Z-turn Board Linux 开发手册

版本 V1.2

2015 年 4 月 22 日

版本记录

版本号	说明	时间
V1.0	初始版本	2014/12/4
V1.1	更新图片 4-1	2014/12/16
V1.2	兼容 XC7Z020	2015/4/22

目 录

第 1 章 产品概述

1.1 产品简介

Z-turn Board 是深圳市米尔科技有限公司推出的一款以 Xilinx Zynq-7010/7020 作为主处理器的嵌入式开发板。Z-turn Board 采用 Xilinx 最新的基于 28nm 工艺流程的 Zynq-7000 All Programmable SoC 平台，将 ARM 处理器和 FPGA 架构紧密集成。该产品拥有双核 ARM Cortex-A9 MPCore 的高性能，低功耗特性，在设计中能更好的满足各种工业需要。

第 2 章 概述及软件资源介绍

2.1 概述

Z-turn Board 提供了丰富的系统资源和软件资源，本手册将从环境搭建开始，一步步介绍如何进行 Z-turn Board Linux 开发。本手册中开发主机上的命令以 Ubuntu 为例进行讲解。

2.2 软件资源

类别	名称	备注	源码
Tool chains	gcc 4.6.1	gcc version 4.6.1 (Sourcery CodeBench Lite 2011.09-50)	
Boot loader	boot.bin	一级引导程序 包括 FSBL 和 u-boot	Yes
Linux Kernel	Linux 3.15.0	专为 Z-turn Board 的硬件制定的 Linux 内核	Yes
Driver	USB OTG	USB OTG 驱动	Yes
	Ethernet	千兆以太网驱动	Yes
	MMC/SD/TF	MMC/SD/TF 卡驱动	Yes
	CAN	CAN 驱动	Yes
	LCD Controller	XYLON LCD 屏驱动	Yes

类别	名称	备注	源码
	HDMI	HDMI 驱动	Yes
	Button	Button 驱动	Yes
	UART	串口驱动	Yes
	LED	LED 驱动	Yes
	GPIO	GPIO 驱动	Yes
	Buzzer	蜂鸣器驱动	Yes
	G-Sensor	三轴传感器驱动	Yes
	Tempreture Sensor	温度传感器驱动	Yes
File system	Ramdisk	Ramdisk 系统镜像	
	Ubuntu Desktop 12.04	tar 归档文件和 SD 卡烧写镜像	

表 1-1

第 3 章 Linux 开发环境搭建

3.1 建立工作目录

拷贝 Z-turn Board 光盘中的资料到主机中：

```
$ mkdir -p <WORKDIR>
$ cp -a <DVDROM>/04-Linux_Source/* <WORKDIR>
```

3.2 设置交叉编译工具

```
$ cd <WORKDIR>/Toolchain
$ tar -xvjf Sourcery_CodeBench_Lite_for_Xilinx_GNU_Linux.tar.bz2
$ export PATH=$PATH:<WORKDIR>/Toolchain/\
CodeSourcery/Sourcery_CodeBench_Lite_for_Xilinx_GNU_Linux/bin
$ export CROSS_COMPILE=arm-xilinx-linux-gnueabi-
```

执行完“export”命令后输入 arm 按 Tab 键来检查是否设置成功，该设置只对当前终端有效，如需永久修改，请将以上 export 命令添加到用户启动脚本文件：~/.bashrc。

3.3 安装工具

安装其他必要工具：

```
$ sudo apt-get install build-essential git-core libncurses5-dev \
flex bison texinfo zip unzip zlib1g-dev gettext \
gperf libsdl-dev libesd0-dev libwxgtk2.6-dev \
uboot-mkimage \
g++ xz-utils
```

第 4 章 Linux 系统编译

4.1 编译 Bootloader

进入 Bootloader 目录，解压 U-boot 源码：

```
$ cd <WORKDIR>/Bootloader
$ tar -jxvf u-boot-xlnx.tar.bz2
$ cd u-boot-xlnx
```

开始编译：

```
$ make ARCH=arm CROSS_COMPILE=arm-xilinx-linux-gnueabi- distclean
$ make ARCH=arm CROSS_COMPILE=arm-xilinx-linux-gnueabi- \
zynq_zturn_config
$ make ARCH=arm CROSS_COMPILE=arm-xilinx-linux-gnueabi-
```

编译完成后，在当前目录下会生成“u-boot”ELF 文件。使用此镜像制作 boot.bin 时需要重命名为“u-boot.elf”。

4.2 编译 Linux 内核

进入 Kernel 目录，解压内核源码：

```
$ cd <WORKDIR>/Kernel
$ tar -xvjf linux-xlnx.tar.bz2
$ cd linux-xlnx
```

开始编译：

```
$ make ARCH=arm CROSS_COMPILE=arm-xilinx-linux-gnueabi- distclean
$ make ARCH=arm CROSS_COMPILE=arm-xilinx-linux-gnueabi- \
zynq_zturn_defconfig
$ make ARCH=arm CROSS_COMPILE=arm-xilinx-linux-gnueabi- uImage
$ make ARCH=arm CROSS_COMPILE=arm-xilinx-linux-gnueabi- dtbs
```

编译完成后，会在 arch/arm/boot 目录下生成 uImage 文件，在 arch/arm/boot/dts/生成 zynq_zturn.dtb 文件。

4.3 制作文件系统

4.3.1 制作 ramdisk

⑴ 挂载 Ramdisk

新建目录 tmp，并将 uramdisk.image.gz 拷贝至该目录

```
$ cd <WORKDIR>/Filesystem
$ mkdir tmp
$ cp uramdisk.image.gz tmp/
$ cd tmp/
```

去掉 mkimage 生成的 64 bytes 的文件头，生成新的 ramdisk.image.gz

```
$ dd if=uramdisk.image.gz of=ramdisk.image.gz bs=64 skip=1
```

gunzip 解压 ramdisk.image.gz 生成 ramdisk.image

```
$ gunzip ramdisk.image.gz
```

新建挂载目录“rootfs”，并将 ramdisk.image 挂载

```
$ mkdir -p rootfs
$ sudo mount -o loop,rw ramdisk.image rootfs
```

进入 rootfs 目录，根据需要做修改

⑵ 重新生成 ramdisk

同步文件系统并卸载 ramdisk

```
$ sync
$ sudo umount rootfs
```

用 gzip 压缩 ramdisk.image，生成 ramdisk.image.gz

```
$ gzip -9 ramdisk.image
```

用 mkimage 添加文件头，生成新的 uramdisk.image.gz 供 u-boot 使用

```
$ mkimage -A arm -T ramdisk -C gzip -n Ramdisk -d ramdisk.image.gz
uramdisk.image.gz
```

删除临时文件 ramdisk.image.gz

```
$ rm ramdisk.image.gz
```

4.3.2 制作 SD 卡镜像

SD 卡镜像 xillinux-1.3.img 在 linux 下可以直接挂载并做修改，修改前应该做好备份。

xillinux-1.3.img 镜像分为两个分区 part0 和 part1，其中 part0 是 FAT32 格式，part1 是 ext4 格式。要挂载这两个分区首先需要计算这两个分区的 offset，然后传入对应的 offset 给

mount 命令再进行挂载。

⑴ 新建目录 sdimage，并解压 sd 卡镜像文件 xillinux-1.3.img.gz 到此目录。

```
$ cd <WORKDIR>/Filesystem
$ mkdir sdimage
$ cd sdimage
$ gunzip xillinux-1.3.img.gz
```

⑵ 查看 xillinux-1.3.img 的分区情况，并计算对应分区的 start offset。

```
$ fdisk –l -u xillinux-1.3.img
Disk xillinux-1.3.img: 0 MB, 0 bytes
255 heads, 63 sectors/track, 0 cylinders, total 0 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000

          Device Boot      Start         End      Blocks   Id  System
xillinux-1.3.img1              63       32129       16033+   b  W95 FAT32
xillinux-1.3.img2           32130     3534299     1751085   83  Linux
```

通过 “Units = sectors of 1 * 512 = 512 bytes” 得知分区表中一个 sector 的大小为 512 bytes，结合“Start”域的数值计算第一个分区的 offset 为：63 * 512 = 32256 bytes，第二个分区的 offset 为：32130 * 512 = 16450560 bytes。

⑶ 挂载 sd 镜像中的 part0 和 part1

新建挂载点

```
$ mkdir -p part0 part1
```

根据上一步的 offset 挂载 part0

```
$ sudo mount -o rw,offset=32256 xillinux-1.3.img part0/
```

挂载 part1

```
$ sudo mount -o rw,offset=16450560 xillinux-1.3.img part1/
```

至此已完成sd镜像中两个分区的挂载，其中part0对应的是FAT32格式的BOOT分区，part1 为 EXT4 格式的文件系统分区。

⑷ 修改文件系统

根据需要进入相应的分区可进行修改，所做的修改将直接保存到 xillinux-1.3.img 文件中。

⑸ 卸载 part0 和 part1 分区，并将修改同步到 xillinux-1.3.img

```
$ sync
$ sudo umount part0 part1
```

新的镜像 xillinux-1.3.img 便可直接用 dd 命令烧写到 SD 卡中。

第 5 章 Linux 系统烧写

提供两种镜像，一个 ubuntu 桌面系统，另一个不带界面的 ramdisk 文件系统，与存储介质关系如下表所示，用户可以根据需要选择。

	Ramdisk	Ubuntu 12.04 Desktop
Qual-SPI Flash	支持	不支持
SD Mrico Card	支持	支持，要求容量 2GB 或 2GB 以上

表 4-1

5.1 烧写 Ramdisk 镜像到 Micro SD（TF）卡

⑴ TF 卡格式化

请使用光盘目录 03-Tools 目录下的 HP USB Disk Storage Format Tool 2.0.6 工具格式化 TF 卡。

① 把 TF 卡插入 USB 读卡器，然后将读卡器跟电脑连接

② 打开 HP USB Disk Storage Format Tool，出现类似提示如下：

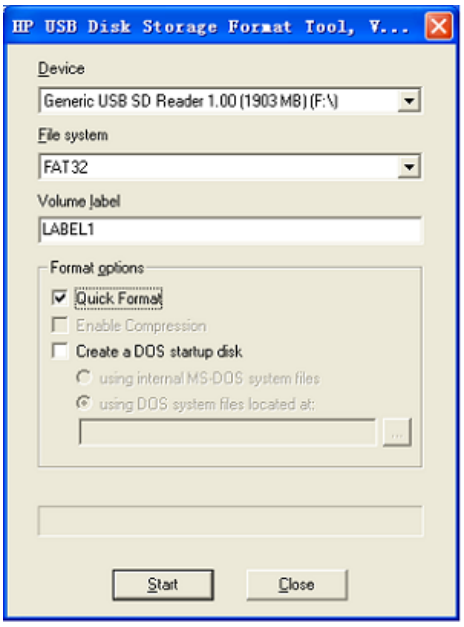

图 4-1

③ 选择“FAT32”系统格式

④ 点击“Start”

⑤ 等待格式化完成，点击“OK”

注意：HP USB Disk Storage Format Tool 会将清除 TF 卡上所有的数据，格式化前请做好备份。

⑵ 映像更新

将所需更新的镜像文件拷贝到 TF 卡上，然后将 TF 卡插入到开发板上的 SD 卡插槽，断开 JP1，连接 JP2，上电重启开发板。

5.2 烧写 Ramdisk 镜像到 Qual-SPI Flash

QSPI Flash 启动只能挂载简单的 Ramdisk 文件系统，可通过两种方法对 QSPI Flash 进行更新：通过 u-boot 命令更新 和 在 Linux 下通过脚本更新。

首先必须准备好待烧写的镜像，并拷贝到 TF 卡的 FAT 分区中：

镜像名称	说明
boot.bin	系统启动程序，包括 fsbl、和 u-boot。具体的制作方法将在《Z-turn Board 可编程逻辑开发手册》中描述
7z010.bit	XC7Z010 Bitstream 文件，U-Boot 将自动判断 SoC 型号选择加载
7z020.bit	XC7Z020 Bitstream 文件，U-Boot 将自动判断 SoC 型号选择加载
uImage	Linux 内核
devicetree.dtb	设备树文件
uramdisk.image.gz	ramdisk 文件系统

表 4-2

⑴ 在 u-boot 中更新 QSPI Flash

断开 JP1，连接 JP2，从 SD 卡启动开发板，u-boot 数秒时按下任意键进入 u-boot 命令行，运行命令“run qspiupdate”更新 QSPI Flash:

```
zynq-uboot> run qspiupdate
Update qspi images from sd card...
- Init mmc...
- Init qspi flash...
SF: Detected W25Q128BV with page size 256 Bytes, erase size 4 KiB, total
16 MiB
- Write boot.bin...
reading boot.bin
```

```
2577740 bytes read in 146 ms (16.8 MiB/s)
SF: 3145728 bytes @ 0x0 Erased: OK
SF: 3145728 bytes @ 0x0 Written: OK
- Write uImage...
reading uImage
3632464 bytes read in 201 ms (17.2 MiB/s)
SF: 5242880 bytes @ 0x300000 Erased: OK
SF: 5242880 bytes @ 0x300000 Written: OK
- Write device tree...
reading devicetree.dtb
17060 bytes read in 17 ms (979.5 KiB/s)
SF: 131072 bytes @ 0x800000 Erased: OK
SF: 131072 bytes @ 0x800000 Written: OK
- Write Ramdisk...
reading uramdisk.image.gz
5294057 bytes read in 290 ms (17.4 MiB/s)
SF: 6160384 bytes @ 0x820000 Erased: OK
SF: 6160384 bytes @ 0x820000 Written: OK
- Done.
```

更新好 SPIFlash 后，连接 JP1、JP2，开发板重新上电后便可从 QSPI 启动。

⑵ 在 Linux 系统下更新 QSPI Flash

启动开发板并进入 ramdisk 文件系统，插入已拷贝好镜像的 TF 卡，系统将自动挂载 TF 卡中的分区到/mnt 目录下，本例的 FAT 分区挂载目录为：/mnt/mmcblk0p1。运行根目录下的脚本 update_qspi.sh 更新 QSPI Flash。

```
Z-turn # ./update_qspi.sh /mnt/mmcblk0p1/
Writing boot.bin Image To QSPI Flash @/dev/mtd0
Erasing block: 6/630 (0%) random: nonblocking pool is initialized
Erasing block: 630/630 (100%)
Writing kb: 2512/2517 (99%)
Verifying kb: 2512/2517 (99%)
Writing uImage To QSPI Flash @/dev/mtd1
Erasing block: 887/887 (100%)
Writing kb: 3544/3547 (99%)
Verifying kb: 3544/3547 (99%)
Writing devicetree.dtb To QSPI Flash @/dev/mtd2
Erasing block: 5/5 (100%)
Writing kb: 16/16 (100%)
Verifying kb: 16/16 (100%)
Writing uramdisk.image.gz To QSPI Flash @/dev/mtd3
Erasing block: 1293/1293 (100%)
Writing kb: 5168/5169 (99%)
```

```
Verifying kb: 5168/5169 (99%)
QSPI flash update successfully!
```

更新好 QSPIFlash 后，连接 JP1、JP2，开发板重新上电后便可从 QSPI 启动。

5.3 烧写 Ubuntu 到 Micro SD（TF）卡

5.3.1 Windows 操作步骤

安装光盘目录下的 Win32DiskImager-0.9.5-install.exe 工具。

解压光盘目录中的中的 xillinux-1.3.img.gz 文件，得到 xillinux-1.3.img 文件，将 TF 卡插入到 PC 机，运行 Win32DiskImager，浏览并选中 xillinux-1.3.img，选中对应的 TF 盘符。然后点击“Write”按钮进行烧写，烧写成功后，会提示成功烧写。

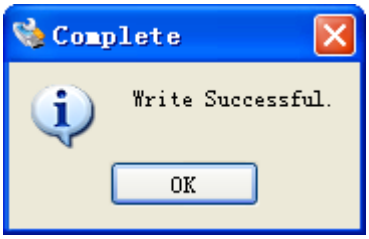

图 4-2

插入新烧写的 TF 卡到 Z-turn 开发板，断开 JP1，连接 JP2，重新上电后便可进入系统。

5.3.2 Linux 操作步骤

运行 Ubuntu，解压 xillinux-1.3.img.gz 得到 xillinux-1.3.img

```
$ gunzip xillinux-1.3.img.gz
```

插入 TF 卡并确认系统识别的 TF 卡盘符，这里以/dev/sdb 为例，若系统自动挂载分区先卸载分区

```
$ sudo umount /dev/sdb*
```

使用 dd 命令烧写镜像 xillinux-1.3.img

```
$ sudo dd if=xillinux-1.3.img of=/dev/sdb
```

完成后可尝试挂载 sdb1 和 sdb2 分区看是否成功。

插入新烧写的 TF 卡到 Z-turn 开发板，断开 JP1，连接 JP2，重新上电后便可进入系统。

第 6 章 Linux 应用程序

Z-turn Board 提供了常用外设的演示程序，程序以及源码都位于*“<WORKDIR>/Examples/”*，请根据目录内的 Makefile 或 README 文件进行编译：

```
$ cd <WORKDIR>/Examples/
$ export CROSS_COMPILE=arm-xilinx-linux-gnueabi-
$ make
```

连接调试串口 J6，设置波特率为 115200，数据位为 8，停止位为 1，无奇偶校验，将对应的可执行程序拷贝到开发板，在开发板上运行程序时需要注意权限，如果无法执行请使用如下命令：

```
# chmod +x <program-to-be-executed>
```

6.1 Led

本例程演示如何使用 Linux API 操作开发板上的 LED，详情请参考源码。

将目录*“<WORKDIR>/Examples/led-test”*中的可执行程序 led-test 拷贝至开发板，把调试串口 J6 连到 PC 上。执行以下命令后按下开发板上三色 LED、D29 和 D30 快速交替闪烁：

```
# ./led-test
[usr_led1] Get trigger: 'none'
[usr_led1] Set trigger to 'none'
[usr_led2] Get trigger: 'none'
[usr_led2] Set trigger to 'none'
[   led_r] Get trigger: 'heartbeat'
[   led_r] Set trigger to 'none'
[   led_g] Get trigger: 'heartbeat'
[   led_g] Set trigger to 'none'
[   led_b] Get trigger: 'heartbeat'
[   led_b] Set trigger to 'none'
```

6.2 Button

本例程演示如何使用 Linux API 读取开发板上的按键，详情请参考源码。

将目录*“<WORKDIR>/Examples/btn-test”*中的可执行程序 btn-test 拷贝至开发板，

把调试串口 J6 连到 PC 上。执行以下命令后，按下开发板的 K1 键，串口会输出相关信息：

```
# ./btn-test
[2972.640160] key_code: 0X66, value: 1
[2972.640160] key_code: 0, value: 0
 [2972.830048] key_code: 0X66, value: 0
[2972.830048] key_code: 0, value: 0
```

6.3 Buzzer

本例程演示如何使用 Linux API 操作开发板上的蜂鸣器，详情请参考源码。

将目录 *“<WORKDIR>/Examples/buzzer-test”* 中的可执行程序 buzzer-test 拷贝至开发板，把调试串口 J6 连到 PC 上。执行以下命令后，蜂鸣器会响，串口会输出相关信息：

```
# ./btn-test
ret: 4, capabilities: 0X2
buzzer dev is found: /dev/input/event0
```

6.4 Acceleration sensor

本例程演示如何使用 Linux API 操作开发板上的加速度传感器，详情请参考源码。

将目录 *“<WORKDIR>/Examples/gsensor-test”* 中的可执行程序 gsensor-test 拷贝至开发板，把调试串口 J6 连到 PC 上。执行以下命令后，晃动开发板，串口会输出相关信息：

```
# ./gsensor-test
[3850.779139] X:   0,   Y:   0, Z:-266
[3850.819431] X:   0,   Y:   0, Z:-279
[3850.859724] X:   0,   Y:   0, Z:-277
[3851.141865] X:   0,   Y: -17, Z:-277
[3851.504610] X:  10,   Y: -17, Z:-277
[3853.802916] X: -32,   Y:  48, Z: -64
[3853.841329] X:  95,   Y: -94, Z: -16
[3853.881621] X:  93,   Y: -80, Z:-407
[3853.921951] X:-282,   Y: 255, Z:-461
[3853.962225] X:-147,   Y: 134, Z:-272
[3854.002555] X: 254,   Y:-247, Z:-104
```

6.5 Temperature sensor

本例程演示如何在用户空间操作开发板上的温度传感器，详情请参考源码。

将目录 *“<WORKDIR>/Examples/tsensor-test ”*中的可执行程序 tsensor-test 拷贝至开发板，把调试串口 J6 连到 PC 上。执行以下命令后，串口会实时输出当前温度值：

```
# ./tsensor-test
Temperature: 35.0 C
```

附录一 售后服务与技术支持

凡是通过米尔科技直接购买或经米尔科技授权的正规代理商处购买的米尔科技全系列产品，均可享受以下权益：

1、6 个月免费保修服务周期

2、终身免费技术支持服务

3、终身维修服务

4、免费享有所购买产品配套的软件升级服务

5、免费享有所购买产品配套的软件源代码，以及米尔科技开发的部分软件源代码

6、可直接从米尔科技购买主要芯片样品，简单、方便、快速；免去从代理商处购买时，漫长的等待周期

7、自购买之日起，即成为米尔科技永久客户，享有再次购买米尔科技任何一款软硬件产品的优惠政策

8、OEM/ODM 服务

如有以下情况之一，则不享有免费保修服务：

1、超过免费保修服务周期

2、无产品序列号或无产品有效购买单据

3、进液、受潮、发霉或腐蚀

4、受撞击、挤压、摔落、刮伤等非产品本身质量问题引起的故障和损坏

5、擅自改造硬件、错误上电、错误操作造成的故障和损坏

6、由不可抗拒自然因素引起的故障和损坏

产品返修：用户在使用过程中由于产品故障、损坏或其他异常现象，在寄回维修之前，请先致电米尔科技客服部，与工程师进行沟通以确认问题，避免故障判断错误造成不必要的运费损失及周期的耽误。

维修周期：收到返修产品后，我们将即日安排工程师进行检测，我们将在最短的时间内维修或更换并寄回。一般的故障维修周期为 3 个工作日（自我司收到物品之日起，不计运

输过程时间），由于特殊故障导致无法短期内维修的产品，我们会与用户另行沟通并确认维修周期。

维修费用：在免费保修期内的产品，由于产品质量问题引起的故障，不收任何维修费用；不属于免费保修范围内的故障或损坏，在检测确认问题后，我们将与客户沟通并确认维修费用，我们仅收取元器件材料费，不收取维修服务费；超过保修期限的产品，根据实际损坏的程度来确定收取的元器件材料费和维修服务费。

运输费用：产品正常保修时，用户寄回的运费由用户承担，维修后寄回给用户的费用由我司承担。非正常保修产品来回运费均由用户承担。

购买请联系：

电话：0755-25622735

传真：0755-25532724

邮箱：sales@myirtech.com

网站：www.myir-tech.com

技术支持请联系：

电话：0755-25622735

传真：0755-25532724

邮箱：support@myirtech.com

网站：www.myir-tech.com