MG2475-ISP Programming Guide

(No. ASW1004)

V1.1

REVISION HISTORY

Version	Date	Description
VER.1.0	2015.3.31	▪ First Version Release
VER.1.1	2015.05.06	▪ Changed company information

CONTENTS

1. INTRODUCTION

MG2475 is a true 2.4 GHz system-on-chip (SOC) designed for low-power and low-cost applications based on industry standards, IEEE802.15.4 and RF4CE. Some special features and peripherals such as peripherals DMA, memory and I/O retention under the power down modes, etc are added to achieve both enhanced performance and low-power. MG2475 uses an ISM band of 2.4 ~ 2.48 GHz. In addition to the standard 250Kbps data-rate specified in IEEE802.15.4, enhanced high data-rate mode (1Mbps) with channel coding is supported.

MG2475 combines an advanced RF transceiver with an industry-standard enhanced 8051 MCU, a baseband PHY, a MAC with AES-128 HW engine, an in-system programmable 64KB flash memory, a 7-KB RAM, and many other application-specific peripherals. For voice applications, the voice encoder/decoder of ADPCM and μ/a-law are embedded.

MG2475 fits best for low-cost and low-power RF4CE remote control applications.

This document explains how to download the program image to FLASH image to FLASH Memory used as code memory of MG2475.

MG2475 can distinguish Normal Mode and Programming Mode(ISP mode) by external pin. For the detailed information of the pin, refer to 'MG2475 Datasheet'.

When MG2475 is operated as ISP mode, it uses UART1 for the connection with the host based on the PC. The following shows the configuration for it.

Port	UART1(P1.0-RX, P1.1-TX)
Baud Rate	115200bps
Data Bit	8-bit
Parity	No Parity
Stop Bits	1-Stop

2. Flow Chart

The following shows the process to download the program image to FLASH Memory used as code memory of MG2475.

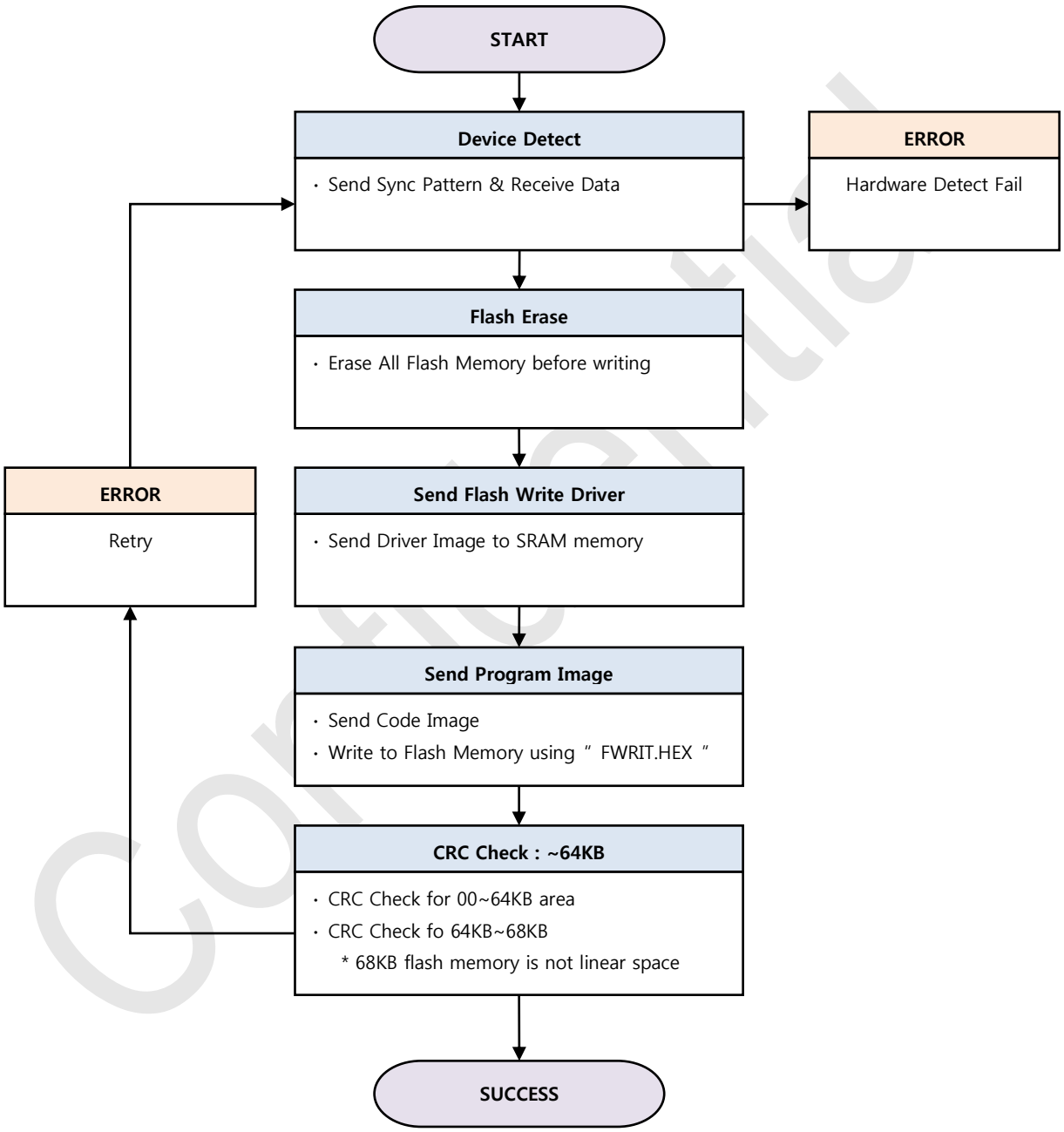


Figure 1. Process of downloading the program image

3. ISP Programming Source File

Followings are Linux and Window source files to develop ISP download program of MG2475. These source files are same to library used for “Device Programmer MD” development. A user can develop MG2475 ISP download program for own environment and purpose.

3.1. FILE Description

	Folder	File		Description	
Linux	Driver	FREAD_B0.HEX		MG2475 driver file for Flash Read	
		FREAD_B1.HEX		MG2475 driver file for Flash Read	
		FREAD_B4.HEX		MG2475 driver file for Flash Read	
		FWRIT.HEX		MG2475 driver file for Flash Write	
		INFO_PT.HEX		MG2475 driver file for Flash Read/Write	
		FLASH	WAFER_SEQ_55.HEX	File for Flash Test	
			WAFER_SEQ_AA.HEX		
		mg2475_isp.c		Main	ISP Run example
		serial.h		serial interface	Modify suitable serial interface for development environment.
		serial.c			
		intelhex.h		Intel hex file utility	Supports read, write, and merge functions of Intel hex file.
		intelhex.c			
		mg2475_rom_isp.h		ISP host handler for internal ROM	
		mg2475_rom_isp.c			
		mg2475_ihx_isp.h		ISP host handler for internal ROM	
		mg2475_ihx_isp.c			
		mg2475_hib_def.h		Hardware Information	Hardware Information define
		mg2475_fsh_check.h		Flash Test	
		mg2475_fsh_check.c			
Windows	Driver	FREAD_B0.HEX		MG2475 driver file for Flash Read	
		FREAD_B1.HEX		MG2475 driver file for Flash Read	
		FREAD_B4.HEX		MG2475 driver file for Flash Read	
		FWRIT.HEX		MG2475 driver file for Flash Write	

		INFO_PT.HEX		MG2475 driver file for Flash Read/Write	
		FLASH	WAFER_SEQ_55.HEX	File for Flash Test	
			WAFER_SEQ_AA.HEX		
	SerialPort	SerialPort.h		Windows Serial	Serial communications file for Window.
		SerialPort.cpp			
	isp	mg2475_isp.cpp		Main	ISP Run example
		serial.h		serial interface	Modify suitable serial interface for development environment.
		serial.cpp			
		intelhex.h		Intel hex file utility	Supports read, write, and merge functions of Intel hex file.
		intelhex.cpp			
		mg2475_rom_isp.h		ISP host handler for internal ROM	
		mg2475_rom_isp.cpp			
		mg2475_ihx_isp.h		ISP host handler for internal ROM	
		mg2475_ihx_isp.cpp			
		mg2475_hib_def.h		Hardware Information	Hardware Information define
		mg2475_fsh_check.h		Flash Test	
		mg2475_fsh_check.cpp			

3.2. Major Functions

3.2.1. isp_cmd(CMD, ADDR, SIZE)

Run ISP command between ISP host and MG2475 ROM.

Syntax	Void isp_cmd(unsigned char cmd, unsigned addr, unsigned size)
Description	Executes ISP command to MG2475 ROM. Please refer to Sec 5 for detailed command.
Parameter	・ cmd: ISP command ・ addr : 16bit address (2byte) ・ size : 16bit length (2byte)
Return Value	void
File	mg2475_rom_isp.h, mg2475_rom_isp.c or mg2475_rom_isp.cpp

3.2.2. isp_sync()

Check ISP mode starting of MG2475.

Syntax	void isp_sync(void)
Description	Check whether MG2475 is in ISP mode or not. TX: 0x55, 0x55, 0x55, 0x55, 0x55, 0x55 RX: 0x55, 0x55, 0x55, 0x55, 0x55, 0x55, 0xFF
Parameter	
Return Value	void
File	mg2475_rom_isp.h, mg2475_rom_isp.c or mg2475_rom_isp.cpp

3.2.3. ihx_fsh_writ(FILE)

Download firmware code to MG2475 code data area. It means this function updates entire firmware of MG2475.

Syntax	void ihx_fsh_writ(const char *fname)
Description	Updates after Flash Mass Erase.
Parameter	· fname : Intel hex 386 file name to write in flash.
Return Value	void
File	mg2475_ihx_isp.h, mg2475_ihx_isp.c or mg2475_ihx_isp.cpp

3.2.4. ihx_fsh_read(FILE)

Read entire code data area of MG2475.

Syntax	void ihx_fsh_read(const char *fname)
Description	Read current firmware code of MG2475.
Parameter	· fname : Intel hex 386 file name to store read data in flash.
Return Value	Void
File	mg2475_ihx_isp.h, mg2475_ihx_isp.c or mg2475_ihx_isp.cpp

3.2.5. ihx_hib_read(FILE, ADDR, SIZE)

Read entire hardware information area of MG2475.

Syntax	void ihx_hib_read(const char *fname, unsigned addr, unsigned size)
Description	Read Hardware Information.
Parameter	· fname : Intel hex 386 file name to store read data in flash. · addr : 0x1000 = 16bit address (2byte) · size : 0x0800= 16bit length (2byte)
Return Value	Void
File	mg2475_ihx_isp.h, mg2475_ihx_isp.c or mg2475_ihx_isp.cpp

3.2.6. intelhex_merge(FILE, BANK0, BANK1, BANK3)

Merge Intel hex files supporting Bank to one hex file.
It is NOT recommended to use f3(BANK3) for user application.

Syntax	void intelhex_merge(const char *fname, const char *f0, const char *f1, const char *f3)
Description	Merge Intel hex file by bank to one Intel hex 386 file.
Parameter	・ fname : Merged Intel hex 386 file name ・ f0 : Intel hex 80 file name in BANK0 area ・ f1 : Intel hex 80 file name in BANK1 area ・ f3 : Intel hex 80 file name in BANK3 area
Return Value	Void
File	intelhex.h, intelhex.c or intelhex.cpp

NOTE : Please refer to **"Code Banking wth Keil uVision"** for detailed Bank function.

3.2.7. fsh_check_mp()

Syntax	void fsh_check_mp(void)
Description	Test FLASH by writing 0x55/AA Pattern on MG2475 Flash.
Parameter	
Return Value	void
File	mg2475_fsh_check.h, mg2475_fsh_check.c or mg2475_fsh_check.cpp

3.3. ISP Programming Example

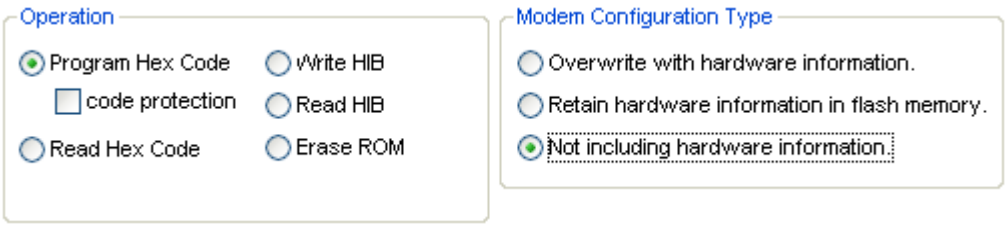
3.3.1. Download Firmware

There are three ways to download MG2475 firmware as follows;

- Not including hardware information
- Overwrite with hardware information
- Retain hardware information in flash memory

3.3.1.1. Not including hardware information

It only downloads firmware hex code. It is same to figure below from Device-Programmer MD.

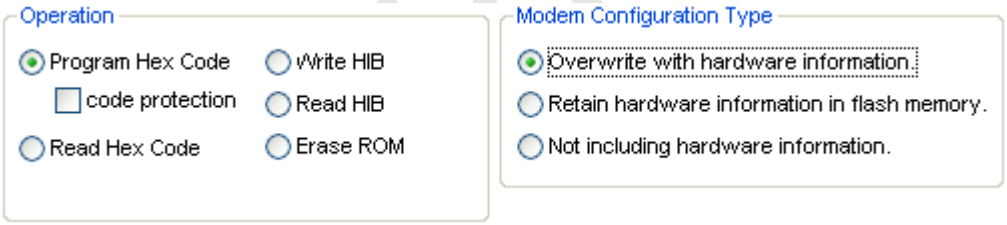


Example : "-fsh_write" option of main()

```
printf("\tCommand : fsh_write\n");
printf("\tFile Name : %s\n", argv[2]);
printf("================================================\n");

c_open_port(TTYS, 115200);
isp_sync();
printf("SYNC:SUCC\n");
fname = (char*)argv[2];
printf("Start...\n");
ihx_fsh_writ(fname);
printf("FLASH_WRITE:SUCC\n");
c_close_port();
```

3.3.1.2. Overwrite with hardware information

It downloads firmware hex code with new hardware information. It is same to figure below from Device-Programmer MD.



Example : "- hib_overwrite" option of main()

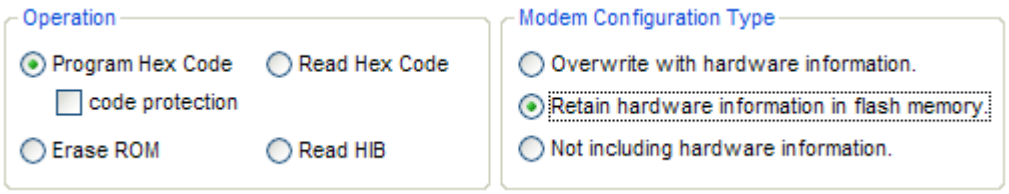
CAUTION : If the example is executed, Hardware Information area of original Intel Hex file is replaced.

```
printf("\tCommand : hib_overwrite\n");
printf("\tFile Name : %s\n", argv[2]);
printf("================================================\n");

int i=0;
int sum=0;
S_HIB HIB;
memset(&HIB, 0, sizeof(HIB));

HIB.MG2475.IEEEAddr[0] = 0x01;      HIB.MG2475.IEEEAddr[4] = 0x01;
HIB.MG2475.IEEEAddr[1] = 0x00;      HIB.MG2475.IEEEAddr[5] = 0x51;
HIB.MG2475.IEEEAddr[2] = 0x00;      HIB.MG2475.IEEEAddr[6] = 0x15;
HIB.MG2475.IEEEAddr[3] = 0x00;      HIB.MG2475.IEEEAddr[7] = 0x00;
HIB.MG2475.ChipId = CHIPID_MG2475;
HIB.MG2475.TxPower = 0;
```

```
    HIB.MG2475.DataRate = DATARATE_250_K;
    HIB.MG2475.StackID = STACKID_NONE;
    HIB.MG2475.Ch = 0x0B;
    HIB.MG2475.PanID[0] = 0x34;                    HIB.MG2475.PanID[1] = 0x12;
    HIB.MG2475.NwkAddr[0] = 0x01;          HIB.MG2475.NwkAddr[1] = 0x00;
    HIB.MG2475.SecLevel;
    HIB.MG2475.PreConfig;
    HIB.MG2475.NetworkKey;
    HIB.MG2475.Rsv_0;
    HIB.MG2475.Rsv_EPID;
    HIB.MG2475.Rsv_1;

    for(i = 0; i < (MG2475_HIB_SIZE-1); i++)
    {
        sum += HIB.Value[i];
    }
    HIB.Value[MG2475_HIB_SIZE-1] = 0 - sum;

    unsigned char *codes = (unsigned char
*)malloc(MG2475_FLASH_BANK_SIZE*4+MG2475_FLASH_INFO_SIZE);
    memset(codes,0xFF,MG2475_FLASH_BANK_SIZE*4+MG2475_FLASH_INFO_SIZE);

    fname = (char*)argv[2];
    intelhex_read(fname, codes);
    memcpy(&codes[MG2475_HIB_OFFSET], HIB.Value, MG2475_HIB_SIZE);
    intelhex_writ(fname, codes);
    free(codes);

    printf("\n");
    printf("IEEE Address : 0x%02X%02X%02X%02X%02X%02X%02X%02X\n",
                        HIB.MG2475.IEEEAddr[7], HIB.MG2475.IEEEAddr[6],
                        HIB.MG2475.IEEEAddr[5], HIB.MG2475.IEEEAddr[4],
                        HIB.MG2475.IEEEAddr[3], HIB.MG2475.IEEEAddr[2],
                        HIB.MG2475.IEEEAddr[1], HIB.MG2475.IEEEAddr[0]);
    printf("\n");

    c_open_port(TTYS, 115200);
    isp_sync();
    printf("SYNC:SUCC\n");
    printf("Start...\n");
    ihx_fsh_writ(fname);
    printf("HIB_OVERWRITE :SUCC\n");
    c_close_port();
```

3.3.1.3. Retain hardware information in flash memory

It downloads firmware hex code with hardware information of existing Flash Memory. It is same to figure below from Device-Programmer MD.



Example : “-hib_retain” option of main()

CAUTION : If the example is executed, Hardware Information area of original Intel Hex file is replaced.

```
printf("\tCommand : hib_retain\n");
printf("\tFile Name : %s\n", argv[2]);
printf("================================================\n");

int i=0;
int sum=0;
S_HIB HIB;
memset(&HIB, 0, sizeof(HIB));

c_open_port(TTYS, 115200);
char* hib_fname = "HIB.hex";
isp_sync();
printf("SYNC:SUCC\n");
printf("Start HIB Read...\n");
ihx_hib_read(hib_fname, MG2475_HIB_OFFSET, MG2475_FLASH_PAGE_SIZE);
printf("HIB_READ:SUCC\n");

printf("Replace HIB...\n");
unsigned char *codes = (unsigned char*)malloc(MG2475_FLASH_BANK_SIZE*4+MG2475_FLASH_INFO_SIZE);
memset(codes,0xFF,MG2475_FLASH_BANK_SIZE*4+MG2475_FLASH_INFO_SIZE);
intelhex_read(hib_fname, codes);
memcpy(HIB.Value, &codes[MG2475_HIB_OFFSET], MG2475_FLASH_PAGE_SIZE);
for(i = 0; i < (MG2475_HIB_SIZE-1); i++)
{
    sum += HIB.Value[i];
}
HIB.Value[MG2475_HIB_SIZE-1] = 0 - sum;

memset(codes,0xFF,MG2475_FLASH_BANK_SIZE*4+MG2475_FLASH_INFO_SIZE);
fname = (char*)argv[2];
intelhex_read(fname, codes);
memcpy(&codes[MG2475_HIB_OFFSET], HIB.Value, MG2475_FLASH_PAGE_SIZE);
intelhex_writ(fname, codes);
free(codes);

printf("\n");
printf("IEEE Address : 0x%02X%02X%02X%02X%02X%02X%02X%02X\n",
                        HIB.MG2475.IEEEAddr[7], HIB.MG2475.IEEEAddr[6],
                        HIB.MG2475.IEEEAddr[5], HIB.MG2475.IEEEAddr[4],
                        HIB.MG2475.IEEEAddr[3], HIB.MG2475.IEEEAddr[2],
                        HIB.MG2475.IEEEAddr[1], HIB.MG2475.IEEEAddr[0]);
printf("\n");

printf("Start WRITE...\n");
ihx_fsh_writ(fname);
printf("HIB_RETAIN :SUCC\n");
c_close_port();
```

3.3.1.4. Code Banking

MG2475 supports Code banking function. In this case, each Hex file of BANK0, BANK1, and

BANK3 is generated. To download these files, they shall be merged into one file.

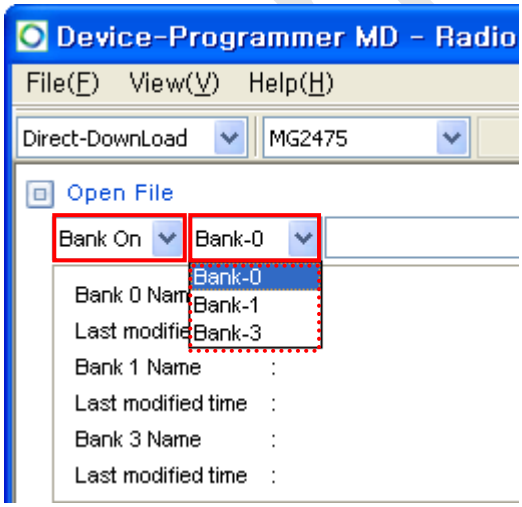
Example : “-merge” option of main()

CAUTION : It is NOT recommended to use f3(BANK3) for user application.

```
const char *f0=NULL;
const char *f1=NULL;
const char *f3=NULL;

printf("\tCommand : merge\n");
printf("\tInput File Name : \t%s\n", argv[3]);
f0 = (char*)argv[3];

if(argc>=5)
{
	printf("\t\t\t\t%s\n", argv[4]);
	f1 = (char*)argv[4];
}
if(argc>=6)
{
	printf("\t\t\t\t%s\n", argv[5]);
	f3 = (char*)argv[5];
}

printf("\tOutput File Name : %s\n", argv[2]);
printf("================================================\n");

intelhex_merge((char*)argv[2], f0, f1, f3);
```

Please refer to Sec 3.3.1.1, Sec 3.3.1.2 and Sec 3.3.1.3 for downloading procedure of merged Hex file of BANK0, BANK1, and BANK3 files.

It is same to figure below from Device-Programmer MD.



Please refer to “Code Banking With Keil uVision” for detailed Code Banking.

3.3.2. Read Hex Code

Read entire Hex code of MG2475 Flash Memory. It is same to figure below from Device-Programmer MD.

Example : "-fsh_read" option of main()

```
printf("\tCommand : fsh_read\n");
printf("\tFile Name : %s\n", argv[2]);
printf("================================================\n");

c_open_port(TTYS, 115200);
isp_sync();
printf("SYNC:SUCC\n");
fname = (char*)argv[2];
printf("Start...\n");
ihx_fsh_read(fname);
printf("FLASH_READ:SUCC\n");
c_close_port();
```

3.3.3. Initialization

Initialize entire Flash Memory of MG2475. It is same to figure below from Device-Programmer MD.

Example : "-erase" option of main()

```
printf("\tCommand : erase\n");
printf("================================================\n");

c_open_port(TTYS, 115200);
//Erase Reference Cell
printf("Erase Reference Cell....");
isp_fsh_rers();
printf("Success\n");
//Mass erase
printf("Mass erase....");
isp_fsh_mers();
printf("Success\n");
printf("ERASE:SUCC\n");
c_close_port();
```

3.3.4. Read Hardware Information

Read only hardware information from MG2475. It is same to figure below from Device-Programmer MD.

Example : "-hib_read" option of main()

```
printf("\tCommand : hib_read\n");
printf("\tFile Name : %s\n", argv[2]);
printf("================================================\n");

//from flash to hib file
c_open_port(TTYS, 115200);
fname = (char*)argv[2];
isp_sync();
printf("SYNC:SUCC\n");
printf("Start...\n");
ihx_hib_read(fname, MG2475_HIB_OFFSET, MG2475_FLASH_PAGE_SIZE);
printf("HIB_READ:SUCC\n");
c_close_port();

//from hib file to buffer
S_HIB HIB;
memset(&HIB, 0, sizeof(HIB));
unsigned char *codes = (unsigned
char*)malloc(MG2475_FLASH_BANK_SIZE*4+MG2475_FLASH_INFO_SIZE);
memset(codes,0xFF,MG2475_FLASH_BANK_SIZE*4+MG2475_FLASH_INFO_SIZE);
intelhex_read(fname, codes);
memcpy(HIB.Value, &codes[MG2475_HIB_OFFSET], MG2475_FLASH_PAGE_SIZE);
free(codes);

printf("\n");
printf("IEEE Address : 0x%02X%02X%02X%02X%02X%02X%02X%02X\n",
                        HIB.MG2475.IEEEAddr[7], HIB.MG2475.IEEEAddr[6],
                        HIB.MG2475.IEEEAddr[5], HIB.MG2475.IEEEAddr[4],
                        HIB.MG2475.IEEEAddr[3], HIB.MG2475.IEEEAddr[2],
                        HIB.MG2475.IEEEAddr[1], HIB.MG2475.IEEEAddr[0]);
```

4. Hardware Information Base (HIB)

Hardware Information is for reading writing in specific Flash Memory after selecting basic information used in the library provided by RadioPulse.

4.1. Start Address

Starting address of Flash is 0x1000.

4.2. Total Size

The size assigned to entire Hardware Information Base is 64bytes.
It actually operates reading/writing from '0x1000' to '0x17FF', 2048 bytes (page unit of chip).Flash page size is 2048byte

4.3. Structure of Hardware Information Base

Name	Byte	Description
IEEE Address	8	64bit IEEE Address. MSB of Address is stored in MSB of IEEE_ADDR
ChipID	1	0x40=MG2475
Transmit Power	1	0~25
Data Rate	1	0= 31.25 Kbps, 1= 62.50 Kbps, 2= 125 Kbps 3= 250 Kbps, 4= 500 Kbps, 5= 1.0 Mbps, 6= 1.3 Mbps, 7= 1.5 Mbps, 8= 2.0 Mbps, 9= 2.6 Mbps, 10= 3.0 Mbps
Stack Identitier	1	0x00=None, 0x01=IEEE 802.15.4 0x02=RF4CE, 0x10=Zigbee2004, 0x11=ZigBee2005, 0x12=ZigBee2006, 0x13=ZigBee2007/ZigBee Pro
RF Channel	1	Initial RF channel.(0x0B~0x1A)
Pan ID	2	16bit PanID. PanID[15:8] = Array[1] PanID[7:0] = Array[0]
Network Address	2	16bit Short(or Network) Address ShortAddr[15:8] = Array[1] ShortAddr[7:0] = Array[0]
Security Level	1	0=No Security, 1=MIC32, 2=MIC64, 3=MIC128 4=ENC, 5=ENCMIC32, 6=ENCMIC64, 7=ENCMIC128
PreConfig-Mode	1	Pre-configured Mode Value
Network Key	16	Network Key for Security MSB of Key is stored in MSB of Network Key.
Reserved_0	8	8Byte reserved buffer
Extended PanID	8	64bit Extended PanID MSB of PanID is stored in MSB of Extended PanID.
Reserved_1	8	8Byte reserved buffer
General Word-0	2	2Byte reserved buffer
General Word-1	2	2Byte reserved buffer
CSUM	1	CheckSum Value =0-(sum value of 63byte in IEEE Address~General Word-1) It means that sum value of Hardware Infromation 64 byte is set as 0.

Flash Address
IEEE_ADDR[7]: 0x1000

IEEE_ADDR[6]: 0x1001
IEEE_ADDR[5]: 0x1002
….
General_Word_1[1] : 0x103D
General_Word_1[0] : 0x103E
CSUM address : 0x103F

5. ISP COMMAND

In ISP mode, MG2475 has a ROM as Boot Loader in the chip. The ISP function is executed by this ROM code in ISP mode.

[Table 1] is a structure for ISP command transmitting.

Table 1. Definition of ISP Command Structure

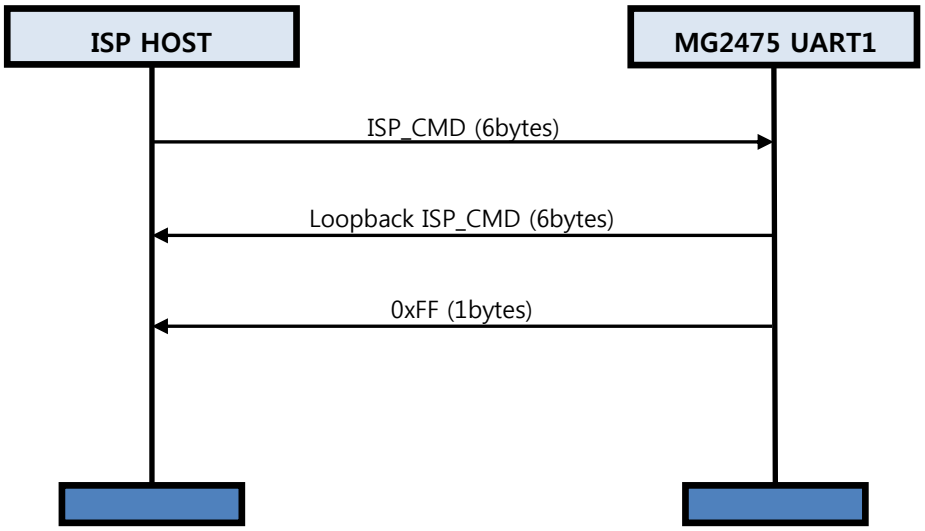
Byte : 1	1	1	1	1	1
SYNC(0x55)	CMD	ADDRH	ADDRL	LENH	LENL
		ADDR		LEN	

- SYNC : Sync Pattern (0x55)
- CMD : Command code supported by MG2475 ROM (Refer to [Table 2].)

Table 2. ISP Command List

Identifier	Description	
0x00	Flash Erase Reference Cell	
0x01	Flash checksum calculation & data compare	
0x02	Flash Bank Select (FBANK SFR setting) - bit 7: SRAM 2KB is mapped to code 0x8000~0xFFFF, - bit 6: SRAN 4KB is mapped to code 0x0000~0x7FFF, - bit 1~0: bank selection	
0x03	Flash Code Execution (Jump to execution address)	
0x04	Read the flash status register	
0x05	XDATA Read	Not Used
0x06	XDATA Write	

- ADDR : 16bit address(2byte)
- LEN : 16bit length (2byte)



ISP_CMD Format Error (isp_cmd[0] != 0x55 or isp_cmd[1] > 0x06

Figure 2. ISP Command Procedure

5.1. Flash Erase Reference Cell - 0x00

Table 3. Definition of ISP Command Structure

Byte : 1	1	1	1	1	1
SYNC	CMD	ADDRH	ADDRL	LENH	LENL
0x55	0x00	Don't Care		Don't Care	

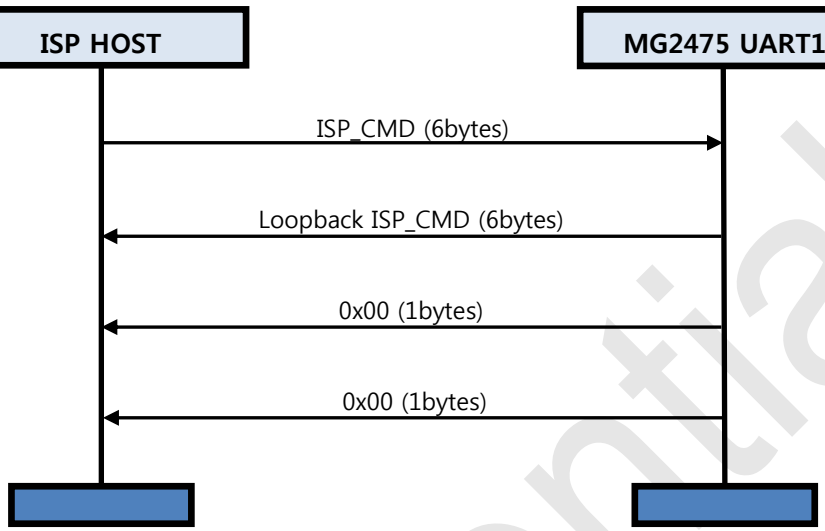


Figure 3. Flash Erase Reference Cell Command Procedure

5.2. Flash Checksum - 0x01

Table 4. Definition of ISP Command Structure

Byte : 1	1	1	1	1	1
SYNC	CMD	ADDRH	ADDRL	LENH	LENL
0x55	0x01	16bit address (2byte)		16bit length (2byte)	

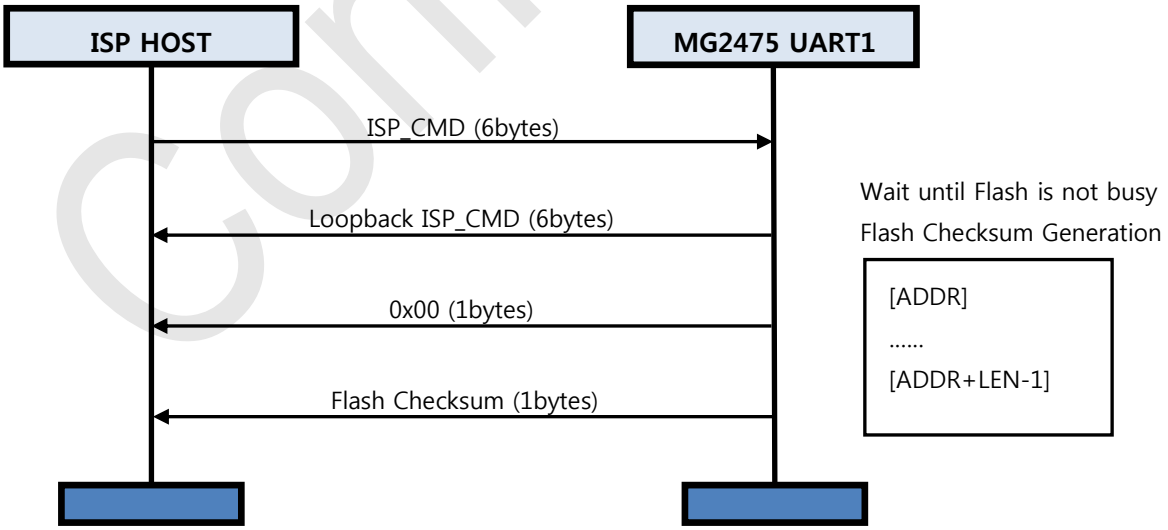


Figure 4. Flash Checksum Command Procedure

5.3. Flash Bank Select - 0x02

Table 5. Definition of ISP Command Structure

Byte : 1	1	1	1	1	1
SYNC	CMD	ADDRH	ADDRL	LENH	LENL
0x55	0x02	Don't Care		Don't Care	8bit FBANK SFR(1byte)

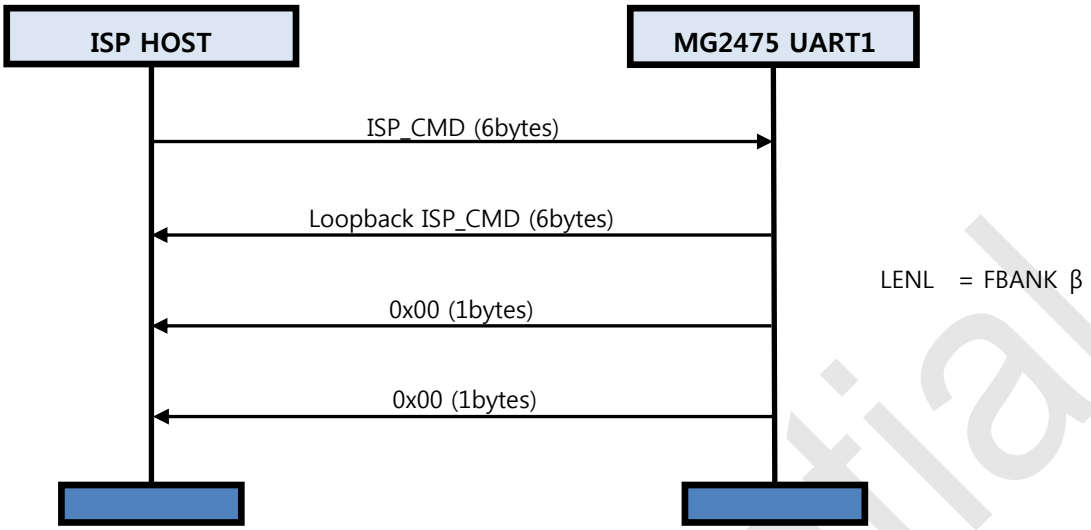


Figure 5. Flash Bank Select Command Procedure

5.4. Flash Code Execution - 0x03

Table 6. Definition of ISP Command Structure

Byte : 1	1	1	1	1	1
SYNC	CMD	ADDRH	ADDRL	LENH	LENL
0x55	0x03	16bit address (2byte)		Don't Care	

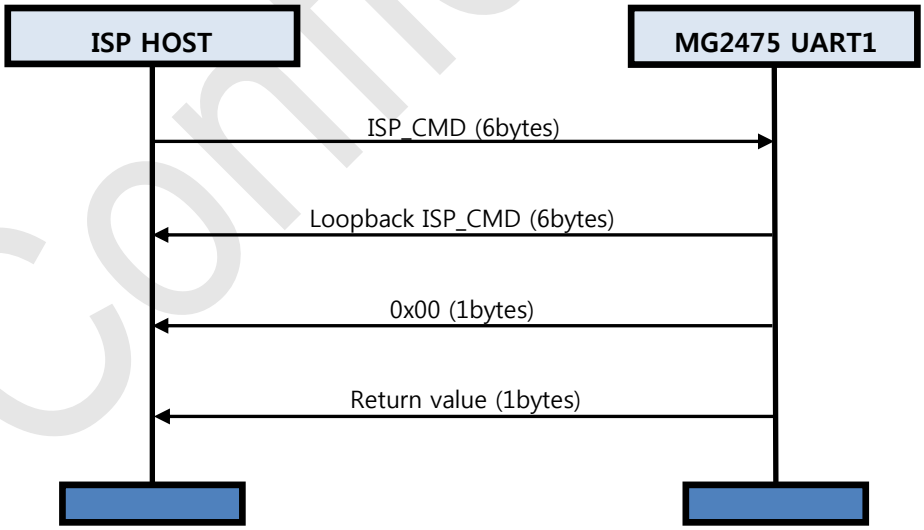


Figure 6. Flash Code Execution Command Procedure

5.5. Flash Status - 0x04

Table 7. Definition of ISP Command Structure

Byte : 1	1	1	1	1	1
SYNC	CMD	ADDRH	ADDRL	LENH	LENL

0x55	0x04	16bit address (2byte)	Don't Care

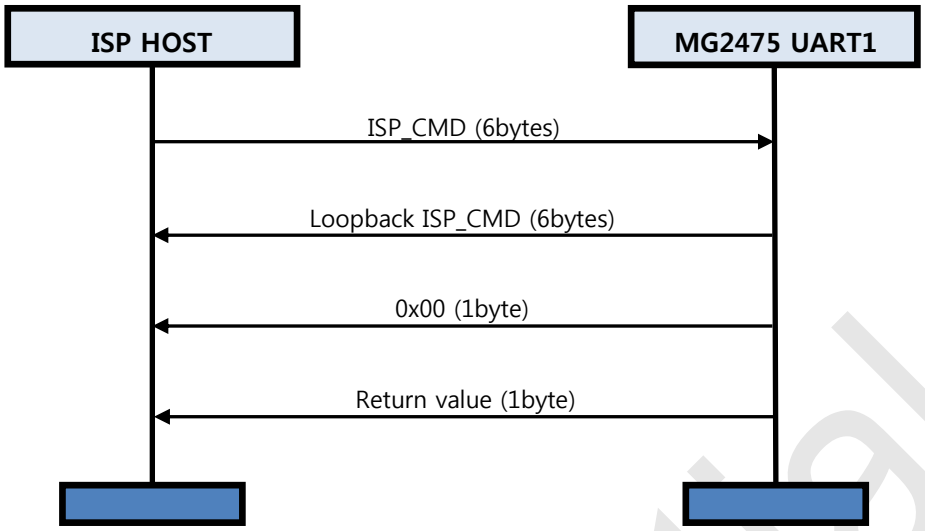


Figure 7. Flash Status Command Procedure

5.6. XDATA Write - 0x06

Table 8. Definition of ISP Command Structure

Byte : 1	1	1	1	1	1
SYNC	CMD	ADDRH	ADDRL	LENH	LENL
0x55	0x05	16bit address (2byte)		16bit length (2byte)	

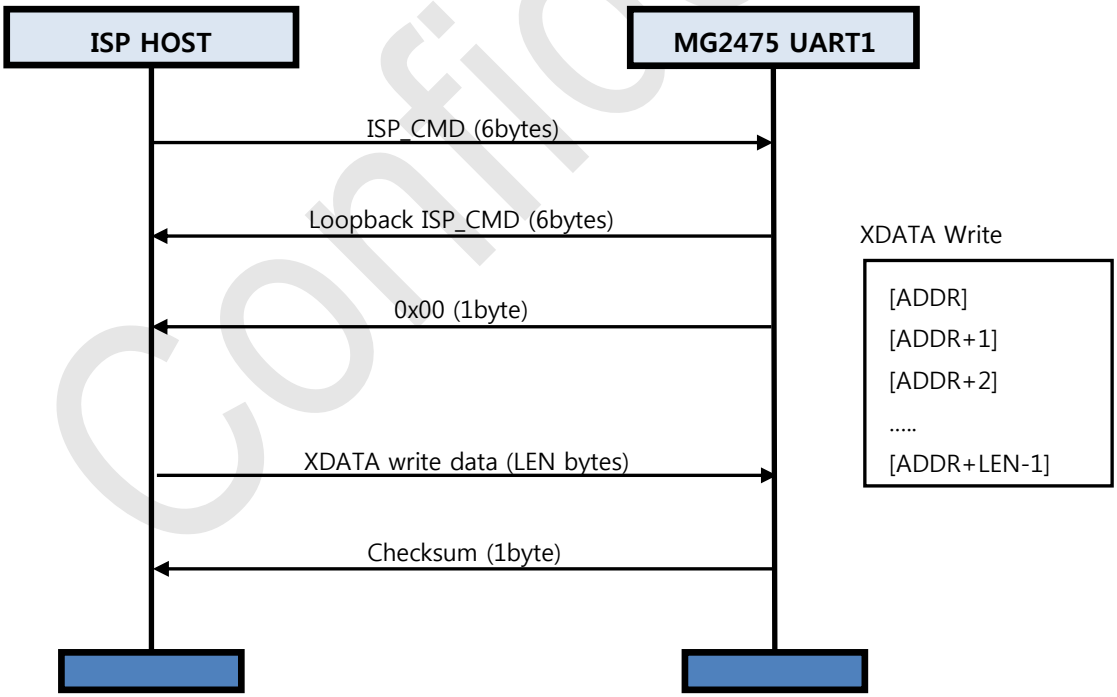


Figure 8. XDATA Write Command Procedure

RadioPulse Inc

U-SPACE #1106, 660, Daewangpangyo-ro, Bundang-gu,
Seongnam-si, Gyeonggi-do, 463-400, Korea

URL: www.radiopulse.co.kr
Tel: +82-70-7113-0925
Fax: +82-70-7113-0987

sue@ixys.net

About RadioPulse Inc.

RadioPulse is a Being Wireless solution provider offering wireless communication & network technologies and developing next generation wireless networking technologies.

The new wireless networking solutions envisioned by RadioPulse will enable user to enjoy wireless technologies with easy interface.