SIOC嵌入式軟體實驗
實驗六：類比—數位轉換(ADC)

WU-YANG
Technology Co., Ltd.

support.wuyang@gmail.com

前言

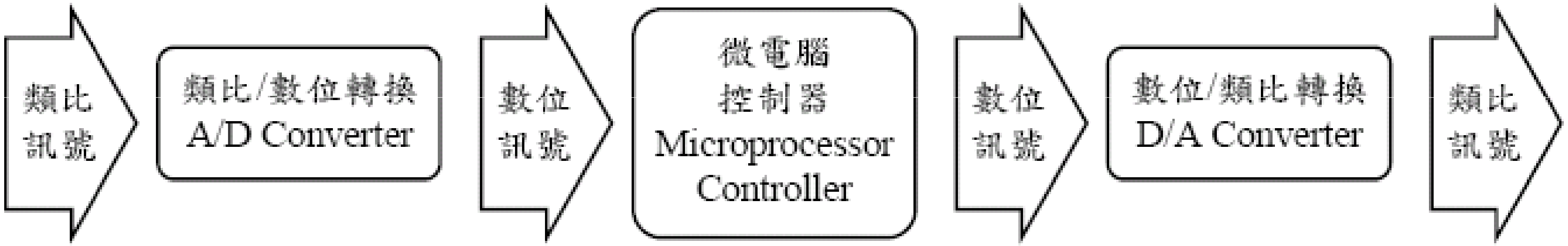
物理世界的觀測皆為連續的類比訊號，若要將其輸入電腦進行數值運算，則必須轉換為數位訊號。本章將介紹如何使用STM32內部的ADC(Analog to Digital)進行訊號轉換，並使讀者瞭解：

- ■ ADC的使用
- ■ ADC與DMA同步運作方式

實驗：

將溫度經由ADC轉換後的結果用VCP輸出到螢幕上顯示

Introduction



類比
訊號

類比/數位轉換
A/D Converter

數位
訊號

微電腦
控制器
Microprocessor
Controller

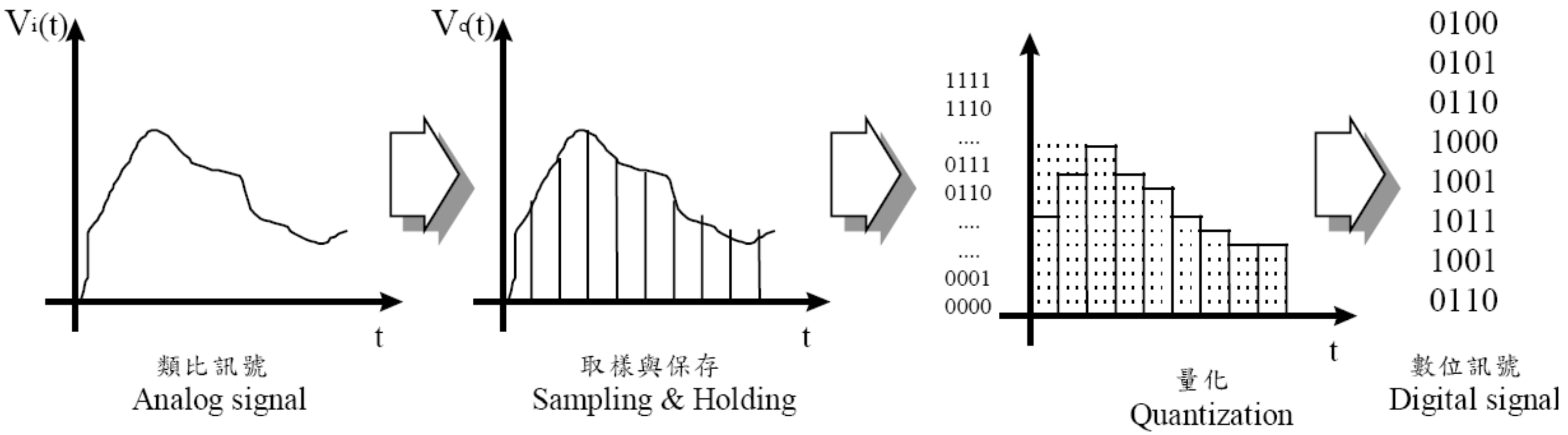
數位
訊號

數位/類比轉換
D/A Converter

類比
訊號

ADC原理

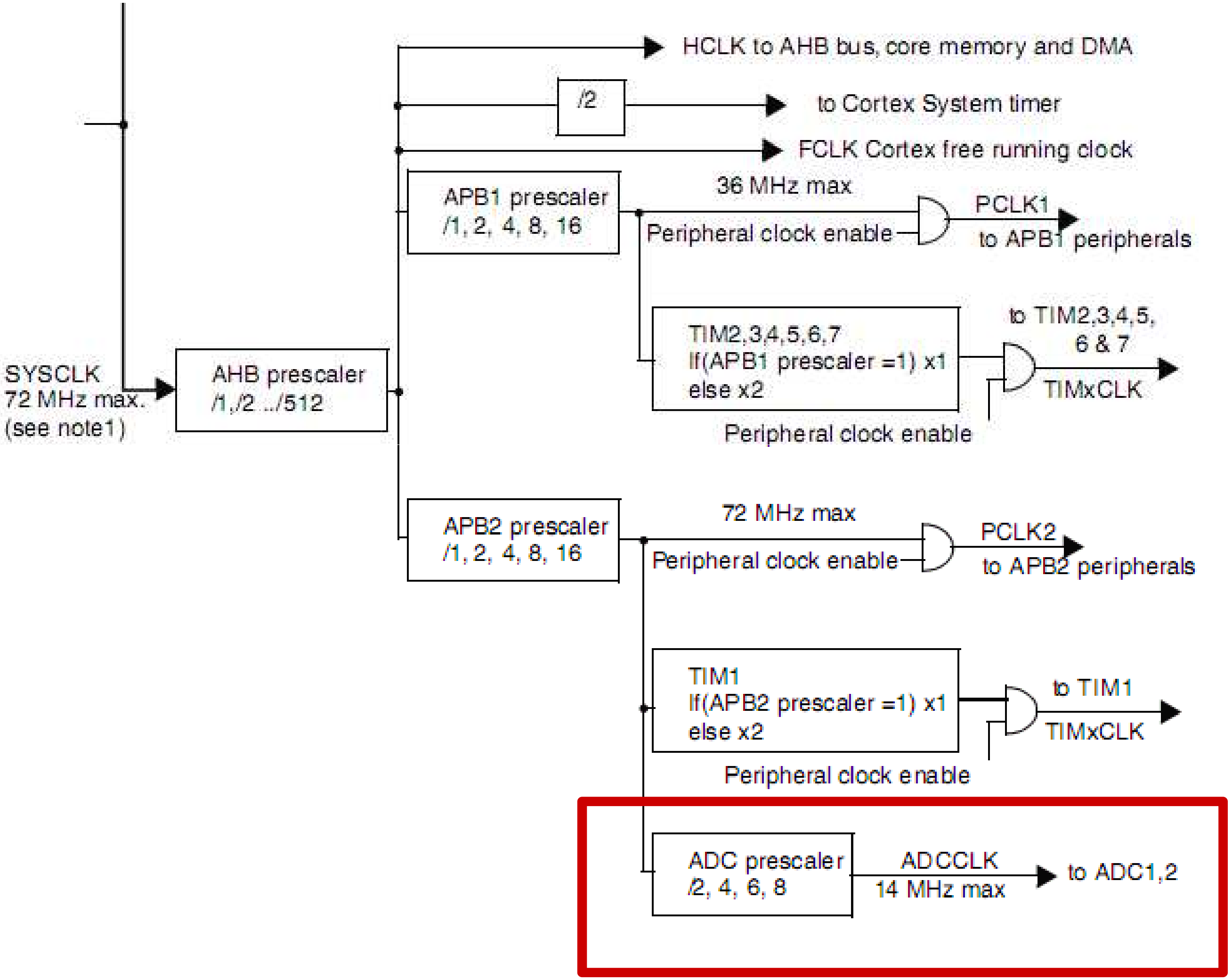
- □ ADC轉換
  - ■ 資料保存(sampling and Holding)
    - □ 取樣率(Sample rate)越高則訊號越不易失真
  - ■ 量化(Quantization)
    - □ 量化的位元數越高則解析度越高



STM32 ADC特色

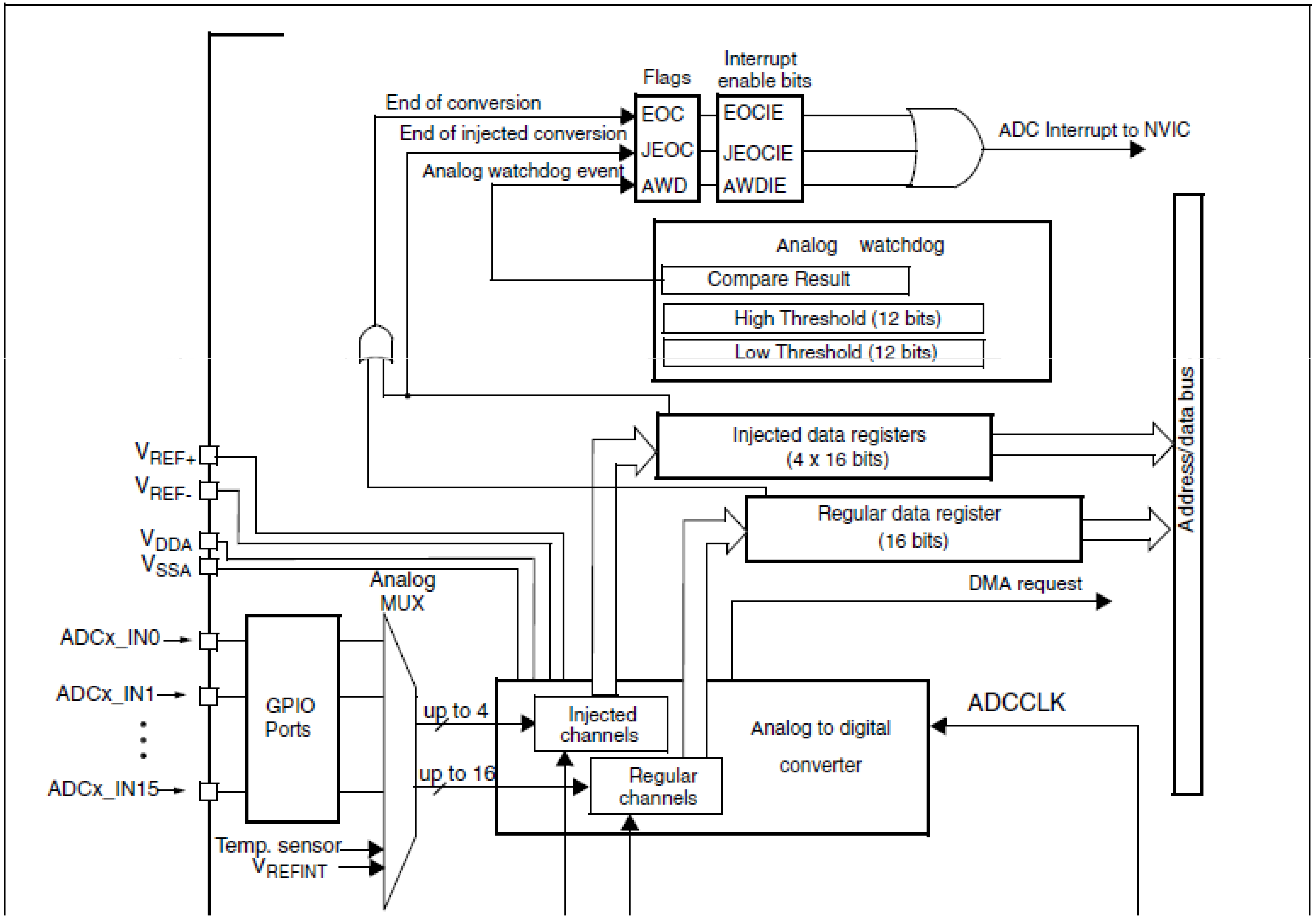
- 12-bit resolution
- Interrupt generation at End of Conversion, End of Injected conversion and Analog watchdog event
- Single and continuous conversion modes
- Scan mode for automatic conversion of channel 0 to channel ‘n’
- Self-calibration
- Data alignment with in-built data coherency
- Channel by channel programmable sampling time
- External trigger option for both regular and injected conversion
- Discontinuous mode
- Dual mode (on devices with 2 ADCs or more)
- STM32F103xx performance line devices: 1 µs at 56 MHz (1.17 µs at 72 MHz)
- ADC supply requirement: 2.4 V to 3.6 V
- ADC input range: VREF- <= VIN <= VREF+
- DMA request generation during regular channel conversion

ADC Clock

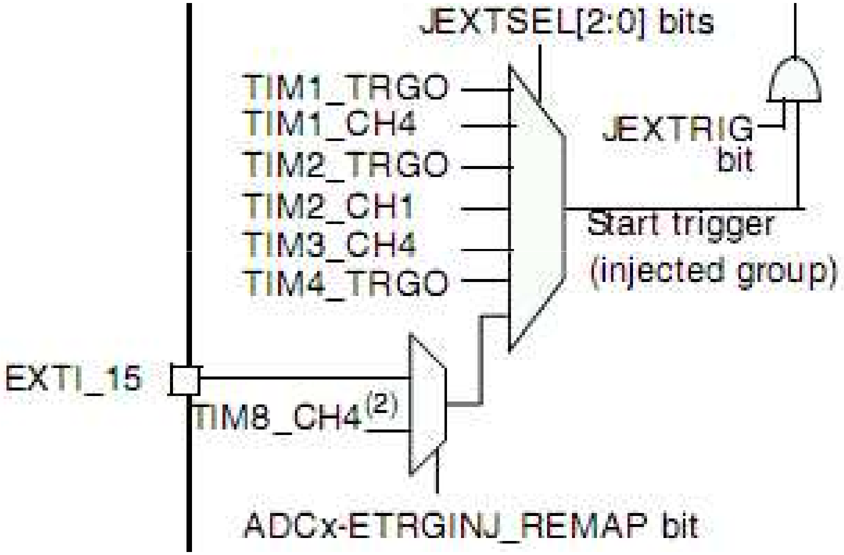

HCLK to AHB bus, core memory and DMA
/2
to Cortex System timer
FCLK Cortex free running clock
APB1 prescaler
/1, 2, 4, 8, 16
36 MHz max
Peripheral clock enable
PCLK1
to APB1 peripherals
TIM2,3,4,5,6,7
If(APB1 prescaler =1) x1
else x2
to TIM2,3,4,5,
6 & 7
TIMxCLK
Peripheral clock enable
SYSCLK
72 MHz max.
(see note1)
AHB prescaler
/1,/2 ../512
APB2 prescaler
/1, 2, 4, 8, 16
72 MHz max
Peripheral clock enable
PCLK2
to APB2 peripherals
TIM1
If(APB2 prescaler =1) x1
else x2
to TIM1
TIMxCLK
Peripheral clock enable
ADC prescaler
/2, 4, 6, 8
ADCCLK
14 MHz max
to ADC1,2

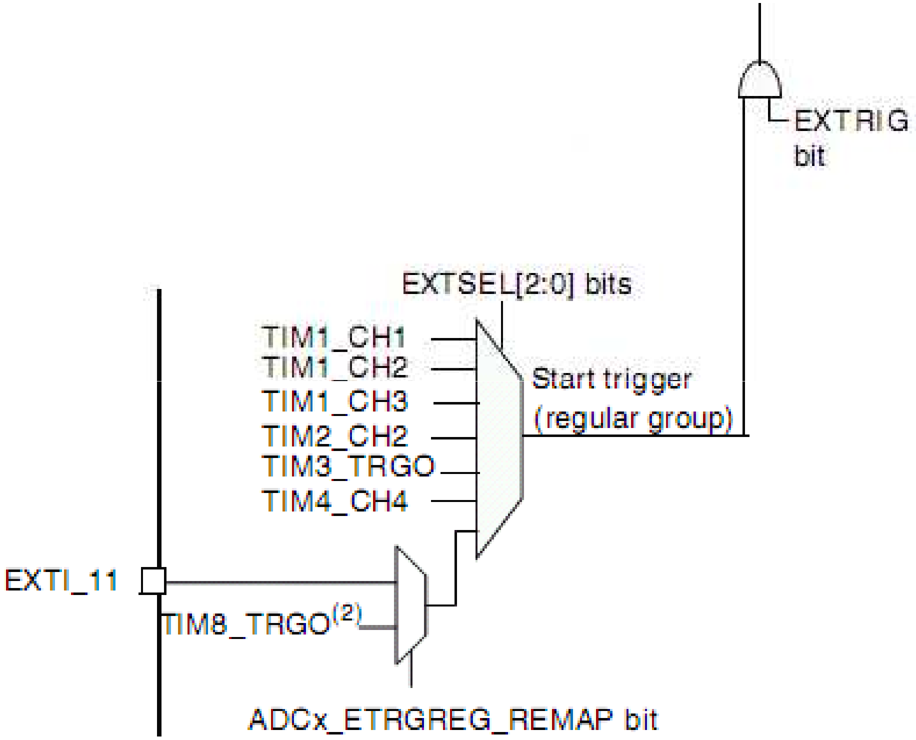
ADC block diagram

Figure 22. Single ADC block diagram



ADC block diagram(cont.)





ADC mapping pin

- STM32F10x8 p27.

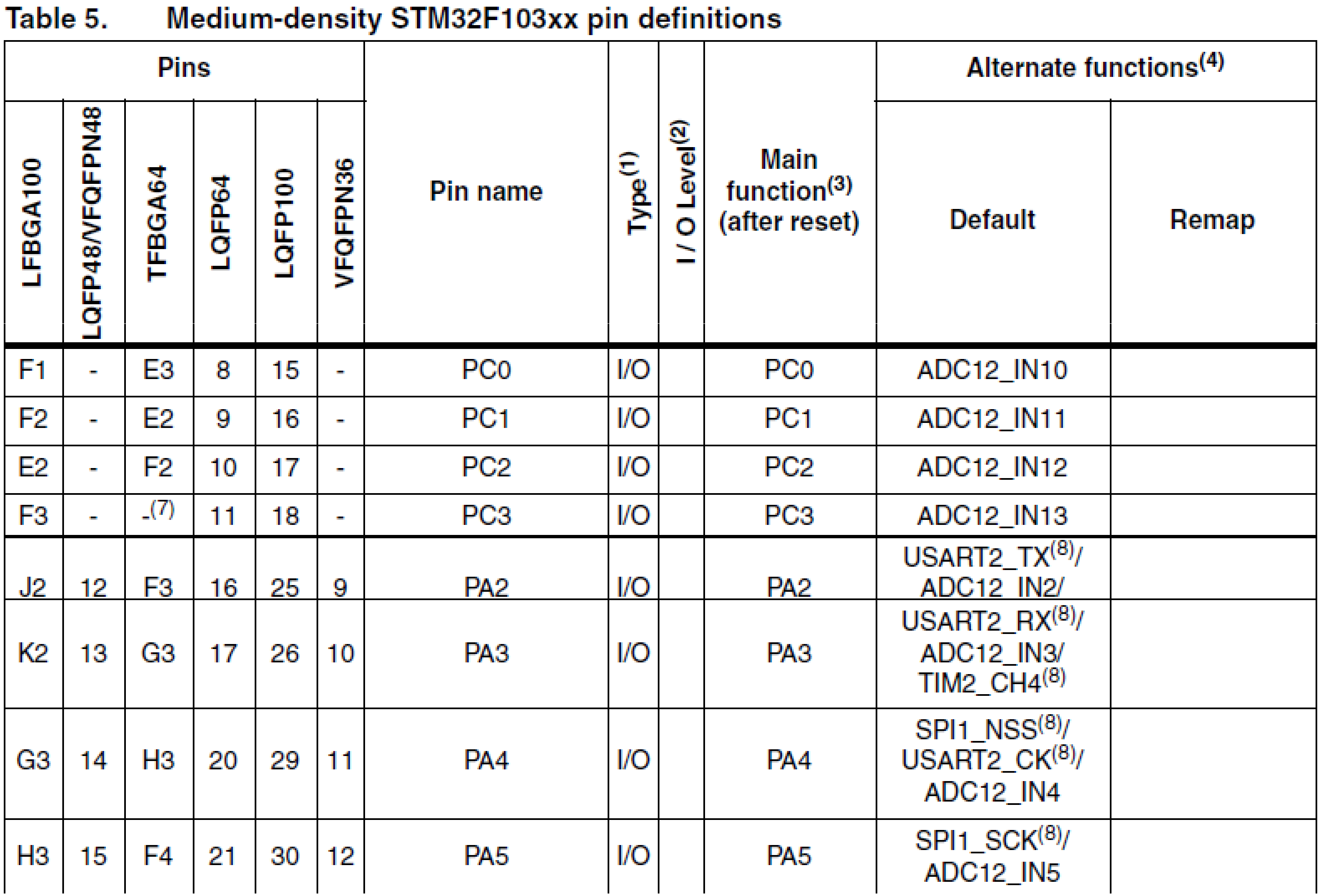
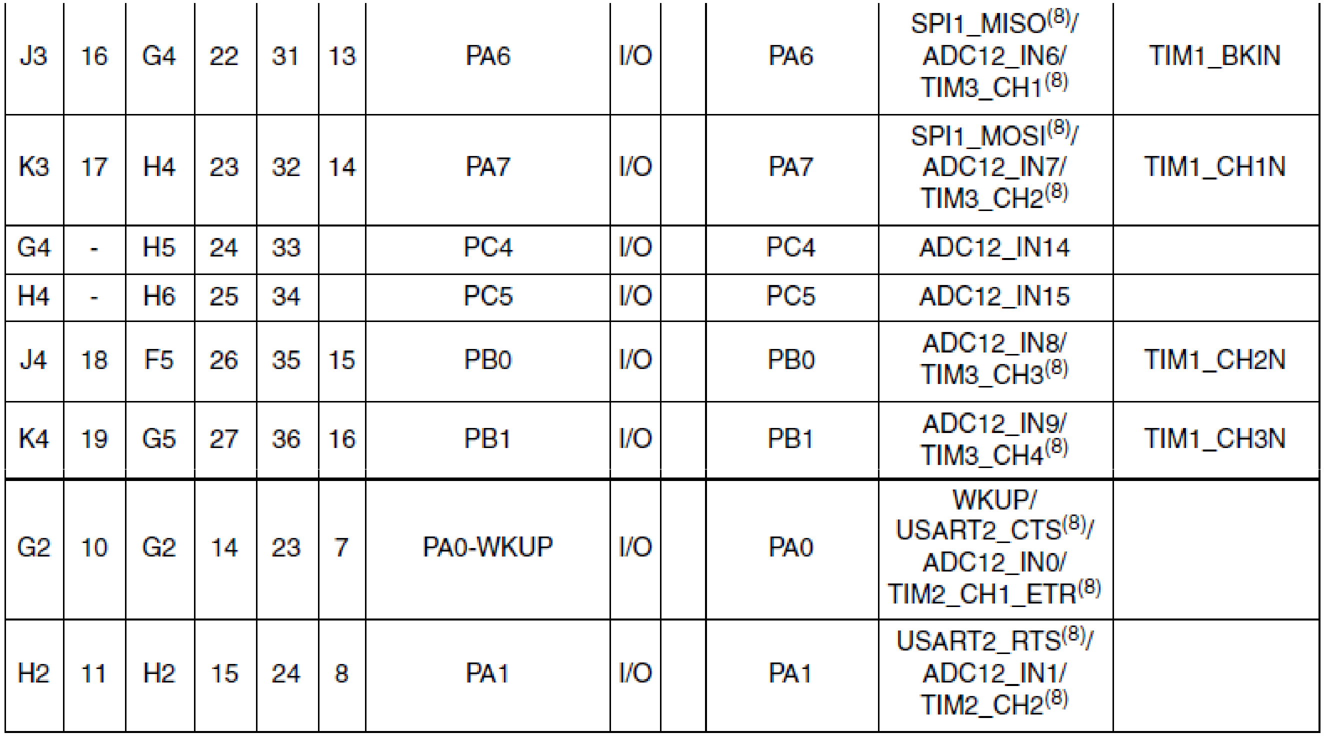


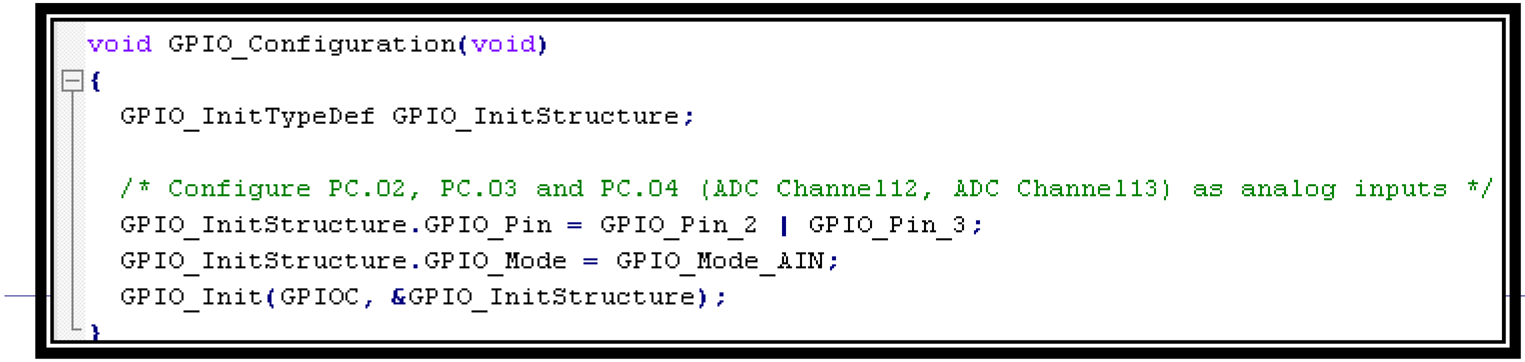
Table 5. Medium-density STM32F103xx pin definitions

Pins						Pin name	Type[1]	I / O Level[2]	Main function[3] (after reset)	Alternate functions[4]	
LFBGA100	LQFP48/VFQFPN48	TFBGA64	LQFP64	LQFP100	VFQFPN36					Default	Remap
F1	-	E3	8	15	-	PC0	I/O		PC0	ADC12_IN10	
F2	-	E2	9	16	-	PC1	I/O		PC1	ADC12_IN11	
E2	-	F2	10	17	-	PC2	I/O		PC2	ADC12_IN12	
F3	-	-[7]	11	18	-	PC3	I/O		PC3	ADC12_IN13	
J2	12	F3	16	25	9	PA2	I/O		PA2	USART2_TX[8]/ ADC12_IN2/	
K2	13	G3	17	26	10	PA3	I/O		PA3	USART2_RX[8]/ ADC12_IN3/ TIM2_CH4[8]	
G3	14	H3	20	29	11	PA4	I/O		PA4	SPI1_NSS[8]/ USART2_CK[8]/ ADC12_IN4	
H3	15	F4	21	30	12	PA5	I/O		PA5	SPI1_SCK[8]/ ADC12_IN5	

ADC mapping pin (cont.)



J3	16	G4	22	31	13	PA6	I/O		PA6	SPI1_MISO(8)/ ADC12_IN6/ TIM3_CH1(8)	TIM1_BKIN
K3	17	H4	23	32	14	PA7	I/O		PA7	SPI1_MOSI(8)/ ADC12_IN7/ TIM3_CH2(8)	TIM1_CH1N
G4	-	H5	24	33		PC4	I/O		PC4	ADC12_IN14	
H4	-	H6	25	34		PC5	I/O		PC5	ADC12_IN15	
J4	18	F5	26	35	15	PB0	I/O		PB0	ADC12_IN8/ TIM3_CH3(8)	TIM1_CH2N
K4	19	G5	27	36	16	PB1	I/O		PB1	ADC12_IN9/ TIM3_CH4(8)	TIM1_CH3N
G2	10	G2	14	23	7	PA0-WKUP	I/O		PA0	WKUP/ USART2_CTS(8)/ ADC12_IN0/ TIM2_CH1_ETR(8)	
H2	11	H2	15	24	8	PA1	I/O		PA1	USART2_RTS(8)/ ADC12_IN1/ TIM2_CH2(8)	



```
void GPIO_Configuration(void)
{
  GPIO_InitTypeDef GPIO_InitStructure;

  /* Configure PC.02, PC.03 and PC.04 (ADC Channel12, ADC Channel13) as analog inputs */
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_2 | GPIO_Pin_3;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AIN;
  GPIO_Init(GPIOC, &GPIO_InitStructure);
}
```

ADC Channel

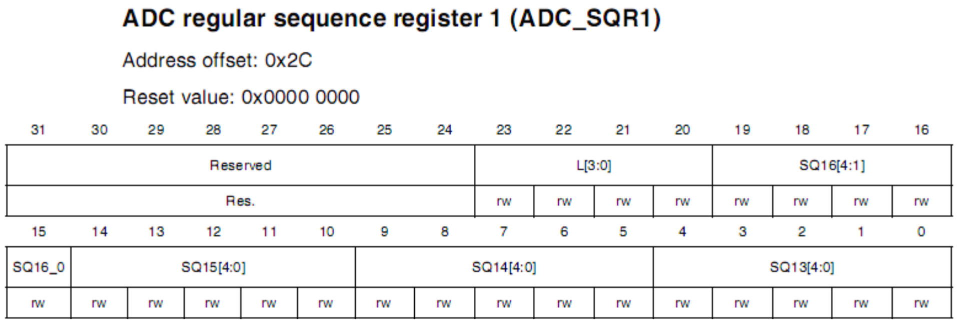
- STM32 將ADC分成兩個通道:
  - Regular channels(規則通道):

    相當於運行的程序，最多包含16個通道。
  - Injected channels(注入通道):

    相當於中斷的程序，最多包含4個通道。
- 當程序正在執行的時候，中斷是用來打斷你的執行順序，因此注入通道的轉換可以打斷規則通道的轉換，在注入通道被轉換完成後規則通道才可以繼續轉換。

ADC register

- □ regular group

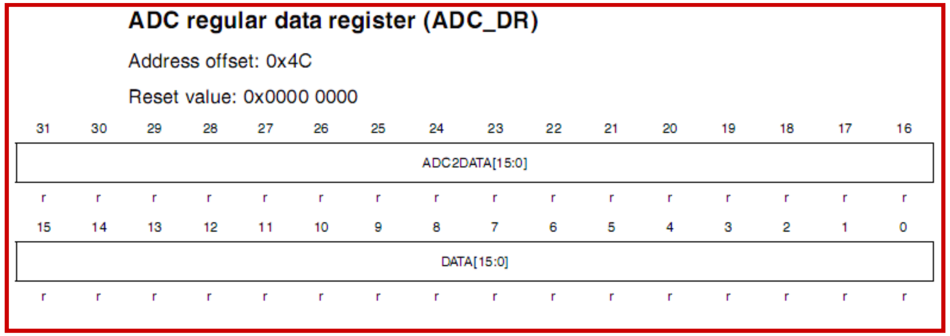


ADC regular sequence register 1 (ADC_SQR1)

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								L[3:0]				SQ16[4:1]			
Res.								rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SQ16_0	SQ15[4:0]					SQ14[4:0]					SQ13[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

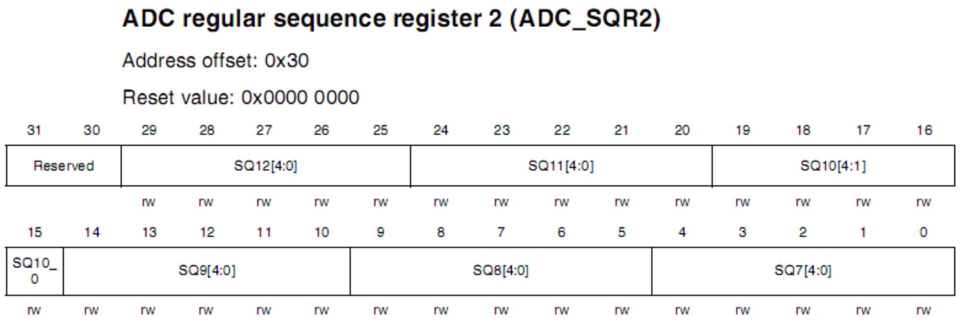


ADC regular data register (ADC_DR)

Address offset: 0x4C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADC2DATA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DATA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

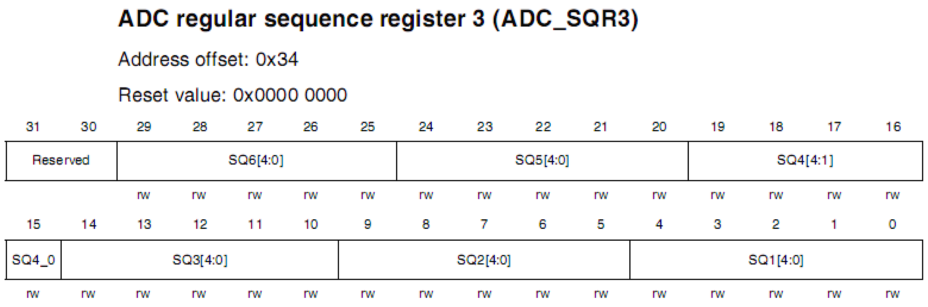


ADC regular sequence register 2 (ADC_SQR2)

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		SQ12[4:0]					SQ11[4:0]					SQ10[4:1]			
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SQ10_0	SQ9[4:0]					SQ8[4:0]					SQ7[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw



ADC regular sequence register 3 (ADC_SQR3)

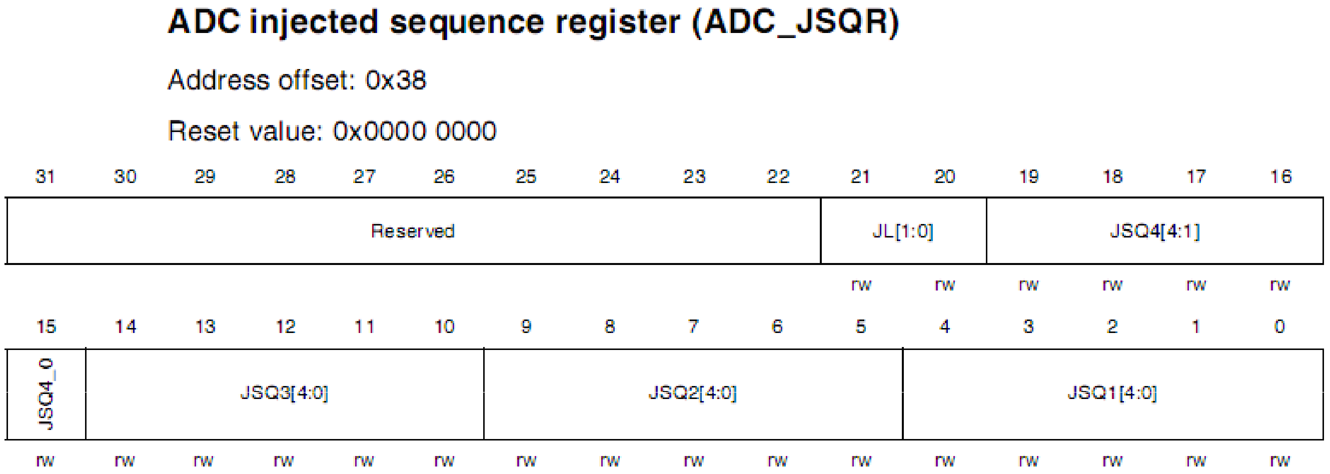
Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		SQ6[4:0]					SQ5[4:0]					SQ4[4:1]			
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SQ4_0	SQ3[4:0]					SQ2[4:0]					SQ1[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

ADC register

- injected group



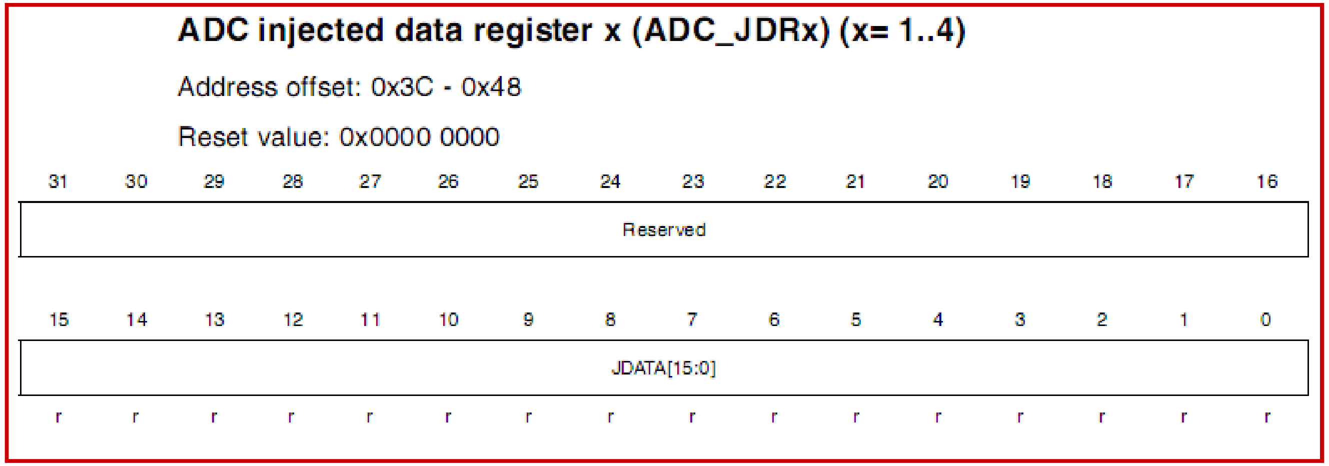
ADC injected sequence register (ADC_JSQR)

Address offset: 0x38

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved										JL[1:0]		JSQ4[4:1]			
										rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JSQ4_0	JSQ3[4:0]					JSQ2[4:0]					JSQ1[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw



ADC injected data register x (ADC_JDRx) (x= 1..4)

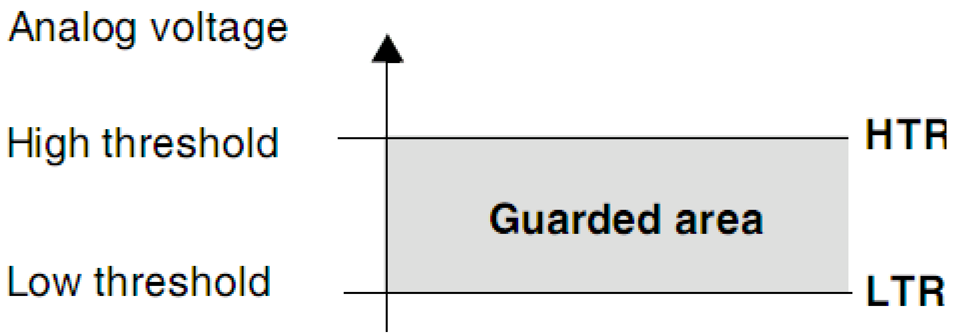
Address offset: 0x3C - 0x48

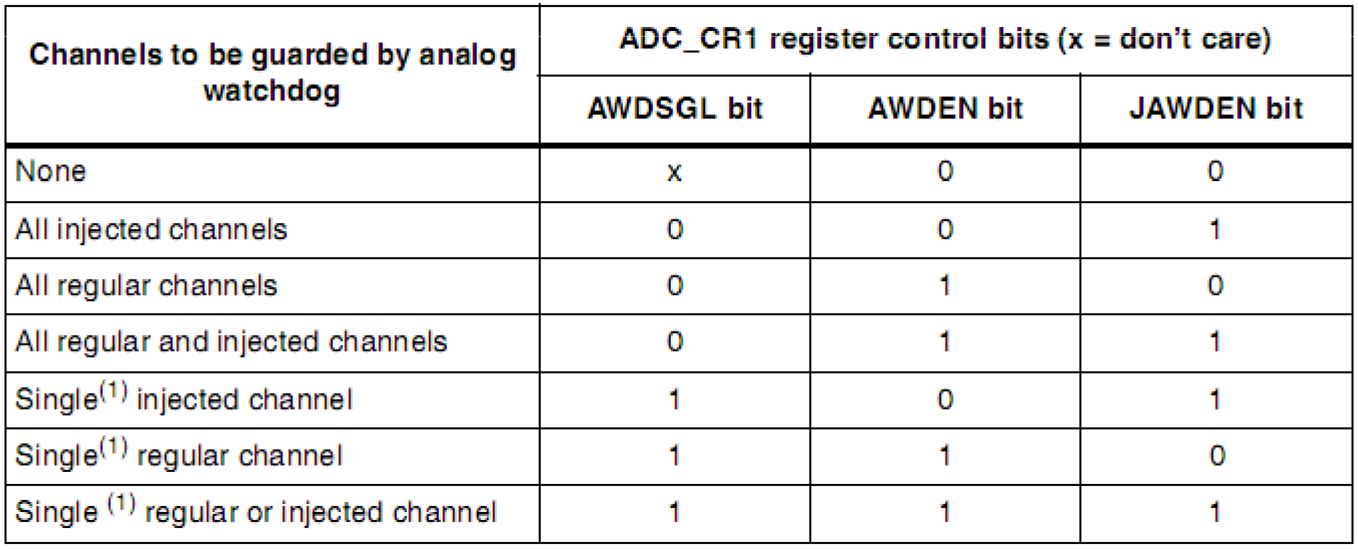
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JDATA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Analog watchdog



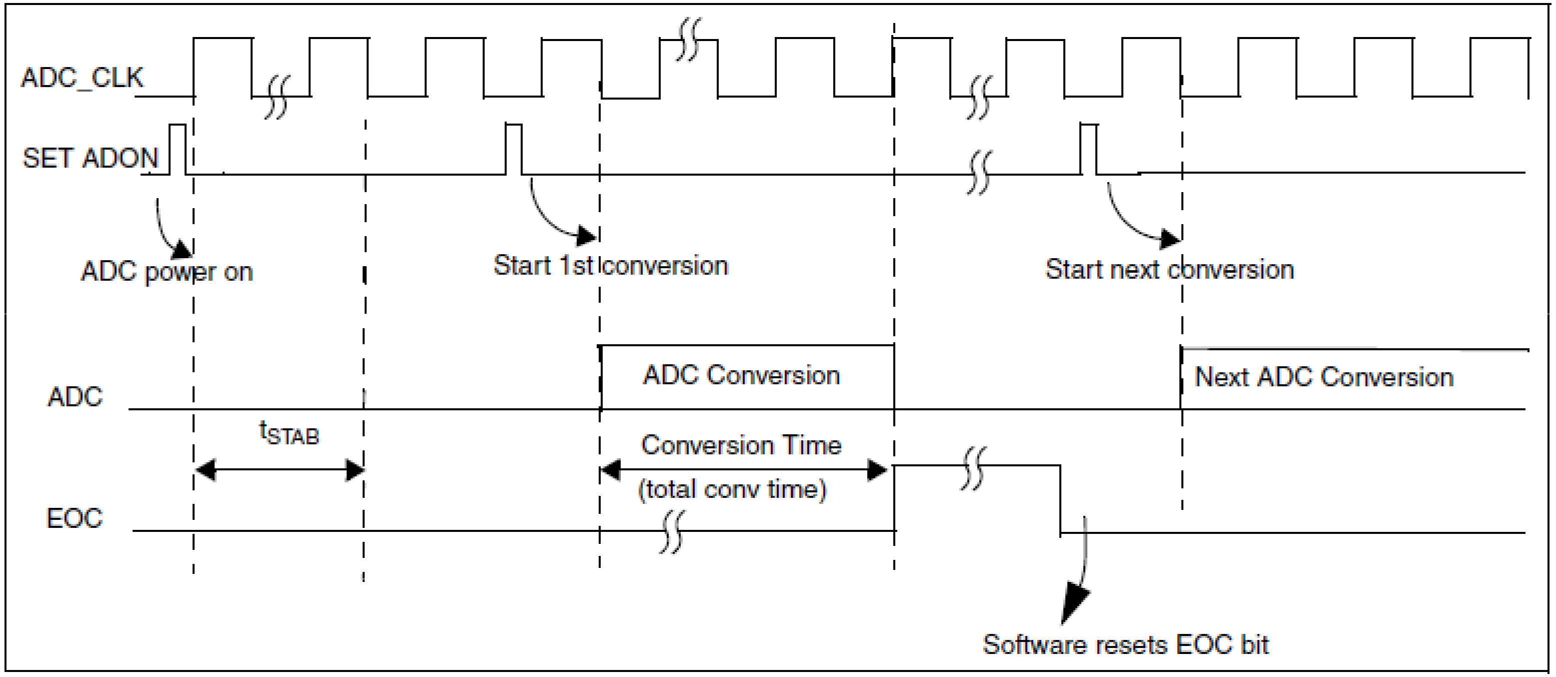


Channels to be guarded by analog watchdog	ADC_CR1 register control bits (x = don't care)		
	AWDSGL bit	AWDEN bit	JAWDEN bit
None	x	0	0
All injected channels	0	0	1
All regular channels	0	1	0
All regular and injected channels	0	1	1
Single[1] injected channel	1	0	1
Single[1] regular channel	1	1	0
Single [1] regular or injected channel	1	1	1

[1] Selected by AWDCH[4:0] bits

Time diagram

Figure 23. Timing diagram



Conversion mode

- □ Single conversion mode
- □ Continuous conversion mode
- □ Scan mode
- □ Discontinuous mode
- □ Dual ADC mode

Discontinuous mode

- Regular group

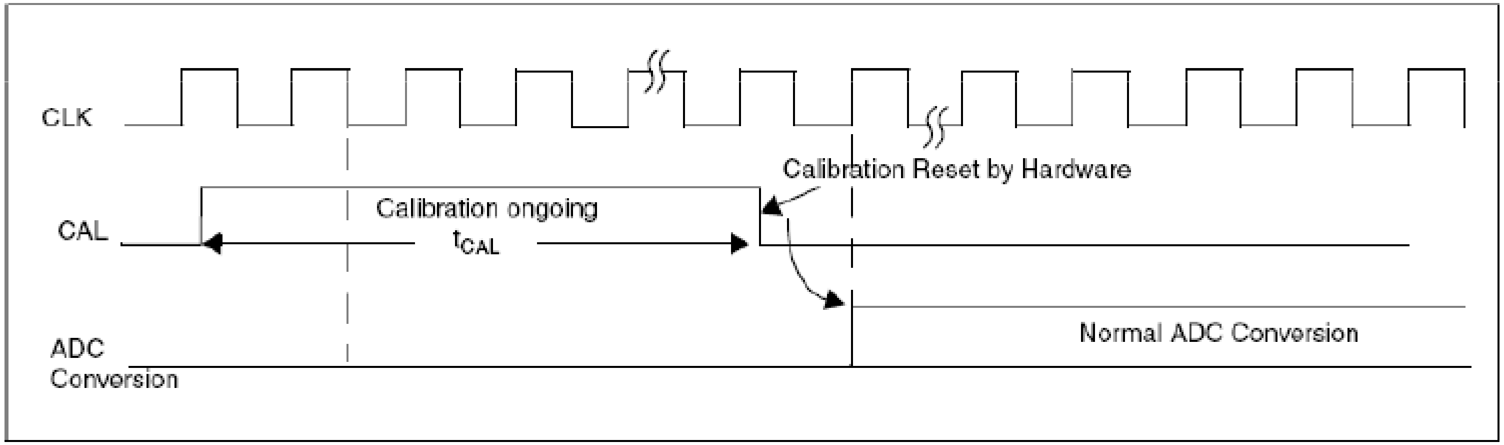
Example:
n = 3, channels to be converted = 0, 1, 2, 3, 6, 7, 9, 10
1st trigger: sequence converted 0, 1, 2
2nd trigger: sequence converted 3, 6, 7
3rd trigger: sequence converted 9, 10 and an EOC event generated
4th trigger: sequence converted 0, 1, 2

- Injected group

Example:
n = 1, channels to be converted = 1, 2, 3
1st trigger: channel 1 converted
2nd trigger: channel 2 converted
3rd trigger: channel 3 converted and EOC and JEOC events generated
4th trigger: channel 1

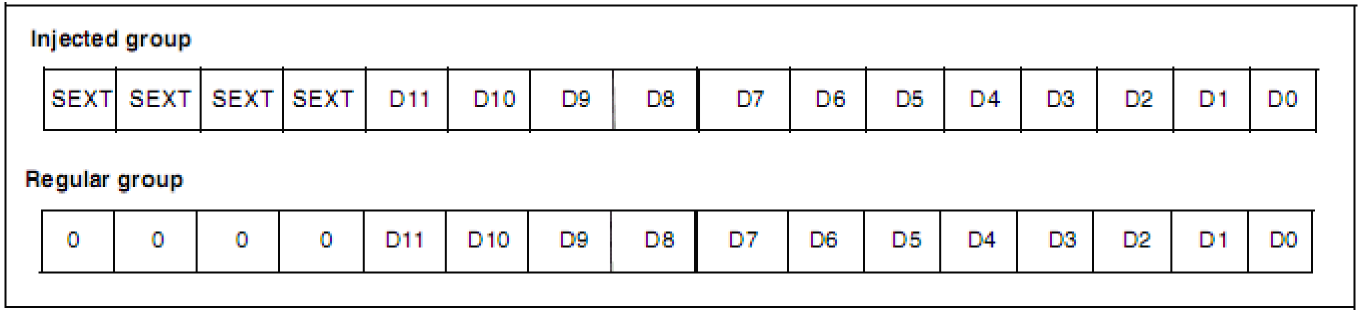
Calibration

- □ ADC內部的校正模式，校正可大幅減少內部精準度的誤差。
- □ 透過ADC_CP2暫存器中的CAL位來啟動校正，校正結束後，CAL位會被硬體復位。



Data alignment

Right alignment of data



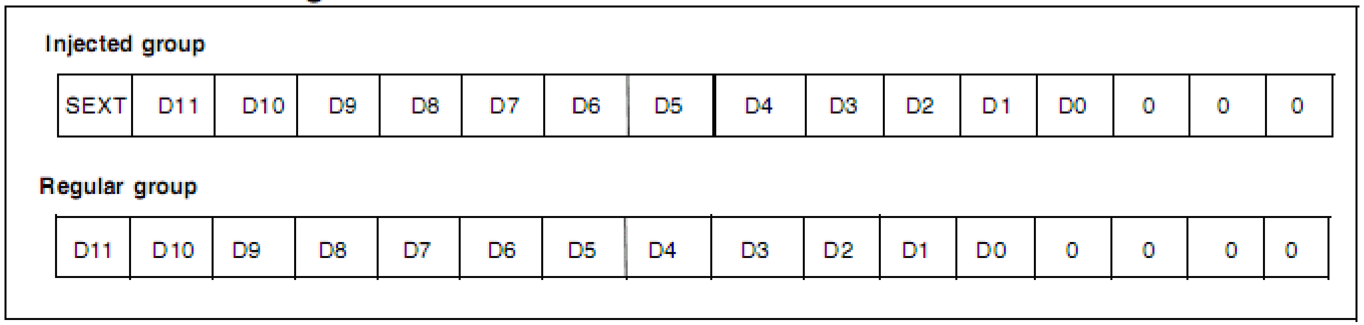
Injected group

SEXT	SEXT	SEXT	SEXT	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Regular group

0	0	0	0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Left alignment of data



Injected group

SEXT	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0

Regular group

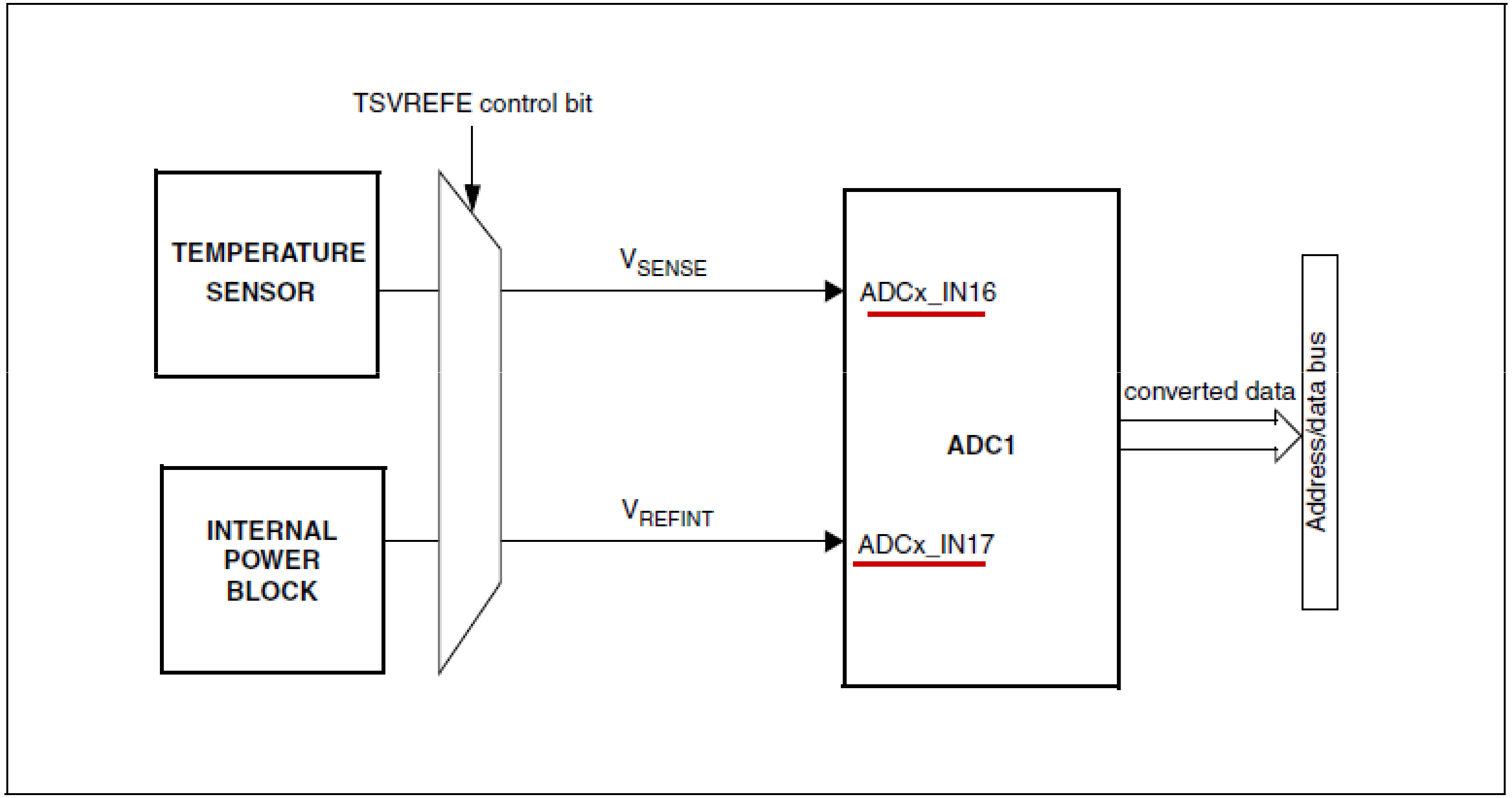
D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0	0

DMA request

- □ 規則通道轉換後的數值儲存在一個唯一的暫存器中，所以當轉換多個規則通道時需要使用DMA，用來避免遺失已經儲存在ADC_DR暫存器的數據。
- □ 只有在規則通道轉換結束後才能產生DMA的請求，並且將轉換後的數據從ADC_DR的暫存器傳輸到用戶指定的目的地位址。

Temperature sensor

Figure 39. Temperature sensor and V_{REFINT} channel block diagram



注:溫度感測器在內部和ADCx_IN16輸入通道相連接，
此通道把感測器輸出的電壓轉換成數字值

Temperature sensor (Con.)

- 讀取溫度的方式:
  1. 選擇ADCx_IN16 輸入通道
  2. 選擇取樣時間大於 2.2 μs
  3. 設置ADC_CR2的TSVREFE位，以喚醒關電模式下的溫度感測器 (使用function ADC_TempSensorVrefintCmd(ENABLE);)
  4. 通過設置ADON位啟動ADC轉換 (使用function ADC_SoftwareStartConvCmd(ADC1, ENABLE);)
  5. 讀ADC暫存器上的VSENSE數據結果
  6. 利用下列公式得出溫度

Temperature (in °C) = $\{(V_{25} - V_{SENSE}) / \text{Avg\_Slope}\} + 25.$

$V_{25} = V_{SENSE}$ 在25°C時的數值

Avg_Slope ＝ 溫度與 V_{SENSE} 曲線的平均斜率(單位爲mV/° C 或 μV/ °C)

Temperature sensor (Con.)

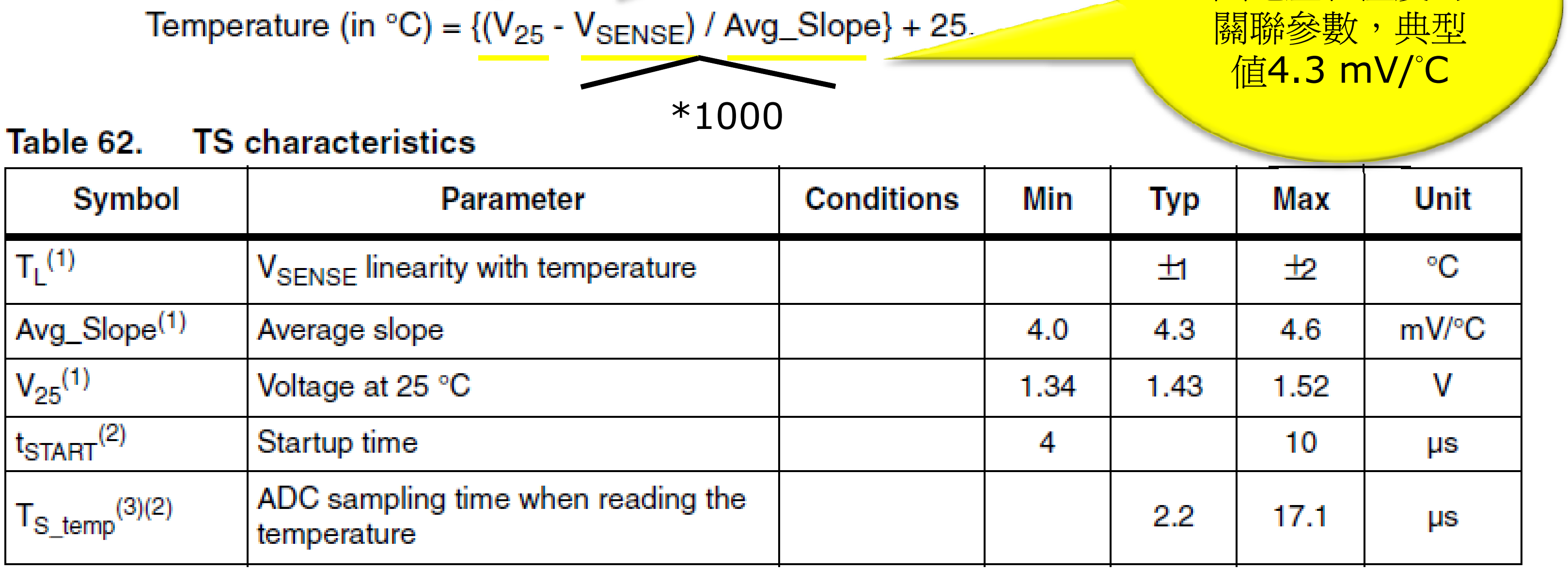
- 溫度公式:

V_{SENSE}:溫度感測器的當前輸出電壓，與 ADC_ConvertedValue 數值轉換成電壓的關係爲:

$$V_{SENSE} = \frac{ADC\_ConvertedValue * Vdd}{Vdd\_convert\_value(0xFFF)}$$

V_{25}:溫度感測器在25度C時的輸出電壓，典型值1.43V

Avg_Slope :溫度感測器的輸出電壓和溫度的關聯參數，典型值4.3 mV/°C



Temperature (in °C) = $\{(V_{25} - \underbrace{V_{SENSE}) / Avg\_Slope}_{*1000}\} + 25$.

Table 62. TS characteristics

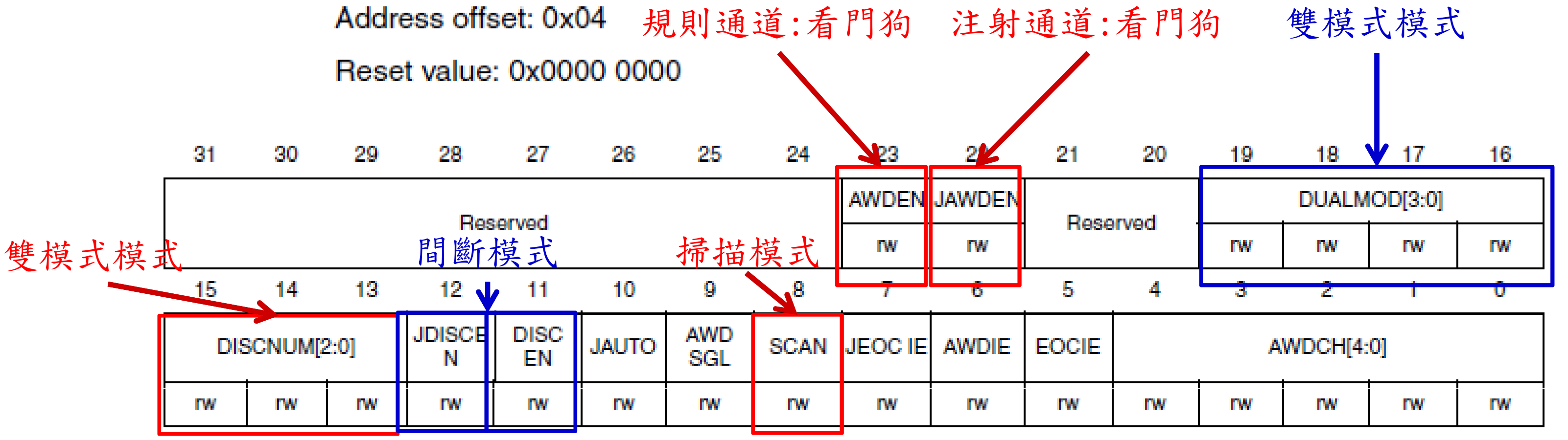
Symbol	Parameter	Conditions	Min	Typ	Max	Unit
T_L(1)	V_{SENSE} linearity with temperature			±1	±2	°C
Avg_Slope(1)	Average slope		4.0	4.3	4.6	mV/°C
V_{25}(1)	Voltage at 25 °C		1.34	1.43	1.52	V
t_{START}(2)	Startup time		4		10	µs
T_{S_temp}(3)(2)	ADC sampling time when reading the temperature			2.2	17.1	µs

ADC control register

11.12.2 ADC control register 1 (ADC_CR1)

Address offset: 0x04

Reset value: 0x0000 0000



規則通道:看門狗

注射通道:看門狗

雙模式模式

雙模式模式

間斷模式

掃描模式

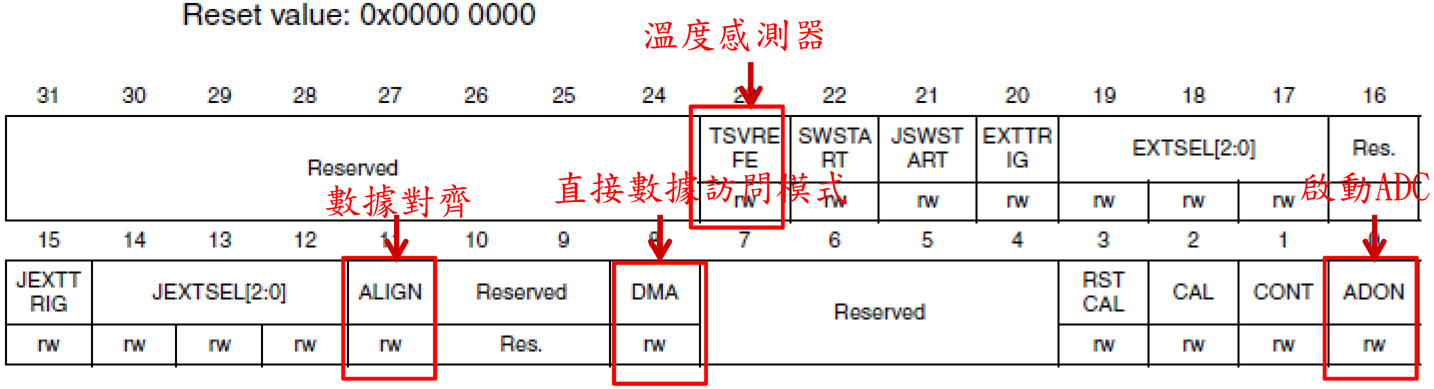
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								AWDEN	JAWDEN	Reserved		DUALMOD[3:0]			
								rw	rw			rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DISCNUM[2:0]			JDISCEN	DISCEN	JAUTO	AWDSGL	SCAN	JEOC IE	AWDIE	EOCIE	AWDCH[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

11.12.3 ADC control register 2 (ADC_CR2)

Address offset: 0x08

Reset value: 0x0000 0000



溫度感測器

數據對齊

直接數據訪問模式

啟動ADC

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								TSVREFE	SWSTART	JSWSTART	EXTTRIG	EXTSEL[2:0]			Res.
								rw	rw	rw	rw	rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JEXTTRIG	JEXTSEL[2:0]			ALIGN	Reserved		DMA	Reserved				RSTCAL	CAL	CONT	ADON
rw	rw	rw	rw	rw	Res.		rw					rw	rw	rw	rw

ADC+DMA實驗

實驗目的

自然界的真實訊號是呈現連續的類比訊號，若要將其輸入電腦中進行數值運算，則必須轉換為數位訊號。本章將將介紹如何使用STM32103ZC內部的Analog to Digital converter (ADC)進行訊號轉換，並使讀者瞭解：

ADC使用方式

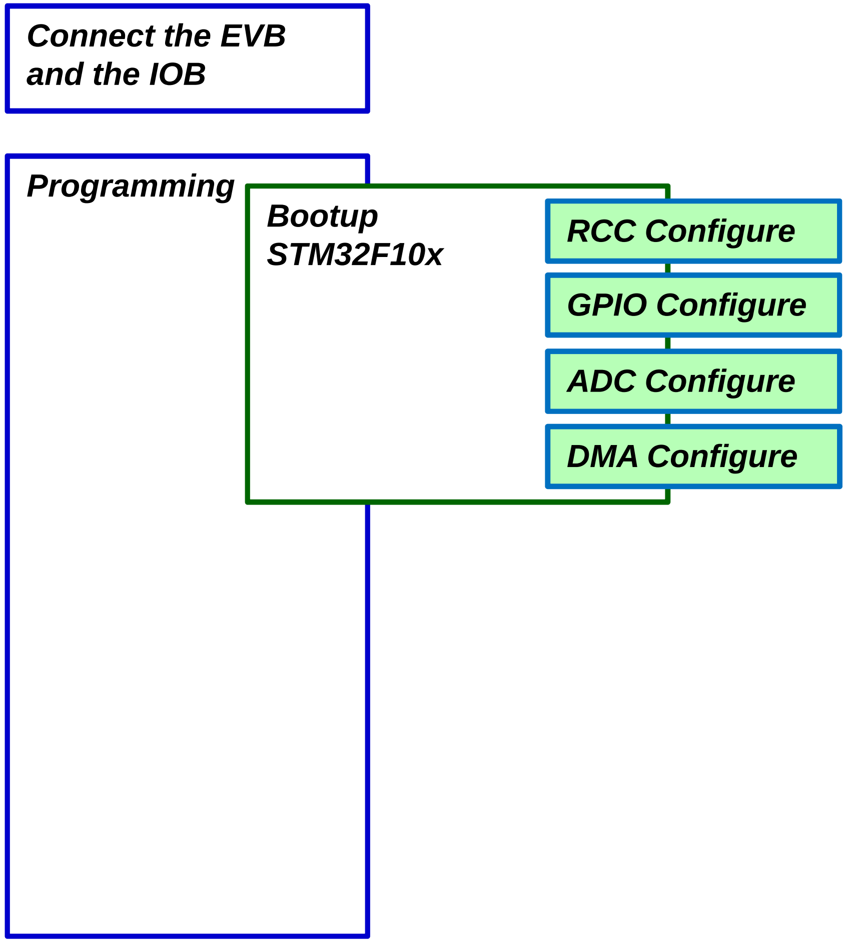
□ ADC與DMA同步運作方式

實做重點

□ 按下DFU Button後，將溫度經由ADC轉換後的結果用VCP輸出到螢幕上顯示。

Development Flow

Embedded Software Side


Connect the EVB and the IOB

Programming

Bootup STM32F10x

RCC Configure

GPIO Configure

ADC Configure

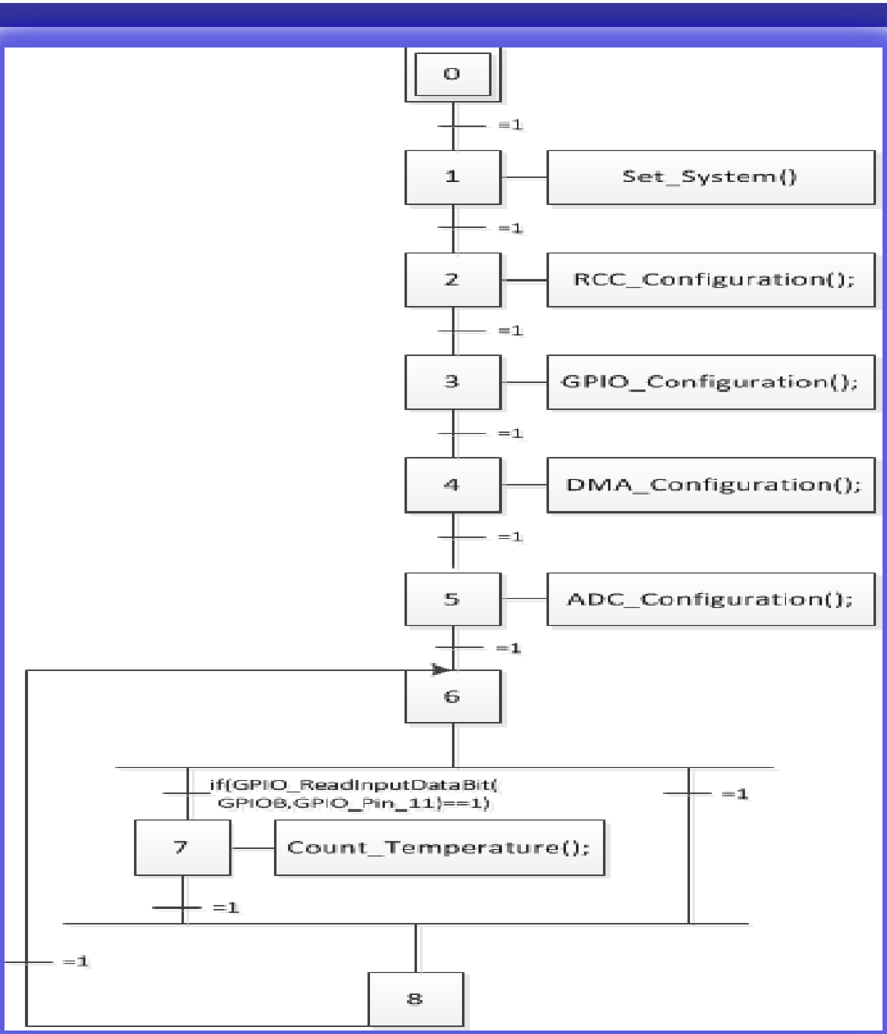
DMA Configure

```
int main(void)
{
/* System USB configuration  */
  Set_System();
  /*pause*/
  getchar();
  /* System clocks configuration  */
  RCC_Configuration();
 /* GPIO configuration */
  GPIO_Configuration();
  /* DMA1 channel1 configuration */
  DMA_Configuration();
  /* ADC1 configuration*/
  ADC_Configuration();

  printf("Configuration finish\n\r");
  printf("Please presee DFU Button to get 
Temperature\n\r");

  while(1)
  {
  /*presee DFU Button to get Temperature*/
    if(GPIO_ReadInputDataBit(GPIOB,GPIO_Pin_11)==1)
  {
           Count_Temperature();
  }
}
```

ADC Grafcet



0

=1

1

Set_System()

=1

2

RCC_Configuration();

=1

3

GPIO_Configuration();

=1

4

DMA_Configuration();

=1

5

ADC_Configuration();

=1

6

if(GPIO_ReadInputDataBit(GPIOB,GPIO_Pin_11)==1)

=1

7

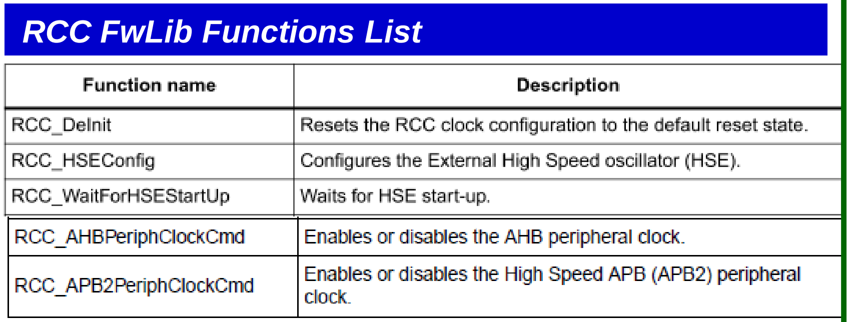
Count_Temperature();

=1

=1

8

Configure RCC



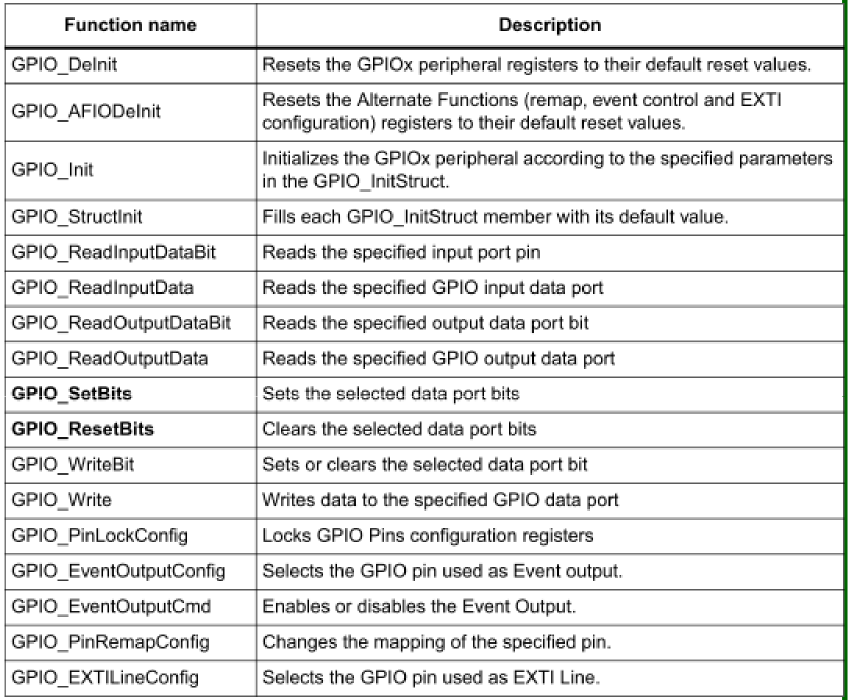
RCC FwLib Functions List

Function name	Description
RCC_DeInit	Resets the RCC clock configuration to the default reset state.
RCC_HSEConfig	Configures the External High Speed oscillator (HSE).
RCC_WaitForHSEStartUp	Waits for HSE start-up.
RCC_AHBPeriphClockCmd	Enables or disables the AHB peripheral clock.
RCC_APB2PeriphClockCmd	Enables or disables the High Speed APB (APB2) peripheral clock.

```
void RCC_Configuration(void)
{
/* Enable peripheral clocks ------------------------------*/
  /* Enable DMA1 clock */
  RCC_AHBPeriphClockCmd(RCC_AHBPeriph_DMA1, ENABLE);
  /* Enable ADC1 clock */
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1, ENABLE);
 /* Enable GPIOB clock */
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);
}
```

Configure GPIO

GPIO FwLib Functions List

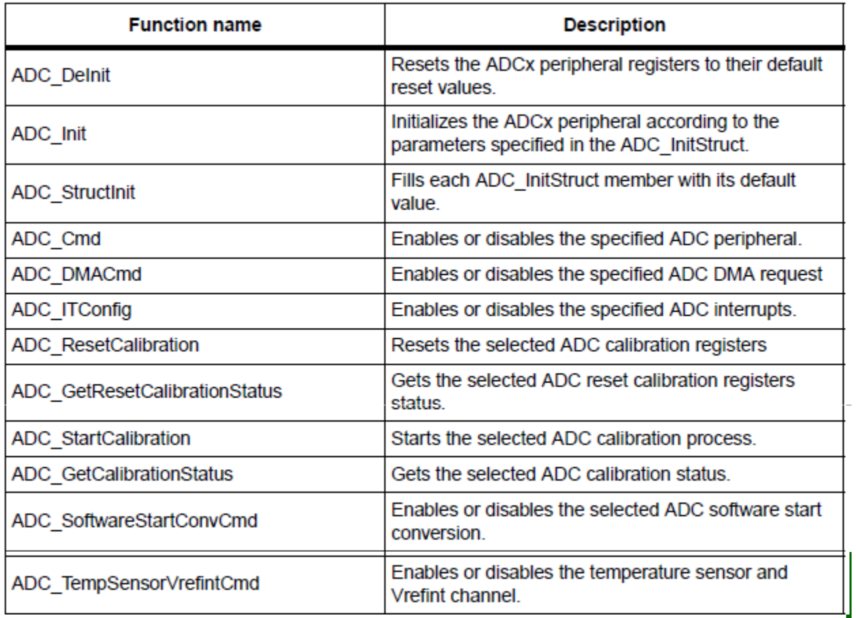


Function name	Description
GPIO_DeInit	Resets the GPIOx peripheral registers to their default reset values.
GPIO_AFIODeInit	Resets the Alternate Functions (remap, event control and EXTI configuration) registers to their default reset values.
GPIO_Init	Initializes the GPIOx peripheral according to the specified parameters in the GPIO_InitStruct.
GPIO_StructInit	Fills each GPIO_InitStruct member with its default value.
GPIO_ReadInputDataBit	Reads the specified input port pin
GPIO_ReadInputData	Reads the specified GPIO input data port
GPIO_ReadOutputDataBit	Reads the specified output data port bit
GPIO_ReadOutputData	Reads the specified GPIO output data port
GPIO_SetBits	Sets the selected data port bits
GPIO_ResetBits	Clears the selected data port bits
GPIO_WriteBit	Sets or clears the selected data port bit
GPIO_Write	Writes data to the specified GPIO data port
GPIO_PinLockConfig	Locks GPIO Pins configuration registers
GPIO_EventOutputConfig	Selects the GPIO pin used as Event output.
GPIO_EventOutputCmd	Enables or disables the Event Output.
GPIO_PinRemapConfig	Changes the mapping of the specified pin.
GPIO_EXTILineConfig	Selects the GPIO pin used as EXTI Line.

```
void GPIO_Configuration(void)
{
  GPIO_InitTypeDef GPIO_InitStructure;
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_11;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_Init(GPIOB, &GPIO_InitStructure);
}
```

Configure ADC

ADC FwLib Functions List

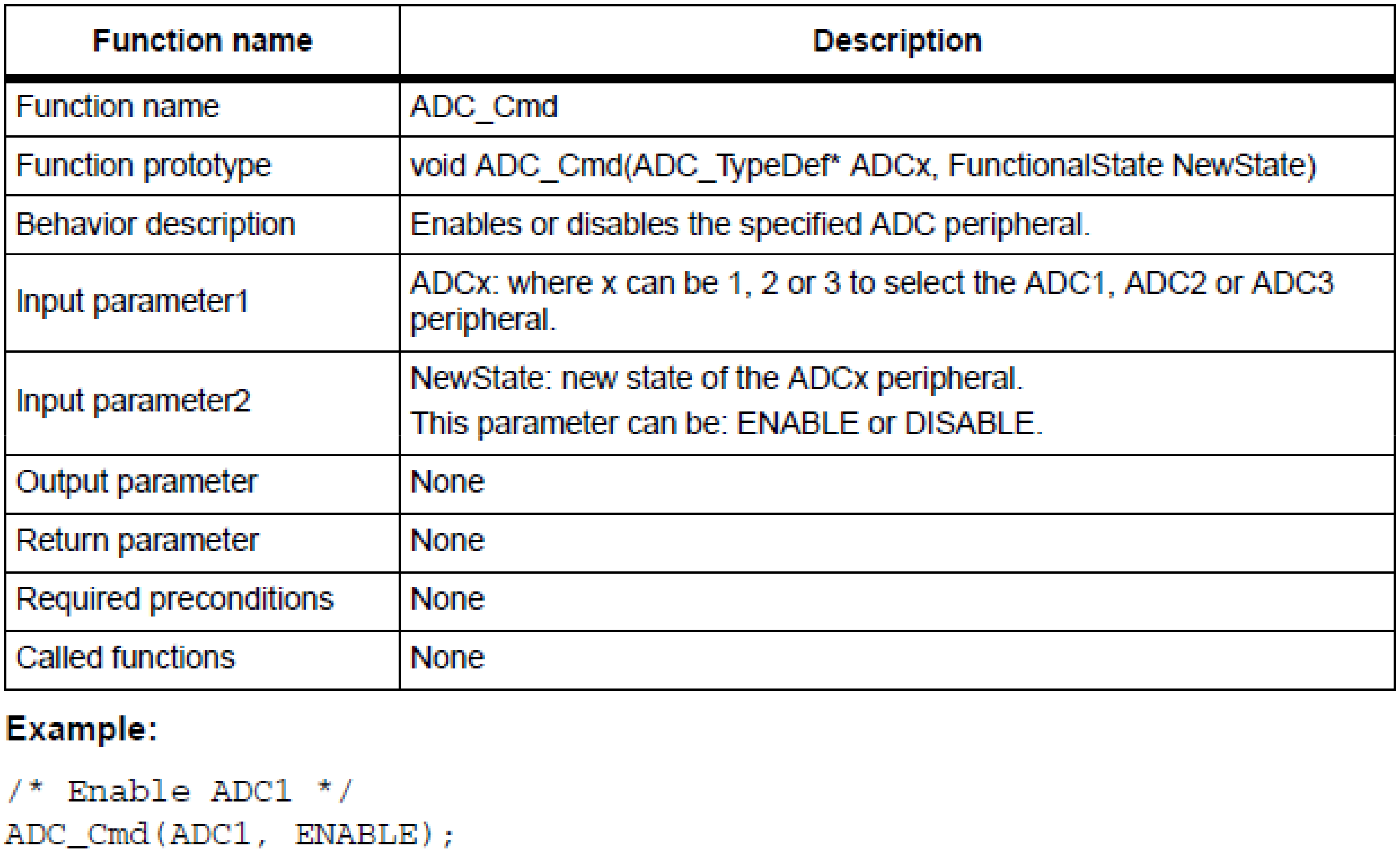


Function name	Description
ADC_DeInit	Resets the ADCx peripheral registers to their default reset values.
ADC_Init	Initializes the ADCx peripheral according to the parameters specified in the ADC_InitStruct.
ADC_StructInit	Fills each ADC_InitStruct member with its default value.
ADC_Cmd	Enables or disables the specified ADC peripheral.
ADC_DMACmd	Enables or disables the specified ADC DMA request
ADC_ITConfig	Enables or disables the specified ADC interrupts.
ADC_ResetCalibration	Resets the selected ADC calibration registers
ADC_GetResetCalibrationStatus	Gets the selected ADC reset calibration registers status.
ADC_StartCalibration	Starts the selected ADC calibration process.
ADC_GetCalibrationStatus	Gets the selected ADC calibration status.
ADC_SoftwareStartConvCmd	Enables or disables the selected ADC software start conversion.
ADC_TempSensorVrefintCmd	Enables or disables the temperature sensor and Vrefint channel.

```
void ADC_Configuration(void)
{
  ADC_InitTypeDef ADC_InitStructure;
  ADC_InitStructure.ADC_Mode = ADC_Mode_Independent;
  ADC_InitStructure.ADC_ScanConvMode = ENABLE;
  ADC_InitStructure.ADC_ContinuousConvMode = ENABLE;
  ADC_InitStructure.ADC_ExternalTrigConv = ADC_ExternalTrigConv_None;
  ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;
  ADC_InitStructure.ADC_NbrOfChannel = 1;
  ADC_Init(ADC1, &ADC_InitStructure);

  /* ADC1 regular channel16 configuration */
  //insert your code
  /* Temperature Enable */
  //insert your code
  /* Enable ADC1 DMA */
  //insert your code
  /* Enable ADC1 */
  //insert your code
  /* Enable ADC1 reset calibaration register */
  //insert your code
  /* Check the end of ADC1 reset calibration register */
  while(ADC_GetResetCalibrationStatus(ADC1));
  /* Start ADC1 calibaration */
  ADC_StartCalibration(ADC1);
  /* Check the end of ADC1 calibration */
  while(ADC_GetCalibrationStatus(ADC1));
  /* Start ADC1 Software Conversion */
  //insert your code
}
```

ADC Cmd function

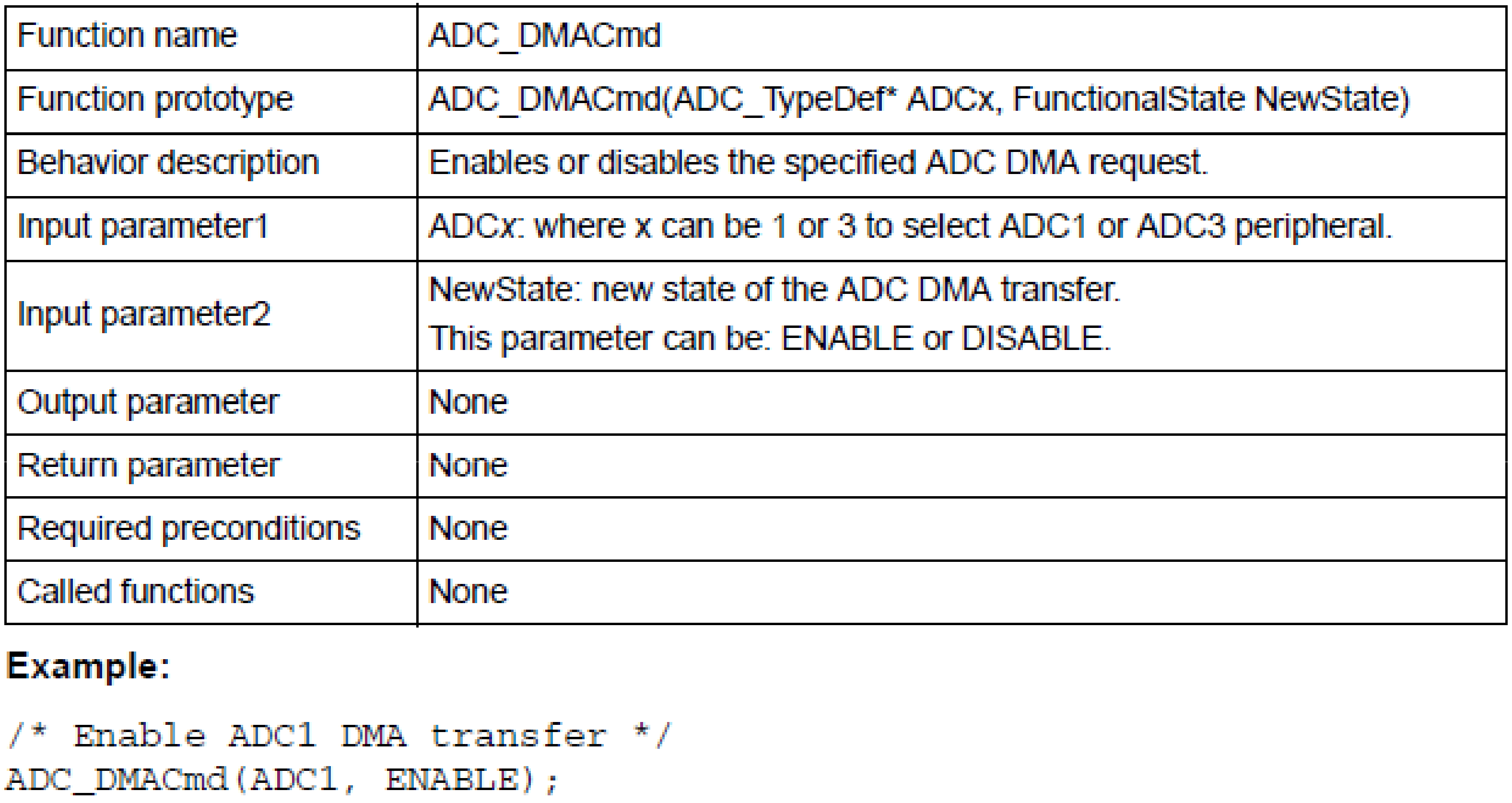


Function name	Description
Function name	ADC_Cmd
Function prototype	void ADC_Cmd(ADC_TypeDef* ADCx, FunctionalState NewState)
Behavior description	Enables or disables the specified ADC peripheral.
Input parameter1	ADCx: where x can be 1, 2 or 3 to select the ADC1, ADC2 or ADC3 peripheral.
Input parameter2	NewState: new state of the ADCx peripheral. This parameter can be: ENABLE or DISABLE.
Output parameter	None
Return parameter	None
Required preconditions	None
Called functions	None

Example:

```
/* Enable ADC1 */
ADC_Cmd(ADC1, ENABLE);
```

ADC DMA Cmd function

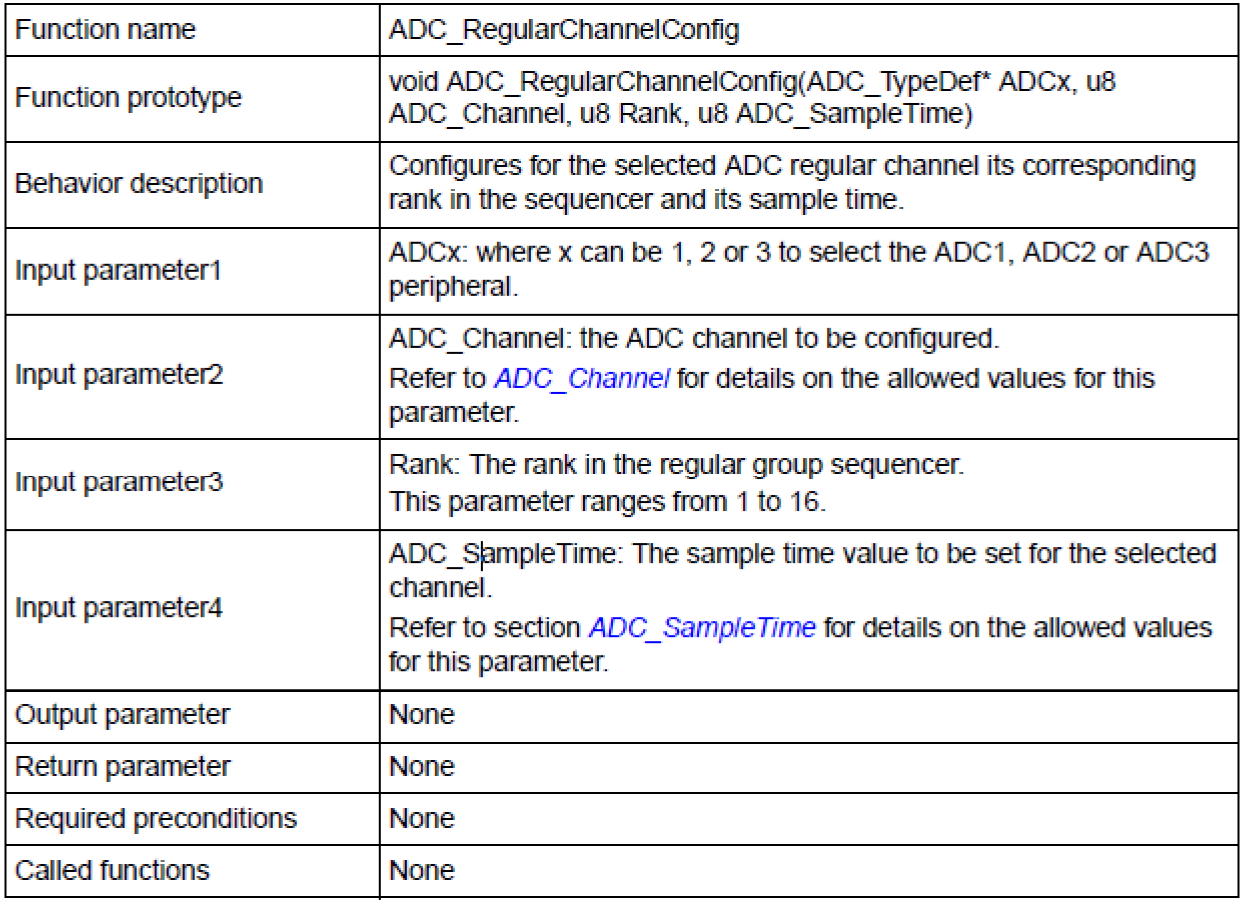


Function name	ADC_DMACmd
Function prototype	ADC_DMACmd(ADC_TypeDef* ADCx, FunctionalState NewState)
Behavior description	Enables or disables the specified ADC DMA request.
Input parameter1	ADCx: where x can be 1 or 3 to select ADC1 or ADC3 peripheral.
Input parameter2	NewState: new state of the ADC DMA transfer. This parameter can be: ENABLE or DISABLE.
Output parameter	None
Return parameter	None
Required preconditions	None
Called functions	None

Example:

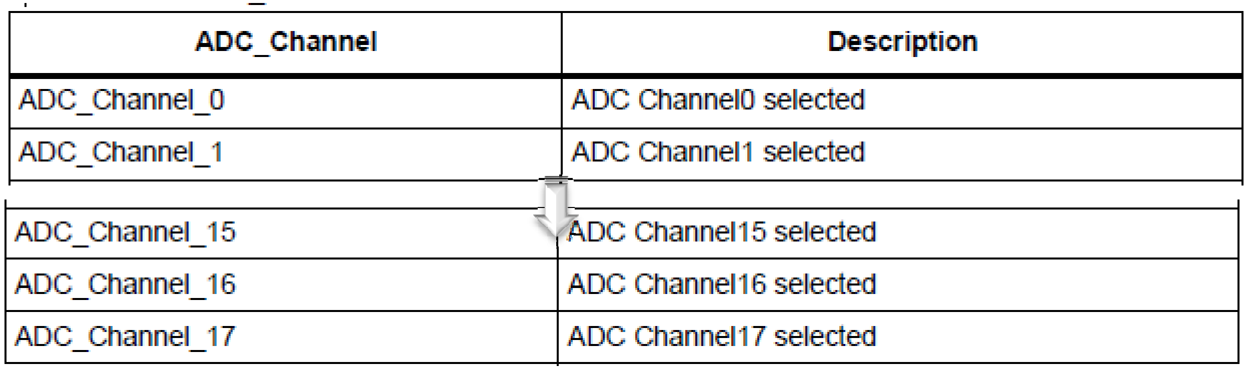
```
/* Enable ADC1 DMA transfer */
ADC_DMACmd(ADC1, ENABLE);
```

ADC RegularChannelConfig function

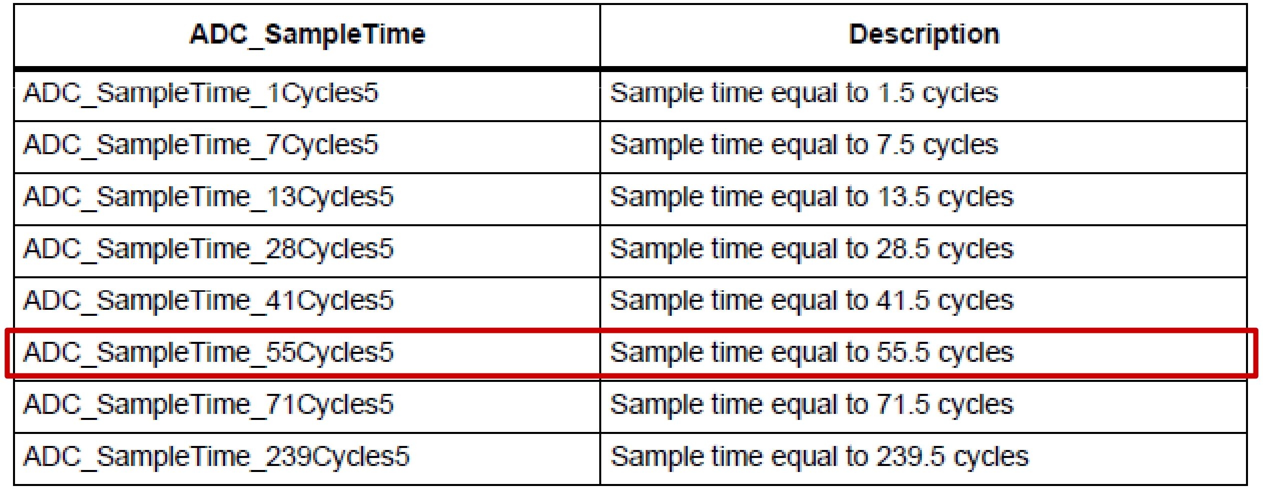


Function name	ADC_RegularChannelConfig
Function prototype	void ADC_RegularChannelConfig(ADC_TypeDef* ADCx, u8 ADC_Channel, u8 Rank, u8 ADC_SampleTime)
Behavior description	Configures for the selected ADC regular channel its corresponding rank in the sequencer and its sample time.
Input parameter1	ADCx: where x can be 1, 2 or 3 to select the ADC1, ADC2 or ADC3 peripheral.
Input parameter2	ADC_Channel: the ADC channel to be configured. Refer to *ADC_Channel* for details on the allowed values for this parameter.
Input parameter3	Rank: The rank in the regular group sequencer. This parameter ranges from 1 to 16.
Input parameter4	ADC_SampleTime: The sample time value to be set for the selected channel. Refer to section *ADC_SampleTime* for details on the allowed values for this parameter.
Output parameter	None
Return parameter	None
Required preconditions	None
Called functions	None

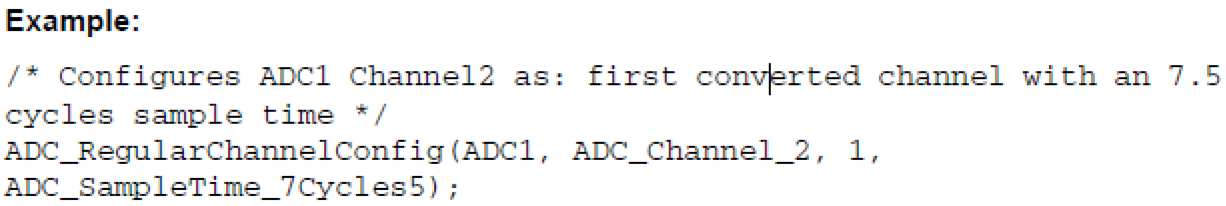
ADC RegularChannelConfig function



ADC_Channel	Description
ADC_Channel_0	ADC Channel0 selected
ADC_Channel_1	ADC Channel1 selected
ADC_Channel_15	ADC Channel15 selected
ADC_Channel_16	ADC Channel16 selected
ADC_Channel_17	ADC Channel17 selected



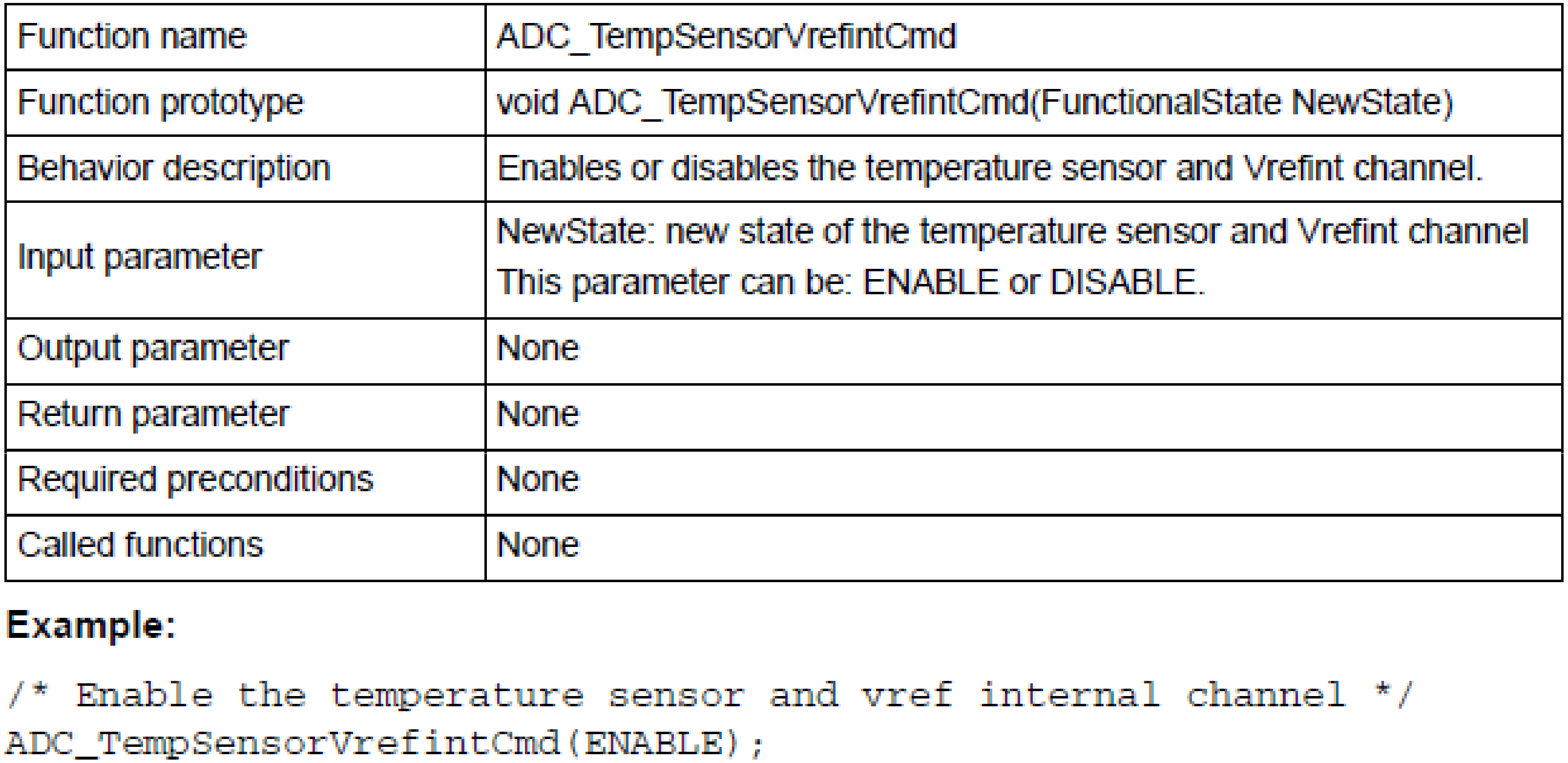
ADC_SampleTime	Description
ADC_SampleTime_1Cycles5	Sample time equal to 1.5 cycles
ADC_SampleTime_7Cycles5	Sample time equal to 7.5 cycles
ADC_SampleTime_13Cycles5	Sample time equal to 13.5 cycles
ADC_SampleTime_28Cycles5	Sample time equal to 28.5 cycles
ADC_SampleTime_41Cycles5	Sample time equal to 41.5 cycles
ADC_SampleTime_55Cycles5	Sample time equal to 55.5 cycles
ADC_SampleTime_71Cycles5	Sample time equal to 71.5 cycles
ADC_SampleTime_239Cycles5	Sample time equal to 239.5 cycles



Example:

```
/* Configures ADC1 Channel2 as: first converted channel with an 7.5
cycles sample time */
ADC_RegularChannelConfig(ADC1, ADC_Channel_2, 1,
ADC_SampleTime_7Cycles5);
```

ADC TempSensorVrefintCmd function

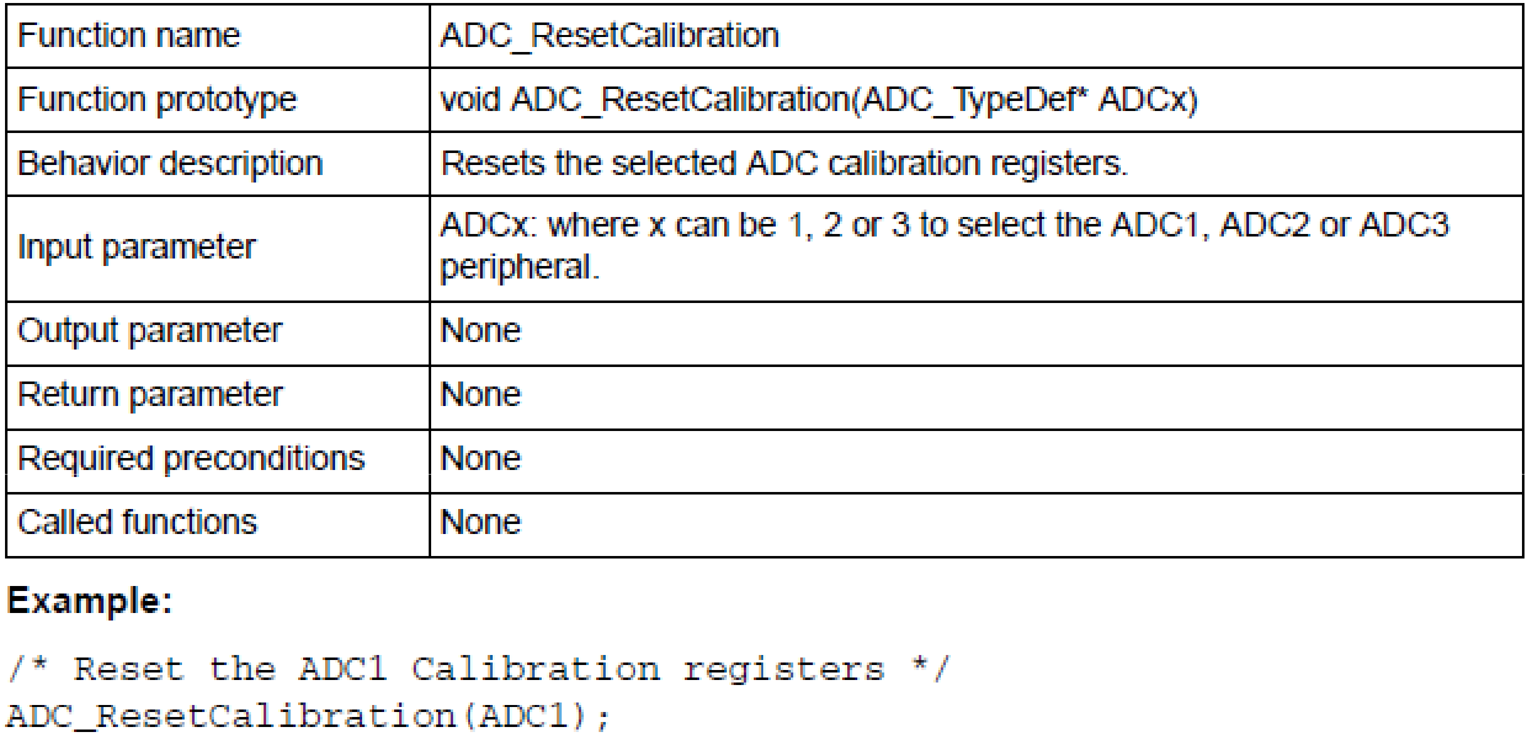


Function name	ADC_TempSensorVrefintCmd
Function prototype	void ADC_TempSensorVrefintCmd(FunctionalState NewState)
Behavior description	Enables or disables the temperature sensor and Vrefint channel.
Input parameter	NewState: new state of the temperature sensor and Vrefint channel This parameter can be: ENABLE or DISABLE.
Output parameter	None
Return parameter	None
Required preconditions	None
Called functions	None

Example:

```
/* Enable the temperature sensor and vref internal channel */
ADC_TempSensorVrefintCmd(ENABLE);
```

ADC ResetCalibration function

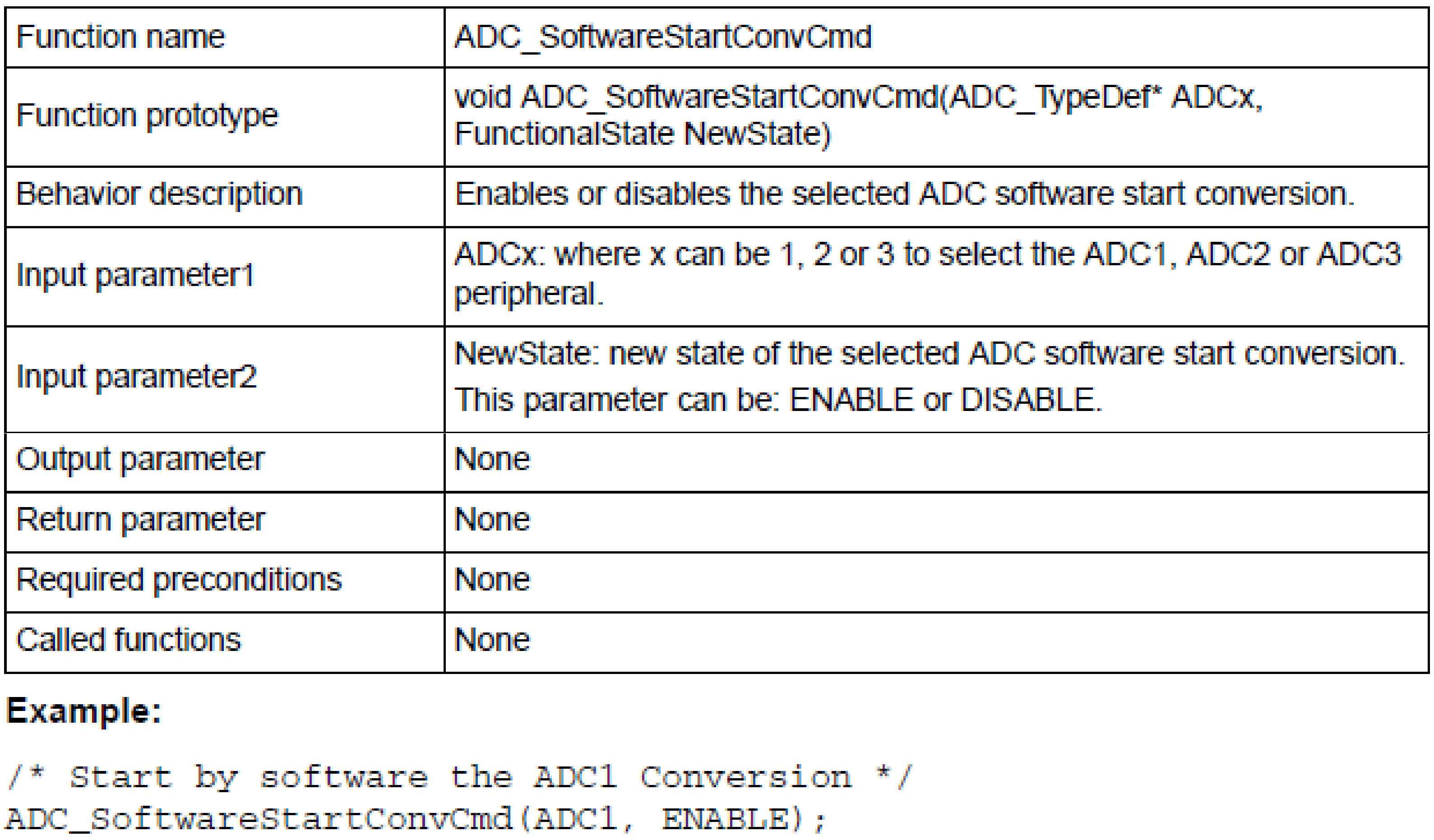


Function name	ADC_ResetCalibration
Function prototype	void ADC_ResetCalibration(ADC_TypeDef* ADCx)
Behavior description	Resets the selected ADC calibration registers.
Input parameter	ADCx: where x can be 1, 2 or 3 to select the ADC1, ADC2 or ADC3 peripheral.
Output parameter	None
Return parameter	None
Required preconditions	None
Called functions	None

Example:

```
/* Reset the ADC1 Calibration registers */
ADC_ResetCalibration(ADC1);
```

ADC SoftwareStartConvCmd *function*

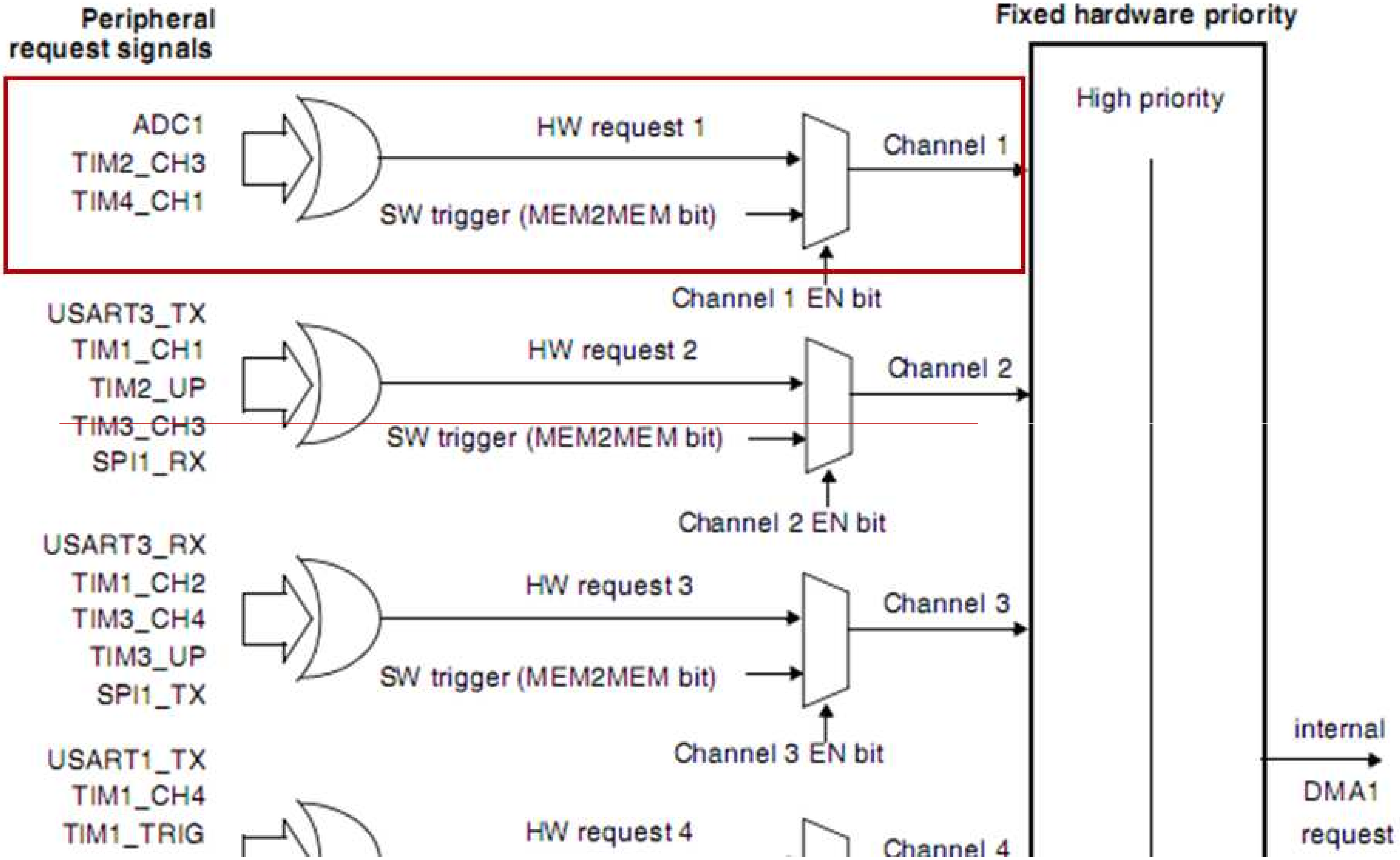


Function name	ADC_SoftwareStartConvCmd
Function prototype	void ADC_SoftwareStartConvCmd(ADC_TypeDef* ADCx, FunctionalState NewState)
Behavior description	Enables or disables the selected ADC software start conversion.
Input parameter1	ADCx: where x can be 1, 2 or 3 to select the ADC1, ADC2 or ADC3 peripheral.
Input parameter2	NewState: new state of the selected ADC software start conversion. This parameter can be: ENABLE or DISABLE.
Output parameter	None
Return parameter	None
Required preconditions	None
Called functions	None

Example:

```
/* Start by software the ADC1 Conversion */
ADC_SoftwareStartConvCmd(ADC1, ENABLE);
```

DMA1 request mapping



Configure DMA

```
void DMA_Configuration(void)
{
  DMA_DeInit(DMA1_Channel1);
  DMA_InitStructure.DMA_PeripheralBaseAddr = ADC1_DR_Address;
  DMA_InitStructure.DMA_MemoryBaseAddr = (u32)&ADCConvertedValue;
  DMA_InitStructure.DMA_DIR = DMA_DIR_PeripheralSRC;
  DMA_InitStructure.DMA_BufferSize = 1;
  DMA_InitStructure.DMA_PeripheralInc = DMA_PeripheralInc_Disable;
  DMA_InitStructure.DMA_MemoryInc = DMA_MemoryInc_Disable;
  DMA_InitStructure.DMA_PeripheralDataSize = 
DMA_PeripheralDataSize_HalfWord;
  DMA_InitStructure.DMA_MemoryDataSize = DMA_MemoryDataSize_HalfWord;
  DMA_InitStructure.DMA_Mode = DMA_Mode_Circular;
  DMA_InitStructure.DMA_Priority = DMA_Priority_High;
  DMA_InitStructure.DMA_M2M = DMA_M2M_Disable;
  DMA_Init(DMA1_Channel1, &DMA_InitStructure);

  /* Enable DMA1 channel1 */
  DMA_Cmd(DMA1_Channel1, ENABLE);
}
```

User code

```
void Count_Temperature(void)
{
//取平均16次的ADCConvertedValue 來計算(比較準確)，
//取溫度必須間隔一下，請使用Delay();
//溫度=Temperature (in °C) = {(V25 - VSENSE) / Avg_Slope} + 25.
//印出現在溫度
}
```

Temperature sensor (Con.)

- 溫度公式:

V_{SENSE} :溫度感測器的當前輸出電壓，與 ADC_ConvertedValue 數值轉換成電壓的關係為:

$$V_{SENSE} = \frac{\text{ADC\_ConvertedValue * Vdd}}{\text{Vdd\_convert\_value(0xFFF)}}$$

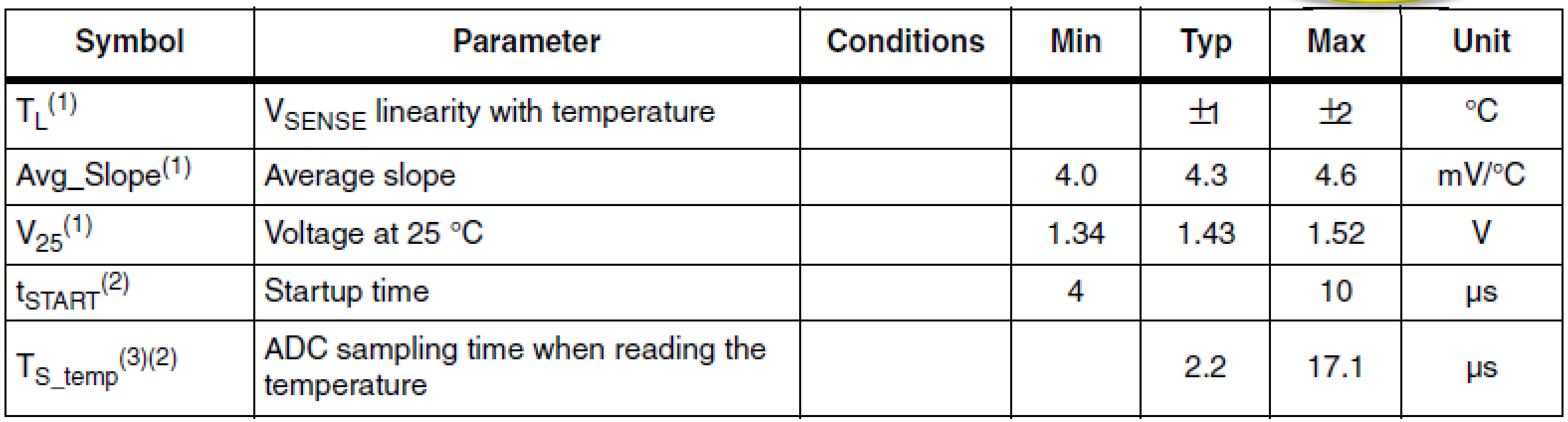
V_{25}:溫度感測器在25度C時的輸出電壓，典型值1.43V

Avg_Slope :溫度感測器的輸出電壓和溫度的關聯參數，典型值4.3 mV/°C

Temperature (in °C) = {(V_{25} - V_{SENSE}) / Avg_Slope} + 25.

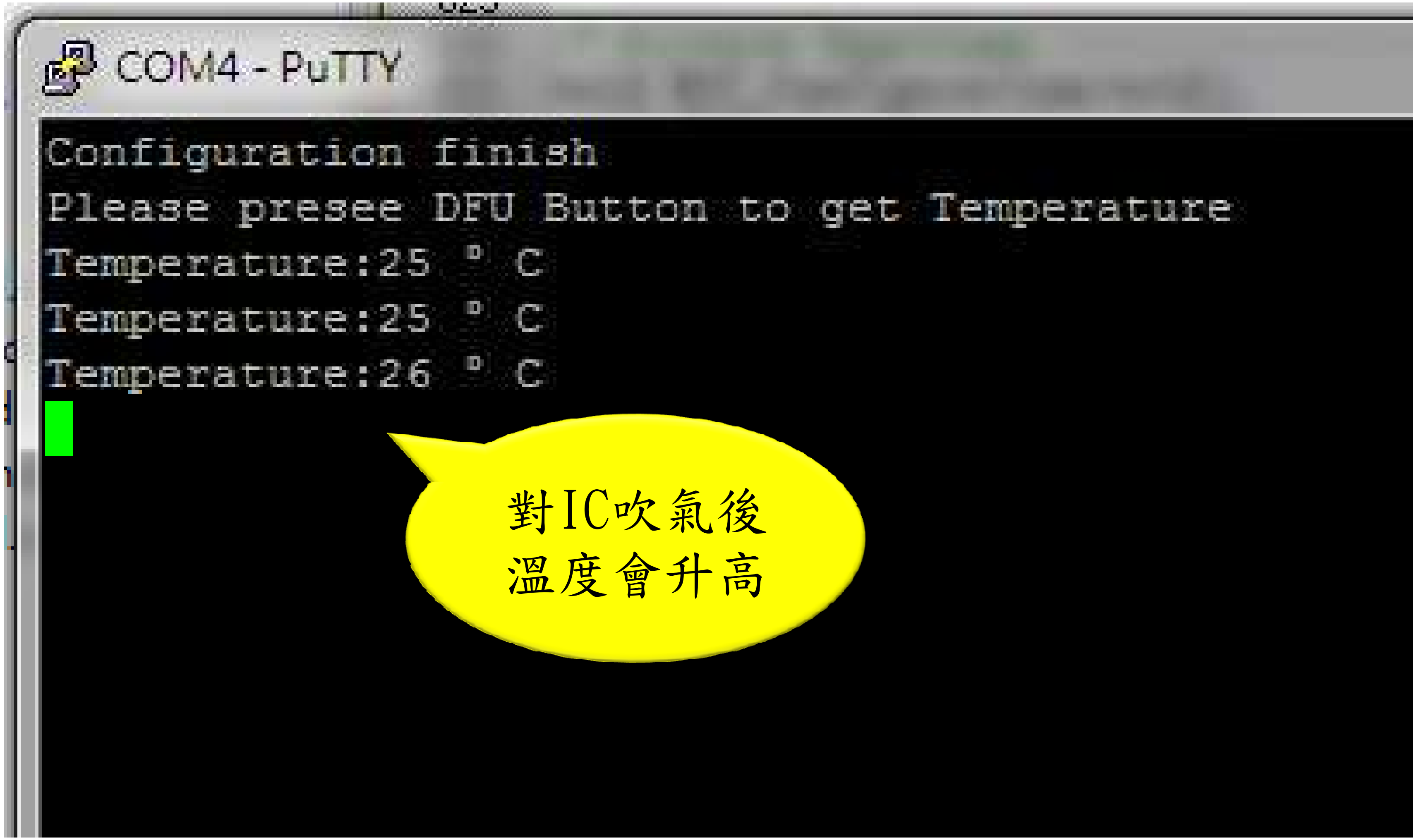
*1000

Table 62. TS characteristics



Symbol	Parameter	Conditions	Min	Typ	Max	Unit
T_L(1)	V_{SENSE} linearity with temperature			±1	±2	°C
Avg_Slope(1)	Average slope		4.0	4.3	4.6	mV/°C
V_{25}(1)	Voltage at 25 °C		1.34	1.43	1.52	V
t_{START}(2)	Startup time		4		10	µs
T_{S_temp}(3)(2)	ADC sampling time when reading the temperature			2.2	17.1	µs

DEMO



```
Configuration finish
Please presee DFU Button to get Temperature
Temperature:25 ° C
Temperature:25 ° C
Temperature:26 ° C
```