MCU Application Note

[Rev 1.1]

Table of contents

Chapter 1 Application Design Considerations

1 Interrupt Considerations

The interrupt vector address of the chip is 04H.

When an interrupt event occurs during the running of the program, the CPU will store the current PC value of the program into the stack, then execute the interrupt program from address 04H until the RETI instruction, retrieve the PC value in the stack and return to the main program to continue execution.

The interrupt protection site and restore site program are used to protect the site information when the program enters the interrupt, such as the value of accumulator W and status register STATUS. If PCLATH is rewritten in the interrupt, you need to add a cache register to protect PCLATH. If the PCLATH value is not rewritten in the interrupt, the protection is not required. Protecting the site and restoring the site program can ensure that the site information is consistent before and after the interruption occurs, and the main program can continue to be executed correctly after the interruption returns.

The recommended conservation site procedures and recovery site procedures are as follows:

```
W_TMP     EQU     0x70    ;Protect the cache register of W value (0x70-0x7F is the common
S_TMP     EQU     0x71    area) ;Protect the cache register of STATUS value
ORG       0004H           ;Pseudo-instruction, define the interrupt program entry address
STR       W_TMP           ;Enter the interrupt program, start to protect the site, store the W value in the cache;
SWAPR     STATUS,W        exchange the high four bits and low four bits of STATUS and store it in W; store the W
STR       S_TMP           value in the S_TMP cache
BANKSEL   PORTA           ;Set BANK to BANK0 where PORTA is
; interrupt handler       located ;Omitted interrupt handler
; interrupt handler       ; Omitted interrupt handler
INT_RET:                  ;Interrupt return label, start to restore scene
SWAPR     S_TMP,W         ;Swap the high four bits and low four bits of S_TMP cache and store it in W;
STR       STATUS          store the W value in STATUS and restore the STATUS value; swap the high
SWAPR     W_TMP, F        four bits and low four bits of W_TMP
SWAPR     W_TMP,W         ;Swap the high four bits and low four bits of W_TMP and store them in
RETI                      W ;Interrupt return
```

W_TMP and S_TMP need to be defined in the address range 0x70-0x7F shared by all BANKs. In case other BANK When entering an interrupt, the interrupt information will be lost.

BANK needs to be set after interrupting the protection site.

If the register in BANK0 is to be processed in the interrupt handler, then BANK needs to be set in BANK0. If the register in BANK1 is to be processed in the interrupt handler, then BANK needs to be set in BANK1. If BANK is not set after entering the interrupt, then the program will generate an error when directly operating the registers in BANK0 after BANK1 enters the interrupt. Similarly, when the program enters the interrupt in BANK0, the operation to the registers in BANK1 will also generate errors.

The above are the matters needing attention when using assembler programming, and there is no need to pay attention to programming in C language.

The C compiler will automatically set interrupt protection context and restore context routines.

2 key wake-up function

When the application needs to save power, it usually adopts the method of sleep and key-press wake-up to deal with it. The key wake-up function requires this IO to have a level change interrupt function. The level change interrupt function of most FMD chips is at the PORTA port, and some chips are falling edge change interrupts (such as the PORTA of the FT61F04X series), and attention should be paid to the level status.

The recommended procedure for interrupt-on-change wake-up is as follows:

```
GIE=0;                  // turn off the total interrupt
ReadAPin = PORTA;       //Read the port status and store it in the cache, set the initial value of the level change interrupt
PAIF =0;                match //clear the level change interrupt flag
IOCA2 =1;               //Enable the level change interrupt function of PA2
PAIE=1;                 //Enable interrupt-on-change
NOP();                  //It is recommended to add 1 to 2 NOP
SLEEP();                instructions//Enter sleep
NOP();                  //add 1 NOP instruction after sleep//read port state and store in cache, eliminate
ReadAPin = PORTA;       level change interrupt matching condition//clear level change interrupt flag
PAIF =0;
IOCA2 =0;               //Close the level change interrupt function of PA2
PAIE = 0;               // Disable level change interrupt
GIE=1;                  / / Open the total interrupt
```

The following points need to be paid attention to when the level change interrupt wakes up:

1) The state of the IO of this button needs to be set in the initialization. For example, if it is set as an input port and the internal pull-up resistor function is turned on, the IO is connected to the ground through a series resistor on the button. In this way, when there is no button, the level state of the port is fixed at high level, and when the button is pressed, the port level changes from high to low, triggering a level change interrupt.

2) From reading port status to going to sleep, the fewer instructions in between, the better. 3) The action of closing the general interrupt GIE before sleep will make the chip continue to execute the program after SLEEP after waking up. 4) If the general interrupt GIE is not closed before sleep, it will jump to the interrupt entry address 04H to execute the interrupt program after waking up, and continue to execute the program after SLEEP after returning from the interrupt. Therefore, it is necessary to add the actions of reading the port status and clearing the level change interrupt flag in the interrupt program, so as not to cause the program to repeatedly enter the level change interrupt and not execute the following program because the condition of the port level change interrupt always matches.

The recommended routine is as follows:

```
void interrupt ISR(void) {          // interrupt function

    if(PAIE && PAIF)                //PA level change interrupt
    {
        ReadAPin = PORTA;           //Read the port state and store it in the cache, eliminate the level change interrupt matching
        PAIF = 0;                   condition //clear the level change interrupt flag
    }
}
```

3 EEPROM Considerations

1) When using the EEPROM program, in order to prevent the instability of reading and writing EEPROM caused by the unstable voltage when powering on, it is necessary to add a delay of about 100ms to the program before operating the EEPROM. If PWRT (power-on delay) is enabled in the compilation options, it only needs to add about 40ms delay in the program before performing EEPROM read and write operations.

2) Before using the EEPROM, it is necessary to initialize the EEPROM, write "0xAA" twice in any address of the EEPROM that is not used, and do not operate to this address in subsequent programs. for example:

```
void SYSTEM_INITIAL (void) {

    EEPROMwrite(0xFF,0xAA);
    EEPROMwrite(0xFF,0xAA);        //Write 0xAA twice in any unused address (0xFF)
}
```

3) The process of writing EEPROM data needs to be carried out according to the following routine steps:

```
void EEPROMwrite(unchar EEAddr, unchar Data) {

    GIE =   0;          //Step A: Write data must turn off interrupt
    NOP();              //Note: When the chip runs at 1T, NOP() should be added here //
    while(GIE);         Step B: Wait for GIE to be 0
    EEADR = EEAddr;     //Step C: Write the address of EEPROM //Step D:
    EEDAT = Data;       Write the data of EEPROM to be written
    EECON1 |= 0x34;     //Step E: Set the three flags WREN1, WREN2, WREN3. //Step F: Set WR
    WR = 1;             to start hardware EEPROM programming //Note: When the chip runs
    NOP();              at 1T, add NOP() here //Step G: Wait EEPROM writing is completed (the
    while(WR);          process is about 2ms) //Step H: writing is completed, enable interrupt
    GIE =   1;
}
```

If the software has enabled the watchdog, the watchdog needs to be cleared after entering the EEPROM write program. If some applications have other functions to be implemented in the interrupt (such as analog PWM output), the interrupt cannot be turned off within 2ms of the EEPROM writing process, and the positions of step H and step G can be exchanged.

4) After writing the EEPROM data, the actual value of the EEPROM should be compared with the target value to be written. If they are equal, it means that the writing is successful, otherwise the writing fails, and the program can take a rewriting operation.

```
EEPROMwrite(0x00,0xCC);          //Write 0xCC to the 00H address of EEPOM
EEReadData = EEPROMread(0x00);//Read 0x00 address EEPROM value
if(EEReadData!=0xCC)             //Judge whether the actual value is equal to 0xCC
{
    EEPROMwrite(0x00,0xCC);      //Rewrite 0xCC to the 00H address of EEPOM
}
```

5) If an external reset, WDT overflow reset, LVR reset or reset caused by an illegal instruction occurs during the writing of the EEPROM, the flag bit EECON1.WRERR will be set to 1. During power-on initialization, this flag bit can be read to determine whether an abnormal situation occurred during the previous EEPROM writing process, and then take corresponding measures.

4 sleep low power consumption

In some battery applications or applications that require low power consumption, the chip must use sleep to reduce the power consumption of the whole machine. The sleep current of the FMD chip can be as low as 3uA, or as low as 0.3uA. When going to sleep, you need to pay attention to the following aspects:

1) The functional peripherals that are not used by the chip should be turned off as much as possible.

2) After the LVD/LVR function is started, there will be a current of about 15uA. During sleep, you can choose to turn off the LVD/LVR function according to the actual situation, and turn it on after waking up from sleep.

3) Ordinary IO should be set according to application needs during sleep. If it is an ordinary output port, it should be set as a power-saving output. If it is an ordinary input port, the input port should be kept at a fixed level and cannot be in a floating state.

4) Whether FOSC is selected as INTOSC in the compilation options. If INTOSC is selected, the CLKO pin of the chip will output a square wave with a frequency of Fosc/4, which will affect the sleep current.

5) Because the pins of some chips are not fully packaged, these IOs also need to be set to avoid being in a state of floating input. 6) The reset pin and IO port of some series of chips are multiplexed, and there is no internal pull-up and pull-down resistor when used as an input port. Special attention should be paid to if this IO is not used, it is necessary to set MCLRE in the compile option to MCLR function, so as to avoid this IO being in the state of floating input. Such as PA3 of FT60F01X and PA5 of FT60F02X.

7) If the application is special, turning off the watchdog function can reduce the current by about 3uA.

5 Internal pull-up/pull-down resistor function

The internal pull-up resistor/pull-down resistor is only effective when the IO is a digital input port. 1) Turn off other analog functions of the IO port first (some chips default to analog ports). 2) Set the corresponding IO as the input port.

3) Turn on the control switch of the corresponding pull-up resistor/pull-down resistor.

4) The PORTA port of some chips also needs to set OPTION.7=0(/PUPA) to enable the pull-up resistor of PORTA.

6 Precautions for PORTA operation

PORTA is an 8-bit bidirectional port. The corresponding in and out direction register is the TRISA register. Conversely, setting a bit to 0 will set the corresponding PORTA port as an output port. When it is set as an output port, the output driver circuit will be turned on, and the data in the output register will be placed into the output port. When the IO is in the input state, read PORTA, and the content of PORTA will reflect the state of the input port. When a write operation is performed on PORTA, the contents of PORTA will be written to the output register.

All write operations are a "read-change-write" process, that is, the process in which data is read, then changed, and then written to the output register. This feature requires special attention when operating some LED digital tubes.

In some applications, it is necessary to set the RDCTRL in the compilation option to LATCH according to the situation, and the value returned by reading PORTA in the output state is the buffer value of PORTA, not the real level value on the PAD.

7 Precautions for using an external crystal oscillator

When using an external crystal oscillator, when the adjacent pin of the MCU crystal oscillator inputs and outputs high-frequency pulses, it will interfere with the crystal oscillator and cause the crystal oscillator to jitter, thereby affecting the timing and causing timing deviation.

The specific adjacent pins can be seen in combination with the package pin diagram: for example, for FT61F145-TRB, PC0 is the adjacent pin of the crystal oscillator. When using an external crystal oscillator, PC0 should not be used as an IO port for high-frequency signal input and output. See the figure below for details:

model	adjacent foot	Crystal pin	adjacent foot	same model
FT60F02X	PA1	PA7,PA6	VDD	
FT61F02X	VDD	PA7,PA6	PA5	
FT61F04X	NC	PA6,PA7	PC5	
FT60F22X	PA5	PA6,PA7	PB5	
FT60F12X	VDD	PA7,PA6	PA5	FT60F11X
FT61F13X	GND	PC1, PC0	PB7	FT62F13X
FT62F08X	PC0	PC1, PB7	GND	FT61F18X
FT61F14X	PC0	PC1, PB7	GND	
FT64F0AX	PC0	PC1, PB7	GND	FT61F0AX

8 Precautions for using the communication interface

1) I2C

Plug-in FT24C02: SDA, SCL are set to open-drain output, need external pull-up

2) SPI

The external FT25C64 is connected to 4 wires, SO, SI, and SCL are set as open-drain outputs, and external pull-ups are required, and CS is driven by ordinary IO

3)USART

The direction register of TX should be set as output, RX should be set as input

9 PWM Application Considerations

Although the double buffering of the period and duty cycle can largely guarantee that the PWM output will not generate glitches, if the software writes the period and duty cycle registers very close to the TIM matching time, especially when the TIM clock frequency is faster than the system clock In some cases, unexpected situations may occur, which may cause the registers to not be expected. Therefore, it is strongly recommended to update the period and duty cycle registers, and only do it in the TIM match interrupt.

Chapter 2 Simulation Precautions

1 Simulation tool version

The simulation tool is divided into three parts, the emulator host computer software (IDE), the emulator (Link), and the firmware FW of the emulator. The version information of the IDE can be viewed in the "menu bar-Help-About..." of the IDE interface, such as 2.1.0.

The latest IDE software can be downloaded from the official website http://www.fremontmicro.com/. Turn off the anti-virus software before installation, and it can be installed on the D drive. The new version of the IDE has a built-in C compiler, so there is no need to call it separately. The version of Link is rarely updated, and the version is printed on the PCB panel, such as Link_V2.5

The firmware FW version information of the emulator can be displayed in the lower right corner of the IDE after the Link is plugged into the computer, such as FW Ver: V0.0.17, which is generally upgraded with the IDE version. If you need to manually upgrade or downgrade, you can click "Menu Bar- Help-Update Firmware..." to update the firmware, and the update file is saved in the IDE under "installation directory\Update\AppUpdate.bin".

2 Simulation connection method

1) Connect the emulator LINK to the computer, use the jumper to select the power supply voltage for the chip in the voltage option in the upper right corner of the emulator. 2) Connect the VCC, GND, CLK, and DAT pins on the right side of the LINK to the VCC, GND, ICSPCLK, and ICSPDAT of the chip. 3) After opening the project, click the Build All button to pop up the compilation option. After selecting the compilation option, click OK. IDE is available via LINK Download the program to the chip.

4) After the download is complete, it will automatically enter the DEBUG interface, and use the icon in the DEBUG toolbar to perform simulation debugging. It should be noted that the emulator needs to re-power the chip when downloading the program, so the target board cannot be connected to an external power supply, and it is best not to have a large capacitor (above 100uF) on the target board that will affect the re-powering of the chip.

ICSPCLK and ICSPDAT should not be externally connected to other components that will affect the debugging communication level status.

3 Simulation download failure processing

The failure of downloading in simulation debugging can be analyzed in two cases. One is that the download fails when using Link to connect to a single chip, and the other is that the download fails when using it for board debugging.

For the first case, the analysis is divided into the following steps:

1) Check if the tool version is up to date.

2) Check whether the link between the Link and the chip is correct and in good contact. 3) Check whether the voltage of LVR in compile option is higher than the output voltage of Link. 4) Check whether the chip model is correct, and replace the chip for testing.

5) Check whether the chip VCC, ICSPCLK, ICSPDAT communication level waveform is normal. For the second case, the analysis is divided into the following steps:

1) Use Link to connect the chip separately, and check according to the first situation. If it can be downloaded, proceed to the next step of judgment. 2) Check whether there is a large capacitance exceeding 100uF between VCC and GND of the chip. 3) Check whether there are capacitive devices or other components on the CLK and DAT pins of the chip that affect the communication level. 4) Check whether the reset pin function of the chip is turned on. Set MCLRE in compile option to PA3/PA5.

4 Simulation method of external power supply

1) Change the voltage selection jumper of the emulator from 3.3V/3.7V/5.0V to External. 2) Connect the VCC terminal of the external power supply to the VCC of the P1 port of the emulator.

3) Connect the GND terminal of the external power supply to the GND of the P1 port of the emulator.

4) Connect the four wires of the Program of the emulator to the VCC, GND, ICSPCLK, and ICSPDAT of the chip on the emulation board. 5) The external power supply voltage must not exceed 5.5V.

When using an external power supply for simulation, it is necessary to control the power supply of the external power supply to the target board through the emulator. The target board and the external power supply cannot be directly connected, and the connection must be controlled through the emulator.

Chapter 3 Programming Precautions

1 Connect to the automatic programming machine

1) Writer is powered by 7.5V/9V power supply.

2) The PROGRAM, BUSY, OK, FAIL, GND and the unmarked pins (3.3V) below the silk screen on the back of the 20PIN horn seat of the programmer are respectively connected to the START, BUSY, OK, NG of the serial port line of the burner , GND, VCC. Be careful not to connect the pin marked VDD to the VCC of the machine.

3) Set the start level, BUSY level, OK level, and NG level to L in the burner option of the burning machine, and the time parameters use the default settings.

2 Read back the chip verification code

1) Place the chip on the programmer according to the programming pins.

2) Press the KEY_MODE button on the writer lightly.

3) The programmer emits a beep and displays "IC's checksum: XXXX" on the display. 4) If this result does not appear, you need to check whether the chip is empty or whether the chip connection is normal.

3 Writer mode setting

The programmer has two functional modes: Write programming mode and Check verification mode. After downloading the program that needs to be verified to the writer, press and hold the KEY_MODE button until the writer's display shows "Write Flash", and enter the writer function selection mode. At this time, the writer mode can be switched by the red button.

There are the following modes under the Write function (the writer will automatically select the burning mode when downloading the program, manual switching is not recommended):

Write Flash:	Burn chip program HEX
Write Flash EEPROM:	Burn chip program HEX and EEPROM.BIN Burn chip program HEX and
Write Fla EEP Rcode:	EEPROM.BIN, burn rolling code Burn chip program HEX and EEPROM.BIN, burn
Write Fla EEP Rd FOSC:	rolling code and calibrate frequency Burn chip program HEX and rolling code
Write Flash Rcode:	
Write Fla Rd FOSC:	Burn the chip program HEX, burn the rolling code and calibrate the frequency
Write Fla FOSC:	Burn the chip program HEX and calibrate the frequency
Write Fla EEP FOSC:	Burn the chip program HEX and EEPROM.BIN and calibrate the
Write EEPROM:	frequency Burn EEPROM.BIN
Write EEPROM R_code:	Burn EEPROM.BIN and rolling code

There are three modes under the Check function:

Check FOSC:	Check the chip frequency (this mode will erase the chip program and need to be reprogrammed after checking) Check the
Check Flash:	program (only applicable to the case where the code is not encrypted: CPB=Disable) Check the checksum of the chip (can be
Check checksum num:	connected to the machine for automatic detection)

4 Writer power supply voltage

There are two ways to power the programmer:

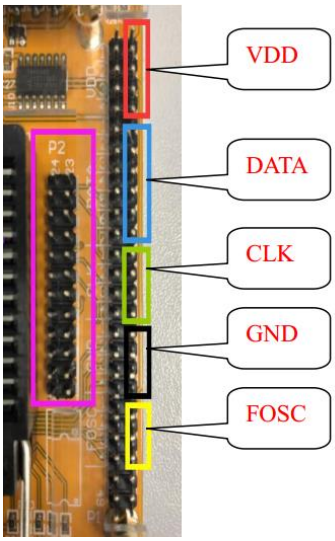
1) Powered by USB, 5V power supply can be connected to USB cable. 2)

Powered by DC power supply, the power supply voltage is 7.5V or 9V.

5 Writer on-board programming

Connect the right pins of VDD, GND, CLK, DAT, FOSC corresponding to the P1 pins on the programmer to the corresponding pins on the target board to start programming.

Refer to the following figure for details:



Some applications require that the reserved programming port does not lead to FOSC to detect the chip frequency. At this time, after importing the program file on the Setting interface of the programmer software, uncheck the Test Frequency option at the bottom, and then program via four-wire.

6 Burning failure processing

The failure of programming can be analyzed in two cases, one is failure when programming the chip alone, and the other is failure when programming the board. For the first case, the analysis is divided into the following steps:

1) Check whether the tool version is the latest and whether the chip model is correct.

2) Check whether the connection between the programmer and the chip is correct and whether the contact is good.

3) Check whether the voltage of LVR in the compilation option is higher than the output voltage of the programmer (the jumper selection on the right side of the screen of the programmer). 4) Check whether the chip VCC, ICSPCLK, ICSPDAT communication level waveform is normal.

5) Check whether the FOSC in the compilation options of the program is the external crystal oscillator mode. If the external crystal oscillator mode is adopted and the FCMEN clock failure detection function is disabled, the chip will fail to start and enter the programming mode.

For the second case, the analysis is divided into the following steps:

1) Use the programmer to connect the chip separately, and check according to the first situation. If it can be burned, proceed to the next step of judgment. 2) Check whether there is a large capacitance exceeding 100uF between VCC and GND of the board chip. 3) Check whether there are capacitive devices or other components affecting the communication level on the CLK and DAT pins of the board chip. 4) Check whether the reset pin function of the on-board chip is turned on. Set MCLRE in compile option to PA3/PA5.

Chapter Four

Notes on IDE use

1 Compilation error prompt

1) When compiling, it prompts "This Language Toolsuite is not Exists"

Generally, it is caused by the C compiler not being installed or the calling path is wrong. After opening a project in the IDE directory, the "menu bar - Project-Select Language Toolsuite..." In the pop-up box, click Browse... to select the compiler program in the compiler installation directory.

2) When compiling, it prompts "File not exist" or "No such file or directory"

Programming in assembly language requires including a header file at the beginning of the program:

include <header file.inc> or #include "header file.inc". The difference between them is the location of the header file:

use #include <header file.inc>, the compiler will check whether there is such a header file in the IDE software directory.

use #include "Header file.inc", the compiler will check whether there is such a header file in the current project directory.

2 Add the header file

Users can add custom header files by include XXX.inc or include XXX.h. After the compilation is passed, the header file will be displayed in the Include File directory of File View. Manually adding directly in the Include File is not supported at the moment.

3 IO functions of ICSPCLK and ICSPDAT

During simulation, LINK will occupy ICSPCLK and ICSPDAT to communicate with the chip, so ICSPCLK and ICSPDAT cannot simulate common IO functions during simulation. ICSPCLK and ICSPDAT can realize normal IO functions only after the program download is completed and the power is turned on again.

4 Turn off the watchdog during simulation

During emulation, if the watchdog function is enabled, please pay attention to clear the watchdog, otherwise, a reset will be generated when the watchdog overflows, so that the IDE cannot obtain the correct PC value, and a simulation error will occur. It is recommended to turn off the watchdog function during simulation debugging, and then turn on the watchdog function when the whole machine needs to be tested.

5 Project compilation failed

1) Caused by the header file.

The header file SYSCFG.H is required to use C language in the IDE, and the chip series .inc header file is included when assembly language is used. 2) Caused by the installation directory of the C compiler.

The win10 system will prohibit the IDE from modifying the files in the C compiler under the system disk. Install the C compiler on a non-system disk or right-click the IDE main program and choose to run the IDE as an administrator to solve this problem.

6 Export EEPROM file

1) Open the project and select Project->Export EEPROM Data in the menu bar to generate a .bin type file. 2) Since IDE does not support online editing of EEPROM files, it is recommended to use third-party text editing software to edit EEPROM files. Take UltraEdit as an example, open the file with UltraEdit, select Edit->Hexadecimal Function->Hexadecimal Edit in the menu bar, you can edit the 256-byte EEPROM data.

7 IDE burning HEX file

At present, the project can only be compiled and downloaded through the Build All of the IDE, and the single HEX file can only be programmed through the programmer. Subsequent IDE will add the function of burning HEX files separately.

8 The check code is different after updating the IDE

Because the early IDE version has opened the function of LVRS=1.8V, so the project with LVRS=1.8V as the compilation option of some programs will be upgraded to 2.0V by default after compiling with the new IDE. The program checksum will include the content in the compilation options, so the checksum will be slightly different from the previous checksum.

If the checksum differs significantly, it is recommended to check whether the C compiler is a limited version (the compilation information will prompt Lite Mode).

9 Library file packaging settings

1) Establish project documentation.

2) Write the program to be packaged as a C file and include it in the project document. 3) After compiling, an lpp file with the same name as the C file will be generated in the project directory. 4) Include the lpp file into the project document LIB File, delete the sub-file and recompile.

chapter Five **Notes on burning software**

1 Writer online

1) The writer is connected to the computer via a USB cable.

2) Open the writer software, the software version number will be displayed in the upper left corner of the software, such as FMDWriter V5.1. 3) The firmware version number will be displayed after DRV Ver under the FMD icon on the upper right of the writer software: such as DRV Ver: V2.1.0. If the version number is not displayed, it means that the writer has not been connected successfully. It is recommended to unplug the USB cable and reconnect.

4) If the firmware version of the programmer is relatively low, after connecting to the programming software, it will prompt that the firmware version has expired. After selecting update, select OK in the pop-up dialog box, wait for about two minutes, and the programmer will automatically complete the update. After the firmware update is complete, you need to re-plug the USB and reconnect. After the reconnection is successful, the current firmware version number will be displayed behind DRV Ver.

5) You can manually update the firmware by clicking the FMD icon on the upper right of the writer software.

2 import file

Click the "Device\Settings" button, and the programmer software will automatically pop up a setting option box. After selecting the series and model of the chip, select the Flash file and EEPROM file in the file below. If you need to burn program files, click Browse to select the Flash file to be burned. If you need to burn EEPROM, click Browse to select the EEPROM file to be burned. Select the program file separately, the programmer only burns the program area of the chip, and does not affect the original EEPROM data of the chip. Select the EEPROM file separately, the programmer only burns the EEPROM area of the chip, and does not affect the original program data of the chip. If you do not select a file, you can only read the chip data through the programming software.

After importing the file, other function settings of the writer:

1) Frequency measurement: 5% (enabled by default)

Test whether the internal frequency deviation of the chip exceeds ±5% through the chip FOSC, if it exceeds ±5%, it is judged as an NG chip. 2) Calibration frequency: 1% (closed by default)

Test whether the internal frequency deviation of the chip exceeds ±1% through the chip FOSC. If it exceeds ±1%, the programmer will calibrate the frequency to within ±1%. If the chip cannot be calibrated within ±1%, it will be judged as an NG chip. Because the frequency deviation range of the factory test of the chip is within ±1.5%, checking this item will cause some normal chips to be judged as NG because they cannot be calibrated within the range of ±1%. Therefore, it is generally not recommended for customers to check this item. If some special applications require that the internal frequency deviation range of the chip does not exceed ±1%, you can check this option. 3) The initial value and maximum value of programming number

It is used to set the programming quantity of the programmer. When the programming chip reaches the maximum value, the programmer will no longer program and the file needs to be re-imported. 4) rolling code

It is used to set the rolling code of the chip, which is stored in the EEPROM area of the chip.

3 Read the check code of the chip

The check code of the chip can be read back through the Key_Mode button on the writer and displayed on the screen of the writer. In addition, the programmer software can also read the contents of the program area and EEPROM area of the chip.

If the chip CPB is encrypted, it can only read EEPROM files. For the read back data, you can click Flash in IC or EEPROM in IC to view the corresponding data.

4 The functions of each button of the programmer software

Auto Program: The programmer will execute "erase", "program programming" and "program verification" in sequence. Only the chip will be erased (Flash and EEPROM can be selected for erasing), and only the chip will be blank checked.

Erase:

Blank Check:

Program: Only execute the burning action on the chip, without checking whether the burning is successful

Verify: Check whether the code in the chip is correct, (if the programming program enables CPB, it cannot be verified) Read out the chip information (if the chip is encrypted, you can only read out the verification code and EEPROM data) Set the number of programming

Read:

Setting:

Rolling Code: Rolling Code Settings

Calibration Frequency: Calibration frequency (checked, the chip will be automatically calibrated if the frequency offset is detected) Test Frequency: Detection frequency (checked, the chip that detects the frequency offset will be treated as NG)

5 The programmer software does not display the FT61F04X chip

After clicking the Device/Load button, a dialog box for selecting the chip model will pop up. If there is no serial number of the current chip, you need to check:

1. Whether the programmer is online normally. (Does DRV Ver show the firmware number of the writer?)

2. Whether the burner is a yellow version burner. Early black/blue/green writers cannot program 4K and above capacity chips. 3. If it is a yellow version, please confirm whether the firmware number of the writer is the latest version.

6 Writer alarm prompt description

Write Flash fail: Burning Flash failed

Write UCFG fail: Burning user configuration failed

Write EEP fail: Burning eeprom failed

Write UID fail: Failed to burn rolling code

NO Flash File: There is no burning program in the burner

No Eeprom File: There is no eeprom program in the burner

Uid Not Loaded: There is no rolling code in the burner

ALL UIDs USED: All rolling codes have been used

Rd CPU_STAT Err: The chip pins are wrongly connected or poor contact

Read ICID Err: The programming chip does not match the HEX corresponding model downloaded by the programmer

ICID_1T MATCH NG: Some versions of FT61F14X, FT61F0AX, FT64F0AX, FT67F0AX do not support 1T operation model

fosc check fail: The frequency measurement fails, the frequency deviation of the core is too large, or the CLKO pin is wrongly connected and the contact is bad, which causes the

fosc cal fail: chip frequency calibration to fail.

Read FCFG Err: If it is an encryption program, it means that the password is not the same as the secret of the chip configuration, and it must be reprogrammed after confirming the passwords of both parties; or the internal information of the chip is wrong.

No ICSeries info: After updating the firmware, it will prompt to let the user import the burning program

Chapter Six Assembly language programming considerations

1 Pseudo-command BANKSEL

The pseudo-instruction BANKSEL is used to set the BANK of the register, and its function is to set the BANK to the BANK where the register behind BANKSEL is located.

Such as FT60F01X and FT60F02X:

```
BANKSEL   PORTA; because PORTA is in BANK0, the instruction is equivalent to BCR   STATUS,5
BANKSEL   TRISA; because TRISA is in BANK1, the instruction is equivalent to BSR   STATUS,5
```

In FT61F02X, there are 3 registers BANK: BANK0, BANK1, BANK2, which need to be set through BIT5 and BIT6 of STATUS, so one BANKSEL is equivalent to two instructions.

```
BANKSEL   PORTA, equivalent to:
    BCR     STATUS,5
    BCR     STATUS,6
BANKSEL   TRISA, equivalent to:
    BSR     STATUS,5
    BCR     STATUS,6
```

This should be paid special attention when it is necessary to accurately calculate the number of instructions.

2 assembler instruction cycles

The assembly instruction cycle refers to the time required to execute an assembly instruction.

According to different series of chips, the instruction cycle time will be different.

Currently we have 1T, 2T, 4T instruction cycles.

The instruction cycle of 1T means that one system clock cycle is required to execute one assembly instruction cycle. The instruction cycle of 2T means that it takes 2 system clock cycles to execute one assembly instruction cycle. The 4T instruction cycle means that it takes 4 system clock cycles to execute an assembly instruction cycle.

For example: FT60F01X is 4T instruction cycle.

When OSCCON.IRCF[2-0]=111 is selected, the internal oscillation frequency Fosc=16M. The system clock cycle is 1/16M seconds. At this time, one instruction cycle is 4*1/16M=1/4M second=0.25us. That is, it takes 0.25us to execute a NOP. FT60F02X is 2T/4T optional instruction cycle. It can be selected as 2T instruction cycle in Tsel of compile option.

When OSCCON.IRCF[2-0]=111 is selected, the internal oscillation frequency Fosc=16M. The system clock cycle is 1/16M seconds. At this time, one instruction cycle is 2*1/16M=1/8M second=0.125us. That is, it takes 0.125us to execute a NOP. Most of the FMD assembly instructions are single-cycle instructions, that is, one instruction occupies one instruction cycle.

Some instructions with the function of jumping to PCL are two-cycle instructions, such as LJUMP, LCALL, RET, RETI, RETW. Part of the judgment jump instruction is a double instruction cycle when the jump condition is met, and a single instruction cycle when the jump condition is not met. Such as BTSS, BTSC, INCRSZ, DECRSZ.

3 Assembly instruction function description

mnemonic	impact sign	cycle	Function Description
BCR R,b	NONE	1	Bit b of register R is cleared to 0
BSR R,b	NONE	1	bit b of register R is set to 1
BTSC R,b	NONE	1 or 2	Skip next instruction if bit b of register R is 0, 2 cycles when jump occurs
BTSS R,b	NONE	1 or 2	Skip next instruction if bit b of register R is 1, 2 cycles when jump occurs
NOP	NONE	1	empty instruction
CLRWDT	/PF, /TF	1	clear watchdog
SLEEP	/PF, /TF	1	sleep command
STR R	NONE	1	store value from accumulator W into register R
LDR R,d	Z	1	Import the value of register R into accumulator W (d=W/0) or register R (d=F/1)
SWAPR R,d	NONE	1	Exchange the high and low four bits of register R and store them in accumulator W (d=W/0) or register R (d=F/1)
INCR R,d	Z	1	Add 1 to the value of register R, and store the result in accumulator W (d=W/0) or register R (d=F/1)
INCRSZ R,d	NONE	1 or 2	Add 1 to the value of register R, and store the result in accumulator W (d=W/0) or register R (d=F/1) If result equals 0, skip next instruction, 2 cycles when jump occurs
ADDWR R,d	C, HC, Z	1	Add the value of the accumulator W to the value of the register R, the result is stored in the accumulator W (d=W/0) or register R (d=F/1) If the result is greater than 255, then C=1
SUBWR R,d	C, HC, Z	1	Subtract the value of the accumulator W from the value of the register R, and store the result in the accumulator W (d=W/0) or register R (d=F/1) If R>=W, then C=1
DECR R,d	Z	1	The value of register R is subtracted by 1 and the result is stored in accumulator W (d=W/0) or register R (d=F/1)
DECRSZ R,d	NONE	1 or 2	The value of register R is subtracted by 1, and the result is stored in accumulator W (d=W/0) or register R (d=F/1) If result equals 0, skip next instruction, 2 cycles when jump occurs
ANDWR R,d	Z	1	The value of register R is ANDed with the value of accumulator W, and the result is stored in accumulator W (d=W/0) or register R (d=F/1)
IORWR R,d	Z	1	The value of register R is ORed with the value of accumulator W, and the result is stored in accumulator W (d=W/0) or register R (d=F/1)
XORWR R,d	Z	1	The value of register R and the value of accumulator W are XORed, and the result is stored in accumulator W (d=W/0) or register R (d=F/1)
COMR R,d	Z	1	The value of register R is reversed, and the result is stored in accumulator W (d=W/0) or register R (d=F/1)
RRR R,d	C	1	The value of register R is shifted right with C bit, and the result is stored in accumulator W (d=W/0) or register R (d=F/1)
RLR R,d	C	1	The value of register R is left shifted with C bit, and the result is stored in accumulator W (d=W/0) or register R (d=F/1)
CLRW	Z	1	clear accumulator W
CLRR R	Z	1	clear register R
RETI	NONE	2	interrupt return
RET	NONE	2	subroutine return
LCALL N	NONE	2	Call subroutine N
LJUMP N	NONE	2	jump to label N
LDWI I	NONE	1	Load the immediate value I into the accumulator W
ANDWI I	Z	1	The value of the accumulator W is ANDed with the immediate value I, and the result is stored in the accumulator W
IORWI I	Z	1	The value of the accumulator W is ORed with the immediate value I, and the result is stored in the accumulator W
XORWI I	Z	1	Exclusive OR the value of the accumulator W with the immediate value I, and store the result in the accumulator W
RETW I	NONE	2	The subroutine returns and stores the immediate value I into the accumulator W
ADDWI I	C, HC, Z	1	The value of the accumulator W is added to the immediate value I, and the result is stored in the accumulator W. If the result is greater than 255, then C=1
SUBWI I	C, HC, Z	1	The value of the accumulator W is subtracted from the immediate value I, and the result is stored in the accumulator W. If I>=W, then C=1

4 Compilation lookup table

Special attention should be paid to the processing of PCLATH when processing the look-up function in assembly instructions. In the assembly, the form data is usually returned through RETW XXX. The commonly used digital tube table lookup program routine is as follows:

```
        LDR     DIG_2,W     ;Assign the second bit to W
        LCALL   GET_CODE    ;Look up the table
        STR     TEMP0       ; Data stored in TEMP0

        ORG     0X300       ; Define table program address to 300H address
GET_CODE:
        LDWI    0X03
        STR     PCLATH      ;Set PCLATH=3 (follow table actual address)
        LDR     BCD,W       ;Look up the table
        ADDWR   PCL,F       ;--GFEDCBA
        RETW    B'00111111' ;0
        RETW    B'00000110' ;1
        RETW    B'01011011' ;2
        RETW    B'01001111' ;3
        RETW    B'01100110' ;4
        RETW    B'01101101' ;5
        RETW    B'01111101' ;6
        RETW    B'00000111' ;7
        RETW    B'01111111' ;8
        RETW    B'01101111' ;9
        RETW    B'00000000' ;OFF
```

5 LJUMP and LCALL instructions

Because of the limitation of PC width, LJUMP and LCALL can only access 2KROM space at most. Chips with a program space within 2K do not need to consider jump errors when using LJUMP and LCALL instructions. When using LJUMP and LCALL to access the space above 2K for chips with a capacity above 2K, PCLATH needs to be set. For chips with more than 4K program space, it is recommended to use C language programming, and the compiler will automatically handle it.

6 Use of labels

When setting the subroutine label or position label in the assembly, you need to pay attention to adding a colon in the label: without the colon, it will be recognized as a macro definition by the compiler. If there is no such macro definition in the header file, the compiler will report an error.

Chapter VII C language programming considerations

1 include the header file

At the beginning of the program, the "syscfg.h" file provided by the compiler must be included with the #include directive to realize the declaration of special function registers and other special symbols. It is not necessary to include the header file of the chip model separately, the IDE will automatically invoke it.

Other subroutine files and header files also need to be included with the #include directive in the Main file.

2 variable type

The types of commonly used variables are as follows: It should be noted that char is an unsigned character variable by default.

type	length (bits)	mathematical expression
bit	1	Boolean variable
char	8	char variable, defaults to unsigned char variable
signed char	8	signed char variable
unsigned char	8	unsigned char variable
int	16	Integer variable, the default is a signed integer variable
signed int	16	signed integer variable
unsigned int	16	unsigned integer variable
long	32	Long integer variable, the default is a signed long integer variable
signed long	32	signed long variable
unsigned long	32	unsigned long variable
float	twenty four	floating point number
double	twenty four	double precision floating point

3 Variable declaration

const: constant variable

If the prefix const is added to the variable definition, then this variable will become a constant variable, which cannot be modified during the running of the program, and is usually used for the definition and declaration of constant arrays.

volatile: resident memory variable

The C compiler will optimize the program, and some variables that have no practical effect in the program will be optimized, and the actual value of the variable cannot be observed during the debugging process. If you want a variable not to be optimized, you need to add a volatile prefix to declare the variable definition. persistent: Non-initialized variables

When the program is powered on and running, all defined variables will be cleared or initial values will be set. But in many practical applications, some variables are not initialized by running the program. At this time, a persistent prefix declaration can be added before the variable definition.

4 Absolute address variables

Some applications need to put 8 bit variables in the same byte for batch operation, which can be done in two ways:

1) By defining a bit field structure and a byte variable combination:

```
union {
    struct {
        unsigned bit0:  1;
        unsigned bit1:  1;
        unsigned bit2:  1;
        unsigned bit3:  1;
        unsigned bit4:  1;
        unsigned bit5:  1;
        unsigned bit6:  1;
        unsigned bit7:  1;
    } oneBit;
    unsigned char allBits;
} TESTFlag;
```

When setting one of the bits, the operation is as follows: `TESTFlag.oneBit.bit5=1;` //bit5 position 1

When clearing bit variables in batches, the operation is as follows: `TESTFlag.allBits=0;` //Bit variables are cleared to 0 in batches

2) Realize by defining the absolute address of the variable

```
volatile unsigned charTESTFlag @ 0x20; // TESTFlag is defined at address 0x20
bit TESTBit0 @((unsigned)&TESTFlag*8)+0;//TESTBit0 corresponds to the 0th bit of
bit TESTBit1 TESTFlag@((unsigned)&TESTFlag*8)+1;//TESTBit1 corresponds to the 1st bit of
bit TESTBit2 TESTFlag@((unsigned)&TESTFlag*8 )+2;//TESTBit2 corresponds to the second bit of
bit TESTBit3 TESTFlag @((unsigned)&TESTFlag*8)+3;//TESTBit3 corresponds to the third bit of
bit TESTBit4 TESTFlag@((unsigned)&TESTFlag*8)+4;//TESTBit4 corresponds to At the 4th bit of
bit TESTBit5 TESTFlag@((unsigned)&TESTFlag*8)+5;//TESTBit5 corresponds to the 5th bit of
bit TESTBit6 TESTFlag@((unsigned)&TESTFlag*8)+6;//TESTBit6 corresponds to the 6th bit of
bit TESTBit7 TESTFlag@(( unsigned)&TESTFlag*8)+7;//TESTBit7 corresponds to the 7th bit of TESTFlag
```

It should be noted that the compiler does not reserve address space for absolutely positioned variables.

The address of the variable TESTFlag in the above routine is 0x20, but the last address 0x20 may be assigned to other variables by the compiler, so an address conflict occurs. Therefore, the absolute address of the user-defined variable needs to be prefixed with a volatile statement when defining the variable to prevent the compiler from automatically optimizing the allocation address.

Such usage will reduce the usage efficiency of RAM, so it is not recommended to define the absolute address of variables in general program design.

chapter eight **Precautions for specific models**

1. FT60F01X

1) Notes on the application of PA3:

PA3 is a unidirectional input port. When used as an input port, it must be pulled up externally.

Only when PA3 is set as a reset pin, the internal weak pull-up will be turned on.

FT60F011A-RB When PA3 is not used in the actual project, it should also be set as a reset pin to reduce power consumption. FT60F010A-URB PA3 is not packaged, so it must also be set as a reset pin

2. FT60F02X

1) PA0, PA1 are set as internal pull-up:

Close the comparator analog function, CxIN is set as a digital IO pin: CMCON0: CM[2:0]=111

3. FT60F12X/FT60F11X

1) CLKO:

After E version CLKO is mapped to PC5 instead of PA6 2) External crystal oscillator

Set external crystal oscillator: FOSC:XT (PA6, PA7 connected to high-speed crystal 4~20Mhz)

A, IESO: Enable (enable double-speed clock), IRCF[2:0]=000, must be set to 000

B, IESO: Disable (double-speed clock not enabled), IRCF can be set at will

3) PWM

When the PWM duty cycle - PWM period = 1, the level of the PWM output will be reversed, and the user needs to write a program to avoid this problem

4. FT61F02X

1) PA0 PA1 PA1 PA3 set input pull-up

A, close the comparator CMCON0 = 0B00000111,

B, Close the analog port ANSEL = 0X00;

C, set as input TRISA = 0B11111111;

D, pull up WPUA = 0xFF;

5. FT61F04X

1) op amp

The op amp can only input negative voltage. If the input signal is positive voltage, it is recommended that customers use FT64F0AX

6. FT61F14X

1) PWM IO mapping [TIM1_CH2]

PB0 cannot output PWM alone, only PA1 can output alone. When PB0 is set, PA1 will also output PWM. When TIM2_CH1 output is enabled, PB0 cannot be configured as TIM1_CH2 output, otherwise PB0 will output TIM2_CH1 waveform. 2) TIM4 is connected to an external crystal oscillator 32.769k for low power consumption applications

If the system clock is HIRC, TIM4 uses LP. When sleeping, the system clock cannot be turned off, and the power consumption is not small. At present, it can be set, enter sleep, the system clock is set to internal 32K, TIM4 interrupt wakes up 6014, 7001, 6006 are the same

The starting capacitor is recommended to be 33pF

7. FT62F08X/FT61F08X

1) Precautions for interrupt function application:

In the C language program, the interrupt function must be reserved, otherwise the compiler will report an error 2) Sector encryption cannot read CHECKSUM

CPB enable, can read CHECKSUM.

If FSECPB0 sector encryption is enabled, CHECKSUM cannot be read. It should be noted that other sector encryption does not affect 3) ADC application notes:

1. ADC calibration, the register must be set before calibration

2. The analog channel 1/4VDD is sampled by resistor division. CHS[3:0] should be switched to other channels before sleep, otherwise it will consume power and cause

The sleep current is relatively large

4) When FT62F088-LRB is used as a touch function, PC7 is occupied by the touch library and must be suspended, and cannot be used as a normal IO port

8. FT62F21X/FT60F21X

1) When outputting the P1D PWM function alone:

The lower 7 bits of P1DDTL cannot be 0 at the same time. For example, when P1DDTL=0X80, P1D will output low level instead of PWM waveform. The same is true when using mapping. This can only be avoided by software at present, and P1CDTH must be assigned a value. must pay attention

9. FT62F13X/FT61F13X

1) PA3, PA5 are not packaged models, they are connected with GND, the software should not set them high, so as not to cause leakage 2) FT61F131B-RB, PB3 (PC0) sealed, PA1 (PA4) sealed, one The IO port must be set as a floating input in order to use another IO function normally, pay attention

3) If the instruction cycle is set to 4T, the EFT anti-interference characteristics of the IO will be stronger