文章编号:1671-4598(2017)08-0120-03 DOI:10.16526/j.cnki.11—4762/tp.2017.08.031 中图分类号:TP273 文献标识码:A

基于 VxWorks 新型映像的三模冗余启动机制研究

邹玉龙，刘 彬，田小莉，郭宗芝，牛 崇
（上海微小卫星工程中心，上海 201210）

摘要：针对 VxWorks 启动映像依赖于 ROM 地址，不能适应航天型号中三模存储要求的缺点，利用 VxWorks 自带的压缩算法，按照航天型号软件任务要求设计了一种基于 VxWorks 内存型映像的新型压缩映像，实现了软件映像与 ROM 区地址无关；并且实现了新型压缩映像的三模冗余引导启动，启动时每一个 bit 数据都从三份映像获取，并进行三取二判断，将最后结果拷贝到内存作为目的映像的 bit 数据，目的映像启动后能够回写正确数据自动纠正空间环境下可能发生的单粒子翻转错误，为软件的可靠启动建立了基础。

关键词：VxWorks；三模冗余；压缩映像；引导启动

Research on Triple Modular Redundancy Boot Mechanism Based on New Type VxWorks Compressed Image

Zou Yulong, Liu Bin, Tian Xiaoli, Guo Zongzhi, Niu Chong
(Shanghai Engineering Center for Microsatellites, Shanghai 201210, China)

Abstract: Aerospace Engineering often requires triple modular redundancy (TMR). While VxWorks bootable image is ROM address relevant and it cannot be used as a TMR type image directly. The new compressed image based on VxWorks compressing algorithm is independent of ROM address. The image can be copied exactly to store in three ROM areas. When system booting, each bit data copied to RAM to make up a complete image is got by comparing three bits in the same position in three same image stored in ROM area. After software boot, the one bit type error which is usually caused by Single Event Upset (SEU) can be corrected by writing back the correct data in image. The software boot reliability benefits from the new type image and TMR booting mechanism greatly.

Keywords: VxWorks; triple modular redundancy (TMR); compressed image; bootloader

0 引言

航天型号软件任务中，ROM 存储空间紧张，要求应用映像大小尽量小，启动时间尽量短，另外对于应用映像的启动可靠性也有较高的要求，对软件映像一般会要求进行三模冗余存储。VxWorks 操作系统由于良好的实时性，稳定性和可裁剪性，在航天领域得到了广泛的应用。但是 VxWorks/Tornado 提供的启动机制并没有直接支持三模冗余，这给软件设计工作带来了一定的困难。设计出一种应用映像占用存储空间小，启动时间短的三模冗余启动机制成为航天型号软件任务中的关键技术要求。

1 VxWorks/Tornado 启动机制分析

VxWorks/Tornado 开发过程中使用的启动机制有如下几种[1]。

1.1 有引导映像的启动机制

1）ROM 区仅存储一份只包含引导功能的引导映像，引导映像往往比较小，启动速度比较快。在引导映像正常启动后，将应用映像再通过其他途径（如网络端口，调试串口端口等）下载到 RAM 中，然后进行一次地址跳转，从 RAM 中开始运行应用映像。由于通过网络端口或者调试串口下载应用映像速度比较快，并且避免了多次烧写 ROM 区而影响 ROM 器件的寿命，在软件开发阶段，往往通过这种机制进行软件调试，提高了软件开发的效率。

2）ROM 区存储一份引导映像和一份未压缩的应用映像，系统上电后首先从引导映像开始运行，在引导映像正常启动后，将应用映像从 ROM 拷贝到 RAM 中，然后从 RAM 中开始运行应用映像。这种机制一般适用于 ROM 存储空间足够并且系统启动时间及稳定性要求不高的场景。

3）ROM 区存储一份引导映像和一份压缩的应用映像，在引导映像正常启动后，将应用映像从 ROM 解压到 RAM 中，解压功能在引导映像中。然后从 RAM 中开始运行应用映像。这种机制一般适用于 ROM 存储空间紧张，但系统启动时间及稳定性要求不高的场景。

1.2 没有引导映像的启动机制

1）ROM 区存储一份未压缩的应用映像，启动时直接从 ROM 区应用映像的代码段开始运行，应用映像将自身拷贝到 RAM 区中，然后从 RAM 区开始运行。这种机制适用于 ROM 存储空间充足的场景。

2）ROM 区存储一份压缩的应用映像，启动时直接从 ROM 区应用映像的代码段开始运行，应用映像将自身解压到 RAM 区中，然后从 RAM 区开始运行。这种机制适用于 ROM 存储空间紧张但对启动速度要求不高的场景。

3）ROM 区存储一份驻留型的应用映像，启动时直接从 ROM 区应用映像的代码段开始运行，应用映像仅将数据段拷贝 RAM 区中。由于软件代码段是在速度比较慢的 ROM 中，软件运行时需要进行比较多的 IO 操作，导致软件运行速度下降。这种机制适用于对运行速度要求不高但 RAM 空间紧张的的场景。

收稿日期:2017-02-17； **修回日期**:2017-03-03。

作者简介:邹玉龙(1984-),男,江苏东台人,硕士研究生,工程师,主要从事卫星软件技术方向的研究。

2　应用场景分析

以某型号 CPU 软件任务为例，软件任务要求应用映像进行三模冗余存储，并且软件启动时间在 10 秒之内，软件启动后能够对存储区域进行自检，如果发现 bit 数据被打翻，则利用三模冗余对错误数据进行纠错。程序存储器件分为 PROM 和 EEPROM 两种，PROM 大小为 32 KB，EEPROM 大小为 1 MB。

将整个系统映像分为引导映像和应用映像两部分。引导映像存储于 PROM 中，不可修改。应用映像存储于 EEPROM 中，可修改。由 Tordando 开发环境编译之后的二进制应用映像大小接近 500 kB，直接对编译后的应用映像三模冗余存储需要 1.5 MB 的存储空间，超过了可用的存储空间大小，这就要求对软件映像进行压缩处理后再进行三模冗余存储。虽然 VxWorks/Tornado 提供的压缩映像启动机制并没有支持三模冗余，但是其压缩解压过程和压缩解压工具和算法有一定的参考价值。

3　新型压缩映像制作

利用 VxWorks/Tornado 的压缩算法对原始应用映像进行压缩，压缩后的应用映像大小为 200 kB 左右，这样应用映像大小足够小，满足了三模冗余存储的要求。由于 VxWorks/Tornado 提供的压缩映像技术是 ROM 地址相关的，不能直接存储三份在 EEPROM 中。我们将 VxWorks/Tornado 的压缩和解压缩功能提取出来，单独对编译后的 VxWorks 原始映像进行压缩，并且将解压缩功能编译到压缩后的应用映像中，使得压缩映像在运行时可以自解压。并且设置压缩后的映像的代码段起始地址为 RAM 空间中某个高地址[2]，这样就做到了压缩映像的起始运行地址与 ROM 地址无关。软件映像可以存储在 EEPROM 中的任意地址处，并且可以复制多份进行冗余存储。

压缩镜像的制作步骤如下：

1）首先将 Tornado 应用工程编译，得到可以直接在 RAM 运行的 vxWorks 原始文件。这一步通过 Tornado 软件界面操作进行。

2）将 Tornado 应用工程编译后的 vxWorks 文件转换为二进制文件格式。其中%AOUTTOBIN1%为 Tornado 开发环境提供的 aoutToBin.exe 软件工具。

命令：

%AOUTTOBIN1% <vxWorks> tmp.bin

3）对 tmp.bin 进行压缩处理。其中%DEFLATE1%为 Tornado 开发环境提供的 deflate.exe 软件工具。

命令：%DEFLATE1% <tmp.bin>tmp.Z

4）将压缩后的 tmp.Z 转换成汇编源文件格式。其中%BINTOASM1%为 Tornado 开发环境提供的 binToAsm.exe 软件工具。

命令：

%BINTOASM1% tmp.Z > vxWorks.Z.s

5）将 vxWorks.Z.s 编译成目标文件。其中%CC1%为 Tornado 开发环境提供的 ccsparc.exe 软件工具。%CFLAGS%为“-g -ansi -nostdinc -DRW_MULTI_THREAD -D_REENTRANT -fvolatile -fno-builtin -DCPU=SPARC -DPRJ_BUILD -g -O0 -Wall -DFLASH_TEST”编译选项。%WIND_BASE%为 Tornado 开发环境的根目录。

命令：

%CC1% %CFLAGS% -I%WIND_BASE%\target\h -c vxWorks.Z.s -o vxWorks.Z.o

6）将解压文件 uncompress.c 编译成目标文件。其中%CC1%，%CFLAGS%与%WIND_BASE%含义同（5）。

命令：

%CC1% %CFLAGS% -I%WIND_BASE%\target\h -c uncompress.c -o uncompress.o

7）将初始化配置文件 start_up.s 编译成目标文件。其中%CC1%，%CFLAGS%与%WIND_BASE%含义同（5）。

命令：

%CC1% %CFLAGS% -c start_up.s - o start_up.o

8）对目标文件进行链接。使得新的应用映像为带硬件初始化功能并且能够自解压的应用映像。RAM_HIGH_ADRS 为 RAM 高地址。其中%LD1%为 Tornado 开发环境提供的 ldsparc.exe 软件工具

命令：%LD1% -n -N -nostartupfiles -static -e _start -Map mapfile -Ttext RAM_HIGH_ADRS start_up.ouncompress.o vxWorks.Z.o %WIND_BASE%\target\lib\libSPARCgnuvx.a -o selfUnCmp

9）将链接后的文件转换成二进制可执行文件。其中%OBJCOPY1%为 Tornado 开发环境提供的 objcopysparc.exe 软件工具。

命令：% OBJCOPY1% - O binary selfUnCmpselfUnCmp.bin

其中 uncompress.c 为自解压模块代码。关键代码如下。Inflate () 为 vxWorks 的解压缩函数。RAM_DST_ADRS 为解压后的程序映像的起始运行位置，也称 RAM 低地址。解压缩模块首先将压缩映像解压到 RAM_DST_ADRS 地址处，然后通过 absEntry (0) 进行跳转到 RAM_DST_ADRS 处运行解压后的映像[3]。

```
void uncompress()
{
volatile FUNCPTR absEntry = (volatile FUNCPTR)RAM_DST_ADRS;
if(inflate (binArrayStart,
RAM_DST_ADRS,
&binArrayEnd - binArrayStart)! =0)
return;
/* and jump to it */
absEntry (0);
}
```

4　三模引导启动过程设计

在 EEPROM 中存储三份一样的压缩映像，并且由于解压过程是在 RAM 中进行，大大提高了解压速度，满足 10 秒内启动的要求。这三份压缩映像由引导程序选择合成一份正确的拷贝到 RAM 空间指定的高地址处，然后从该高地址处开始运行。

由于引导映像是存储在 PROM 中，不可更改，并且 PROM 可用空间比较小，所以引导映像的功能必须尽可能的简单。除了基本的硬件初始化外，引导映像只做一件事情，就是将存储在 EEPROM 中的三模冗余应用映像合成一份正确的

拷贝到 RAM 某高地址处，最后跳转到该 RAM 高地址处运行[4]。为了有效利用三模冗余的应用映像，引导映像在进行三取二选择时，首先简单的对三模的三个字节进行相等比较，然后取两个字节相等的作为判定结果，如果发生了同一地址处的 3 个映像的字节都不相等的情况时，按照 bit 级进行判定。例如拷贝应用映像的某个字节数据时，对这一字节的每一个 bit 进行三取二的判定，由于 bit 只有 0 和 1 两种情况，这样就避免了三个字节都不相等而无法判定的问题，提高了系统启动的可靠性，即只要不是某 bit 位两份映像或者三份映像都发生错误，即使三个字节都不相等，都能判定出正确的数据。

引导启动过程核心代码如下。

```
//三模拷贝压缩映像到内存区。
for(cnt=0u; cnt<IMAGE_SIZE; cnt = cnt+1u)
{
des = (UINT8 *)(RAM_HIGH_ADDRESS + cnt);
dtA = IO_READB(IMAGE_FIRST_ADDRESS + cnt); /* get three image addr value */
dtB = IO_READB(IMAGE_SECOND_ADDRESS + cnt);
dtC = IO_READB(IMAGE_THIRD_ADDRESS + cnt);
judgeImgValue(cnt, des, dtA, dtB, dtC, &errorByteCnt); /* compare three image value */
}
//判断 3 个字节是否相同并记录错误字节数
void judgeImgValue(const UINT32 index, UINT8 *des, UINT8 dtA, UINT8 dtB, UINT8 dtC, UINT8 *Errordes)
{
if((dtA == dtB)&&(dtB == dtC))
{
*des = dtA;
}
else if((dtC == dtB)&&(dtB != dtA)) /* A Error */
{
*des = dtB;
saveErrorFlag(Errordes,index);
}
else if((dtA == dtC)&&(dtC != dtB)) /* B Error */
{
*des = dtA;
saveErrorFlag(Errordes,index);
}
else if((dtA == dtB)&&(dtB != dtC)) /* C Error */
{
*des = dtA;
saveErrorFlag(Errordes,index);
}
else /* three code not same */
{
*des = TMR8Bit(&dtA, &dtB, &dtC);
saveErrorFlag(Errordes,index);
}
}
//如果 3 个字节都不相同则调用该函数进行 bit 判定
UINT8 TMR8Bit(UINT8 *first, UINT8 *seconde, UINT8 *third)
{
UINT8 nCnt = 0;
UINT8 nTmp = 0u;
UINT8 nTmp1 = 0u;
UINT8 nTmp2 = 0u;
UINT8 nTmp3 = 0u;
UINT8 nResult = 0u;
for(nCnt = 0; nCnt < 8; nCnt = nCnt + 1)
{
nTmp1 = (((*first)>>nCnt)&0x1);
nTmp2 = (((*seconde)>>nCnt)&0x1);
nTmp3 = (((*third)>>nCnt)&0x1);
if((nTmp1 == nTmp2)||(nTmp1 == nTmp3))
{
nTmp = nTmp1;
}
else
{
nTmp = nTmp2;
}
nTmp = nTmp << nCnt;
nResult = nResult | nTmp;
}
return nResult;
}
```

应用映像启动后需要进行自检，对于 EEPROM 区域中可能出现的数据存储错误进行纠正，这时可以利用 RAM 区中已经存在的正确软件映像进行纠错。如果将 RAM 区中压缩映像的每一字节与 EEPROM 中每一压缩映像的字节进行比较，如果不一致，则用 RAM 区中的正确数据覆盖 EEPROM 中的错误数据，虽然这样也可以达到纠正错误数据的目的，但是比较耗时，影响了实时系统的启动速度。实际工程中为了提高启动速度，在引导启动过程中将 EEPROM 中的软件映像数据向 RAM 区进行拷贝时记录了出错的数据的字节下标，在系统启动后通过检查这些错误字节下标，就可以有针对性的对发生错误的字节进行修复，而不用再次检查一遍所有 3 份软件映像的数据正确性。如果没有错误，则直接往下运行程序，避免了无意义的重复检查错误过程。为了软件能够长期稳定运行，设置了关于 EEPROM 中三份应用映像的 CRC 校验值作为遥测数据，软件运行时定期对 EEPROM 中应用映像进行 CRC 校验，当遥测数据中观测到三份应用映像的 CRC 校验数据发生改变时，可以通过特定的指令控制应用软件进行自检，及时纠正错误数据，避免由于长期不维护，造成 EEPROM 中三模冗余应用映像 bit 错误过多而无法自主修复。整个启动过程如图 1 所示。

5 测试与验证

5.1 正确映像启动

1）测试过程：将正确的应用层软件映像拷贝到 EEPROM 的 3 个不同位置，接着从引导地址 0x00000000 处启动后，观察调试串口的输出，并记录启动过程所需的时间。

2）测试结果：调试串口正常输出运行信息，没有错误记录报告，整个系统启动时间约为 8 s，符合要求。

5.2 同一位置 1 个字节错误类型映像启动

1）测试过程：对 3 份应用层映像分别在第 1 个映像的第

（下转第 126 页）

选系统移植花生分级等算法奠定了基础。常用的几何特征量有：面积、周长、形心、长轴和短轴。

图 9 的分辨率为 2 048×1 800，根据轮廓上所有点的坐标值及上面的算法来计算花生的几何参数。计算的结果如表 2 和 3 所示。

表 2 花生形心坐标表 像素点

标号	1	2	3	4	5	6	7	8	9
X	313.40	391.17	501.05	857.95	895.90	839.33	1039.77	1310.28	1537.36
Y	1610.71	885.15	327.63	833.89	298.27	1180.36	1442.74	602.44	1312.93

表 3 花生几何参数表 像素点

标号	1	2	3	4	5	6	7	8	9
面积	52698	40845	40634	46792	59746	34820	53605	47649	59492
周长	974.85	834.61	801.55	859.11	1029.60	707.51	953.41	873.15	990.95
长轴	368.91	326.51	287.59	323.39	394.16	228.88	364.04	324.98	382.69
短轴	185.70	160.81	180.71	184.70	194.14	194.25	189.92	187.50	199.80

在不同的生产需求中，可依据上表 3 中花生的几何参数，对其进一步筛选。

5 结论

本文设计了基于 DSP 和 FPGA 的图像选别系统，分析了硬件结构和数据传输的过程，采用仿真工具 MATLAB 研究了有关的图像处理算法。以破损花生检测为试验对象，分析 RGB 颜色模型，实现了花生破损区域的识别。改进了中值滤波算法。采用形态学运算，消除了图像中的点/孔状噪声，平滑了物体的边界轮廓。采用 8 邻域链码算法和连通域检测算法，实现了花生边界轮廓的检测，计算了花生内部破损区域的面积。综上提出了破损花生检测的算法并移植到 DSP 上进行测试。花生筛选时花生级别也是重要的考虑因素，故本文在最后根据花生轮廓曲线，计算出了花生的几何特征参数，为进一步筛选花生奠定了基础。

参考文献：

[1] 章 炜. 机器视觉技术发展及其工业应用 [J]. 红外，2006，27 (2)：11－17.

[2] 张五一，赵强松，王东云. 机器视觉的现状及发展趋势 [J]. 中原工学院学报，2008，19 (1)：9－12，15.

[3] 林茂先. 新型杂粮色选机的应用 [J]. 粮食加工，2014 (10)：24－27.

[4] 张 麟. 光电色选机及其应用 [J]. 农机与食品机械，1997 (5)：24－27.

[5] 张德丰. 数字图像处理 [M]. 北京：人民邮电出版社，2015.

[6] 刘志敏，杨 杰. 基于数学形态学的图像形态滤波 [J]. 红外与激光工程，1999 (4)：10－15，33.

(上接第 122 页)

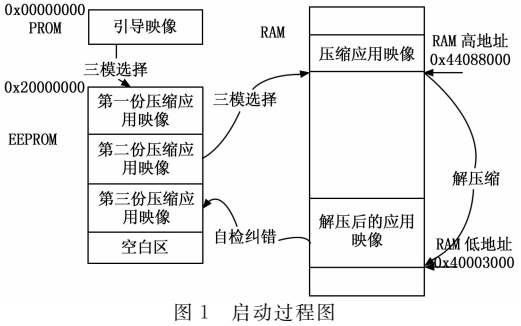

图 1 启动过程图

1 个字节，第 2 个映像的第 100K 个字节，第 3 个映像的第 200K 个字节处制造错误，然后将 3 份错误的应用层映像拷贝到 EEPROM 的 3 个不同位置，接着从引导地址 0x00000000 处启动后，观察调试串口的输出，并记录启动过程所需的时间。系统启动 1 分钟后再从 EEPROM 中读取 3 份应用层映像数据与正确映像分别进行比较，检查是否正确的进行了映像修复。

2）测试结果：调试串口输出软件运行信息，并且输出错误报告，记录了 3 处字节错误，并且修复完成。整个系统启动时间约为 8 秒。3 份从 EEPROM 中拷贝的软件映像与正确的软件映像数据二进制一致。结果表明错误数据已经都被纠正。

5.3 同一位置多个字节错误但 bit 位置不同错误类型映像启动

1）测试过程：对第 1 个应用层软件映像的第 1 个字节的 bit0－bit2 进行取反操作，对第 2 个应用层软件映像的第 1 个字节的 bit3－bit5 进行取反操作，对第 3 个应用层软件映像的第 1 个字节的 bit6－bit7 进行取反操作。对第 1 个应用层软件映像的第 100K 个字节的 bit0－bit1 进行取反操作，对第 2 个应用层软件映像的第 100K 个字节的 bit2－bit4 进行取反操作，对第 3 个应用层软件映像的第 100K 个字节的 bit5－bit7 进行取反操作。对第 1 个应用层软件映像的第 200K 个字节的 bit0－bit2 进行取反操作，对第 2 个应用层软件映像的第 200K 个字节的 bit3－bit4 进行取反操作，对第 3 个应用层软件映像的第 200K 个字节的 bit5－bit7 进行取反操作。然后将 3 份错误的应用层映像拷贝到 EEPROM 的 3 个不同位置，从引导地址 0x00000000 处启动后，观察调试串口的输出，并记录启动过程所需的时间。系统启动 1 分钟后再从 EEPROM 中读取 3 份应用层映像数据与正确映像分别进行比较，检查是否正确的进行了映像修复。

2）测试结果：调试串口输出软件运行信息，并且输出错误报告，记录了 3 处字节错误，并且修复完成。整个系统启动时间约为 8 秒。3 份从 EEPROM 中拷贝的软件映像与正确的软件映像数据二进制一致。结果表明错误数据已经都被纠正。

6 结论

本文分析了基于 VxWorks 系统进行工程设计的几种启动机制，并针对航天型号软件任务设计了基于 VxWorks 的新型三模冗余压缩映像的引导启动机制，给出了测试验证结果。该技术已经在某航天型号软件上应用，软件在轨运行情况良好。

参考文献：

[1] River W. Tornado 用户指南 [M]. 王金刚，等译. 北京：清华大学出版社，2003.

[2] 周治国，崔国辉，刘志文. 基于 S3C2440 NandFlashVxWorks 启动及性能分析 [J]. 计算机工程与应用，2010，46 (9S)：164－166.

[3] 黄海宇，姜连祥，杨勤荣，等. 内存受限系统下的 VxWorks 映像压缩算法 [J]. 计算机测量与控制，2010，18 (2)：419－421.

[4] 康凤举，段晓军，吴成富，等. 基于 VxWorks 的无人机飞控计算机快速启动导引技术研究 [J]. 计算机测量与控制，2010，18 (6)：1446－1448.