HY16F Series(AndeSight Linker Script)

Flash Memory Address Definition 应用说明书

目录

1 简介

HY16F 允许用户根据需要，对 FLASH 空间划分不同的区域，用于存放不同的程序段。针对不同的程序段自定义区域起始地址，程序区域的起始地址在链接脚本内定义。程序编译的每一个链接过程都由链接脚本控制，链接脚本名为 Linker Script。链接脚本主要用于控制 SECTIONS 内各程序地址空间的分布，一般 Linker Script 是以 LD 作为文件的后缀名。本文介绍如何修改 LD 档，根据需求，将用户的函数或变量定义为固定的绝对地址位置，在程序上如何使用该绝对地址来定义函数或变量。

2 操作步骤

2.1 修改 LD 档

在 LD 档找到 SECTIONS 部分定义程序地址，由于 Flash 的 0X90000~0X9042F 是作为固定程序段不允许用户使用，因而 FUNCTIONA 是从 0X90430 开始；假如要定义四个固定地址的程序区域，如'FUNCTIONA' ,ADDRESS =0X90430；'FUNCTIONB'，ADDRESS=0X91090；定义中断矢量的区域'FUNCTIOND',ADDRESS=0X91000；定义固定数据区域'FUNCITONE',ADDRESS=0X9FE00;按照格式添加到 LD 档的 SECTIONS 内；这里定义区域的大小要根据程序大小需要来定义，这样可以最大限度给利用 FLASH 空间；

```
SECTIONS
{
  .start              :
  {
    *(.start)
  }> ROM

  .Eric_init                :
  {
    KEEP(*(.Eric_init))
  } > FLASH
  FUNCTIONA   0x90430 :                  //定义 BOOTLOADER 区域
  {
    *(.FUNCTIONA)
  } > FLASH
  FUNCTIOND   0x91000 :                 //定义中断矢量区域
  {
    *(.FUNCTIOND )
  } > FLASH
  FUNCTIONB   0x91090 :                //定义 main code（用户程序）区域
  {
    *(.FUNCTIONB )
```

```
    } > FLASH
    FUNCTIONE   0x9FE00 :              //定义数据区域
    {
      KEEP(*(.FUNCTIONE ))
    } > FLASH
    .text :
    {
      *(.nds32_init .text .stub .text.* .ex9.itable)
      . = ALIGN(4);
    } > FLASH
    …
    …
}
```

2.2 程序编写

在 LD 定义 FUNCTIONA/FUNCTIONB，程序上，定义一个 Flash 自我烧录程序 SelfBurnLoop(void)在 FUNCTIONA，main（）code 在 FUNCTIONB 区域，将一个数据 ARRAY[1024]定义在 FUNCTIONE,ADDRESS=0X9FE00，程序的撰写如下：

使用 LD 档定义好的区域地址对函数进行声明，使用关键字：**__attribute__**，声明格式如下：

Type **__attribute__ ((section(“绝对地址标识符 “))) 函数名;**

如：将 int SelfBurnLoop(void) 定义到‘FUNCTIONA‘区域，声明如下：

```
int __attribute__ ((section ("FUNCTIONA"))) SelfBurnLoop(void);
```

SelfBurnLoop(void)的程序撰写，该区域程序目前定义在 0X90430~0X91000，撰写的程序要注意控制空间大小，否则会覆盖后面程序出现异常错误警报。在此区域目前一个主要应用时撰写 ISP 在线更新程序，可以写 IIC/UART 接口的 BOOTLOADER 程序，所有需要将函数设定在此区域，函数声明都如下：

```
int __attribute__ ((section ("FUNCTIONA"))) SelfBurnLoop(void);//函数绝对地址声明
int SelfBurnLoop(void)                                          //函数
{
    return 1;
}
```

main()定义 FUNCTIONB，地址从 0X91000 开始，其他的用户程序都写成子函数模式，main()调用其他的子函数:

```
int __attribute__ ((section ("FUNCTIONB")))main(void); //函数绝对地址声明
int main(void)                                          //函数
{
    return 0;
}
```

定义个常数在固定区域 FUNCTIONE，地址从 0X9FE000 开始，用户可以通过 ISP 功能独立更新该区域数据：

```
const unsigned char DefaultData3[256] __attribute__ ((section ("FUNCTIONE"))) =
{
0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0A, 0x0B, 0x0C, 0x0D, 0x0E, 0x0F,
0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x17, 0x18, 0x19, 0x1A, 0x1B, 0x1C, 0x1D, 0x1E, 0x1F,
0x20, 0x21, 0x22, 0x23, 0x24, 0x25, 0x26, 0x27, 0x28, 0x29, 0x2A, 0x2B, 0x2C, 0x2D, 0x2E, 0x2F,
0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x38, 0x39, 0x3A, 0x3B, 0x3C, 0x3D, 0x3E, 0x3F,
0x40, 0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48, 0x49, 0x4A, 0x4B, 0x4C, 0x4D, 0x4E, 0x4F,
0x50, 0x51, 0x52, 0x53, 0x54, 0x55, 0x56, 0x57, 0x58, 0x59, 0x5A, 0x5B, 0x5C, 0x5D, 0x5E, 0x5F,
0x60, 0x61, 0x62, 0x63, 0x64, 0x65, 0x66, 0x67, 0x68, 0x69, 0x6A, 0x6B, 0x6C, 0x6D, 0x6E, 0x6F,
0x70, 0x71, 0x72, 0x73, 0x74, 0x75, 0x76, 0x77, 0x78, 0x79, 0x7A, 0x7B, 0x7C, 0x7D, 0x7E, 0x7F,
0x80, 0x81, 0x82, 0x83, 0x84, 0x85, 0x86, 0x87, 0x88, 0x89, 0x8A, 0x8B, 0x8C, 0x8D, 0x8E, 0x8F,
0x90, 0x91, 0x92, 0x93, 0x94, 0x95, 0x96, 0x97, 0x98, 0x99, 0x9A, 0x9B, 0x9C, 0x9D, 0x9E, 0x9F,
0xA0, 0xA1, 0xA2, 0xA3, 0xA4, 0xA5, 0xA6, 0xA7, 0xA8, 0xA9, 0xAA, 0xAB, 0xAC, 0xAD, 0xAE, 0xAF,
0xB0, 0xB1, 0xB2, 0xB3, 0xB4, 0xB5, 0xB6, 0xB7, 0xB8, 0xB9, 0xBA, 0xBB, 0xBC, 0xBD, 0xBE, 0xBF,
0xC0, 0xC1, 0xC2, 0xC3, 0xC4, 0xC5, 0xC6, 0xC7, 0xC8, 0xC9, 0xCA, 0xCB, 0xCC, 0xCD, 0xCE, 0xCF,
0xD0, 0xD1, 0xD2, 0xD3, 0xD4, 0xD5, 0xD6, 0xD7, 0xD8, 0xD9, 0xDA, 0xDB, 0xDC, 0xDD, 0xDE, 0xDF,
0xE0, 0xE1, 0xE2, 0xE3, 0xE4, 0xE5, 0xE6, 0xE7, 0xE8, 0xE9, 0xEA, 0xEB, 0xEC, 0xED, 0xEE, 0xEF,
0xF0, 0xF1, 0xF2, 0xF3, 0xF4, 0xF5, 0xF6, 0xF7, 0xF8, 0xF9, 0xFA, 0xFB, 0xFC, 0xFD, 0xFE, 0xFF
}
```

定义中断矢量的绝对地址，作为中断函数的跳转入口，防止程序在更新单独更新 main code 部分程序会由于找不到中断矢量的入口地址；

先定义中断矢量的绝对地址：

```
void __attribute__ ((section ("FUNCTIOND")))HW0_ISR();    //define HW0 address
```

在中断矢量函数里包含用户的中断处理程序，用户中断处理程序全部在 main code 区域完成，可以节省空间：

```
void HW0_ISR()   //define HW0 include the corresponding interrupt vector
{
    UART_ISR();
}
```

用户中断处理程序，编译在main code区域：

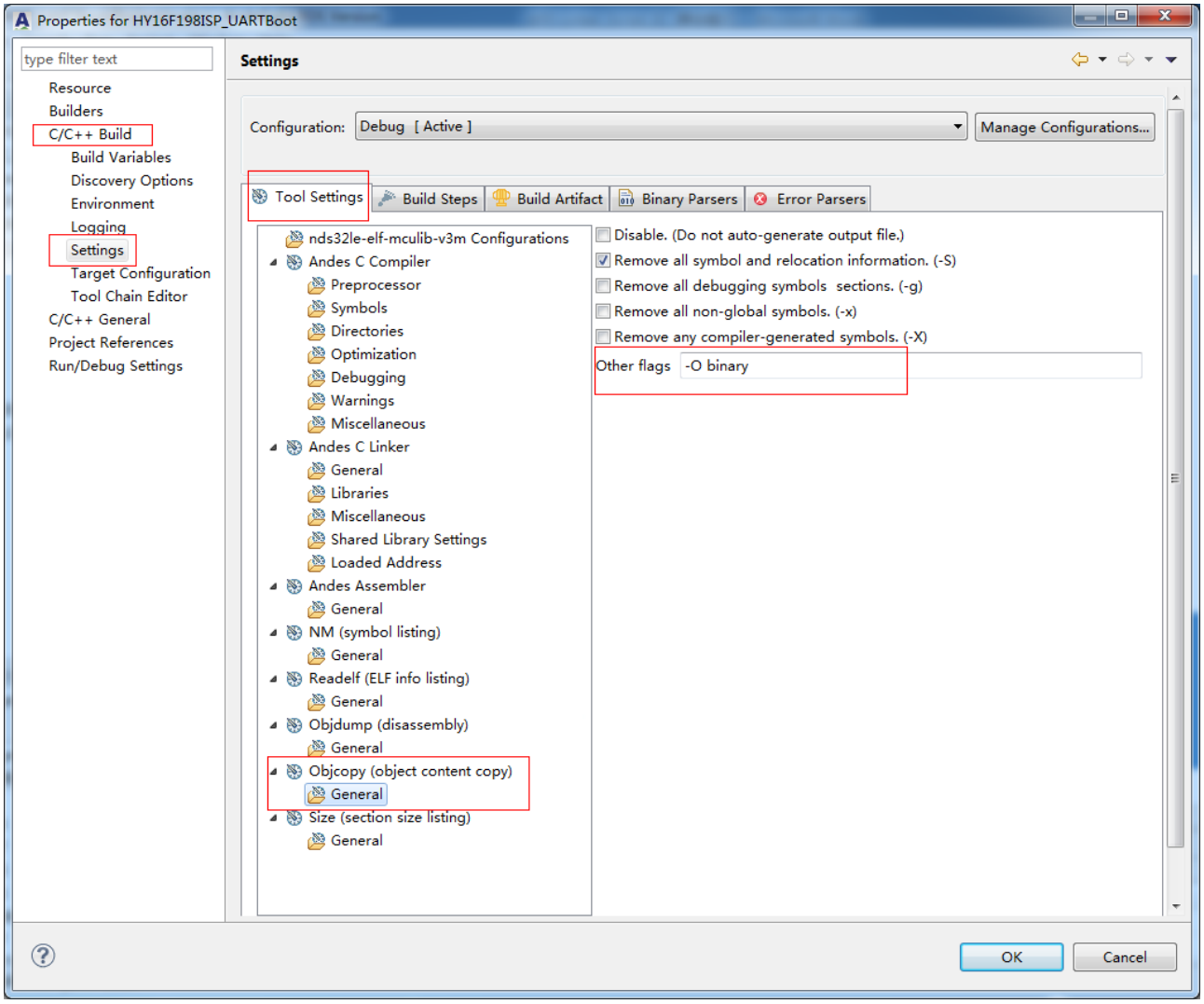
```
void UART_ISR(void)   //the UART interrupt operation
{

}
```

2.3 工程设定

在工程设定，要针对自定义的 FLASH 区域进行编译，产生相对应的 BIN 档，操作如下：

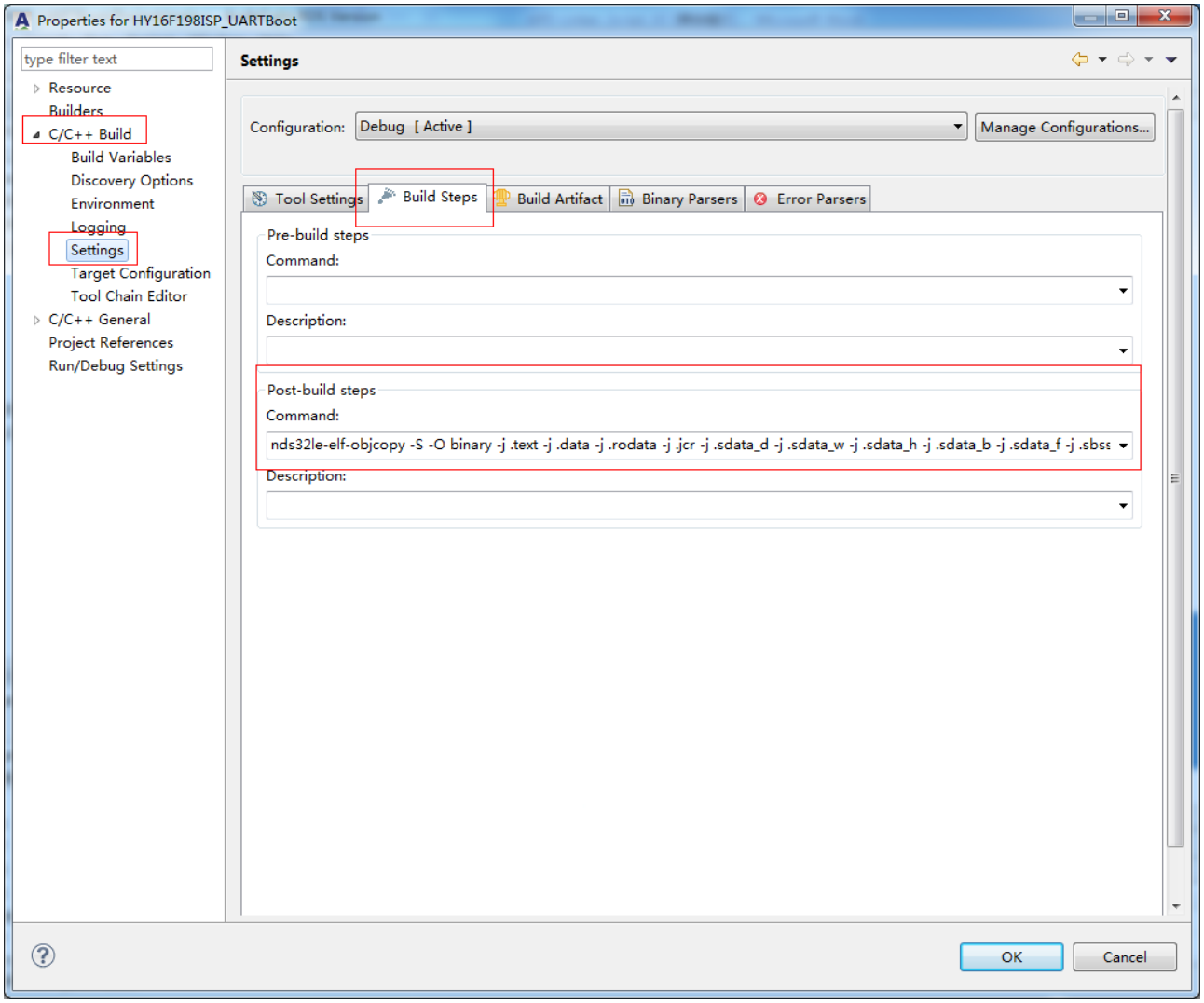
A） 设定 objcopy：

C/C++ Build -> Settings -> Objcopy(object content copy) -> General -> Other flags



B） 设定编译产生对应的 BIN 档：

C/C++ Build -> Settings -> Build Steps -> Post-build steps -> Command



根据 LD 档定义的区域，及用户需要产生的 BIN，撰写对应的命令，多个命令之间用分号隔离，命令格式如下：

nds32le-elf-objcopy -S -O binary -j .text -j .data -j .rodata -j .jcr -j .sdata_d -j .sdata_w -j .sdata_h -j .sdata_b -j .sdata_f -j .sbss_f -j .sbss_b -j .sbss_h -j .sbss_w -j .sbss_d -j .bss -j FUNCTIONA -j FUNCTIONB -j FUNCTIOND -j FUNCTIONF -j FUNCTIONE -j .Eric_init ${ProjName}.adx output/${ProjName}.bin;

该命令式将 FUNCTIONA/FUNCTIONB/FUNCTIOND/FUNCTIONF/FUNCTIONE 编译生成一个 BIN 档，名称为‘工程名.BIN’；

nds32le-elf-objcopy -I binary -O ihex output/${ProjName}.bin output/${ProjName}.hex;

HYChcekSum output/${ProjName}.bin;

该命令式将 FUNCTIONA/FUNCTIONB/FUNCTIOND/FUNCTIONF/FUNCTIONE 编译产生一个 HEX 档、输出 chencksum；

nds32le-elf-objcopy -S -O binary -j .text -j .data -j .rodata -j .jcr -j .sdata_d -j .sdata_w -j .sdata_h -j .sdata_b -j .sdata_f -j .sbss_f -j .sbss_b -j .sbss_h -j .sbss_w -j .sbss_d -j .bss -j FUNCTIONB -j FUNCTIOND -j FUNCTIONF -j FUNCTIONE ${ProjName}.adx output/${ProjName}_APP.bin;

该命令式将 FUNCTIONB/FUNCTIOND/FUNCTIONF/FUNCTIONE 编译生成一个 BIN 档，名称为‘工程名_APP.BIN’。

nds32le-elf-objcopy -I binary -O ihex output/${ProjName}_APP.bin output/${ProjName}_APP.hex;
HYChcekSum output/${ProjName}_APP.bin

该命令式将 FUNCTIONA/FUNCTIONB/FUNCTIOND/FUNCTIONF/FUNCTIONE 编译产生一个 HEX 档、输出 chencksum；

3 Demo Code 及相关档案


HY16F198ISP_UARTBoot-20160302.zip HY16F198B UART BOOTLOADER 及 LD 档撰写范例

4 参考文献

[1] http://www.hycontek.com/page2-HY16F.html，HY16F1XX资料
[2] http://www.hycontek.com/，纮康科技股份有限公司

5 修订纪录

以下描述本文件差异较大的地方，而标点符号与字形的改变不在此描述范围。

版本	页次	变更摘要	更新日期
V01	All	初版发行	2016/03/04
V02	Page4	原描述*(.FUNCTIONE)新增修正为 KEEP(*(.FUNCTIONE))	2016/08/23
	All	说明书描述 FUNCTIONG 修正为 FUNCTIONE，在此说明书内 FUNCTIONE 代表的是自定义 Data 区块	