bus. Users do not need to concern this configuration.

- **[ ]** Disable **ICache**
- **[ ]** Disable **DCache**

  Users can disable ICache and/or DCache by setting the configurations above.

- **[ ]** Disable **Write Buffer**

  Users can disable the cache write buffer by setting this configuration.

- **[ ] Force write through D-Cache**

  Users can force the FA626TE cache to enter the write-through mode. The default mode is the write-back mode.

- **[*] Idle mode support**

  Users can use this configuration to enable the CPU idle mode.

- **[*] BTB support**

  Users can use this configuration to enable the Branch Target Buffer (BTB).

Please refer to the FA626TE data sheet for the detailed CPU functions.

2.2.2.2 Making Kernel

For the first time users of making a Linux kernel, users may clean all the object files and recreate the dependency. The following is an example.

 # make clean

FA626TE-Linux provides the shell script, “build”, for users to easily make the kernel.

 # ./build

It creates the final kernel image, **mbootpImage,** and the kernel ELF file, **vmlinux.** Users may modify **“**build” for the following reason:

- Copy the output image to a specified directory: Modify the command, “cp <source> <target>”, in the **build** file to meet the requirement.

```
[root@Harry linux-2.6.28-fa]# cat build
Make zImage
sudo cp arch/arm/boot/zImage /tftpboot/mbootImage
sudo cp –f system.map /tftpboot/
sudo cp –f vmlinux /tftpboot/
sudo cp –f vmlinux.map /tftpboot/
sudo chown nobody:nobody /tftpboot/mbootImage
```

Figure 2-3. File Content of “build”

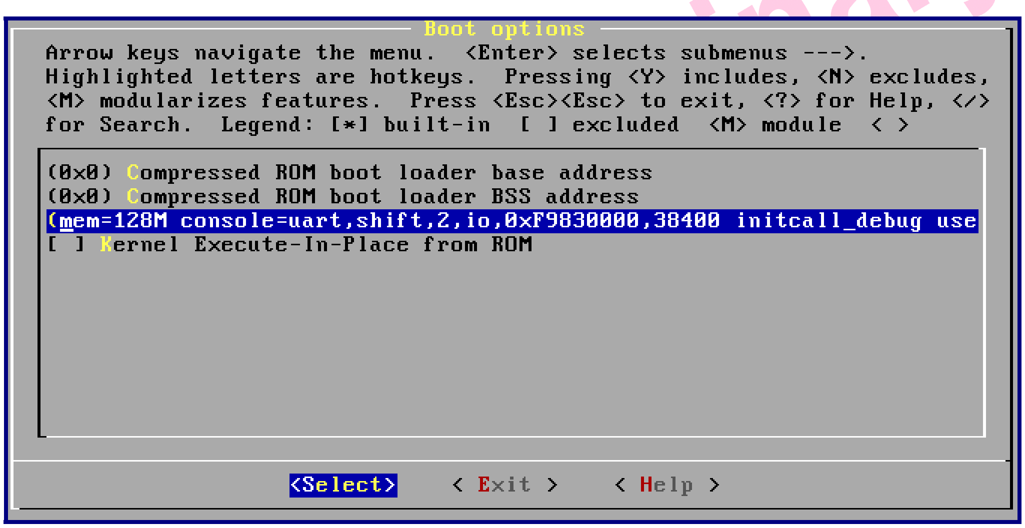


Figure 2-4. Configuring Boot Options

2.2.3 Building U-BOOT

U-BOOT is used by FA626TE-Linux as the OS loader.

2.2.3.1 Configuring U-BOOT

The FA626TE U-BOOT maintains the configuration file, *GM.h*, to configure different hardware environments. The *GM.h* file is located at: **/usr/src/arm-linux-2.6.28/u-boot-2008.10/include/configs/GM8126.h**. Users can modify this file depending on the circumstances.

In addition, users can modify the MAC address to download the Linux code to the MediaCreative platform. At this stage, users should modify the definition, CONFIG_ETHADDR, of the MAC address in **u-boot-2008.10/include/configs/GM8126.h** (As shown in Figure 2-5) and the variable, ftmac110_mac_addr, in **u-boot-2008.10/driver/net/ftmac110.c** (As shown in Figure 2-6).

```
#define CONFIG_ETHADDR 00:40:25:00:00:01
#define CONFIG_NETMASK 255.255.255.0
#define CONFIG_IPADDR 192.168.68.200
#define CONFIG_SERVERIP 192.168.68.201
```

Figure 2-5. Modifying MAC and IP for U-BOOT in config_GM8126.h

```
static char ftmac110_mac_addr[] = {0x00, 0x42, 0x70, 0x00, 0x30, 0x52};
```

Figure 2-6. Modifying MAC for U-BOOT in ftmac110.c

2.2.3.2 Making U-BOOT

Once GMAC and IP are modified, users can build U-BOOT with the following commands:

 # cd /usr/src/arm-linux-2.6.28/u-boot-2008.10

 # ./make_8126

These two commands will create the file, *u-boot.bin*, in the folder shown in the first line. Users should follow the instruction to burn U-BOOT into the GM8126 Flash and write the specific image, **u-boot.bin**, to the flash address 0x10200000 (For the NOR system). However, to upgrade SPI NOR or NAND flash, the PCTOOL or SPI/nand command should be used. Please refer to the flash user guide for programming SPI NOR or NAND flash.

2.3 FA626TE-Linux OS Loader – U-BOOT

U-BOOT is a well-known OS loader in the Linux world that is capable of loading images from a terminal protocol (Such as **Kermit**) and booting the Linux kernel. It provides the flash utilities and Ethernet TFTP transfer functions.

2.3.1 Running U-BOOT

The FA626TE-Linux distribution package provides the **U-BOOT** code to perform the following tasks:

- Programming flash
- Transferring data from PC to the target by UART (Kermit) or Ethernet (TFTP)
- Loading or branching Linux kernel

Note: Users can run the **U-BOOT** code from flash or via ICE.

2.3.1.1 Running U-BOOT from Flash

If the boot code (rom.bin) and the U-BOOT code are ready in flash, users can run U-BOOT from flash by the keystroke of “ESC” at the reset time. Please refer to Appendix E for the information about the CPE boot flow.

2.3.1.2 Running U-BOOT via ICE

Users may run U-BOOT via ICE by following the procedure below:

- Connecting the FA626TE target to the user PC with JTAG ICE
- Opening the OPENice debugger and load **u-boot.bin** to the memory address, 0x0

- Setting PC to 0x0, and run

2.3.2 U-BOOT Environment Variables

U-BOOT maintains a number of environment variables for various functions. These environment variables can be displayed by using the following command.

=> printenv

Users can set the environment value using the command, “setenv name value”, where “name” denotes the environment variable name and “value” denotes the value to be set. The following command is an example of setting the IP address environment.

=> setenv ipaddr 192.168.68.48

Table 2-4 lists the environment variables used in this document.

Table 2-4. Environment Variables

Environment Variable	Description
ipaddr	The target IP address
serverip	The TFTP server IP address
ethaddr	The MAC address value
netmask	The IP address netmask value

2.3.3 U-BOOT Command Reference

The most commonly used commands in U-BOOT are listed in Table 2-5. Users can type “help” on the U-BOOT terminal to display the command list.

Table 2-5. Commonly Used U-BOOT Commands

Command	Description
printenv	Print the environment variables of U-BOOT
setenv [env] [value]	Set the environment variable
go [address]	Jump to the specific address
flinfo	Print the flash information
erase [address 1] [address 2]	Erase the flash from address 1 to address 2
cp.b [source] [destination] [size]	Program the flash from source to destination
loadb [address]	Load binary to specific address by using the Kermit protocol
tftp [address] [image]	TFTP transfer of images to the specific address

2.4 Booting FA626TE-Linux

FA626TE-Linux can be booted on the target system by using one of the following two booting scenarios:

- Booting FA626TE-Linux via ICE
- Booting FA626TE-Linux from flash

2.4.1 Boot FA626TE-Linux via ICE

Users can load the FA626TE-Linux kernel image to a specific address via ICE and jump. The procedure is as below:

- Load **mbootpImage** to the address, 0x4000000 (Recommended) by the OPENice debugger (Please refer to Appendix B for the procedure)
- Set PC to **0x4000000** on the OPENice debugger and run
- Before running Linux, the system should finish the initialization already.

2.4.2 Booting FA626TE-Linux from Flash

Users can automatically boot the Linux kernel from flash. To boot the Linux kernel from flash, the specified boot code and U-BOOT code are required. Please follow the steps below to automatically boot the FA626TE-Linux from flash (In the case of 16-bit bus width):

- Prepare the boot code: Write the specific images, **nsboot.bin** and **rom.bin**, to flash
- Prepare the U-BOOT code: Write the specific image, **u-boot.bin**, to the flash address by PCTOOL (If the system is SPI NOR or NAND, please use PCTOOL or other methods to program to the flash. Please refer to the flash user guide).
- Prepare the Linux kernel image: Write the specific image, **mbootpImage**, to the flash address. The address definition is not listed. If the system is SPI NOR or NAND, please use PCTOOL or other methods to program to the flash. Please refer to the flash user guide.
- Figure 2-7 is an example of PCTOOL upgrading the images.

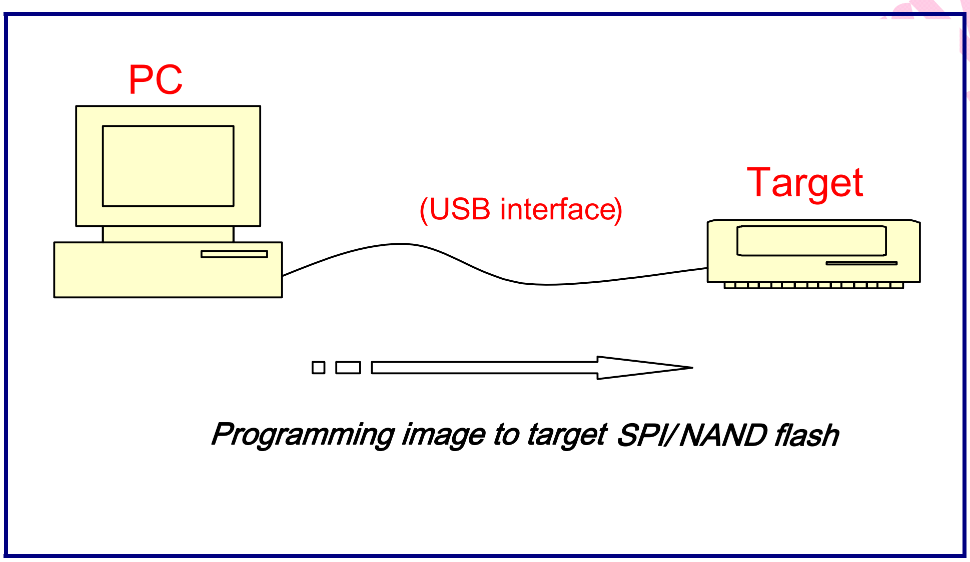


Figure 2-7. Upgrade Images by PCTOOL

2.4.3 Booting FA626TE-Linux by U-BOOT

At most parts of the development stage, users need to perform iterations to modify the code and/or download the code until the results are satisfied. In such circumstances, users need U-BOOT to download and run the code. The required procedure is as follows:

- Set the tftp server on the user Linux host and /etc/xinetd.d/tftp as shown in Figure 2-8.

```
Service tftp
{
   disable         = no
   socket_type     = dgram
   protocol        = udp
   wait            = yes
   user            = root
   server          = /usr/sbin/in.tftpd
   server_args     = -c –u nobody –s /tftpboot
   per_source      = 11
   cps             = 100 2
}
```

Figure 2-8. tftp Setting

- Use the Linux making shell (build) within this package to compile the code, generate the Linux code, and place it into the folder, /tftpboot. To run the Linux code, please follow the steps below:
  1. Reset the GM8126 target
  2. Select item, “ESC”, to enter U-BOOT
  3. Make sure that the IP addresses of the Linux host and GM8126 are correct (printenv), as shown in Figure 2-9.
  4. Type the command, **“tftp 0x4000000 mbootpImage”**, to download the code, as shown in Figure 2-10.
  5. Boot up Linux by using the command, **“go 0x4000000”**. Users can see the Linux boot-up message on the screen, as shown in Figure 2-11.

```
U-Boot 2008.10 (Jan  5 2009 - 19:20:06)

DRAM:  128 MB
Manufacturer ID: 0001
Device ID: 007E
Spansion Mirror Bit flash: addr[28] = 22  addr[30] = 01
Flash: 16 MB
*** Warning - bad CRC, using default environment

In:    serial
Out:   serial
Err:   serial
Net:   FTGMAC#0
=> printenv
bootdelay=3
baudrate=38400
ethaddr=00:84:14:72:61:69
ipaddr=10.0.1.52
serverip=10.0.1.51
gatewayip=10.0.1.51
netmask=255.0.0.0
ethact=FTGMAC#0
ver=U-Boot 2008.10 (Jan  5 2009 - 19:20:06)

Environment size: 188/262139 bytes
=>
```

Figure 2-9. Environment Settings of U-BOOT

```
=> setenv ipaddr 192.168.68.19;setenv serverip 192.168.68.18;tftp 0x4000000 mboo
tpImage;go 0x4000000
FTMAC with ICPLUS PHY IP1001 support
HALF
PHY_SPEED_100M
Using eth0 device
TFTP from server 192.168.68.18; our IP address is 192.168.68.19
Filename 'mbootpImage'.
Load address: 0x4000000
Loading: t RD_REQ, file: mbootpImage
#################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         ###############################################
done
Bytes transferred = 11181388 (aa9d4c hex)
```

Figure 2-10. Downloading Linux Code from U-BOOT

```
Linux version 2.6.28 (root@Harry) (gcc version 4.4.0 (Faraday C/C++ Compiler Rel
ease 20100325) ) #358 Mon Sep 27 09:12:59 CST 2010
CPU: FA626TE [66056261] revision 1 (ARMv5TE), cr=0000397f
CPU: VIPT aliasing data cache, VIPT aliasing instruction cache
Machine: Faraday GM8126
Warning: bad configuration page, trying to continue
MSG: meminfo->nr_banks = 1
MSG: bank start = 0, size = 134217728, node = 0
Memory policy: ECC disabled, Data cache writeback
Memory: high memory occupis 55296K bytes
Built 1 zonelists in Zone order, mobility grouping on.  Total pages: 32512
Kernel command line: mem=128M console=uart,shift,2,io,0xF9830000,38400
Early serial console at I/O port 0xf9830000 (options '38400', shift 2)
console [uart0] enabled
PID hash table entries: 512 (order: 9, 2048 bytes)
GM Clock: CPU = 533 MHz, AHBCLK = 266 MHz, PLL1CLK = 800 MHz, PLL2CLK = 540 MHz
console handover: boot [uart0] -> real [ttyS0]
Dentry cache hash table entries: 16384 (order: 4, 65536 bytes)
Inode-cache hash table entries: 8192 (order: 3, 32768 bytes)
Memory: 128MB = 128MB total
Memory: 67584KB available (3090K code, 159K data, 3652K init)
Calibrating delay loop... 526.33 BogoMIPS (lpj=263168)
Mount-cache hash table entries: 512
CPU: Testing write buffer coherency: ok
```

Figure 2-11. Linux Message during Boot-up

2.5 FA626TE-Linux Internals

This section introduces the implementation of FA626TE-Linux.

2.5.1 I/O Address Mapping

Because FA626TE-Linux uses the MMU-based Linux kernel, it requires the I/O memory mapping to access the I/O peripheral. Most of the I/O addresses of the common IPs mapping on FA626TE-Linux is defined in "**<TOPDIR>/**arch/arm/mach-GM/platform-GM8126/spec.c", such as the DDR controller, DMA controller, and interrupt controller.

The most important items on map_desc are “virtual I/O address” and “physical I/O address”. “IO_ADDRESS()” is defined in “**<TOPDIR>/**arch/arm/mach-GM/include/mach/platform-GM8126/spec.h”.

Note: Most of the module drivers use “ioremap()” to allocate its own virtual address instead of using “spec.h”. If the physical addresses of the IPs are not specified in “spec.h”, they will be “platform_io.h”.

Users may modify “spec.c” and “spec.h” when:

- The peripheral device has been added or removed.
- The physical I/O address or size of the device has been changed.

2.5.2 Reserved memory

Usually, the module drivers require large and contiguous physical memory. However, in the embedded system, it is difficult to prevent the memory from fragementation. For this purpose, GM has pre-allocated a large memory and use particular mechanisms to manage the memory. The GM memory manager is called “Frammap”. The reserved memory size for the Frammap manager is defined in ***<TOPDIR>/****arch/arm/mach-GM/include/mach/platform-GM8126/memoy.h*, which defines the high memory zone size for the system DDR only. For other DDRs, the entire DDR size is allocated to be used by drivers during the system initialization. While inserting the Frammap module, users can decide the reserved size of the real memory. Please refer to the relevant Frammap documents for details.


```
#define HIGH_MEM_SIZE     0x3600000
```

```
/ # cat /proc/frammap/ddr_info
Memory Statistic: DDR:0, each block size = 256K
MemBase      MemEnd      Next Avail    AllocBlks    FreeBlks    MemSize
0x30000000   0x65ffffff  0x31800000    6            210         0x3600000
```

2.5.3 Virtual Memory Range

In *pgtable.h*, a macro defines its starting address for virtual memory.

```
#ifndef VMALLOC_START
#define VMALLOC_OFFSET          (8*1024*1024)
#define VMALLOC_START           (((unsigned long)high_memory + VMALLOC_OFFSET) &
~(VMALLOC_OFFSET-1))
#endif
```

The ending address is defined in **<TOPDIR>/**arch/arm/mach-GM/include/mach/platform-GM8126/vmalloc.h.

#define VMALLOC_END 0xE0000000

Note: The end address is configurable if users want to expand the virtual memory range.

2.5.4 mbootpImage Flow

FA626TE-Linux provides **mbootpImage** to run the Linux kernel. **mbootpImage** comprises four components, as shown in Figure 2-12.

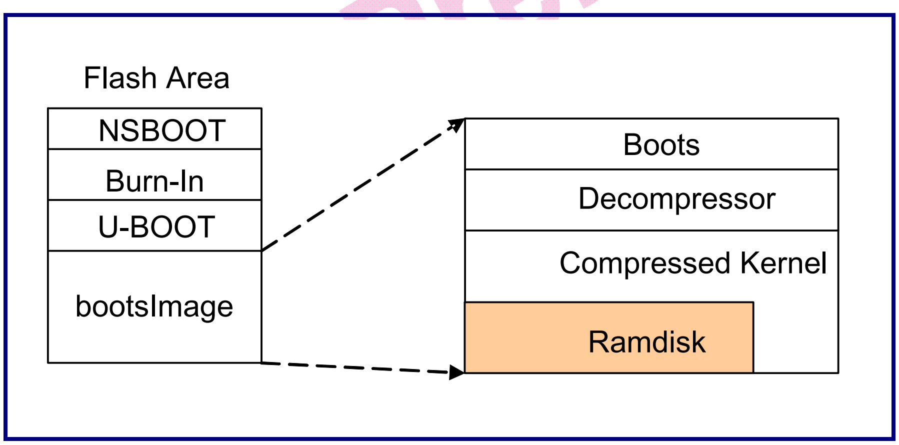


Figure 2-12. mbootpImage

- Boots: This component is in charge of copying bootsImage to the memory in the NOR system. However, this action is not allowed in the SPI NOR/NAND system. Both the boot loader (U-Boot) and Burn-In play a key role in copying the linux image to the memory.
- Decompressor: The code that is in charge of decompressing "Compressed Kernel".
- Compressed Kernel: The image of compressed kernel.
- Ramdisk: The file system that is provided for the Linux kernel.

2.6 FA626TE-Linux Debugging

The FA626TE-Linux distribution provides users with information on how to debug the Linux kernel code and user application programs.

2.6.1 Debugging the Kernel

OPENice is implemented to debug the Linux kernel in the development host by ICE (JTAG interface). OPENice supports the ELF Dwarf-2 debug format, which can also be generated by the GCC compiler.

The detailed debugging procedure is as follows:

- Load **mbootpImage** to DDR 0x4000000
- Set PC to 0x4000000 on the OPENice debugger. Clear the "**Vector catch**" and "**Semihosting**" options on the OPENice debugger
- Run for a few seconds and then stop the OPENice from running (This is used to set the breakpoint at 0x8054). Please run OPENice until kernel is decompressed to 0x8000. The best timing to stop running OPENice is when the string, "Uncompressing Linux…", is observed on the terminal.
- Set the breakpoint to 0x8054 on the OPENice debugger and run
- After stopping the OPENice debugger and catching the breakpoint at 0x8054, users can use "Step in" on the OPENice debugger until reaching the address 0xc0008080
- Launch "Load debug symbols" on OPENice to load the **vmlinux** image
- Debug the Linux kernel

Note: The source level debugging only supports the C code. The assembly code is not supported.

2.6.2 Debugging Application Programs

FA626TE-Linux has precompiled gdb to debug the application programs. Users can also build their versions of gdb with the GM cross compiler.

Th GM precompiled version is based on gdb 7.2. Users can acquire the debugger in arm-linux-2.6.28-fa/usr/gdb/ directory.

2.6.3 Debugging Application through Ethernet

Assume that the target board with an IP address, 192.168.68.38, the procedure for debugging the user application is as follows:

1. Compile the user application with the option, "**-g**". For example, the execution file name is *fail_example* and the source code is "fail_example.c":
   # arm-none-linux-gnueabi-gcc –static -g fail_example.c -o fail_example

2. Let **arm-none-linux-gnueabi-gdbserver** starts to listen to the gdb client on the host

```
/mnt/nfs/works/trunk/arm-linux-2.6.28/user/gdb # ls
CVS                              arm-none-linux-gnueabi-gdbserver
arm-none-linux-gnueabi-gdb       test
/mnt/nfs/works/trunk/arm-linux-2.6.28/user/gdb # ./arm-none-linux-gnueabi-gdbser
ver 192.168.68.38:2345 test/fail_example
Process test/fail_example created; pid = 79
Listening on port 2345
```

3. On the host, execute **arm-none-linux-gnueabi-gdb**

```
[mars@dream gdb]$ ./arm-none-linux-gnueabi-gdb test/fail_example
GNU gdb (GDB) 7.2
Copyright (C) 2010 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "--host=i686-pc-linux-gnu --target=arm-none-linux-gnueabi".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/mars/works/trunk/arm-linux-2.6.28/user/gdb/test/fail_example...done.
(gdb) target remote 192.168.68.38:2345
Remote debugging using 192.168.68.38:2345
0x00008120 in _start ()
(gdb) c
Continuing.

Program received signal SIGSEGV, Segmentation fault.
0x00008234 in bar () at fail_example.c:6
6           tmp = *ptr;
(gdb) list
1       void bar(void)
2       {
3           int *ptr = 0;
4           int tmp = 0;
5
6           tmp = *ptr;
7       }
8
9       void foo(void)
10      {
(gdb) bt
#0  0x00008234 in bar () at fail_example.c:6
#1  0x00008254 in foo () at fail_example.c:11
#2  0x00008264 in main () at fail_example.c:16
(gdb) 
```

GM recommends users to perform unit tests and integration tests in order to prevent from debugging, which is always time consuming.

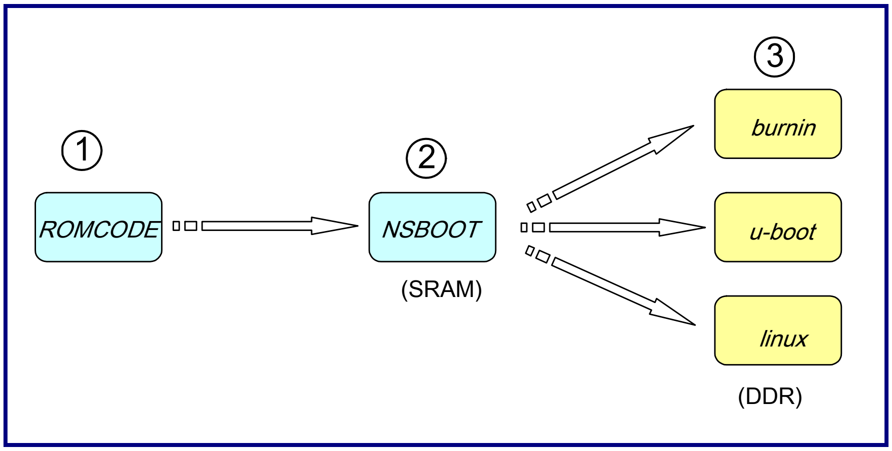
2.7 SPI/NAND Flash Boot Loader

The content of the SPI/NAND flash should not be read as a memory device. In order to read the boot image from the SPI/NAND flash for booting, an embedded ROM code is responsible for loading the boot loader, which is called "nsboot", from the SPI/NAND flash into SRAM for execution. SRAM is a readable/writeable memory in MCP100 of GM8126. GM8126 is equipped with an internal SRAM buffer to support the SPI/NAND flash boot loader. When booting, the first page of SPI/NAND flash will be read by the embedded ROM code for verification purpose. The subsequent boot loader body stored in the SPI/NAND flash memory will be fully loaded into SRAM for execution. For the current design, the image size of nsboot cannot exceed 24K bytes.

In general, the SPI/NAND boot code will copy the content of the SPI/NAND flash to SDRAM. Once the copy is completed, the main program will be executed on SDRAM.

2.7.1 Bloot Loader Booting Sequence

- When booting, the embedded ROM code is first brought up and reads the jumper setting to know which flash type will be accessed. Currently, there are two flash types: SPI NOR and NAND. The mebedded ROM code will then check if there are valid images in the flash. ROMCODE will break the booting procedure and enter the firmware update mode if any false image is found.
- Once the images are verified, the SPI/NAND will boot and execute the next block that will be loaded into SRAM.To simplify, the SPI boot and NAND boot are combined as "NSBOOT".
- NSBOOT will read the information stored in the flash to recognize the next image to be executed (Currently it should be burn-in). NSBOOT will then copy the image body to DDR (SDRAM) according to the size of the image in the image header of the loaded image.
- Once the copy is completed, the loaded image will be executed on SDRAM.
  (The loaded image can be burnin, u-boot, linux, or any other customer image.)


③
①
②
ROMCODE
NSBOOT
burnin
u-boot
linux
(SRAM)
(DDR)

Chapter 3

Device Driver

This chapter contains the following sections:

3.1 Timer Driver

For GM8126, all the timing interrupts are IRQ. However, for Linux 2.6.28, IRQ provides high resolution timing, including the system timing. At least two timers are needed in GM8126: Timer0 and Timer1. The related source files of the timers and the system timer are listed in Table 3-1.

Table 3-1. Related Timer Source Files

File Name	Description
<TOPDIR>/arch/arm/mach-GM/fttimer010.c	This file implements the low-level timer functions.

3.2 Interrupt Driver

Table 3-2. Related Interrupt Source Files

File Name	Description
<TOPDIR>/arch/arm/kernel/irq.c	This file implements the kernel patch irq functions.
<TOPDIR>/arch/arm/ach-GM/ftintc010.c	This file implements the low-level interrupt controller functions.

3.3 Serial Driver

3.3.1.1 Serial I/O Resource Overview

The GM serial I/O resource and memory are listed in the GM8126 data sheet. Please refer to Table 3-3 for the UART clock setting. The UART clock is divided from PLL2.

Table 3-3. GM8126 UART Clock

UART clock	PLL2/div MHz

3.3.1.2 Source File Overview

FA626TE-Linux provides the UART driver to implement the the tty/console driver. The related files on Linux are shown in Table 3-4. Please note that <TOPDIR> is the top path of the kernel source tree.

Table 3-4. File List of UART Source

File Name	Description
<TOPDIR>/drivers/serial/serial_cs.c	This file implements low-level UART functions.
<TOPDIR>/include/asm/arch-GM/platform-GM8126/serial.h	This file is the header file of serial port definition.

3.3.2 Serial Driver Guide

In order for users to activate the UART function, certain options, such as serial port, setting the baud rate to 38400, and selecting UART0 as the kernel console port, should be selected for kernel configuration. Figure 3-2 depicts the serial port (Main Menu → Character Devices).

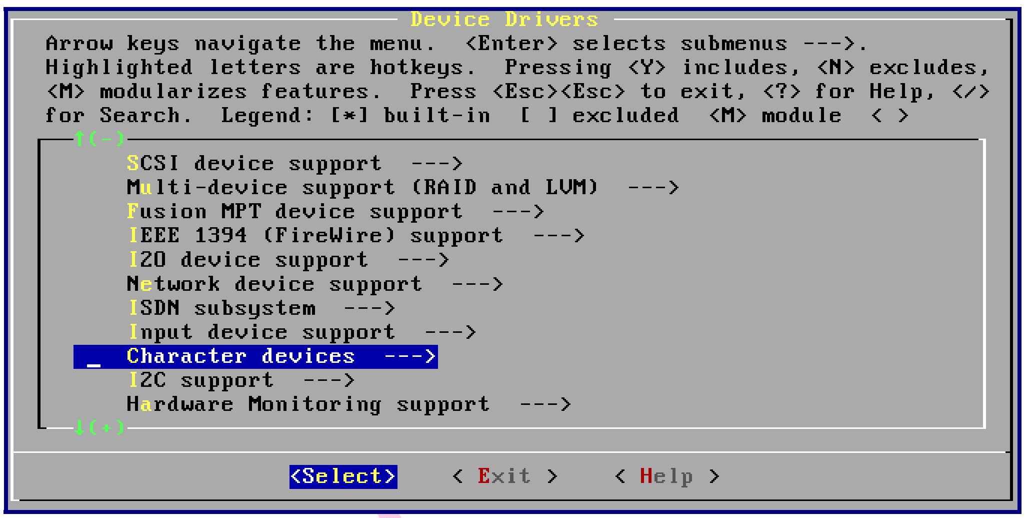


Figure 3-1. Kernel Configuration: Device Drivers

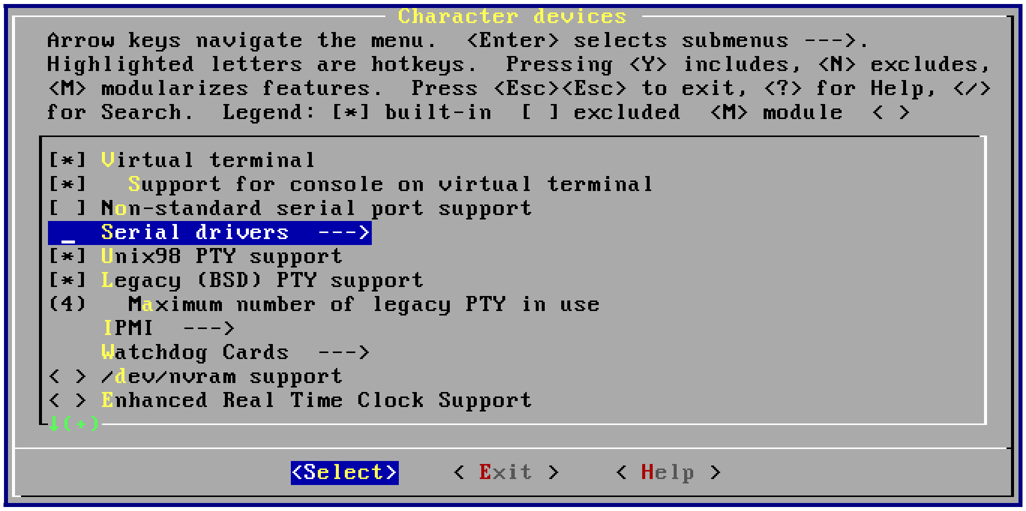


Figure 3-2. **Kernel Configuration: Character Devices**

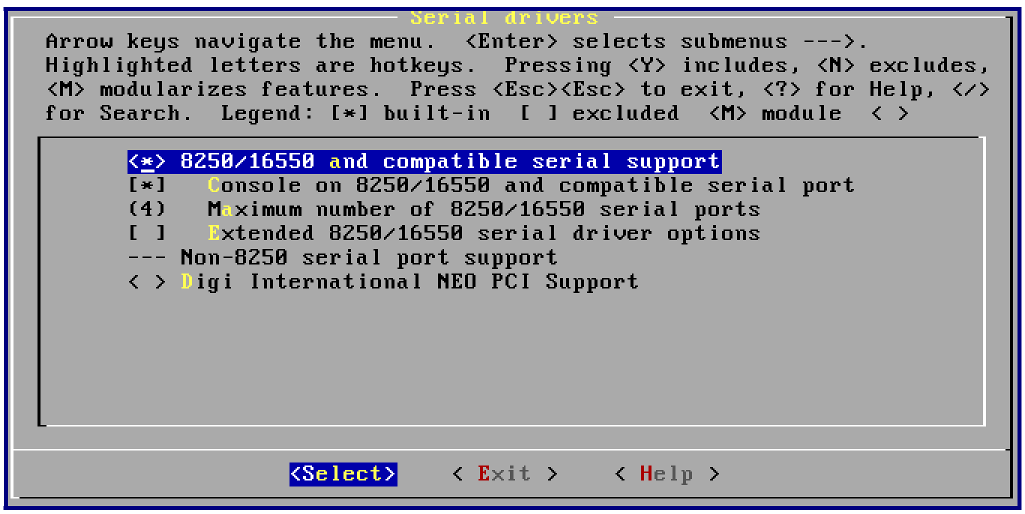


Figure 3-3. **Kernel Configuration: CPE Serial Port Support**

After compiling the Linux kernel with the options listed above, Linux will run and the terminal will show the booting message, as shown in Figure 3-4. Users will see the description of UART in the booting message if the procedure is successfully followed. Please note that the terminal on the host must be set to the same baud rate as the target.

```
Serial: 8250/16550 driver4 ports, IRQ sharing disabled
serial8250: ttyS0 at I/O 0xf9830000 (irq = 9) is a 16550A
serial8250: ttyS1 at I/O 0xf9840000 (irq = 10) is a 16550A
serial8250: ttyS2 at I/O 0xf9850000 (irq = 20) is a 16550A
serial8250: ttyS3 at I/O 0xf9880000 (irq = 21) is a 16550A
```

Figure 3-4. Linux Booting Message: Serial

3.3.3 Driver Internals

This section describes the internal design of the Linux driver , including features, architecture, and functions.

3.3.3.1 Driver Features

The Linux UART driver uses the device name, "**/dev/ttyS**". The major number of ttyS is 4 and the minor number starts from 64. Table 3-5 lists the device name and the device number.

Table 3-5. UART Device Number

Device Name	Major Number	Minor Number
/dev/ttyS0	4	64
/dev/ttyS1	4	65
/dev/ttyS2	4	66
/dev/ttyS3	4	67

The UART device driver does not implement any DMA feature; instead, it implements the interrupt routine to serve the Rx FIFO data.

3.3.4 References

- GM UART and IrDA Controller Specification
- Linux Device Drivers, 2nd Edition:
  Available at http://www.oreilly.com/catalog/linuxdrive2/chapter/book/index.html

3.4 Driver ko in lib/Modules and User Guide

3.4.1 Introduction

For different EVB combinations, users can use the insmod drivers based on certain rules. All device driver modules are provided in SDK in the following path:
/usr/src/arm-linux-2.6.28/target/rootfs-cpio/lib/modules/

The hardware devices and related device driver modules are listed in Table 3-6.

Table 3-6. List of Hardware Devices and Related Device Driver Modules

Device Driver Module	Description
cpe-wdt.ko	WatchDog timer driver
ds1307.ko	DS1307 RTC IC driver
dvr_common.ko/dvr_dec.ko/dvr_disp.ko/dvr_enc.ko	DVT drivers
irdet.ko	IRDET driver
keyscan.ko	Keyscan driver
favc_common.ko/favc_drv.ko	H.264 encoder/decoder drivers
fcapx.ko	Video Capture drivers to receive the video data from the ITU-R BT.656 interface
fgpio.ko	GPIO control driver
fi2c.ko	I^2C control driver
flcd200-common.ko/flcd200-pip.ko	LCD control drivers
fmcp_drv.ko/fmjpeg_drv.ko	JPEG engine drvier
frammap.ko	Memory mapping driver
fscaler0.ko	Scaler control driver
ftdi210.ko	3D De-interlace and De-noise filte driver
ftmac100.ko	Ethernet control driver
ftsdc010.ko	SDC control driver
security.ko	AES/DES/TDES cipher control driver
snd_fi2s_xxx.ko	Driver to transmit audio data between SSP controller and audio codec
snd_ftssp010.ko	SSP control driver
videograph.ko	Video Graph driver

In the arm-linux system, all the modules exist on the "/lib/modules/" path. The "insmod" command is used to insert the device driver modules. These modules depend on each other.

The related user guides are listed in Table 3-7.

Table 3-7. List of Related User Guides

Device Driver Module	Reference User Guide
Capture	GM8126_Capture_User_Guide
LCD	GM8126_LCD_User_Guide
Deinterlace/Denoise	GM8126_DN_User_Guide
Frammap	GM8126_Frammap_User_Guide
SPI/NOR flash	GM8126_Flash_User_Guide
GPIO	GM8126_GPIO_User_Guide
I2C	GM8126_I2C_User_Guide
RTC	GM8126_RTC_User_Guide
SAR ADC	GM8126_SAR_ADC_User_Guide
WatchDog	GM8126_Watchdog_User_Guide
DVR	GM8126_DVR_User_Guide
USB OTG	GM8126_USB_OTG_User_Guide
SD/MMC Card	GM8126_SD_Card_User_Guide
SPI	GM8126_SPI_User_Guide
Scaler	GM8126_Scaler_User_Guide