Enhanced RedBoot
for the ARM9315A Processor
Revision 1.1

Introduction

Windows CE customers have been experienced unreliable TFTP operation when booting from the network. Debugging the cause of this failure has proved frustrating to developers since source to the EBoot network stack is not available .

The goal of this project was therefore to create an open source bootloader that is capable of network booting Windows CE 5.0, and to make it robust to network disconnections by using a back-up disk.

Existing Work

Cirrus Logic provides a port of the eCos-2.0 RedBoot bootloader [9] for the ARM9315a. This bootloader is capable of booting both Linux and Windows CE 4.2, and can executed independently of user intervention through a scripted flash interface.

This alone does not fulfill the project requirements for 3 reasons:

First, at the time of this writing, 12/12/06, the bootloader is not capable of booting CE 5.0 [5]. Attempting to use the old method [10] results in a silent failure and erased RAM.

A patch was submitted was 3/29/05 for booting CE 5.0 bin files [6] on a custom edb9315 platform using the rival GNU platform, Das-Uboot [7]. This method fails to boot CE on our target, however, perhaps due to differences between the specific OAL [8] it was designed for, and the one used by Cirrus.

Second, the existing scripting interface is limited. Conditional execution is needed for the case when TFTP fails, and the standard script interface only provides simple unbranched execution. The only distinction made between the success and failure of a function call is the memory it operated on and print statements.

Third, the Cirrus port has no disk support, which prevents any type of removable media from being used. This eliminates the possibility of a non-network back up disk except for on-board flash. This is not always practical, however, due to limited write cycles and size.

Enhancements

To overcome the limitations of the existing bootloader, the following features were added:

- External Builds.
- IDE support
- FAT16 reads
- Enhanced Scripting
- Configuration Files
- CE 5.0 compatibility

External Builds

The existing build environment provides a single directory in which the bootloader, OS, and windowing environment are created from a single build script. To speed development time, code specific to the RedBoot environment was extracted into a separate standalone project.

This project is built using the top level build.sh script, which creates an output directory, redbuild, where it stores all created files. The typical output file is redbuild/install/bin/redboot.bin, which is an image suitable for use with the open source download utility [11].

The bootloader project has successfully been built using a Linux Debian-testing system, with the Cirrus recommend arm compiler:
http://arm.cirrus.com/files/tools/arm-elf-gcc-3.2.1-full.tar.bz2.
It should be compatible with most modern Linux distributions and Cygwin.

IDE support

RedBoot provides functions for interfacing with standard IDE controllers with PIO through it's fs/ide.c module. By porting low level function calls back from the Cirrus Linux port for the processor, basic IDE/CF functionality was realized. The low level function calls are implemented as macros in hal/arm/arm9/ep93xx/include/hal_ide.h The creation of this layer allows the use of the RedBoot *disks* command for viewing the recognized partitions on a hard disk.

To provide a way to automatically back up an OS to disk, the standard RedBoot I/O structure was expanded to allow write calls. Commands for doing raw I/O to disk partitions were then added to the disk layer and exported out to the user by adding the new commands rawread and rawwrite. These commands read and write directly to contiguous sectors of a partition, allowing simple data storage and retrieval.

FAT16

The RedBoot load command can call the disk module, which calls a file system module determined by the partition type. Implementing standard file systems calls allows the high level operating system to reconfigure the bootloader by modifying configuration files.

To support Windows systems, a FAT file system [1] was implemented in fs/fat32.c. Currently this module has only supports FAT16 reads with DOS filenames.

Enhanced Scripting

Scripting was enhanced by adding a pass/fail return value to all the commands, defined in redboot.h. A new script command was then created, in script.c, which parses an area of virtual memory.

The executed script contains a series of RedBoot commands which are executed one line at at time.

It also supports the following special characters:

=
The line following a = are to be echoed to the screen and not executed.

#
The line following a # are a comment and are ignored

?*jumplen*
If the command following these two special characters succeeds, ignore the next *jumplen* (1-9) commands (echoes and comments don't count)

&
end script

The script will terminate and resume normal execution on a & character, blank line, or syntax error.

The same interface was also used to replace the original method of executing a RedBoot script from flash, so flash scripts can now use conditionals as well.

Configuration Files

To further increase CE compatibility, the iniparse command was added in iniparse.c. Iniparse reads Windows style INI files [3] and interprets them as follows:

1. All possible variables are initialized to default values based upon what is stored in a constant structure called *param_desc.*
2. The files variable names are compared with the names defined in *param_desc.*
3. If the variable names match, they are converted to a 32 bit value based upon a conversion function specified in the *param_desc* structure.
4. The 32 bit values are written into the boot_args region for CE [3]
5. The boot arg region is parsed and RedBoots parameters are modified.

File variables with no match in *param_desc* are ignored.
Parsing ends when a line of zero length is reached or after 100 lines.
See the iniparse command in appendix A for a table of recognized variables.

CE 5.0 compatibility

By comparison with the eboot source, and debugging of some of CE's early initialization sequence, It was determined there were multiple reasons that the existing RedBoot was incapable of booting CE 5.0.

The first step to insure compatibility was to change the memory map to create a virtual=physical mapping for the RAM. In this way RedBoot can still run in the same memory region. When the image boots, it will “fall through” into the correct physical location when it turns off the MMU.

In addition to the new memory map, a *ce* command was added, which performs some boot initializations needed by CE 5.0:

- Reads a CE Binary Image file [6]
- Verifies their checksum and returns a failure if corrupt.
- Copies the image into the address location specified by the file
- Writes to the scratchpad register, telling CE not to initialize the DRAM
- Turns off the ethernet to prevent DMA corruption of the CE memory area.
- Disables Interrupts.
- Invalidates and disables the cache
- Jumps to the beginning of the CE build.

Note that it does not jump to the starting location as defined by the bin standard[6], as this address appears to be incorrect (offset by 2) in the Cirrus builds. It instead starts at the beginning of the image which contains a jump to the correct location.

Appendix A – Command Additions

This section details summarizes the commands added to the Cirrus RedBoot port, and explains their syntax.

load --

Extract data from various mediums into RAM.
This command is common to most RedBoot implementations but the available options vary. The standard Cirrus bootloader provides load through TFTP, the disk option is now provided by the IDE and File system layers.

arguments
[-v] [-d] [-r] [-m [tftp | disk]] [-h *server_IP_address*] [-f *location*] [-b *location*] [*file_name*]

-v
Display a small spinner (indicator) while the download is in progress. This is just for feedback, especially during long loads.

-d
Decompress data stream (gzip data) (untested)

-r
Raw (or binary) data. -b or -f must be used

-m tftp
Transfer data via the network using TFTP protocol, this is assumed if no -m option is given.

-m disk
Transfer data from a local disk, the file system layer to use is determined by the partition type. Currently only FAT16 formats and DOS file names are supported.

-h *server_IP_address*
`The server_IP_address argument is an IPV4 address of the TFTP server in dot-decimal notation. If this is not specified, the current global server address is used (set by flash config before ini parse and  ini variables after)`

-b *location*
Virtual Address in RAM to load the data to.

-f *location*
Virtual Address in flash to load the data. (untested)

file_name
The name of the file on the TFTP server or the local disk. Details of how this is

specified for TFTP are host-specific. For local disk files, the name must be in *disk*: *filename* format. The disk portion must match one of the disk names listed by the *disks* command.

rawread --

Raw copy data from a disk partition to RAM.
This command is new to RedBoot.

arguments
[-b `location`] [-l `length`] [`file_name`]

-b
Virtual Address in RAM to copy into.

-l
Number of bytes to copy

file_name
The name must be in *disk*: *filename* format. The disk portion must match one of the disk names listed by the *disks* command. The *filename* portion is currently ignored.

rawwrite --

Raw copy data from RAM to a disk partition.
This command is new to RedBoot.

arguments
[-b `location`] [-l `length`] [`file_name`]

-b
Virtual Address in RAM to copy from.

-l
Number of bytes to copy

file_name
The name must be in *disk*: *filename* format. The disk portion must match one of the disk names listed by the *disks* command. The *filename* portion is currently ignored.

disks --

List disk partitions that RedBoot has identified.

arguments
none

script --

Execute a script of RedBoot commands.
See the *Enhanced Scripting* section for a more detailed explanation and Appendix B for some scripting examples.

arguments
[-b `location`]

-b
Virtual Address in RAM to begin execution from

iniparse --

Parse a Windows INI file for recognized variables.
Recognized variables are stored in the CE driver globals region (0xC0008000-0xC0020000).
RedBoot parameters are updated to the parsed values or defaults.

Variable Name	Type	Physical Address	Default	Modifies
BAUDRATE	integer	0xC000800C	9600	Baud Rate
COMPORT	integer	0xC0008010	1	Com Port
TIMEOUT	integer	0xC0008014	1	TFTP timeout
RETRY	integer	0xC0008018	1	TFTP retries
NET_IP	dot-decimal	0xC000801C	192.168.100.120	IP address
GATEWAY	dot-decimal	0xC0008020	0.0.0.0	Gateway
SER_IP	dot-decimal	0xC0008024	192.168.100.11	Server IP

arguments
[-b `location`]

-b
Virtual Address in RAM to begin parsing

ce --

Boot a ce 5.0 image stored in RAM.
The storage and execution locations must not overlap.

arguments
[-b `location`]

-b
Virtual address of the start of the binary image in RAM.

Appendix B – Sample boot scripts

Two example scripts are provided for booting CE 5.0 and Cirrus's Linux Implementation. The scripts attemp to load an image from a TFTP server. If the load succeeds the image is raw written to the second IDE partition. If the load fails the image is raw read from the same partition.

CE Boot Script

```
#Load an INI file from the first IDE partition
load -r -b 0x100000 -m disk hda1:NQX.INI
#parse the file and update boot_args and RedBoot with defaults or values found
iniparse -b 0x100000
########################TFTP########################
#display “attempt a tftp load”
=attempt a tftp load
#attempt to load NK2.bin from the TFTP server specified by defaults or INI
#jump ahead 2 command statements if this succeeds
?2 load -v -r -b 0x2000000 NK2.bin
########################FAILURE########################
=tftp unsuccessful, load the bin from the binary partition and execute
#raw read  0x1000000 bytes from the second disk partition into  0x2000000
rawread -b 0x2000000 -l 0x1000000 hda2:NK.bin
#execute the bin image at now at 0x2000000
ce -b 0x2000000
########################SUCCESS########################
=tftp success! write to disk and boot from RAM
#write  0x1000000 from the tftp location 0x2000000 to the second disk partition
rawwrite -b 0x2000000 -l 0x1000000 hda2:NK.bin
ce -b 0x2000000
&(end)
```

Linux Boot Script

```
=set tftp timeout to 1 second 1 retry
tftpconfig -t 1 -r 1
########################TFTP########################
=attempt a tftp load
?3 load -h 10.0.2.60 -v -r -b 0xa00000 ramdisk.gz
########################FAILURE########################
=tftp unsuccessful, load the RAMDISK from the binary partition and execute
rawread -b 0xa00000 -l 0x9c7418 hda2:disk
#load the kernel from a flash partition
fis load zImage
exec -r 0xa00000 -s 0x9c7418 -c "root=/dev/ram console=ttyAM"
########################SUCCESS########################
rawwrite -b 0xa00000 -l 0x9c7418 hda2:disk
fis load zImage
exec -r 0xa00000 -s 0x9c7418 -c "root=/dev/ram console=ttyAM"
&(end)
```

Appendix C – Sample INI file

```
; ------------------------------------------------------
; Debug
COMPORT = 1
; baudrate 9600, 19200, 38400, 57600, 115200
BAUDRATE = 57600
; ------------------------------------------------------
; Network
NET_IP = 10.0.2.30
; Subnet Mask
SUBNET = 255.255.255.0
;
; Default Gateway
GATEWAY = 10.0.2.1
;
; TFTP Server IP Address
SER_IP = 10.0.2.60
; ---------------------------------
; The amount of Time between retries
TIMEOUT = 3
; Total Number of Retries for the device
RETRY = 3
UNRECOGNIZEDVAR1 = 20
UNRECOGNIZEDVAR1 = 5
Unrecognized statements are ignored.
Parsing will end after a 100 lines or a zero length line.
```

References

[1] Microsoft Extensible Firmware Initiative FAT32 File System Specification
http://www.microsoft.com/whdc/system/platform/firmware/fatgen.mspx

[2] RedBoot Users Guide
http://ecos.sourceware.org/docs-latest/redboot/redboot-guide.html

[3] INI file
http://en.wikipedia.org/wiki/INI_file

[4] Creating Driver Globals and Boot Args
http://msdn2.microsoft.com/en-us/library/ms903586.aspx

[5] Cirrus CE Forum:Does not boot downloaded nk.nb0 on EDB9315A
http://arm.cirrus.com/forum/viewtopic.php?t=631

[6] Window CE Binary Image Data Format
http://msdn2.microsoft.com/en-us/library/ms924510.aspx

[7] Scian's Das-Uboot CE 5.0 patch: Re: [U-Boot-Users] Win CE?
http://sourceforge.net/mailarchive/message.php?msg_id=11324955

[8] OEM Adaptation Layer
http://msdn2.microsoft.com/en-us/library/aa917005.aspx

[9] Cirrus EP9315
http://www.cirrus.com/en/products/pro/detail/P1052.html

[10] Cirrus AN254 -
Booting Windows CE Using RedBoot from a Windows-Based Host PC
(no longer available, see [5])

[11] EDB9315A Technical Reference Manual:
Appendix B. Programming Linux Images into Flash from a Windows® PC
http://www.cirrus.com/en/pubs/manual/EDB9315A_Tech_Ref_Manual.pdf