Boot Loader Manual

Boot loader version: 00.00.15

REVISION HISTORY

Revision	Date	Description
0.0.1	2003/05/06	Initial draft
0.0.2	2004/2/9	Fixed for linux development
0.0.3	2004/3/1	Remove rootdir Add configuration option 'a'
0.0.15	2004/11/18	Synchronize to the last loader version.

1. Objective

This document is created to describe the loader program and corresponding initialization, driver, file system, and download modules. The main function of the loader program is to load the runtime image from the flash to SDRAM, decompress it, and start execution of runtime code. Besides, the loader program provides the download and update functionality.

2. File List

The source tree contains four directories.

2.1. src directory

This directory contains source codes, including C and assembly files.

- □ bdinfo.c: Board information, e.g. MAC address, access.
- □ calloc.c: Simple and small memory allocation management.
- □ dev.c: Proprietary device interface.
- □ filesystem.c: File system.
- □ flashdrv.c: Flash driver.
- □ loader.c: Main routine.
- □ ns16550.c: NS16550 compatible UART driver.
- □ uart.c: Uart function interface and queue management.
  - ■ xmodem.c: XMODEM download.
- □ lx4180 directory: Lexra 4180 CPU specific source code.
  - ■ genexcpt.c: General exception handler.
  - ■ crt0.s: CPU initialization and startup.
  - ■ lx4180.s: CPU specific utilities.
  - ■ vectors.s: Reset and exception vectors.
  - ■ c_data.c: static variables for .s files.
- □ rtl8650 directory: RTL8650 ASIC specific source code.
  - ■ int.c: Interrupt dispatcher.

  - phy.c: PHY register access.
  - swCore.c: Switch core access interface.
  - swNic_poll.c: Switch NIC polling mode driver.
  - swTable.c: Switch core table access.
  - swUtil.c: String utilities for switch corresponding files.
  - tick.c: Tick timer driver.
  - vlanTable.c: VLAN table access.
  - initmem.s: Memory controller driver.

- □ gzip directory: GZIP decompression utility.
  - gzip.c, gziputil.c, inflate.c, unzip.c, crypt.h, getopt.h, gzip.h, lzw.h, revision.h, tailor.h.
- □ tftpnaive directory: BOOTP and TFTP utility.
  - arp.c, bootp.c, busywait.c, icmp.c, ip.c, net.c, tftp.c, udp.c, net.h, tftpnaive.h.
- □ libc directory: libc utility.
  - stdlib/strtol.c, stdlib/atoi.c, string/memcmp.c, string/bzero.c, string/memcpy.c, string/memset.c, string/strchr.c, string/strcpy.c, string/strlen.c, string/strncmp.c, sys-include/_syslist.h, sys-include/COPIED, sys-include/_ansi.h, sys-include/machine/fastmath.h, sys-include/machine/ansi.h, sys-include/machine/ieeefp.h, sys-include/machine/malloc.h, sys-include/machine/setjmp-dj.h, sys-include/machine/setjmp.h, sys-include/machine/stdlib.h, sys-include/machine/termios.h, sys-include/machine/time.h, sys-include/machine/types.h, sys-include/alloca.h, sys-include/ar.h, sys-include/argz.h, sys-include/assert.h, sys-include/ctype.h, sys-include/dirent.h, sys-include/envz.h, sys-include/errno.h, sys-include/fastmath.h, sys-include/fcntl.h, sys-include/grp.h, sys-include/ieeefp.h, sys-include/langinfo.h, sys-include/limits.h, sys-include/locale.h, sys-include/malloc.h, sys-include/math.h, sys-include/newlib.h, sys-include/paths.h, sys-include/process.h, sys-include/pthread.h, sys-include/pwd.h, sys-include/reent.h, sys-include/regdef.h, sys-include/search.h, sys-include/setjmp.h, sys-include/signal.h, sys-include/sys/stat-dj.h, sys-include/sys/stat.h, sys-include/sys/stdio.h, sys-include/sys/syslimits.h, sys-include/sys/time.h, sys-include/sys/timeb.h, sys-include/sys/times.h, sys-include/sys/types.h, sys-include/sys/unistd.h, sys-include/sys/utime.h, sys-include/sys/wait.h

2.2. inc directory

This directory contains header files.

- board.h: Board corresponding definitions.
- flash_map.h: Flash block map for file system.
- ns16550.h: NS16550 compatible UART definitions.
- rtl_dev.h: Proprietary device interface definitions and function prototypes.
- rtl_errno.h: Error number definitions.
- flashdrv.h: Flash driver definitions and function prototypes.
- rtl_image.h: Root directory and image header definitions for file system.
- rtl_types.h: Proprietary type definitions.
- semaphore.h: POSIX compatible semaphore definitions and prototypes.
- uart.h: UART driver definitions.
- compiler directory: Compiler specific header file.
  - ghs \ rtl_depend.h: Green Hills compiler specific definitions.
  - gnu \ rtl_depend.h: GNU compiler specific definitions.
- lx4180 directory: Lexra 4180 CPU specific header file.
  - eregdef.h: CPU specific register definitions.
- rtl8650 directory: RTL8650 ASIC specific header files.
  - asicRegs.h: Peripheral and switch core register definitions.
  - phy.h: PHY access definitions.
  - swCore.h: Switch core interface prototypes.
  - swNic_poll.h: Switch NIC driver prototypes.
  - vlanTable.h: VLAN table access definitions.

2.3. bin directory

This directory contains binary utility programs and third-party libraries and header files.

- packbin: Pack and pad output images.
- packer: Pack loader image for EEPROM.

2.4. doc directory

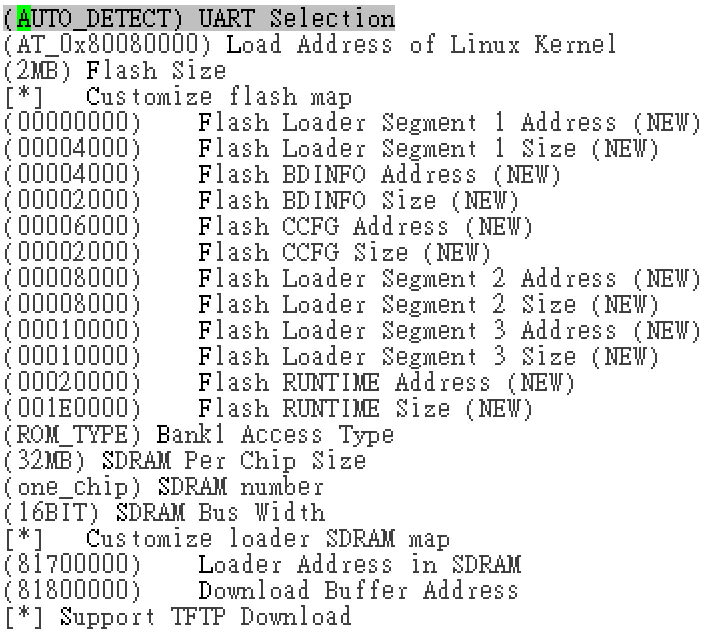
This directory contains documentation files of loader.

3. Build Loader

You may have different types of loader package. For different package, we have different way to build loader image.

3.1. Whole uClinux Distribution

If you have the whole uClinux distribution, you should first type 'make menuconfig' in the uClinux root directory. Then, you will see the following screen:



```
(AUTO_DETECT) UART Selection
(AT_0x80080000) Load Address of Linux Kernel
(2MB) Flash Size
[*]   Customize flash map
(00000000)    Flash Loader Segment 1 Address (NEW)
(00004000)    Flash Loader Segment 1 Size (NEW)
(00004000)    Flash BDINFO Address (NEW)
(00002000)    Flash BDINFO Size (NEW)
(00006000)    Flash CCFG Address (NEW)
(00002000)    Flash CCFG Size (NEW)
(00008000)    Flash Loader Segment 2 Address (NEW)
(00008000)    Flash Loader Segment 2 Size (NEW)
(00010000)    Flash Loader Segment 3 Address (NEW)
(00010000)    Flash Loader Segment 3 Size (NEW)
(00020000)    Flash RUNTIME Address (NEW)
(001E0000)    Flash RUNTIME Size (NEW)
(ROM_TYPE) Bank1 Access Type
(32MB) SDRAM Per Chip Size
(one_chip) SDRAM number
(16BIT) SDRAM Bus Width
[*]   Customize loader SDRAM map
(81700000)    Loader Address in SDRAM
(81800000)    Download Buffer Address
[*] Support TFTP Download
```

The detailed descriptions are followed:

- **UART Selection:** Choice which UART you hardware layouted.
- **Loader Address of Linux Kernel**: Choice the starting SDRAM address of you kernel. You can specify the address you want, and you loader also must know that address you choice. Use default value is suggested, and never change it unless you know what are you doing.
- **Flash Size:** Choice the flash size on your target.
- **Customize Flash Map:** The above figure is the default setting for 2MB flash. You may rearrange your flash map according your need. For example, you need a larger CCFG section. Then, you can enlarge ‘CCFG Size’ and adjust Loader Segment 2 and Loader Segment 3. The example is listed as follow.

(00006000)	Flash CCFG Address
(0000A000)	Flash CCFG Size
(00010000)	Flash Loader Segment 2 Address
(00010000)	Flash Loader Segment 2 Size
(00000000)	Flash Loader Segment 3 Address
(00000000)	Flash Loader Segment 3 Size

We note that the starting address of each section must be assigned to the flash block, and leave zero offset and zero size for unused Loader Segment field. We note that you must confirm all the sections you defined will NOT be overlapped.

- **Bank1 Access Type:** RTL865X reserves flash bank1 for extension. Two types of access timing diagram are supported. One is ROM-type for compatible with flash, and the other one is IO-type for compatible to PCMCIA.

- **SDRAM Per Chip Size:** Choice the size of your SDRAM. We note that this is per-chip size, not total chip size.

- **SDRAM Number:** Choice how many SDRAM chips will be on your target board.

- **SDRAM Bus Width:** Choice 16-bit or 32-bit. Contact your hardware engineer for more information.

- **Custom Loader SDRAM Map:** You can specify the loader execution area and download buffer. Criteria are followed:

  1. Buffer Download Address must be large enough to download image (in development stage) from network. The image may be runtime code file or loader image file, whose name are called as “*.bix”.

  2. Loader code will NOT be overwritten by the extracting kernel code.

  The default addresses will show up after you choose SDRAM size.

- **Support TFTP Download:** You do not need TFTP functionality any more when you enter mass-product stage. Therefore, disabling this function can reduce loader image size.

After you configure the above setting, you can build loader by type 'make clean && make' in the 'loader_srcroot' directory.

3.2. Stand-alone Loader

If you only obtain a 'loader_srcroot.tgz' file, you may follow the steps to build loader.

1. Type 'tar -zxvf loader_srcroot.tgz' to uncompress archive.

2. Change directory to the product directory, e.g. 'cd loader_srcroot'.

3. If you want to change configuration, edit ‘inc/linux/autoconf.h’ file. The detailed descriptions are listed in section 3.1.

4. Type 'make clean && make' to build loader.

3.3. Output files

After the building procedure, you will get the following files in the 'loader_srcroot/' directory.

'`ldr.bix`' -- file used for BOOTP/TFTP

'`ldr.rom`' -- file used for EEPROM programmer

'`ldr.out`' -- file used for MULTI (ICE)

In the following chapter, we will introduce how to put loader into flash.

4. Program Loader Image into Flash

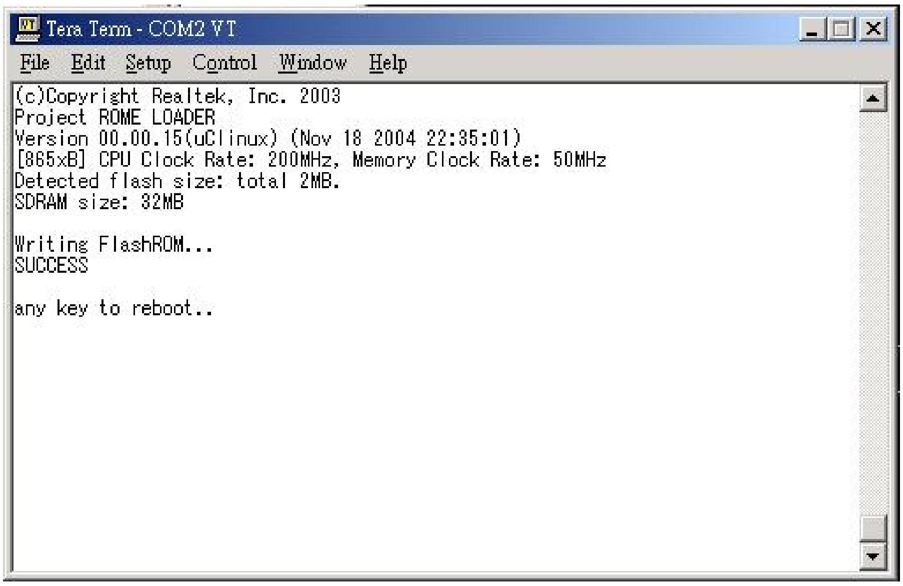
We introduce three methods to program loader image into flash.

4.1. EEPROM Programmer

This is an easy way. All you need is ‘ldr.rom’ file. In Makefile, we will automatically generate this file by the ‘packer’ utility (see Chapter 7 for more details). The parameters of ‘packer’ are designed for our default configuration for 2MB flash. If you rearranged the loader address in flash (in menuconfig), you must also change the parameter of ‘packer’ in Makefile.

Now, copy your ‘ldr.rom’ to your EEPROM programmer. Choice your flash vendor and type, open the ‘ldr.rom’ file, and program it. We note that packer has byte-swapped the file since ‘-2’ parameter is specified. So you don’t need to byte-swap the file.

After programmed, mount flash on your target board, and turn on the board. If everything is OK, you will see the following screen:



```
(c)Copyright Realtek, Inc. 2003
Project ROME LOADER
Version 00.00.15(uClinux) (Nov 18 2004 22:35:01)
[865xB] CPU Clock Rate: 200MHz, Memory Clock Rate: 50MHz
Detected flash size: total 2MB.
SDRAM size: 32MB

Writing FlashROM...
SUCCESS

any key to reboot..
```

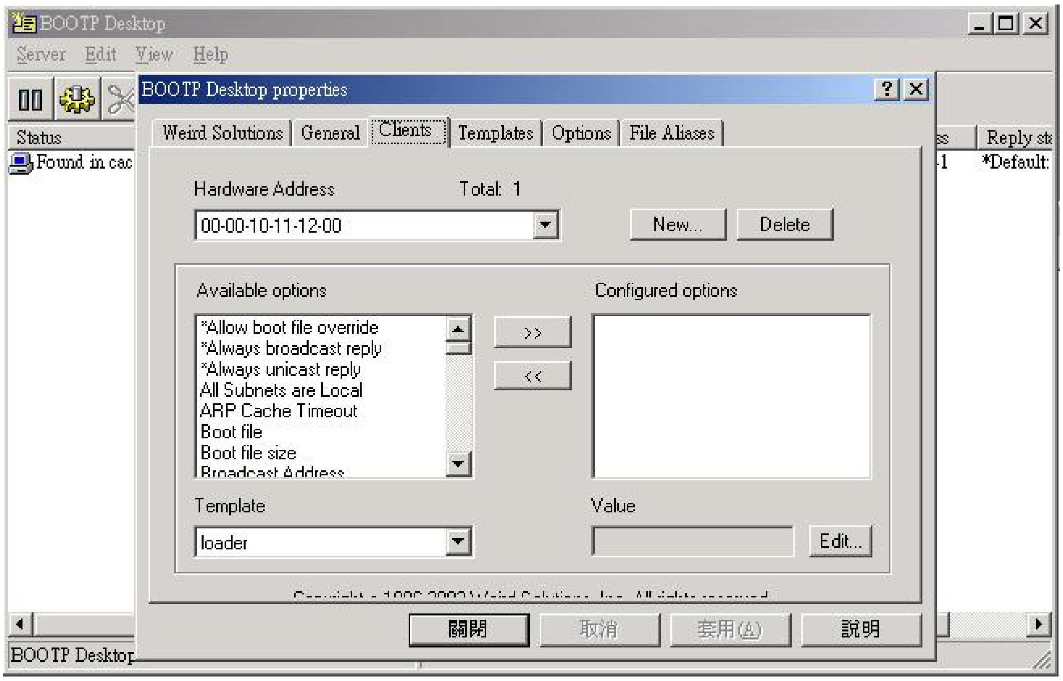
If you see that, it works now.

4.2. BOOTP/TFTP Utility

Once you had burned loader into flash, you can upgrade your loader from network, rather than EEPROM programmer. It is convenient for engineers in development stage. We note that you must enable ‘TFTP Download’ functionality in menuconfig (see section 3.1).

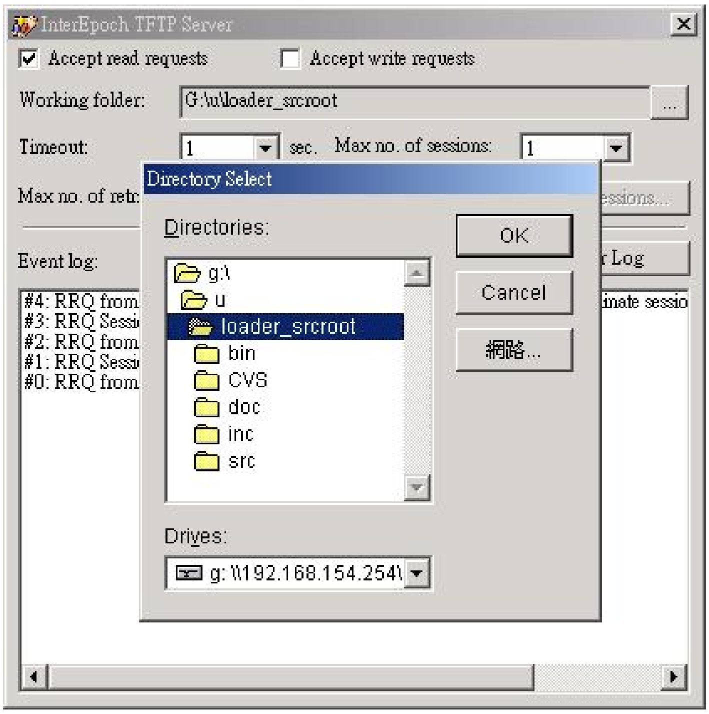
You need a BOOTP server and a TFTP server installed on your development platform. These servers provide target board’s IP address and the loader image file. The following figures take ‘BOOTP Desktop’ and ‘InterEpoch TFTP Server’ for example.

In this method, you need ‘ldr.bix’ file.



Hardware Address: 00-00-10-11-12-00
Total: 1
Template: loader
Value:

Configure BOOTP Desktop. The most important options are ‘**Boot file**’ and ‘**IP Address**’.



☑ Accept read requests ☐ Accept write requests

Working folder: G:\u\loader_srcroot

Timeout: 1 sec. Max no. of sessions: 1

Max no. of retr

Event log:

```
#4: RRQ from
#3: RRQ Sessi
#2: RRQ from
#1: RRQ Sessi
#0: RRQ from
```

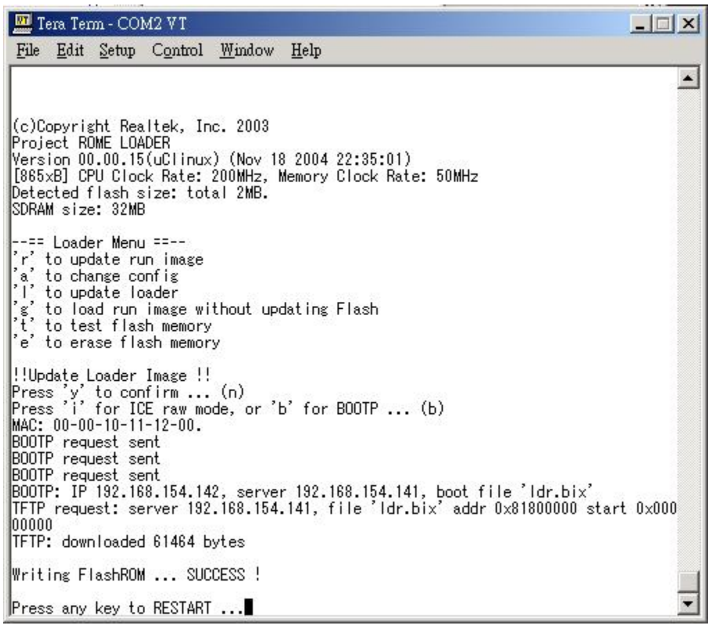
Directories:

```
g:\
  u
    loader_srcroot
      bin
      CVS
      doc
      inc
      src
```

Drives: g: \\192.168.154.254\

Configure TFTP Server. The most important thing is selecting ‘Working folder’, where ‘ldr.bix’ is located in.

Now, turn on your target board. After loader menu shows up, press ‘l’ to update loader. Then, loader will upgrade itself. A typical screenshot is followed.



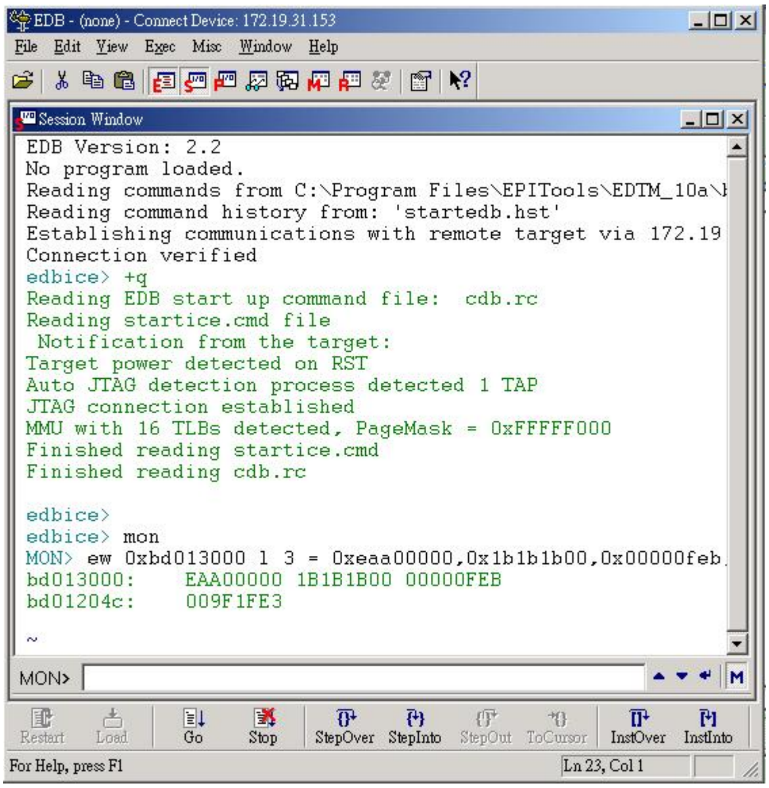
```
(c)Copyright Realtek, Inc. 2003
Project ROME LOADER
Version 00.00.15(uClinux) (Nov 18 2004 22:35:01)
[865xB] CPU Clock Rate: 200MHz, Memory Clock Rate: 50MHz
Detected flash size: total 2MB.
SDRAM size: 32MB

--== Loader Menu ==--
'r' to update run image
'a' to change config
'l' to update loader
'g' to load run image without updating Flash
't' to test flash memory
'e' to erase flash memory

!!Update Loader Image !!
Press 'y' to confirm ... (n)
Press 'i' for ICE raw mode, or 'b' for BOOTP ... (b)
MAC: 00-00-10-11-12-00.
BOOTP request sent
BOOTP request sent
BOOTP request sent
BOOTP: IP 192.168.154.142, server 192.168.154.141, boot file 'ldr.bix'
TFTP request: server 192.168.154.141, file 'ldr.bix' addr 0x81800000 start 0x000
00000
TFTP: downloaded 61464 bytes

Writing FlashROM ... SUCCESS !

Press any key to RESTART ...
```

If you see that, it works now.

4.3. Use ICE to Burn Loader

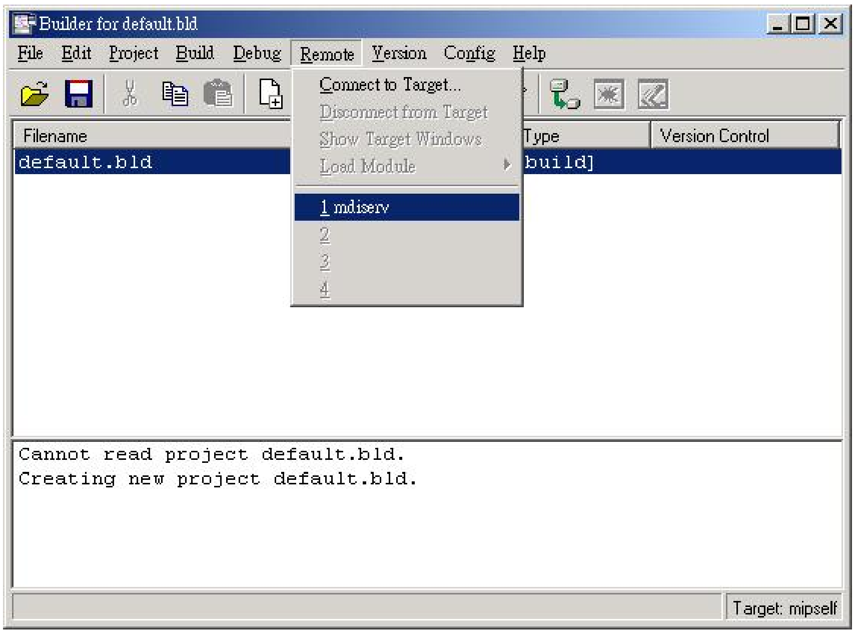
If you have an ICE machine, and essential software, EPI EDB and GreenHill MULTI, another way to program loader into flash is proposed as following. You need two files: ‘ldr.out’ and ‘ldr.bix’.

Connect your ICE and target board with JTAG connector. Then, turn on your ICE and target board. Invoke EPI EDB, and configure MCR register (see below).



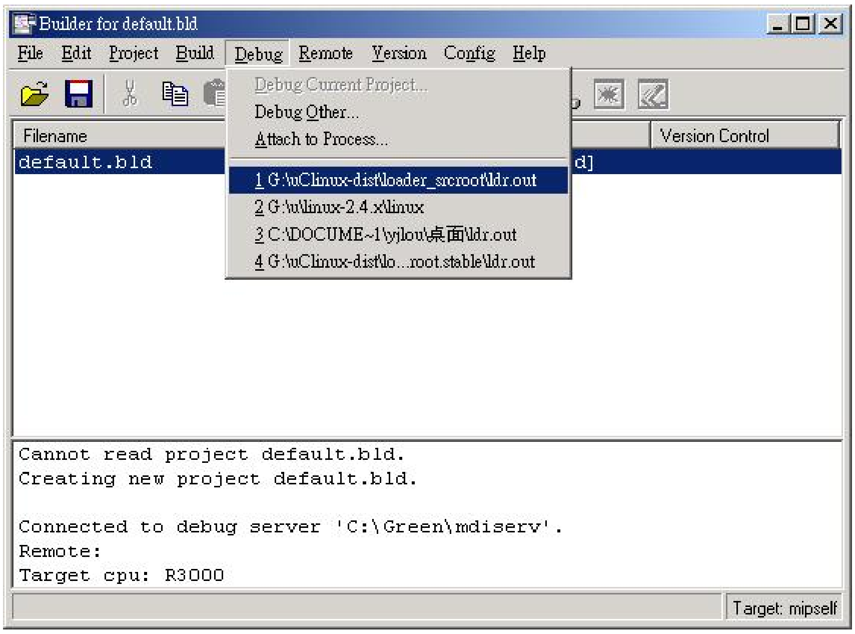
```
EDB Version: 2.2
No program loaded.
Reading commands from C:\Program Files\EPITools\EDTM_10a\
Reading command history from: 'startedb.hst'
Establishing communications with remote target via 172.19
Connection verified
edbice> +q
Reading EDB start up command file:  cdb.rc
Reading startice.cmd file
 Notification from the target:
Target power detected on RST
Auto JTAG detection process detected 1 TAP
JTAG connection established
MMU with 16 TLBs detected, PageMask = 0xFFFFF000
Finished reading startice.cmd
Finished reading cdb.rc

edbice>
edbice> mon
MON> ew 0xbd013000 l 3 = 0xeaa00000,0x1b1b1b00,0x00000feb
bd013000:    EAA00000 1B1B1B00 00000FEB
bd01204c:    009F1FE3

~
```

Then, close EDB EPI. Invoke GreenHill MULTI. Connect to target.



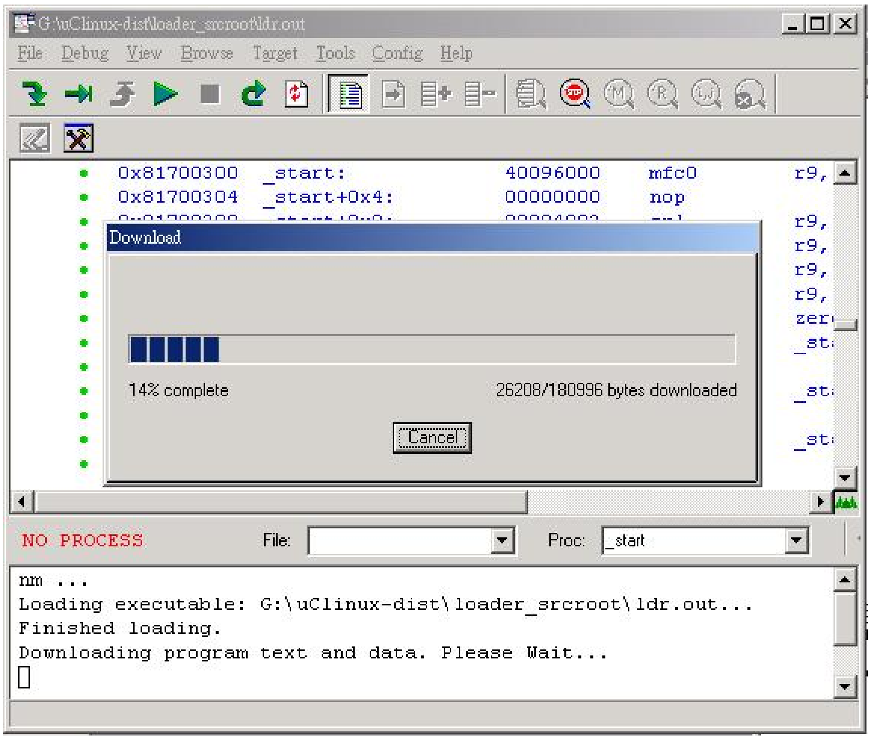
```
Cannot read project default.bld.
Creating new project default.bld.
```

Then, open the ‘ldr.out’ file. You may choose ‘Debug Other’ to lookup the file.


Builder for default.bld
File Edit Project Build Debug Remote Version Config Help
Debug Current Project...
Debug Other...
Attach to Process...
1 G:\uClinux-dist\loader_srcroot\ldr.out
2 G:\u\linux-2.4.x\linux
3 C:\DOCUME~1\yjlou\桌面\ldr.out
4 G:\uClinux-dist\lo...root.stable\ldr.out
Filename
Version Control
default.bld
d]
Target: mipself

```
Cannot read project default.bld.
Creating new project default.bld.

Connected to debug server 'C:\Green\mdiserv'.
Remote:
Target cpu: R3000
```

Start the loader by press ‘F5’.

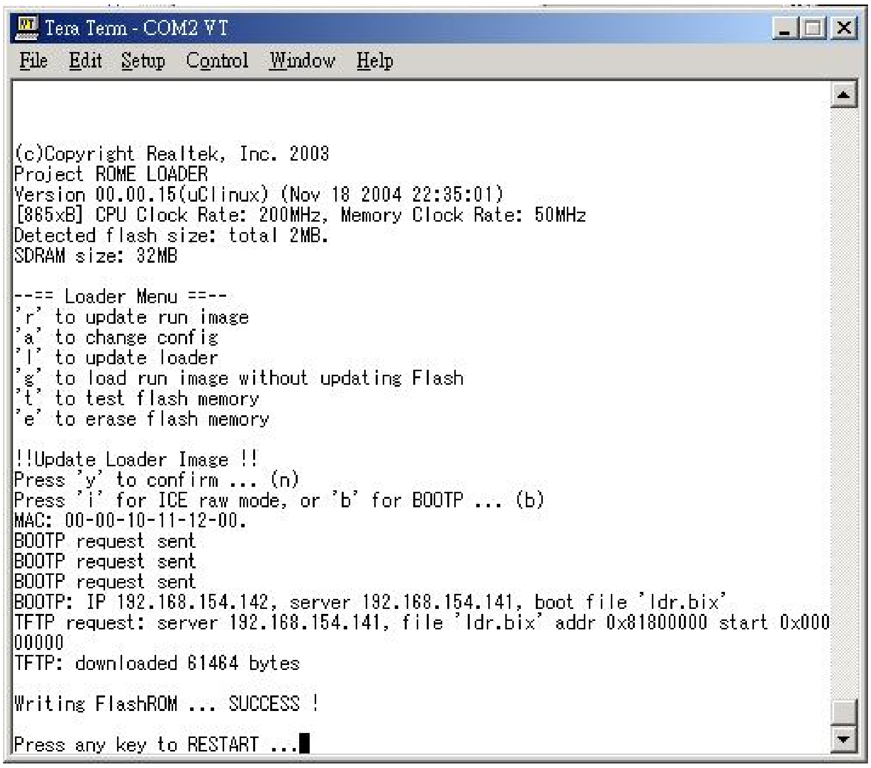

G:\uClinux-dist\loader_srcroot\ldr.out
File Debug View Browse Target Tools Config Help
Download
Cancel
NO PROCESS
File:
Proc: _start

```
0x81700300  _start:       40096000  mfc0  r9,
0x81700304  _start+0x4:   00000000  nop
```

14% complete 26208/180996 bytes downloaded

```
nm ...
Loading executable: G:\uClinux-dist\loader_srcroot\ldr.out...
Finished loading.
Downloading program text and data. Please Wait...
```

After download completed, you can see the loader menu in your terminal. Press ‘l’ key to download and update loader. Remember to prepare BOOTP server and TFTP server.



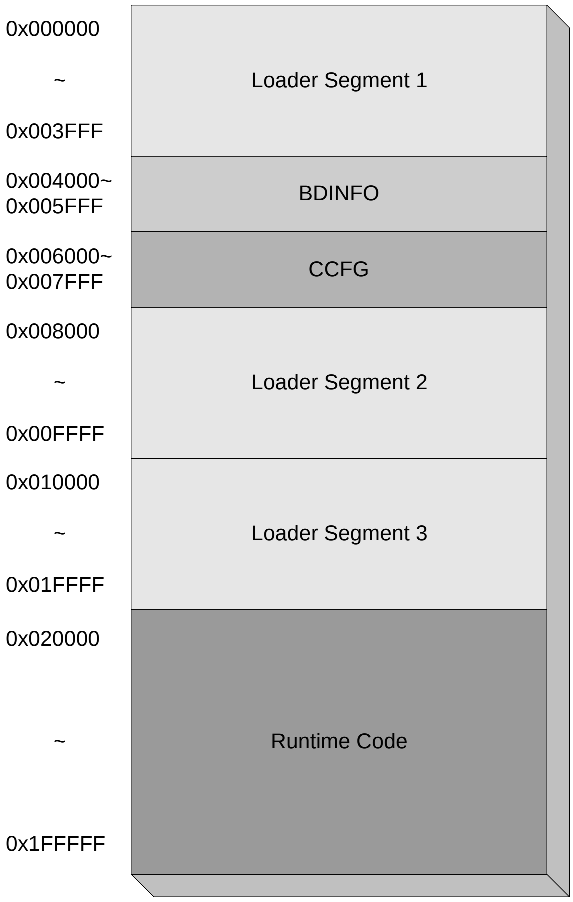
```
(c)Copyright Realtek, Inc. 2003
Project ROME LOADER
Version 00.00.15(uClinux) (Nov 18 2004 22:35:01)
[865xB] CPU Clock Rate: 200MHz, Memory Clock Rate: 50MHz
Detected flash size: total 2MB.
SDRAM size: 32MB

---== Loader Menu ==---
'r' to update run image
'a' to change config
'l' to update loader
'g' to load run image without updating Flash
't' to test flash memory
'e' to erase flash memory

!!Update Loader Image !!
Press 'y' to confirm ... (n)
Press 'i' for ICE raw mode, or 'b' for BOOTP ... (b)
MAC: 00-00-10-11-12-00.
BOOTP request sent
BOOTP request sent
BOOTP request sent
BOOTP: IP 192.168.154.142, server 192.168.154.141, boot file 'ldr.bix'
TFTP request: server 192.168.154.141, file 'ldr.bix' addr 0x81800000 start 0x000
00000
TFTP: downloaded 61464 bytes

Writing FlashROM ... SUCCESS !

Press any key to RESTART ...
```

If you see that, it works now.

5. Flash/SDRAM Map

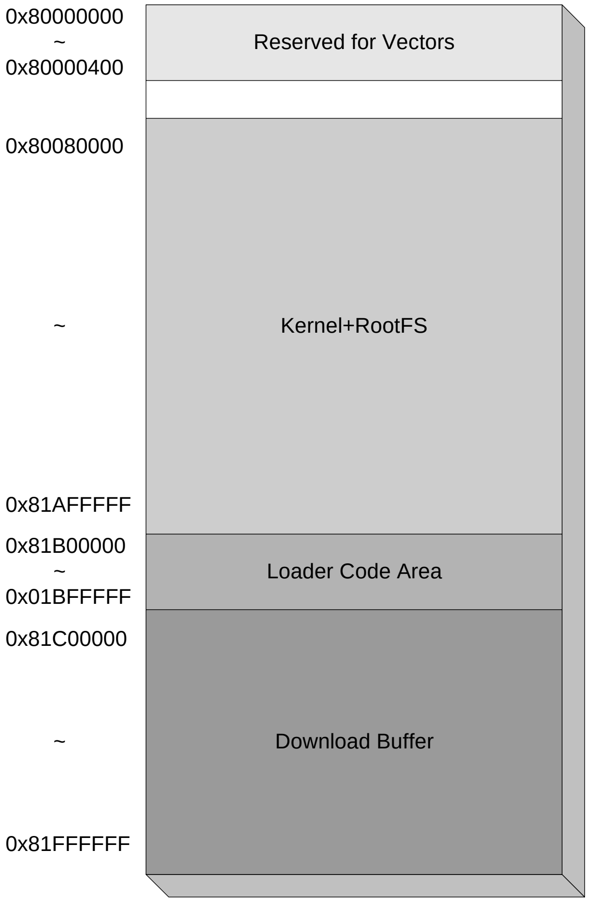
5.1. Default Flash MAP

The flash map is arranged for 2MB flash. If you want to change, use ‘make menuconfig’ in ‘uClinux-dist’ directory to change it.



5.2. Default SDRAM MAP

The SDRAM map is arranged for 32MB. If you want to change, use ‘make menuconfig’ in ‘uClinux-dist’ directory to change it.



6. Currently Supported Flash

We now support the following flash chips and their compatible product:

- AM29LV800BB
- AM29LV800BT
- AM29LV160BB
- AM29LV160BT
- AM29LV320DB
- AM29LV320DT
- M29W320DB
- MX29LV640BB
- M29W320DT
- MX29LV320AB

- MX29LV320AT
- MX29LV640BT
- I28F640J3A
- I28F128J3A
- I28F160C3B

7. Packer

‘Packer’ utility can pack your loader into image form for EEPROM programmer.

Usage: bin/packer [options] outputfile address={filename[,offset[,limit]]|hex:[hex:[...]]} ...

Example:

```
bin/packer -2 images/mp.rom \
          0x00000=loader_srcroot/ldr.bin,0x0,0x4000 \
          0x04000=00:00:10:11:12:00 \
          0x06000=CCFG \
          0x08000=loader_srcroot/ldr.bin,0x4000,0x18000 \
          0x20000=images/run.bix
```

In this example, packer will pack image according to the flash map in section 5.1.