NetBurner

Application Note

How to Increase User Flash Storage

1. Introduction

All NetBurner hardware platforms provide a default value of 8Kbytes of on-chip flash storage space for user/application use. This number is based on the sector size of some flash memory components. One method to obtain more flash storage space is to use the area of flash memory beyond the point where the application is stored.

Other methods of flash storage include the Embedded Flash File System (EFFS) external SD card interface.

2. Flash Memory Usage for 512kb Devices

The NetBurner Mod5270, Mod5282, SB70, and SB72 (not the SB72EX) have a total of 512k byte of flash memory allocated as follows:

Description	**Size in Bytes**	**Starting Address**
Boot Monitor	16k	0xFFC0 0000
System Parameters	8k	0xFFC0 4000
User Parameters	8k	0xFFC0 6000
Application	480k	0xFFC0 8000

To add more user storage space, we can use some of the application space. The application created by the NetBurner tools is a compressed image, and is typically less than 200k bytes in size. Each time you compile your project, the compiler output will display the compressed size of your application.

In this example we will reserve an additional 64k of flash for user storage. The flash sector size on the devices listed above are 4k, so you can use any multiple of 4k bytes.

Description	**Size in Bytes**
Boot Monitor	16k
System Parameters	8k
User Parameters	8k
Application	416k
User Storage	64k

To implement this new storage area, we need to:
1. Identify the start address of the user storage area
2. Create functions to read/write the data
3. Modify the application linker script so it knows the new ending address

2.1 Flash Memory Usage on 2MB Flash Platforms

The Mod5234, Mod5272 and SB72EX each have 2MB of flash memory. These flash chips have a boot block providing the same size flash sectors for the monitor, system parameters and user parameters. The major difference is that the flash chip sector size is 64k instead of 4k, so any area you allocate must be in 64k blocks.

Description	**Size in Bytes**	**Starting Address**
Boot Monitor	16k	0xFFC0 0000
System Parameters	8k	0xFFC0 4000
User Parameters	8k	0xFFC0 6000
Application	1968k	0xFFCE 0000

3. Implementation

The user storage area is used in the following manor:

1. Read the current stored data (you may want to use a structure)
2. Make any modifications
3. Erase the user flash area
4. Write the modified data to flash

3.1 Define the Start of the User Storage Area

The functions to read/write data need to know the address in flash where the data is located. 512k is equivalent to 0x80000 hexadecimal. Our starting flash memory address is 0xFFC00000, which means that the end of the 512k is:
0xFFC00000 + 0x80000 = 0xFFC80000.

If we subtract 64k from that number we have: 0xFFC80000 – 0x10000 = 0xFFC70000. This is the starting address of our new user storage area. In this application note we refer to the starting address of the user storage area as "BigParamBase".

The name and value of BigParamBase is passed to the application through the sys.ld linker script. The location of sys.ld depends on what platform you are using. The path is c:\nburn\<platform>\lib\sys.ld. The sys.ld file for the Mod5270 is shown below, with the addition of the BigParamBase as the last line:

```
vector_base =0x20000000;
sim = 0x40000000;
gConfigRec = 0xffc04000;
UserParamBase = 0xffc06000;
_RESET_ADDR = 0xffc00008;
BigParamBase = 0xffc70000;
```

3.2 Function to Read the Stored Data

Since any type of data can be stored in this area, the function to retrieve the data simply returns a pointer to the data location in flash memory. Once your application has the pointer, you format it as appropriate (eg. in a structure).

```
void * Get64KUserParameters()
{
   return (void *)BigParamBase;
}
```

3.3 Function to Write the Data to Flash

Before using this function, make sure you have read the previous stored data and made your modifications to it! The function will erase the flash sectors of the user storage area, then do a byte copy of the data array pointed to by the *pCopyFrom void pointer. The critical sections ensure the data is not modified during the writing process.

```
extern char BigParamBase[65536];

int Save64KUserParameters( void * pCopyFrom,int len)
{
   if (len > 0xFFFF) return 0;
   if (len < 0) return 0;
   USER_ENTER_CRITICAL()
   FlashErase( (void *)&BigParamBase, len );
   FlashProgram( (void *)&BigParamBase, pCopyFrom, len);
   USER_EXIT_CRITICAL();
   return 1;
}
```

3.4 Specify the End of Application Storage Space

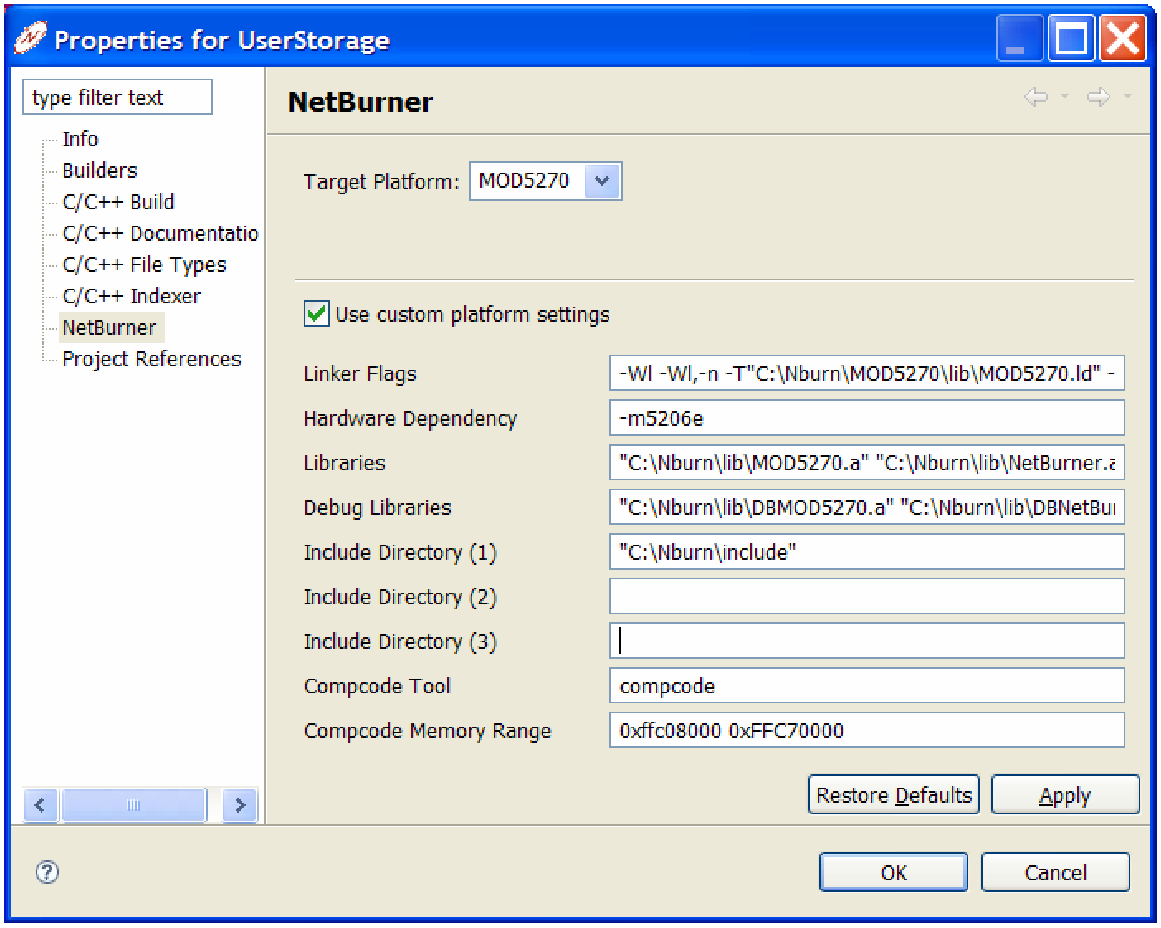
The final step is to tell the NetBurner tools where the flash application space ends. The default value is the end of flash, which we specified above to be 0xffc8000. Since we are using the last 64k as user storage space, the application now must end at 0xffc7000.

To make this change in NBEclipse:

1. Right-click on your project and select Properties
2. Select NetBurner from the Properties dialog box
3. Click and enable the checkbox for “Use custom platform settings”

4. Modify the “Compcode Memory Range” to specify the new end address for the application. In this case, 0xffc70000.
5. Rebuild your application.

A screen shot of the modified properties dialog box is shown below:



Target Platform: MOD5270

[✓] Use custom platform settings

Linker Flags	-Wl -Wl,-n -T"C:\Nburn\MOD5270\lib\MOD5270.ld" -
Hardware Dependency	-m5206e
Libraries	"C:\Nburn\lib\MOD5270.a" "C:\Nburn\lib\NetBurner.a
Debug Libraries	"C:\Nburn\lib\DBMOD5270.a" "C:\Nburn\lib\DBNetBu
Include Directory (1)	"C:\Nburn\include"
Include Directory (2)	
Include Directory (3)	
Compcode Tool	compcode
Compcode Memory Range	0xffc08000 0xFFC70000

If you are using the command-line tools, you would modify the c:\nburn\<platform>\make\ldflags.mk file COMPCODE entry as shown below:

COMPCODEFLAGS = 0xffc08000 0xffc70000

Note that changes to ldflags.mk will affect all your <platform> projects.