How to Program to Flash Memory in W7100A

Version 1.2

For more information, visit our website at http://www.wiznet.co.kr

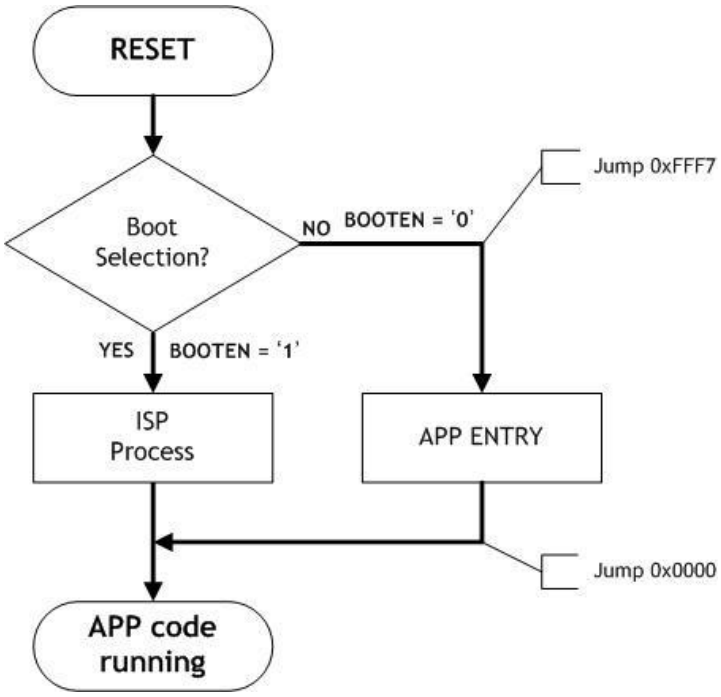
Table of Contents

1 Introduction

The W7100A internally has Boot ROM and Flash Memory, working as Code Memory. The Boot ROM automatically uses the ISP function to drive the W7100A. Therefore the user cannot modify the code for Boot ROM. However, the Flash Memory can be used as Code Memory.

The W7100A automatically executes the Boot code of the Boot ROM when its system is reset. Then the next process differs depending on the BOOTEN pin. This manual will explain how to program the code to Flash Memory.

Detailed execution is as below.



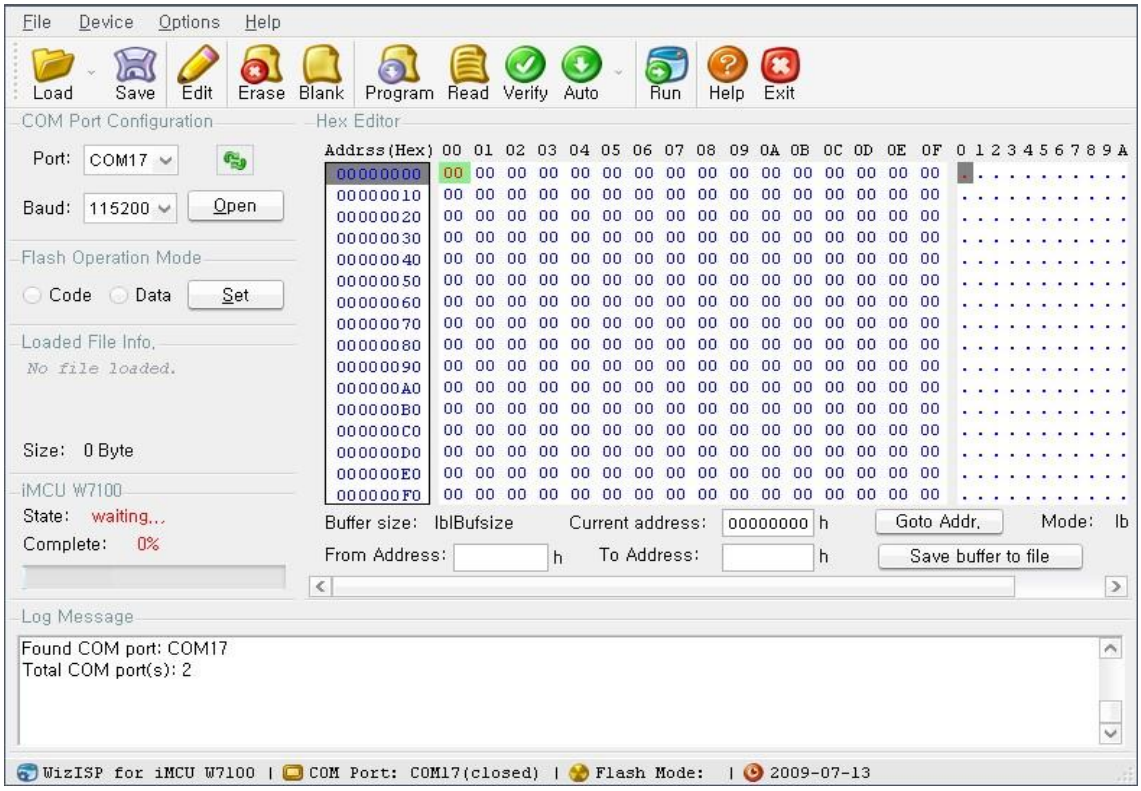
<Fig.1> Boot Sequence Flowchart

As showed in Fig. 1 above, after Reset, there are two different ways of execution, depending on the BOOTEN pin of W7100A. One execution is using the ISP function in the Boot ROM, by setting the BOOTEN pin to '1'. The other execution is jumping to Flash Memory without using the Boot ROM, by setting the BOOTEN pin to '0'.

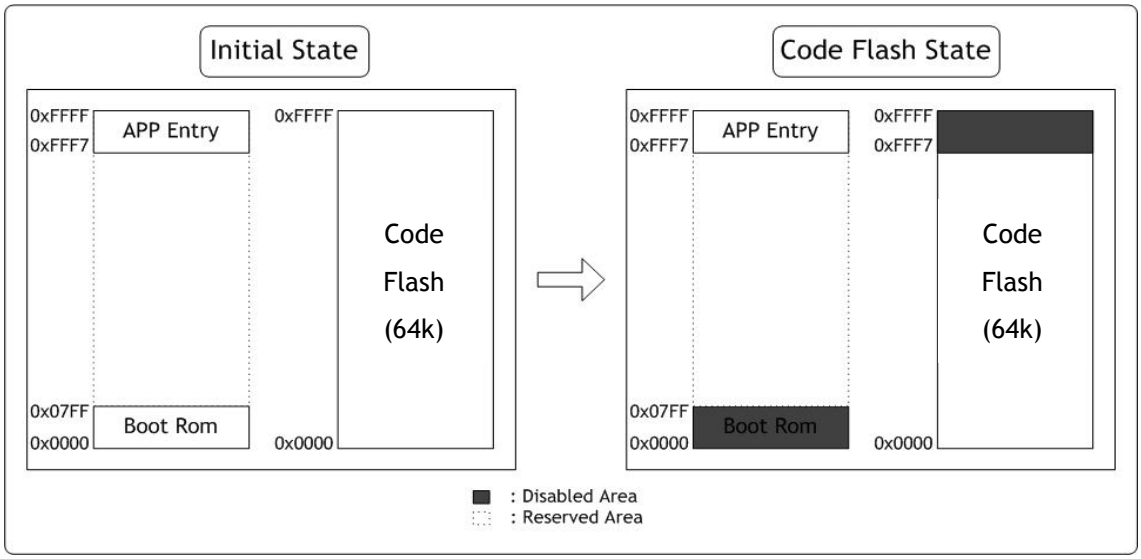
User can program Flash with ISP Mode or modify the ISP Process in APP Mode. For more details on execution of APP Mode(BOOTEN pin = '1'), refer to the '2.1 Code Memory' of the W7100A Datasheet.

2 ISP Mode

When the BOOTEN pin is set to '1', the state of W7100A is set as ISP Mode, and the user can use the ISP Program of WIZnet. During this mode, user can write or read the Code Flash and Data Flash by using the ISP Program of WIZnet. Please refer to the 'WizISP Program User Guide' for more details.



<Fig.2> ISP program for W7100A



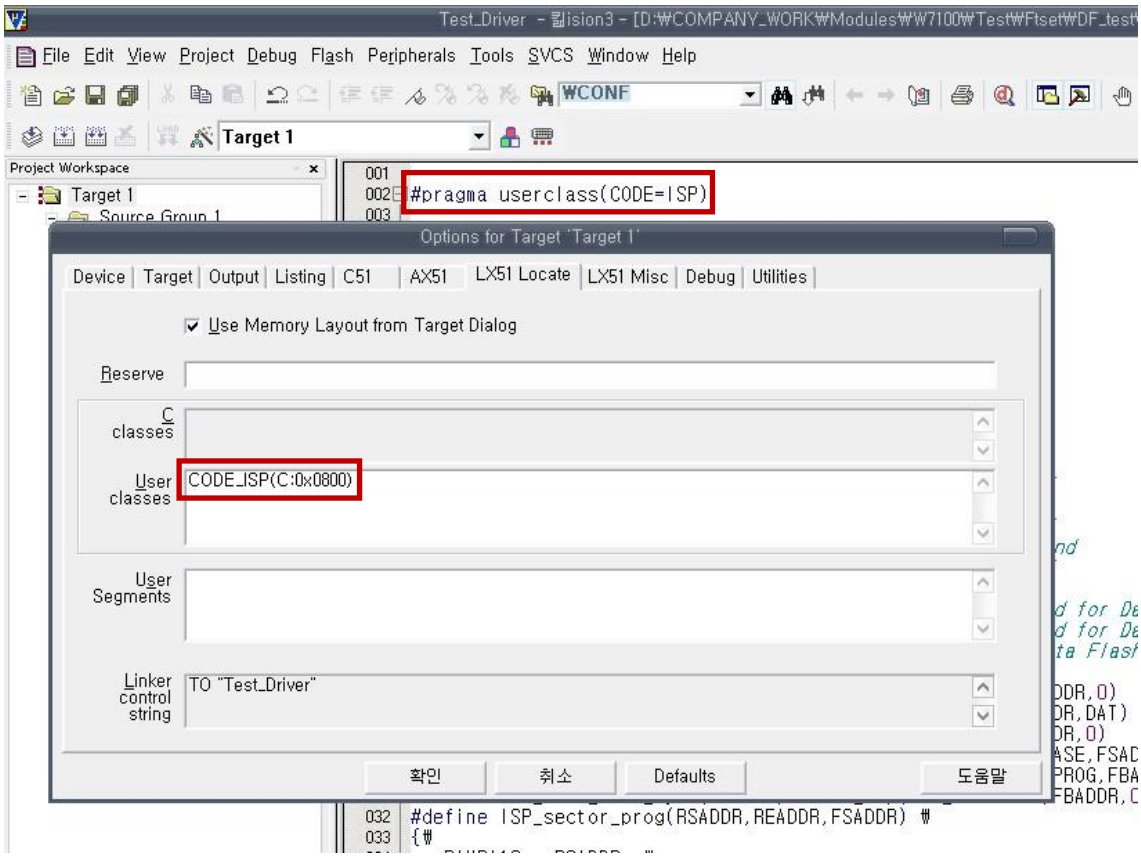
<Fig.3> Code Memory Map Switching

3 ISP Process in APP Mode

Usually writing and erasing data, using the ISP Program, in Code / Data Flash is the general way. But since the ISP function is already in the Boot ROM, the user can call and use the ISP function in the Flash. To do this, the W7100A must execute in APP Mode. If ISP Mode was on, user must change to APP Mode by rebooting. When calling the ISP function, please note that the routine for calling the ISP function must not take place in the overlapped area. As in the 'Initial State' of Fig., the overlapped area is the area where the Boot ROM area and Flash area overlaps. Therefore, the user must program the code above the overlapped area (0x0000 ~ 0x07FF). If the ISP function calling routine is placed in the overlapped area (0x0000 ~ 0x07FF), an error will occur because there will be no return address during the switching process between Boot ROM and Flash memory. In order for the program to execute properly, user must insert the following code in the 'Keil uVision3' compiler to prevent the overlapping.

Insert paragraph 'CODE_ISP(C:0x0800)' in Project menu => 'Options for Target 'project name'' => 'LX51 Locate' => 'User classes'. And insert below code to the function() which contains calling the ISP function. This process can change depending on the compiler.

```
#pragma userclass(CODE = ISP)
```



<Fig.4> Overlapping prevention of code memory

If the W7100A executes by Flash code, W7100A will complete the memory map switching, as shown in

the 'Code Flash State' of Fig. 3, and the Boot ROM will be disabled. Therefore, to use the ISP function in the Boot ROM, the Boot ROM must be reactivated by switching back the memory map. To do this, user must set the ISPEN bit of WCONF (0xFF) to '0', and the Boot ROM will be activated. Then the user can call the defined ISP function, corresponding to the ISPID, to erase or write data in Flash.

The basic example for using ISP function in Flash is as below.

Note: In this section, the whole example code is based on the 'Keil uVision3' compiler.

```
#define ISP_ENTRY 0x07FD
unsigned char do_isp(unsigned char isp_id, unsigned short isp_addr, unsigned char isp_data)
{
    TMPR0 = EA;              // backup EA
    EA = 0;                  // disable EA
    WCONF &= ~(0x40);        // Enable ISP Entry
    ISPID = isp_id;
    ISPADDR16 = isp_addr;
    ISPDATA = isp_data;
    ((void(code*)(void))ISP_ENTRY)();      // call ISP Entry
    WCONF |= 0x40;                         // Disable ISP Entry
    EA = TMPR0;                            // restore EA
    return ISPDATA;
}
```

The 'do_isp' function is executing the ISP function with isp_id, isp_addr, and isp_data.

User must make sure that no other interrupts occur during the ISP function procedure like above. Set the ISPEN bit to '0' to activate to Boot ROM and ISP ENTRY. Then, write each ISPADDR and ISPDATA corresponding to the defined ISPID. After that, the ISP function in the Boot ROM should be executed by calling the ISP ENTRY. If the ISP ENTRY is called, the ISP function will execute corresponding to the ISPID the user has set. When the execution of ISPID is done, the ISP ENTRY will automatically return to the routine, which the ISP function was called. If the user sets the ISPEN bit to '1' after the return, both the Boot ROM and ISP ENTRY that were activated will deactivate. And then, the Flash can be completely used as Code Memory again. Finally return all interrupt settings to original state to complete the whole process.

Defined ISPIDs corresponding to each command are as below.

```
#define ISP_SERASE      0x30      // Sector Erase Command
#define ISP_MERASE      0x10      // Chip Erase Command
#define ISP_BPROG       0xA0      // Byte Program Command
#define ISP_STATUS      0x70      // ISP Status Command
#define ISP_BREAD       0x00      // Byte Read Command
```

```
#define ISP_DATAERASE          0xD0       // Sector Erase Command for Data FLASH
#define ISP_DATAPROG           0xD1       // Sector Program Command for Data FLASH
#define ISP_DATAREAD           0xD2       // Read Command for Data FLASH
```

Example of real command using above ISPID for Code Flash is as below.

```
#define ISP_set_reboot()       (WCONF |= 0x80)       //Reboot After ISP operation
#define ISP_chip_erase()       do_isp(ISP_MERASE,0,0)
#define ISP_sector_erase(FSADDR)       do_isp(ISP_SERASE,FSADDR,0)
#define ISP_write_byte(FBADDR,DAT)       do_isp(ISP_BPROG,FBADDR,DAT)
#define ISP_read_byte(FBADDR)       do_isp(ISP_BREAD,FBADDR,0)

#define ISP_chip_prog(RSADDR,READDR,FSADDR) \
{\
 RAMBA16 = RSADDR;\                    //RSADDR: RAM Start Address
 RAMEA16 = READDR;\                    //READDR: RAM End Address
 do_isp(ISP_CHIPROG,FSADDR,0);\        //FSADDR: FLASH Start Address
}\
#define ISP_sector_prog(RSADDR,READDR,FSADDR) \
{\
 RAMBA16 = RSADDR;\
 RAMEA16 = READDR;\
 do_isp(ISP_SECPROG,FSADDR,0);\
}\
```

And the example of real command using above ISPID for Data Flash is as below.

```
#define ISP_data_sector_erase()       do_isp(ISP_DATAERASE,0,0)
#define ISP_data_sector_read(RSADDR)\
{\
 RAMBA16 = RSADDR;\                    //RSADDR: RAM Start Address
 do_isp(ISP_DATAREAD,0,0);\
}\
#define ISP_data_sector_prog(RSADDR) \
{\
 RAMBA16 = RSADDR;\
 do_isp(ISP_DATAPROG,0,0);\
}\
```

Since the 'ISP_chip_erase() command' erases the whole Flash data, and can also erase the execution codes which will execute after returning to Flash, do not use the 'ISP_chip_erase() command' except

using ISP Program.

Using the above command, erase from 0x00 to 1sector and write the value 0xAA and read in 0x00, the code example is:

```
ISP_sector_erase(0x00);
ISP_write_byte(0x00,0xAA);
ISP_read_byte(0x00);
```

The example code for Data Flash, as shown below, describes that the user can erase 1sector and write 1sector to the value 0x00 ~ 0xFF. Since the Data Flash of W7100A has 256Byte (1 sector), there is no needs about the RAM End Address for the ISP_data_sector_prog or ISP_data_sector_read.

```
uint16 data i = 0;
uint8 xdata temp1[256];
uint8 xdata temp2[256];
for(i = 0; i < 0x100; i++)
{
    winf[i] = (uint8)i;
}
ISP_data_sector_prog((uint16)temp1);
ISP_data_sector_read((uint16)temp2);
```

Document History Information

Version	Date	Descriptions
Ver. 0.9Beta	Sep, 2009	Release with W7100 launching
Ver. 0.92	Nov, 2010	Modify the example code
Ver. 1.0	May, 2011	Release with W7100A launching
Ver. 1.1	Mar, 2012	Fixed the ISPIDs at *3. ISP Process in APP Mode* (P. 6)

Copyright Notice

Technical Support: support@wiznet.co.kr
Sales & Distribution: sales@wiznet.co.kr

For more information, visit our website at http://www.wiznet.co.kr