

# 如何减少 *Nios* 程序代码量



zrtech

[WWW.ZR-TECH.COM](http://WWW.ZR-TECH.COM)

# 引言

用过 NIOS 的朋友都会体会到 NIOS 那笨拙的编译器编译出的代码有多么庞大。没有 SDRAM 的 FPGA 系统想架构 NIOS 是一件多么困难的事情，因为仅仅一句 `printf` 就要占去几十 K 的存储空间，真是让人望而却步。今天我们这个专题就是为那些没有 SDRAM 却又想试试 NIOS 强大功能的朋友们准备的。

## 如何减少代码量

废话少说，切入正题：NIOS 减小代码量的方法有很多，大家按照如下设置，就可以轻松将自己的 C 代码优化到最小尺寸。

### 1. 采用 `alt_main()` 作为程序入口

NiosII 处理器的启动可采用两种方式：自动初始化和用户自定义初始化。ANSI C 标准定义应用程序可以通过调用 `main()` 来开始执行。在调用 `main()` 之前，应用程序假定运行环境和所有的服务系统都被初始化并准备运行。初始化可以被硬件抽象层（HAL）系统库自动执行。程序员不需要考虑系统的输出设备以及如何初始化每一个外设，HAL 会自动初始化整个系统。

HAL 提供的系统初始化代码按以下启动顺序运行：

- ① 启动指令和数据高速缓冲存储器；
- ② 配置堆栈；
- ③ 配置全局指针；
- ④ 通过链接器提供的 `_bss_start` 和 `_bss_end` 来零初始化 BSS 层，`_bss_start` 和 `_bss_end` 是开始和结束 BSS 的命令；
- ⑤ 如果当前系统没有启动下载器，就复制 `.rwdata`、`.rodata`，或者剩下的部分到 RAM；
- ⑥ 调用 `alt_main()`。

如果不调用 `alt_main()` 函数，则系统默认运行步骤如下：

- ① 调用 `ALT_OS_INIT()` 来执行任何操作系统所特有的初始化。如果 HAL 是在操作系统里运行的，那么初始化 `alt_fd_list_lock` 命令。它可以控制访问 HAL 文件系统，初始化中断控制器并执行中断；
- ② 调用 `alt_sys_init()` 函数，以初始化系统里所有的驱动装置和软件组成部分；
- ③ 重新设置 C 标准 I/O 通道（`stdin`，`stdout`，`stderr`），以使用合适的器件
- ④ 调用 `main()`；
- ⑤ 调用 `exit()`，`main()` 的返回代码作为 `exit()` 的输入。

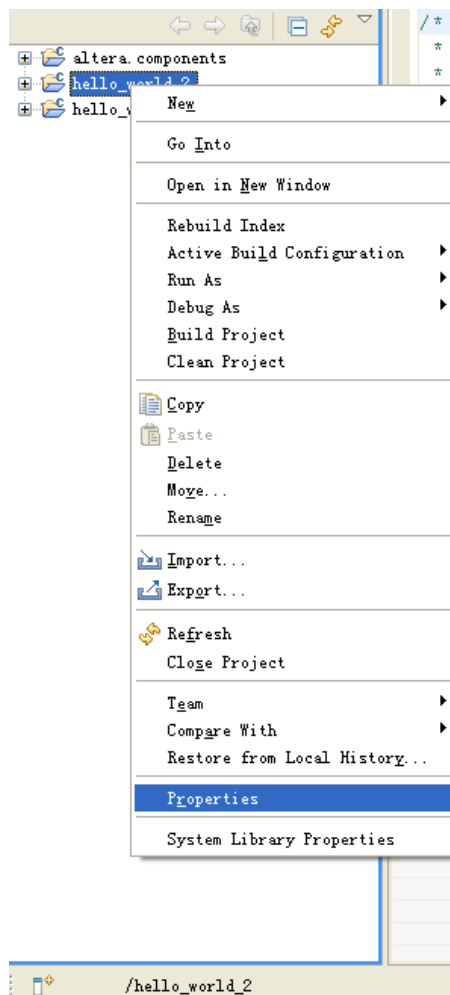
看看吧，如果不编写自己的 `alt_main()` 函数，NIOS 将会默认执行所有的初始化过程，显然耗费了大量的初始化代码。所以如果我们够牛，或者我们的程序使用的硬件足够容易，能手动初始化任何所用的硬件。则可以采用 `alt_main()` 函数完全控制系统的初始化，手动编写初始化系统的代码以减小代码量，其格式如下

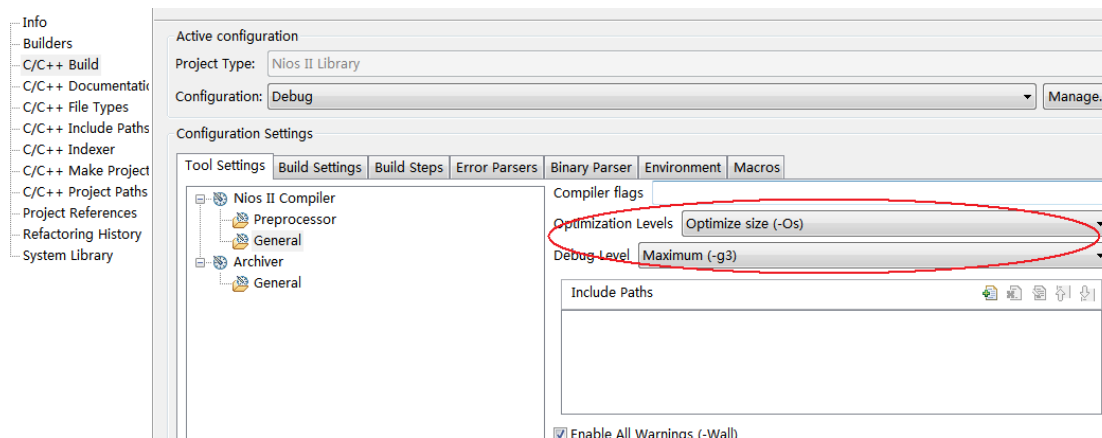
```
int main (void) __attribute__ ((weak, alias ("alt_main")));  
int alt_main (void)  
{  
    .....  
}
```

使用独立式编程环境会增加 NiosII 程序编写的复杂性。独立式编程环境的主要作用在于减小代码量，但要使用这种方法，需要对 NiosII 处理器的外设和驱动编写都非常熟悉才行。所以啊，为了少出问题，最好大家谨慎使用。比如如果使用了 `alt_main`，则中断控制器必须手动初始化，在 `alt_main` 开始加入 `alt_irq_init (ALT_IRQ_BASE)` 才行。但是当用 `main` 代替 `alt_main` 时，不需要 `alt_irq_init` 中断也能运行。所以使用好这一招还是比较麻烦的。不过别担心，在 NiosII IDE 中也可以通过某些选项来减小 HAL 系统库容量，从而达到减小代码量的目的，那可是完全傻瓜化，比使用独立式编程环境容易得多。

## 2. 打开编译器优化选项

在 `nios2-elf-gcc` 编译器中使用“-O3”选项，代码可以被最大限度地优化，包括代码的大小和执行速度。需要注意的是，编译器优化可能会带来一些意想不到的结果。另外，必须在用户工程和系统库中都使用-O3 选项，如图所示。

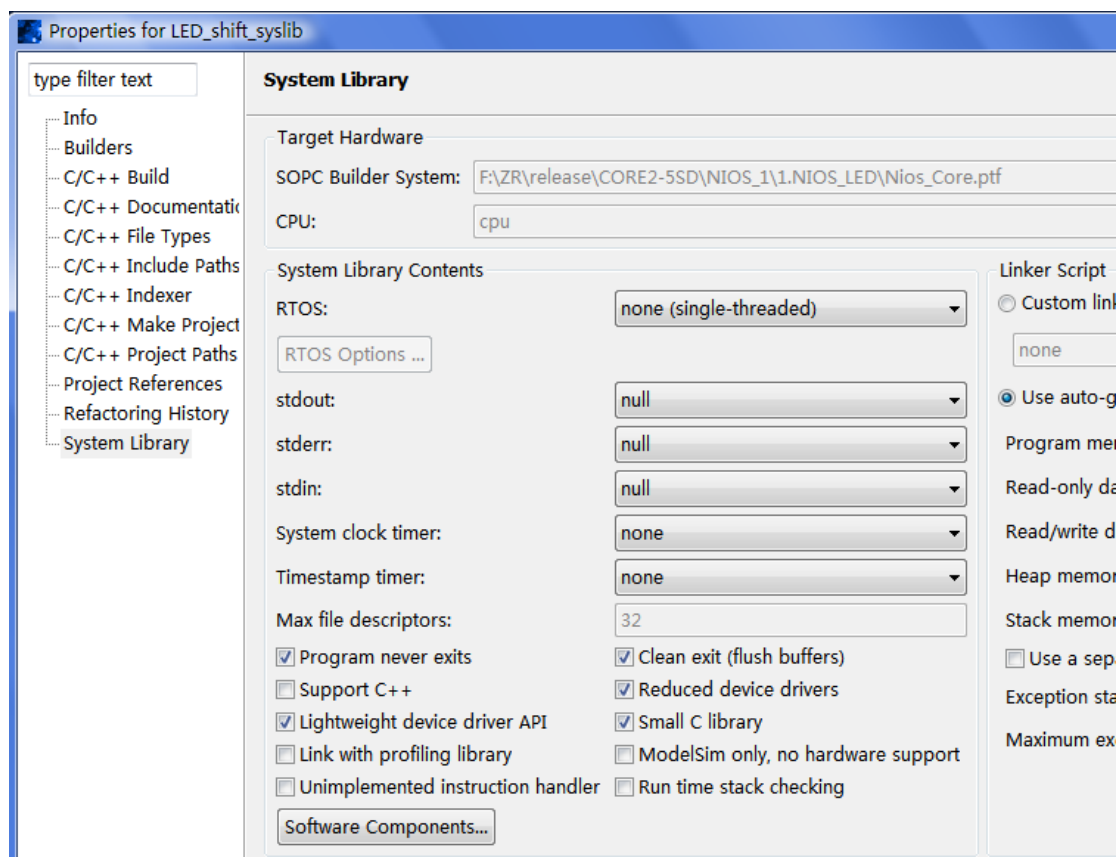




### 3 使用小封装的驱动库与 C 语言库等

HAL 为处理器的外设提供了两种驱动库：一种是执行速度快，但代码量大的版本；另一种是小封装版本。默认情况下，HAL 系统使用的是代码量大的版本。可以选择 **Reduced device drivers** 选项来选择小封装版本，从而减小代码量。

完整的 ANSI C 标准库通常不适用于嵌入式系统，HAL 提供了一系列经过裁减的新的 ANSI C 标准库，占用非常小的代码量。可以选择 **Small C library** 选项来选择新的 ANSI C 标准库，此外还推荐勾选如图配置，**Program never exits** 表示程序永远不会结束，则系统可以省去 `exit` 代码，还有最好使用纯 C 语言编程，不要使用 C++。

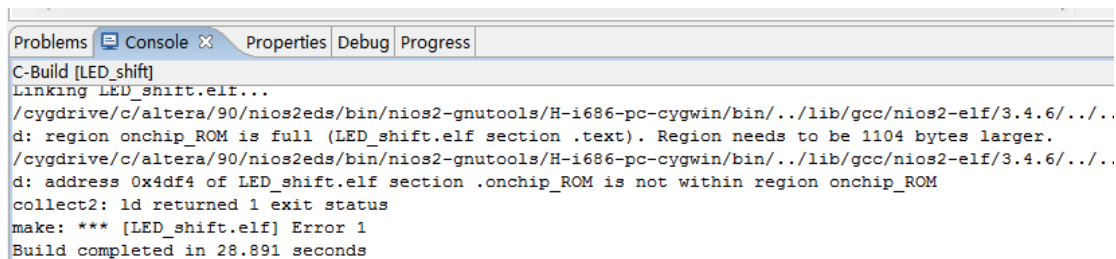


## 4 去掉不使用的驱动库

当 NiosII 系统中有外设时，NiosII IDE 认为这些设备需要驱动，因此在 HAL 系统中加入了相应的驱动库。如果在用户的程序中并不需要使用到这些外设，也可以在初始化时不加载这些驱动库。当用户的程序并没有使用到 NiosII 系统中某些设备时，应在系统中将这些设备完全移除。这样，既可以减小软件代码量，又可以减少占用的 FPGA 资源。

# 测试与总结

拿个例子测试一下吧，最简单的流水灯吧。不去优化它，编译一下....  
晕死，怎么那么大，4KB 的片内存储器都放不下，疯了，居然还差 1104B....



```

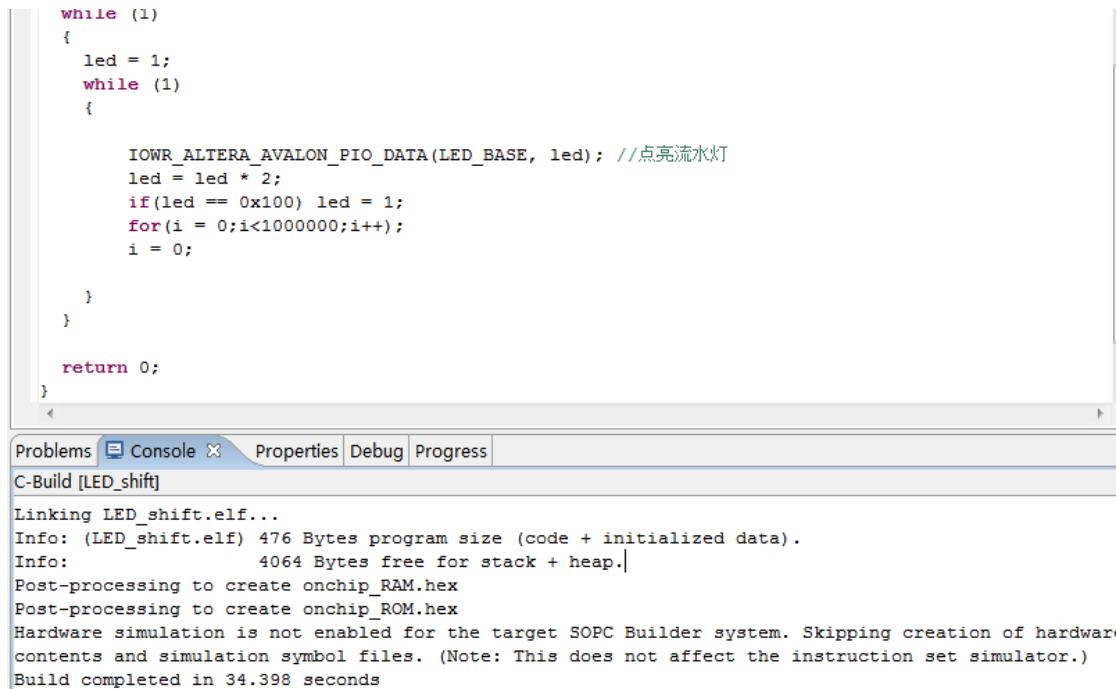
Problems Console Properties Debug Progress
C-Build [LED_shift]
Linking LED_shift.elf...
/cygdrive/c/altera/90/nios2eds/bin/nios2-gnutools/H-i686-pc-cygwin/bin/./lib/gcc/nios2-elf/3.4.6/./...
d: region onchip_ROM is full (LED_shift.elf section .text). Region needs to be 1104 bytes larger.
/cygdrive/c/altera/90/nios2eds/bin/nios2-gnutools/H-i686-pc-cygwin/bin/./lib/gcc/nios2-elf/3.4.6/./...
d: address 0x4df4 of LED_shift.elf section .onchip_ROM is not within region onchip_ROM
collect2: ld returned 1 exit status
make: *** [LED_shift.elf] Error 1
Build completed in 28.891 seconds

```

没事，我们有绝招！

按照上面说的，全部设置完毕，编译....

看看我们的优化成果吧，哈哈，仅用了 400 多字节的代码与初始化数据，对于 ONCHIP 存储器来说也是小菜一碟，没有 SDRAM 的板子也能跑起 NIOS 啦！



```

while (1)
{
    led = 1;
    while (1)
    {

        IOWR_ALTERA_AVALON_PIO_DATA(LED_BASE, led); //点亮流水灯
        led = led * 2;
        if(led == 0x100) led = 1;
        for(i = 0;i<1000000;i++);
        i = 0;

    }
}

return 0;
}

```

```

Problems Console Properties Debug Progress
C-Build [LED_shift]
Linking LED_shift.elf...
Info: (LED_shift.elf) 476 Bytes program size (code + initialized data).
Info: 4064 Bytes free for stack + heap.
Post-processing to create onchip_RAM.hex
Post-processing to create onchip_ROM.hex
Hardware simulation is not enabled for the target SOPC Builder system. Skipping creation of hardware
contents and simulation symbol files. (Note: This does not affect the instruction set simulator.)
Build completed in 34.398 seconds

```

**注意：**虽然优化可以有效减少代码量，但有的时候也会带来意想不到的问题。所以大家慎用。遇到异常时，先关闭优化，CLEAN 代码后再次编译，测试一下是否是优化所致。

## 相关信息

---

关于其他的相关信息，请访问以下网站

■ 购买本教程配套的开发套件，子卡或下载线缆：

<http://www.zr-tech.com>

■ 心得交流与问题互助：

<http://www.zr-tech.com/bbs>

## 版权信息

---

■ 本文档手册为ZRTech（[www.zr-tech.com](http://www.zr-tech.com)）原创资源，享有完全版权。

■ 任何收存和保管本文档各版本的单位和个人，未经本公司同意，不得随意复制、抄录、修改本文档的部分或者全部内容。

■ 转载本文档时请务必保证此文档的完整性。文档必须包含本版权信息。不得将转载作品以任何形式谋取商业利益，也不得向任何第三方提供，否则视为侵权。