Milkymist BIOS reference manual

Sébastien Bourdeauducq

June 2010

1 Presentation

Using an "execute-in-place" schema, the system boots from the NOR flash, mapped at the reset vector for the softcore CPU. It contains the BIOS which is in charge of booting the board as well as providing debugging features.

The Milkymist BIOS is based on FreeMAC (`http://lekernel.net/prism54/freemac.html`), originally written for running on the ARM embedded processors of Prism54 Wi-Fi cards. It has undergone major modifications since then.

The BIOS loads firmware from an external source (memory card, Ethernet network and UART). The BIOS supports direct booting of Linux kernels. If no suitable boot medium is found, it starts a debug shell on the UART, which allows to do basic system operations such as writing to registers. The `help` command lists the commands available with the BIOS release you are currently using.

All transmissions over the UART are made at 115200 bps or 230400 bps, with 8 bits per character, without parity, and with 1 stop bit. The second DIP switch on the board enables the 230400 bps rate when it is set, which is convenient when transferring large firmware images such as Linux kernels.

2 Starting Linux

The BIOS can directly start versions of Linux specifically made for running on Milkymist boards. Such modified Linux kernels are distributed separately.

- **The Linux kernel itself** is always loaded at the beginning of the SDRAM (0x40000000) and executed from there.
- **Kernel command line parameters** are written to SDRAM and their address is set in CPU register `r1` before the kernel is started.
- **Initial ramdisk (initrd)** is optionally loaded into SDRAM and its start and end addresses are set in CPU registers `r2` and `r3`, respectively.

3 Boot from the UART

The first boot medium tried by the BIOS is the UART, using a special protocol. This enables firmware replacement during development by just connecting the serial cable and not switching a memory card between a computer and the device.

The device initiates an UART boot session by sending the ASCII string `sL5DdSMmkekro` followed by a carriage return. This string is completely random, but has been encoded in ASCII so it does not garble the output of terminals which are not aware of the boot protocol.

Upon reception of this magic string, the UART boot program running on the host computer acknowledges it by returning the ASCII string `z6IHG7cYDID6o` followed by a carriage return. This puts the device in firmware reception mode, and further commands are sent in binary using the protocol described below.

The host sends command to the device, which acknowledges them or reports errors by sending one ASCII character after each command. Those characters and their meanings are listed in the table below:

`K`	Command successful
`C`	CRC error, retry the command
`U`	Unknown command ID
`E`	Generic error

The host must wait for a reply from the device before sending the next command. However, it can retry the command after a timeout.

All the commands sent by the host are frames using this common format:

Length	**CRC16**	**Command ID**	**Payload**
1 byte	2 bytes	1 byte	0-255 bytes

(As in the whole document, all multi-byte numbers are big-endian unless otherwise specified)

The length field is the length of the payload. The CRC16 is computed on the command ID followed by the payload.

The following commands are implemented:

Command ID	**Description**
0x00	Abort session. No payload. The BIOS continues to the next boot medium.
0x01	Load to memory. The payload is a 32-bit start address followed by the data.
0x02	Jump. The payload is the 32-bit jump address.
Linux-specific commands	
0x03	Set Linux kernel command line parameters address. The payload is the 32-bit address. This address is written to the CPU register r1 before the kernel is started (using a jump command).
0x04	Set Linux kernel initrd start address. The payload is the 32-bit address. This address is written to the CPU register r2.
0x05	Set Linux kernel initrd end address. The payload is the 32-bit address. This address is written to the CPU register r3.

4 Network boot

To debug large pieces of software, such as Linux, it is possible to boot from the network using TFTP. The board has the fixed IP address 192.168.0.42 and tries to connect to the TFTP server at 192.168.0.14. It attempts to download the files `boot.bin`, `cmdline.txt` and `initrd.bin`, which are handled in the same way as when doing a filesystem boot.

5 Boot from the memory card

If the device has not received an acknowledgement for UART boot after a certain period of time, it then attempts to boot from the memory card.

For this purpose, it searches for a file named `BOOT.BIN` on the first FAT partition. It then loads it into the beginning of the SDRAM, and, if this was successful, transfers execution to the SDRAM.

Linux support. If the filesystem also contains files named `CMDLINE.TXT` and `INITRD.BIN`, they are loaded into SDRAM at offsets `0x1000000` (16MB) and `0x1002000` (16MB + 8KB) respectively. The addresses are then passed to the kernel (loaded from `BOOT.BIN` or `ALTBOOT.BIN`) using CPU registers `r1`, `r2` and `r3` (see above). The files must be small enough so that they do not overlap in memory.

Copyright notice