FCM32F0xx/FCM32H0xx Application Notes

(For D Version IC)

FCM32F0xx/FCM32H0xx is a MCU Chip based on the Cortex M0 core, it's compatible with the S**32F0xx series. However, there are some differences between them due to the production process and circuit design.

This document applies to FCM32x030, FCM32x031 and FCM32x042.

Contents

1. Similarities

- M0 Core
- Pin Assignment
- Memory map
- Development environment (Keli/IAR etc.)
- Writer tools
- Firmware library / Example code
- Same frequency performance
- Timing compatibility

2. Differences

Product	STM32F0	FCM32F0	FCM32H0
VDD	2.0~3.6V	1.8~5.5V	1.8~5.5V
VDDA	2.0~3.6V	1.8~5.5V	1.8~5.5V
Operating Temperature Range		-40~125	-40~125
Coremark Performance	103.8@48MHz	103.7@48MHz	211.3@96MHz
CPU/AHB/APB Max. Frequency	48MHz	48MHz	100MHz
Flash Operating Frequency	24MHz	24MHz	32MHz
Flash Turbo	-	-	√
TIM Clock	48MHz	48MHz	200MHz
SPI Max. Frequency	24MHz	24MHz	50MHz
UART Max. Baud Rate	6Mbps	6Mbps	12.5Mbps
SPI2 Act-as I2S	x	√	√
Hardware Division Arithmetic	x	√	√
Hardware Square Root Arithmetic	x	√	√
HDMI-CEC	√	x	x
Boot loader	USART/IIC/USB	USART	USART
Working Current(HSI8MHz)	4.9mA		3.7mA
Working Current (HSI48MHz)	22mA		15.0mA
Working Current (HSI8+PLL96MHz)	-	-	27.5mA
Stop Current(I_{DD}+I_{DDA})	6.2uA	6.2uA	6.2uA
Standby Current(I_{DD}+I_{DDA})	3.6uA	6.0uA	6.0uA

Note:

1. Use S**32F042 as reference.
2. The test conditions are 3.6V/25C, LSI/IWDG OFF and VDDA Monitor ON.
3. S**32F0 supports different boot ports according to the different models. Not all devices support the ports listed in the table.

3. Notes

3.1 FCM Device Recognition

FCM32F0/H0 series MCU models can be read at the address below in the information block to distinguish them from other manufacturers.

Address	Content	Description
0x1FFF_F790	0x46433332	'FC32' ASCII code

0x1FFF_F794	0x0046xxxv/
	0x0048xxxv
		46 = 'F', 48 = 'H'
		xxx=type, eg. 030
		v=version, eg. A/B/C

3.2 ADC

3.2.1 ADC Accuracy

In some applications, the ADC accuracy is not enough, because the ADC electrical characteristics of FCM are not the same as S**. For the same sampling time, the input impedance (R_{AIN}) of FCM is lower. The R_{AIN} table is listed in the specification already.

Either solution or a combination of the below two:

1) Adjust the sampling time according to the external input impedance.
2) Reduce the external input impedance.

3.2.2 Software Triggers a Multi-channel Discontinuous Conversion doesn't work properly

When an ADC needs to discontinuously convert multiple channels (DISCEN=1, CONT=0) and it's triggered by software (it means: only one channel conversion for each trigger), the ADC does not work properly and the result of each conversion is the first channel in the sequence.

Solution:

Using a software-controlled channel switch mode, only one channel is allowed at a time before starting the conversion.

For example, the channels need to be converted are 4, 16, and 17, in the red box are the added code.

1) After the ADC configuration is complete, set the channel to be the last channel (17 is the last channel in this example) (the original program set CHSELR to 4/16/17 in the ADC_Config () firmware).

```
/* Configure the ADCx peripheral */
ADC_Config();
ADC1->CHSELR = 1<<17; //  (1) set ch to last want to converted
```

2) Before starting ADC conversion, set the channel want to be converted.

```
//  set only one ch every convert start
if (ADC1->CHSELR & (1<<17))
  ADC1->CHSELR = 1<<4;
else if (ADC1->CHSELR & (1<<4))
  ADC1->CHSELR = 1<<16;
else
  ADC1->CHSELR = 1<<17;
if (HAL_ADC_Start(&AdcHandle) != HAL_OK)
{
  Error_Handler();
}
```

3.2.3 ADC Bit Selection

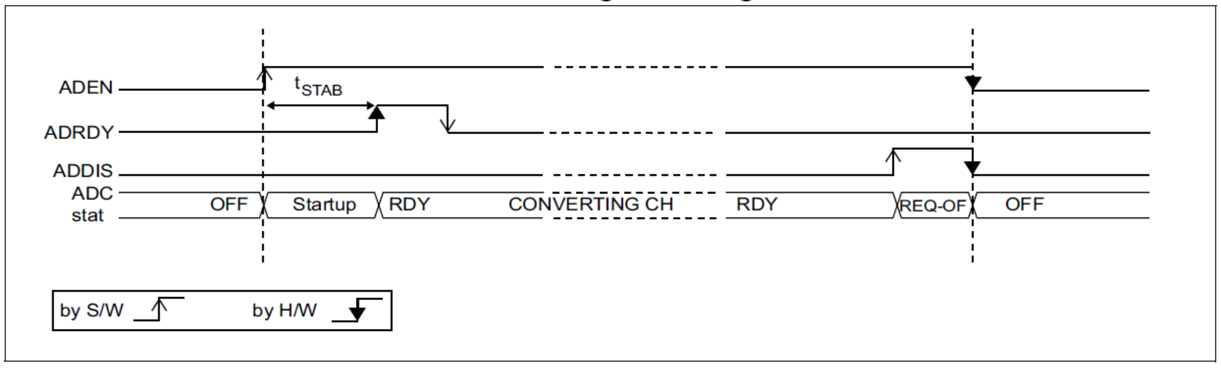
ADC supports data bits to be set to 6/8/10/12, but the ADC conversion time does not decrease with the reduction of bits. The conversion time is fixed to the conversion time of 12 data bits.

3.2.4 ADC Trigger be lost before ADC RDY

After the ADC allowed, it needs to wait the time of t_{STAB} before allowing the conversion, during this period, the ADC trigger is invalid (including software trigger).

The timing allowed by the ADC is as follows:

Enabling/disabling the ADC



Therefore， after the ADC allowed, and the ADC software trigger is executed at once, ADC will not initiate the conversion. As shown below, some users' program do not follow this rule.

Solution:

Wait for the ADC to be READY before starting the conversion, using the code in the red box instead of the previous line.

```
  ADC_Cmd(ADC1 , ENABLE);              //使能ADC1
//  while(!ADC_GetFlagStatus(ADC1 , ADC_FLAG_ADEN));     //ADC_FLAG_ADRDY
  while(!ADC_GetFlagStatus(ADC1 , ADC_FLAG_ADRDY));
  ADC_StartOfConversion(ADC1);     //启动转换
}
```

3.3 VBAT Power Supply

The FCM MCU series do not provide backup battery power in Power Down Mode, and when the main VDD is not provided, the VBAT will continue to supply power to the VDD by a diode inside the IC. Therefore, for applications where power consumption is important, other power supply measures should be taken.

3.4 SPI

3.4.1 SPI Receives Errors in Bits 9 to 16

When the SPI is set to 9 to 16 bits, the 8-bits MSB of 16-bit data is always be 00.
Solution:
Use 8-bit data read mode, read twice. If DMA is used for receiving, the size and length of DMA data are set to 8 bits both.
Demonstration code (assuming high bit first)：

```
uint16_t tmp;
uint16_t data;
data = *((__IO uint8_t *)&SPIx->DR);    //  The lower 8 bits are read first in 8-bit mode
tmp = *((__IO uint8_t *)&SPIx->DR);     //  The higher 8 bits are ready first in 8-bit mode
data |= (tmp<<8);                       //  Merge
```

3.4.2 SPI Slave Receives Error

When SPI operates in slave mode, SCK is same as master frequency /2, and the received data is back-to-back (no gaps between bytes), then receive meet error.

Solution：

Requires SCK<= main frequency /3, in order to keep the correction when receiving back-to-back data from the master.

3.5 CAN

3.5.1 FCM CAN is not binary compatible with S**T

Using the CAN library provided by FCM to recompile.

3.5.2 CAN Clock Accuracy Requirements

When using the CAN function, the MCU clock source must use crystal oscillator and can't use internal RC OSC.

3.6 RCC

3.6.1 TIM1 Input Clock Requires 2x PLLCLK (New function of FCM)

TIM1 peripherals can use 2*PLLCLK as a clock source as follows:
It is implemented by the TIM1SW bit of RCC_CFGR3. When 2x PLLCLK is selected, SYSCLK/HCLK/PCLK must be the same frequency.
That is, without any frequency division.
RCC_CFGR3
Address: 0x30
Reset value: 0x0000 0000
[9]: TIM1SW, 1 = select 2x PLLCLK as TIM1 clock source

3.6.2 Clock Configuration Fails When HSE is used

When the firmware configured the clock to use HSE, HSERDY was not waited after the HSE was started, resulting in a clock configuration failure.
The reason is that the reset voltage of FCM32 is about 1.6V, and some crystal is difficult to startup under this voltage, and HSERDY signal will not appear until VDD rises to a certain voltage. In this way, when VDD rises slowly, MCU has started to execute clock configuration program and carry out overflow timeout count at 1.6V voltage. There is no HSERDY signal before the count overflow, so the clock configuration failed

Solution：

Modify the definition of HSE overflow time in the header file to more larger:

```
#define HSE_STARTUP_TIMEOUT      ((uint16_t)0xffff) /*!< Time out for HSE start up */
```

The typical phenomenon of this fault is: there is no reaction after power-on, and the nRST pin of the MCU is shorted to the ground once, then the MCU works normally.

3.6.3 HSI Accuracy is not Meet the Spec over Whole Temperature

When the ambient temperature is lower than <0℃ or higher than >60℃， the accuracy of the internal HSI OSC may exceed 3% which affects serial port communication.

Solution：

1) When the ambient temperature range is wide requirement, use an external crystal oscillator
2) Adopt automatic baud rate

3.7 USART

3.7.1 ISR->TC bit is Wrong Setting

ISR<TC>bit, during the data transmission process, if the CR1<TE> clears to 0, TC bit will be incorrectly set, this can result in data loss in some applications that rely on the TC bit to interrupt sending data (If the interval between clearing and setting on TE bit is shorter than the sending time of one byte, the TC is repeatedly set to 1 during the sending time of one byte, as a result, the TC bit enters an error interruption, and data written to the TDR is lost during this period).

Solution：

In an application that uses interrupt routine of TC bit for data sending, query ISR<TXE> bit before writing the sending data.

```
void USARTx_IRQHandler(void)
{
        …
        if (USART_GetITStatus(USARTx, USART_IT_TC) != RESET) //Sending interruption
        {
                USART_ClearITPendingBit(USARTx, USART_IT_TC); // Clear the TC interrupt flag
                if (USARTx->ISR & USART_ISR_TXE)                    // query TXE
                {
                        USARTx->TDR = *buf;
                        …
                }
        }
}
```

3.7.2 When Using HSI, the Baud Rate is not Accuracy enough over Whole Temperature

The HSI accuracy may be affected by the temperature by more than 3%, so the serial port communication may be error.

Solution：

The baud rate adaptive algorithm is adopted.

3.7.3 When Automatic Baud Rate Detection is used, Subsequent Data cannot be received

When automatic baud rate detection is enabled, subsequent data cannot be received after the detection is successful.

Solution：

After successful detection, turn it off (USART->CR2.ABREN clears to 0).

3.8 Hardware Computing Unit (FCM added)

This MCU series add with hardware division and integer root operations, see the relevant documentation.

3.9 FLASH

3.9.1 H0xx Using Flash Turbo (FLASH acceleration function)

In the FCM32H0xx series, the FLASH Turbo module is used to replace the prefetch-buf of the F0xx series. The On and Off of the Flash Turbo module is still controlled by FLASH_ACR.PRFTBE bit, and the cache operation is completely hardware without software intervention.

3.9.2 FLASH Waiting Cycle

The FLASH speed of FCM32x0xx is 32MHz, and when the MCU main frequency exceeds this value, the correct FLASH waiting cycle must be set according to this speed (it can also be set follow 24MHz). Some users' programs set the main frequency to 48MHz, but did not set the waiting cycle (the default is 0), resulting in program execution errors.

Solution：

Use the correct clock configuration functions; Or set the FLASH wait period correctly before the clock configuration of the original program.

3.10 Unique ID (UID)

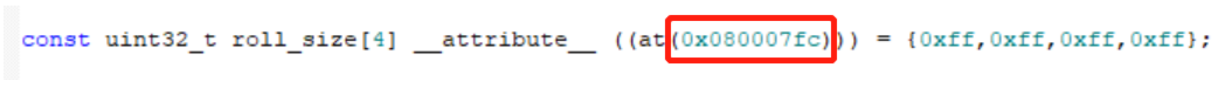
All series of FCM32x0xx contain a unique ID whose address begins at 0x1FFF_F7AC with 3 consecutive words (96 bits).

3.11 Failed to write the roll code

When the function of roll code writing is enabled in some types of writer, roll code writing fails.

Solution:

Set the roll address within the first 32K of FLASH, keep that address in the source program, initialize to full FF, and recompile.
For example, if the roll code is 4 bytes and the start address is 0x080007fc, add the following code to the source program:



```
const uint32_t roll_size[4] __attribute__ ((at(0x080007fc))) = {0xff,0xff,0xff,0xff};
```

3.12 PWR CR reg is Incorrectly Reset

PWR-> CR register is only powered on, power off reset, other reset will not reset the register.
When using the PVD function (PVD is turned on) and relying on it to generate an interrupt, if PVD has detected that the supply voltage is lower than the set value (PWR->CSR.PVDO=1), reset MCU and re-enable PVD, it will not generate a PVD rising edge interrupt (because PVDE was turned on before the MCU reset and always be 1 after that).

Solution:

Before enabling PVD, disable it (PWR-> CR.PVDE write 0 first)

3.13 I2C

When I2C1->SCL pin is mapped to PF1, the pin is unresponsive and I2C1 does not work properly.

Solution:

Do not use the AF1 feature of PF1.

4. Version History

Date	Revision	Author	Changes
2021/7/29	0.10	Dick Hou	First edition
2021/7/30	0.11		Added 3.8 UID
2021/8/9	0.12		Added instruction for ADC bit setting
2021/8/11	0.13		Added SPI related description; Section adjustment
2021/8/25	0.14		Added USART section
2021/11/4	0.15		Added 3.11 chapter of roll code burning
2022/1/24	0.16		Added description of 3.6.3 HSI temperature drift
2022/2/23	0.17		Added section 3.7.2
2022/5/23	0.18		Simplification3.7.2 Added section 3.2.4 Added section 3.9.2
2022/9/5	0.19		Added Description for 3.6.2 Added section 3.7.3 Added section 3.12
2023/3/2	0.20		Added section 3.4.2
2023/3/27	0.21		Added section 3.13

5. Others