Text LCD / Analog to Digital Converter Driver

WU-YANG

Technology Co., Ltd.

Declared Version

Training Only	
Declare	
Document Version	1.00
Release Date	2009.06.20
Document Title	Text LCD / Analog to Digital Converter Driver
Exercise Time	■ Lecture 30 minutes ■ Operating 60 minutes
Platform	■ MIAT_STM32 ■ MIAT_IOB
Peripheral	■ Text LCD ■ VR
Author	■WU-YANG Technology Co., Ltd.

實驗目的

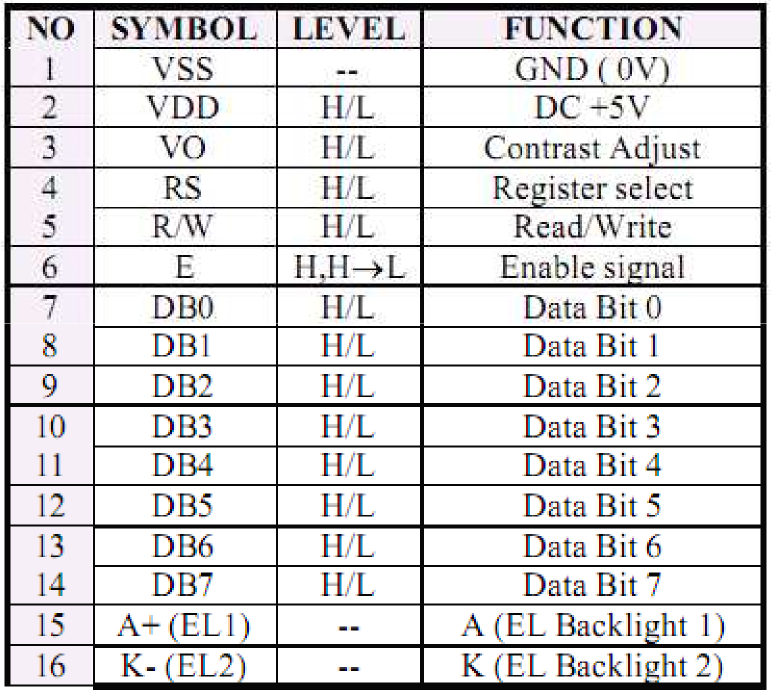
- □ 瞭解ARM A/D Converter 運作方式，並將轉換後的數位資料顯示於Text LCD上。

實驗原理

- Text LCD Control
- Development Flow
- ARM Configure

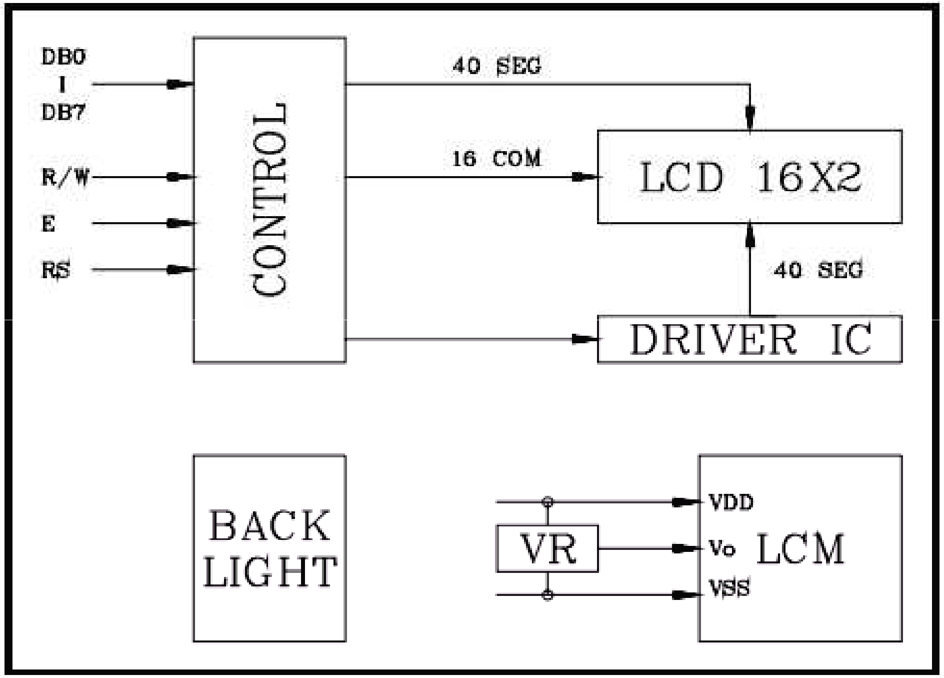
Text LCD Control(1)

Text LCD Pin Define



NO	SYMBOL	LEVEL	FUNCTION
1	VSS	--	GND (0V)
2	VDD	H/L	DC +5V
3	VO	H/L	Contrast Adjust
4	RS	H/L	Register select
5	R/W	H/L	Read/Write
6	E	H,H→L	Enable signal
7	DB0	H/L	Data Bit 0
8	DB1	H/L	Data Bit 1
9	DB2	H/L	Data Bit 2
10	DB3	H/L	Data Bit 3
11	DB4	H/L	Data Bit 4
12	DB5	H/L	Data Bit 5
13	DB6	H/L	Data Bit 6
14	DB7	H/L	Data Bit 7
15	A+ (EL1)	--	A (EL Backlight 1)
16	K- (EL2)	--	K (EL Backlight 2)

Text LCD Block Diagram



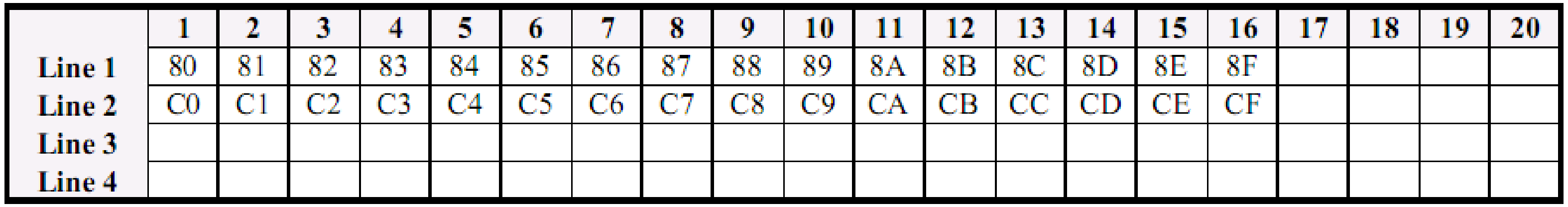
Text LCD Control(2)

- 1.固定字型ROM，稱為CG（Character Generator）ROM。

CG ROM內儲存著192個5x7點矩陣的字型，這些字型均已固定，例如我們將"A"寫入LCD中，就是將 "A" 的ASCII碼41H寫至DDRAM中，同時至CG ROM中將 "A"的字型點矩陣資料找出來而顯示在LCD上。

- 2.資料顯示RAM，稱為DD（Data Display）RAM。

DD RAM內用來儲存寫至LCD內部的字元，DD RAM的位址分佈從00H到67H，分別代表LCD的各行位置，如下表所示，例如我們要將 "A"寫入第2行的第1個位置，就先設定DD RAM位址為40H，而後寫入41H至LCD即可。



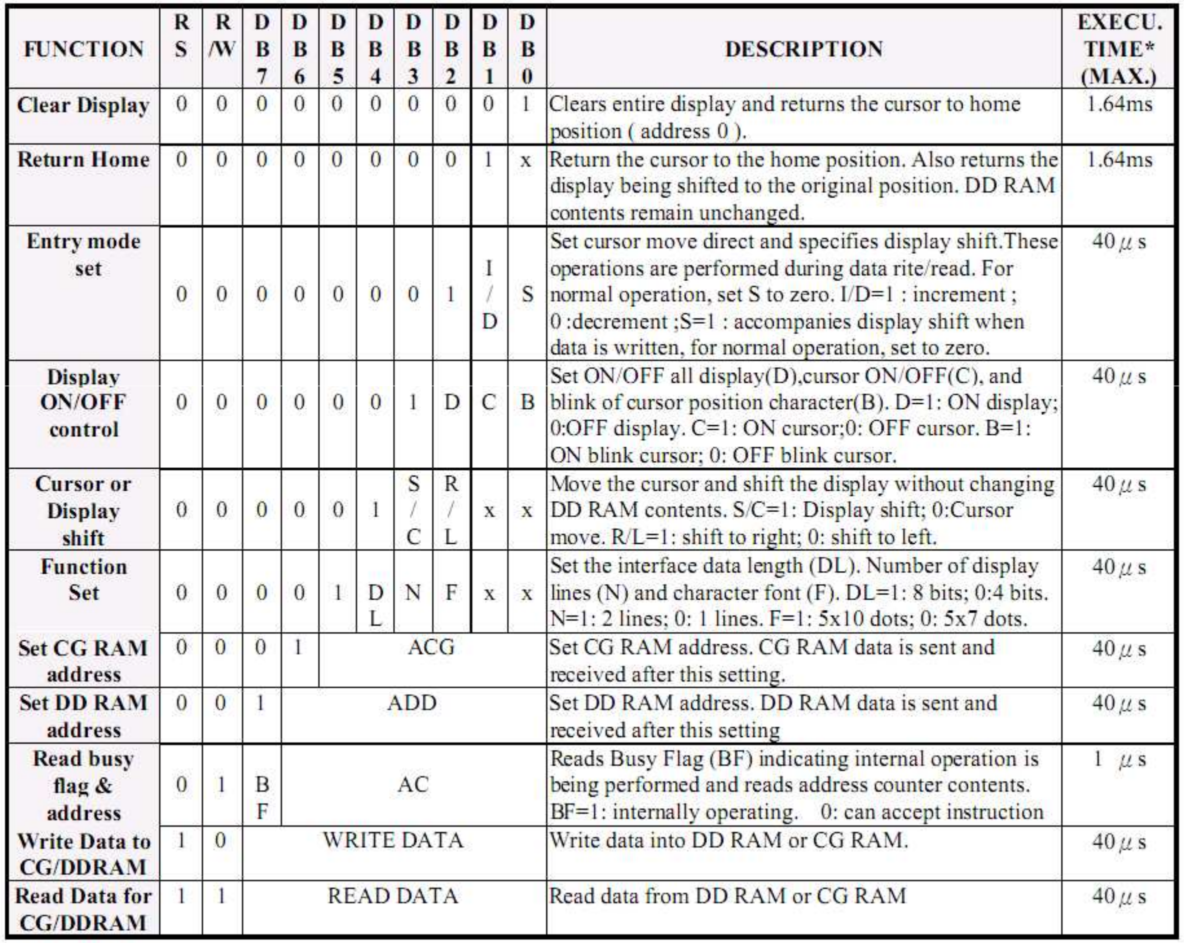
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Line 1	80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F				
Line 2	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF				
Line 3																				
Line 4																				

- 3.使用者自訂字型RAM，稱為CG RAM。

此區域只有64位元組，可由使用者將自行設計的字型寫入LCD中，一個字的大小為5x8點矩陣，共可以儲存8個字型，其顯示碼為00H到07H。

Text LCD Control(3)

Instruction Set



FUNCTION	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	DESCRIPTION	EXECU. TIME* (MAX.)
Clear Display	0	0	0	0	0	0	0	0	0	1	Clears entire display and returns the cursor to home position (address 0).	1.64ms
Return Home	0	0	0	0	0	0	0	0	1	x	Return the cursor to the home position. Also returns the display being shifted to the original position. DD RAM contents remain unchanged.	1.64ms
Entry mode set	0	0	0	0	0	0	0	1	I/D	S	Set cursor move direct and specifies display shift.These operations are performed during data rite/read. For normal operation, set S to zero. I/D=1 : increment ; 0 :decrement ;S=1 : accompanies display shift when data is written, for normal operation, set to zero.	40 μ s
Display ON/OFF control	0	0	0	0	0	0	1	D	C	B	Set ON/OFF all display(D),cursor ON/OFF(C), and blink of cursor position character(B). D=1: ON display; 0:OFF display. C=1: ON cursor;0: OFF cursor. B=1: ON blink cursor; 0: OFF blink cursor.	40 μ s
Cursor or Display shift	0	0	0	0	0	1	S/C	R/L	x	x	Move the cursor and shift the display without changing DD RAM contents. S/C=1: Display shift; 0:Cursor move. R/L=1: shift to right; 0: shift to left.	40 μ s
Function Set	0	0	0	0	1	DL	N	F	x	x	Set the interface data length (DL). Number of display lines (N) and character font (F). DL=1: 8 bits; 0:4 bits. N=1: 2 lines; 0: 1 lines. F=1: 5x10 dots; 0: 5x7 dots.	40 μ s
Set CG RAM address	0	0	0	1	ACG						Set CG RAM address. CG RAM data is sent and received after this setting.	40 μ s
Set DD RAM address	0	0	1	ADD							Set DD RAM address. DD RAM data is sent and received after this setting	40 μ s
Read busy flag & address	0	1	BF	AC							Reads Busy Flag (BF) indicating internal operation is being performed and reads address counter contents. BF=1: internally operating. 0: can accept instruction	1 μ s
Write Data to CG/DDRAM	1	0	WRITE DATA								Write data into DD RAM or CG RAM.	40 μ s
Read Data for CG/DDRAM	1	1	READ DATA								Read data from DD RAM or CG RAM	40 μ s

Text LCD Function List(1)

- init_lcd()

  LCD初始化，設定LCD游標是否出現、資料傳送方式等
- write_com(unsigned char c)

  將參數C寫入指令暫存器
- write_data(unsigned char c)

  將參數C寫入資料暫存器
- print(char line, char *str)

  將LCD清空並把*str印在LCD上，line是決定要印在哪一行
- prline1(char x, char w)

  LCD不清空，將字元w印在第一行的x位置
- prline2(char x, char w)

  LCD不清空，將字元w印在第二行的x位置

Text LCD Function List(2)

□ clear(void)
清除LCD

□ home(void)
游標移至原始位置

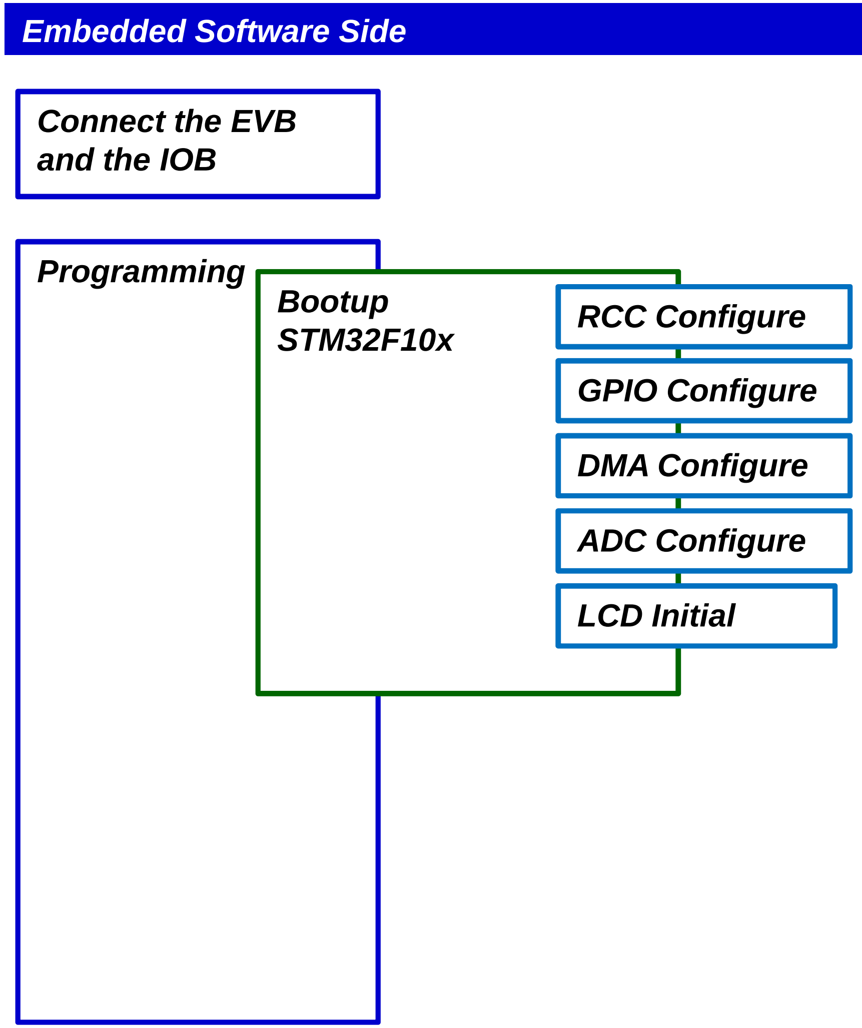
□ setCursor(char index)
設定游標位置

□ shiftDisplayLeft(void)
左移Display

□ shiftDisplayRight(void)
右移Display

□ pf4h(unsigned int value)
將整數轉為hex並輸出於LCD上

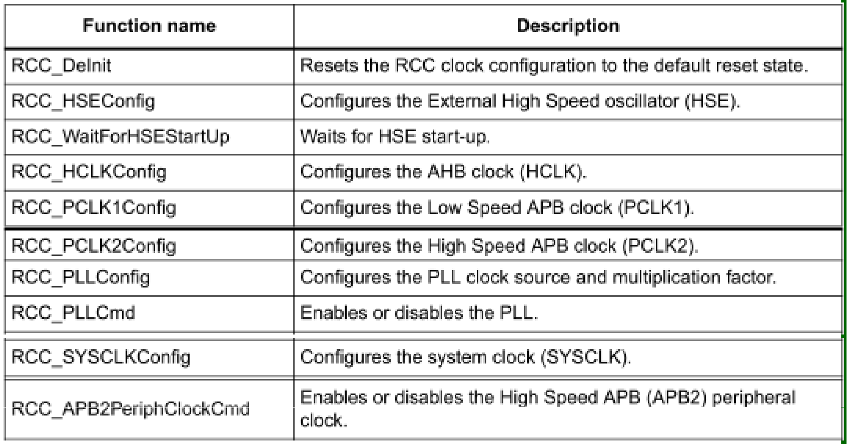
Development Flow



```
int main(void)
{

  #ifdef DEBUG
    debug();
  #endif

  /* System clocks configuration ----------------------*/
  RCC_Configuration();

  /* NVIC configuration --------------------------------*/
  NVIC_Configuration();

  /* GPIO configuration --------------------------------*/
  GPIO_Configuration();

  /* DMA1 channel1 configuration ----------------------*/
  DMA_Configuration();

  /* ADC1 configuration --------------------------------*/
  ADC_Configuration();

  /* 初始化 LCD 介面  */
  init_lcd();

   clear();
   while(1)
   {
      pf4h(ADCConvertedValue);
      delay(10000);
   }
}
```

Configure RCC

RCC FwLib Functions List

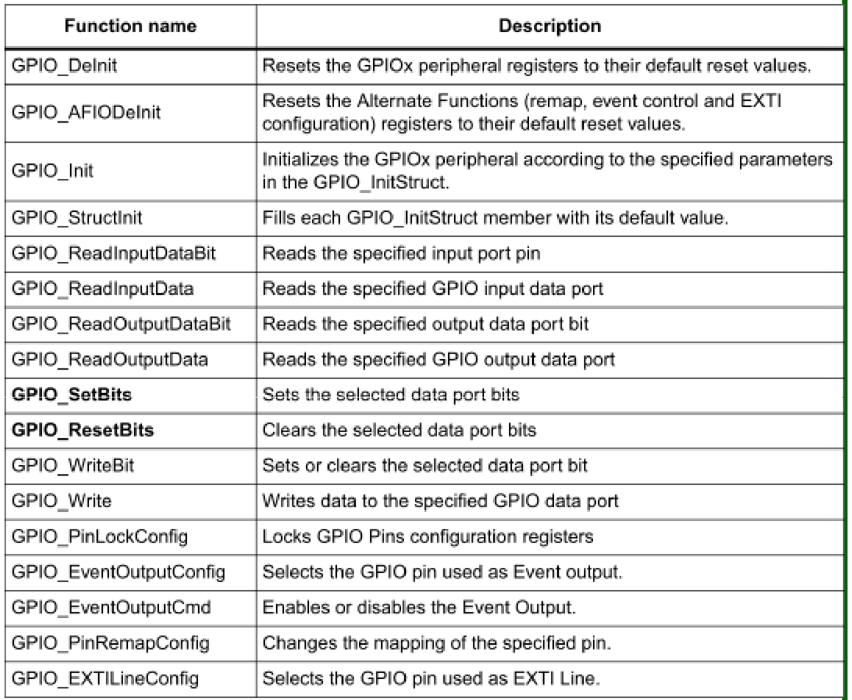


Function name	Description
RCC_DeInit	Resets the RCC clock configuration to the default reset state.
RCC_HSEConfig	Configures the External High Speed oscillator (HSE).
RCC_WaitForHSEStartUp	Waits for HSE start-up.
RCC_HCLKConfig	Configures the AHB clock (HCLK).
RCC_PCLK1Config	Configures the Low Speed APB clock (PCLK1).
RCC_PCLK2Config	Configures the High Speed APB clock (PCLK2).
RCC_PLLConfig	Configures the PLL clock source and multiplication factor.
RCC_PLLCmd	Enables or disables the PLL.
RCC_SYSCLKConfig	Configures the system clock (SYSCLK).
RCC_APB2PeriphClockCmd	Enables or disables the High Speed APB (APB2) peripheral clock.

```
void RCC_Configuration(void)
{
  /* RCC system reset(for debug purpose) */
  RCC_DeInit();
  /* Enable HSE */
  RCC_HSEConfig(RCC_HSE_ON);
  /* Wait till HSE is ready */
  HSEStartUpStatus = RCC_WaitForHSEStartUp();
  if(HSEStartUpStatus == SUCCESS) {
    /* Enable Prefetch Buffer */
    FLASH_PrefetchBufferCmd(FLASH_PrefetchBuffer_Enable);
    /* Flash 2 wait state */
    FLASH_SetLatency(FLASH_Latency_2);
    /* HCLK = SYSCLK */
    RCC_HCLKConfig(RCC_SYSCLK_Div1);
    /* PCLK2 = HCLK */
    RCC_PCLK2Config(RCC_HCLK_Div1);
    /* PCLK1 = HCLK/2 */
    RCC_PCLK1Config(RCC_HCLK_Div2);
    /* ADCCLK = PCLK2/4 */
    RCC_ADCCLKConfig(RCC_PCLK2_Div4);
    /* PLLCLK = 8MHz * 7 = 56 MHz */
    RCC_PLLConfig(RCC_PLLSource_HSE_Div1, RCC_PLLMul_7);
    /* Enable PLL */
    RCC_PLLCmd(ENABLE);
    /* Wait till PLL is ready */
    while(RCC_GetFlagStatus(RCC_FLAG_PLLRDY) == RESET) { }
    /* Select PLL as system clock source */
    RCC_SYSCLKConfig(RCC_SYSCLKSource_PLLCLK);
    /* Wait till PLL is used as system clock source */
    while(RCC_GetSYSCLKSource() != 0x08) {  }
  }
/* Enable peripheral clocks ------------------------------*/
  /* Enable DMA1 clock */
  RCC_AHBPeriphClockCmd(RCC_AHBPeriph_DMA1, ENABLE);
  /* Enable ADC1 and GPIOA clock */
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1 | RCC_APB2Periph_GPIOA,
ENABLE);
  /* Enable GPIOC clock */
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC, ENABLE);
}
```

Configure GPIO

GPIO FwLib Functions List

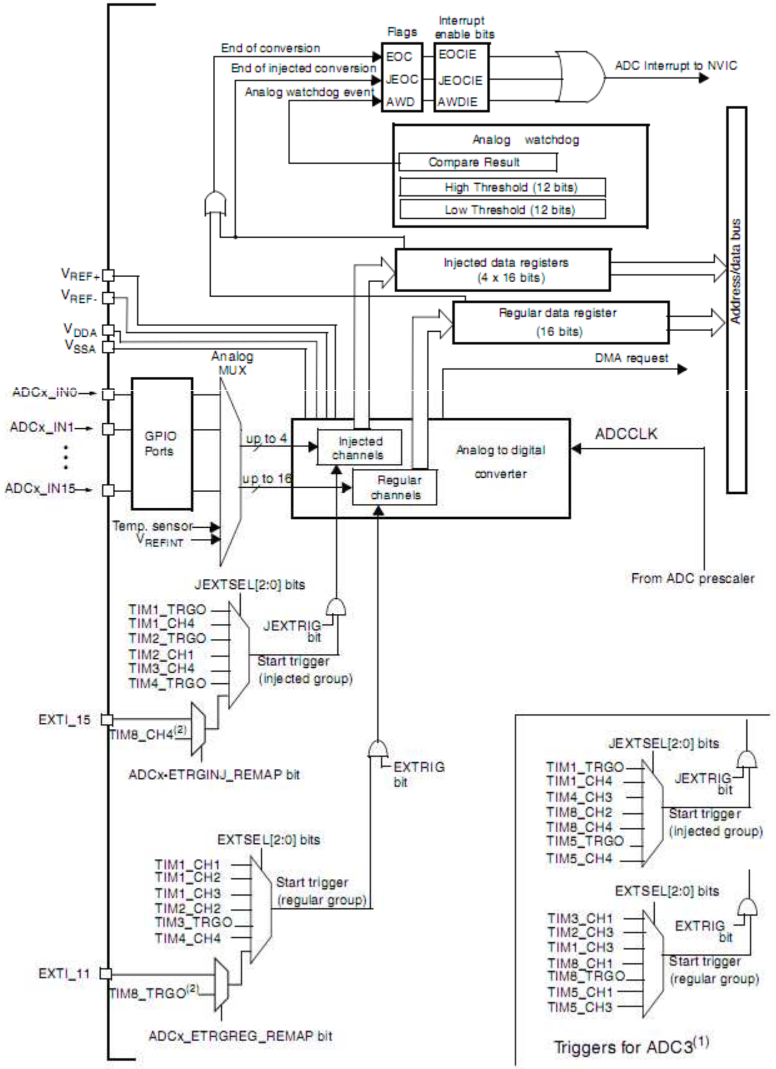


Function name	Description
GPIO_DeInit	Resets the GPIOx peripheral registers to their default reset values.
GPIO_AFIODeInit	Resets the Alternate Functions (remap, event control and EXTI configuration) registers to their default reset values.
GPIO_Init	Initializes the GPIOx peripheral according to the 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;


  /*Configure PA.1 (ADC Channel1) as analog input --------------*/
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_1;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AIN;
  GPIO_Init(GPIOA, &GPIO_InitStructure);



  /* Configure IO connected to GPIOC ********************/
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0 | GPIO_Pin_1 | GPIO_Pin_2
                              | GPIO_Pin_3 | GPIO_Pin_4 | GPIO_Pin_5
                              | GPIO_Pin_6 | GPIO_Pin_7 | GPIO_Pin_8
                              | GPIO_Pin_9 | GPIO_Pin_10 ;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_Init(GPIOC, &GPIO_InitStructure);
}
```

ARM ADC Control

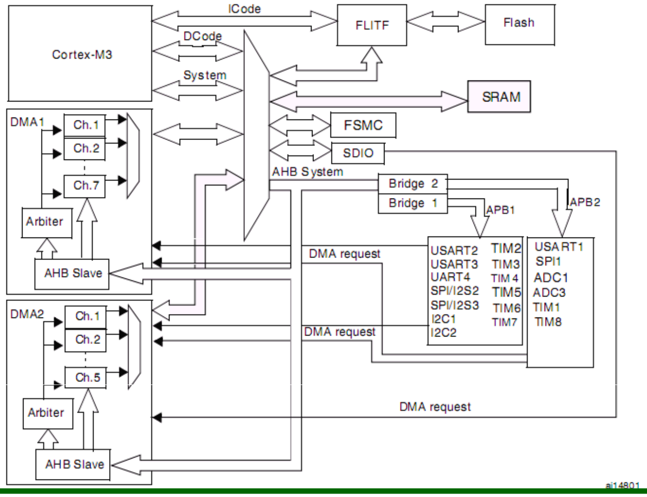
ADC Block Diagram



```
void ADC_Configuration(void)
{
  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 channel1 configuration */
  ADC_RegularChannelConfig(ADC1, ADC_Channel_1, 1,
ADC_SampleTime_55Cycles5);

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

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

  /* Enable ADC1 reset calibration register */
  ADC_ResetCalibration(ADC1);
  /* Check the end of ADC1 reset calibration register */
  while(ADC_GetResetCalibrationStatus(ADC1));

  /* Start ADC1 calibration */
  ADC_StartCalibration(ADC1);
  /* Check the end of ADC1 calibration */
  while(ADC_GetCalibrationStatus(ADC1));

  /* Start ADC1 Software Conversion */
  ADC_SoftwareStartConvCmd(ADC1, ENABLE);
}
```

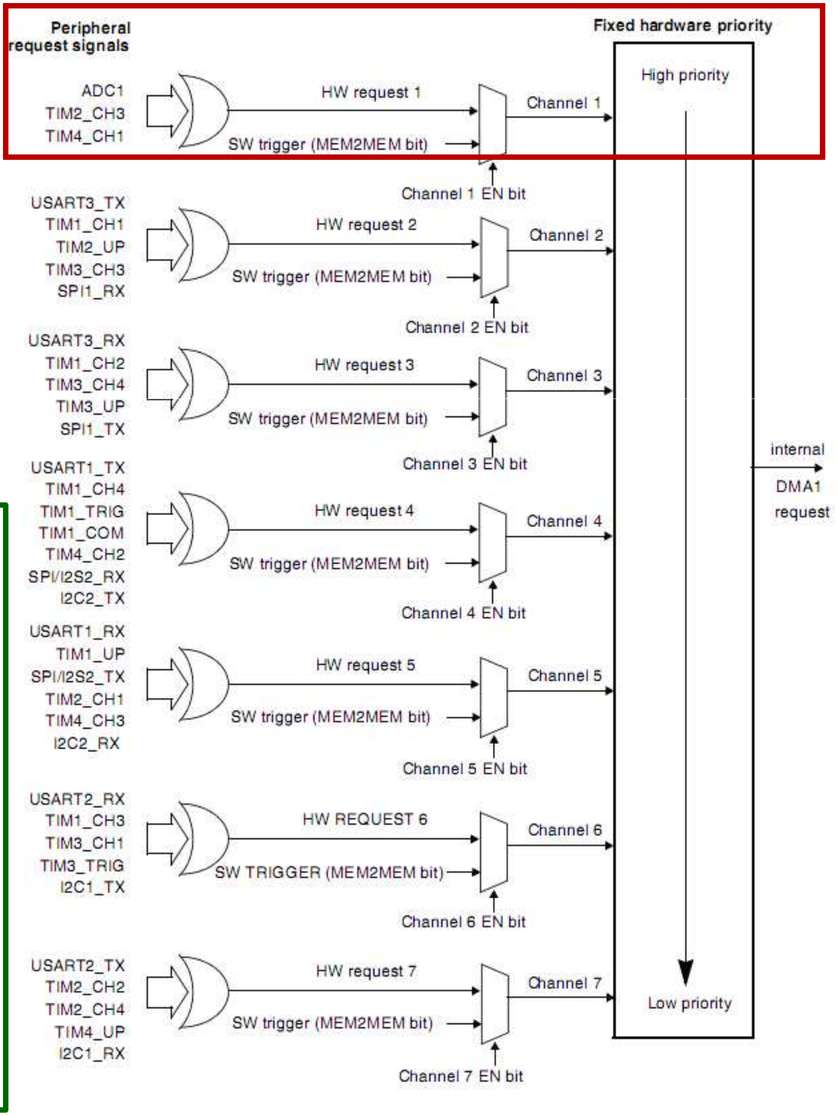
ARM DMA Control

DMA Block Diagram



```
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);
}
```

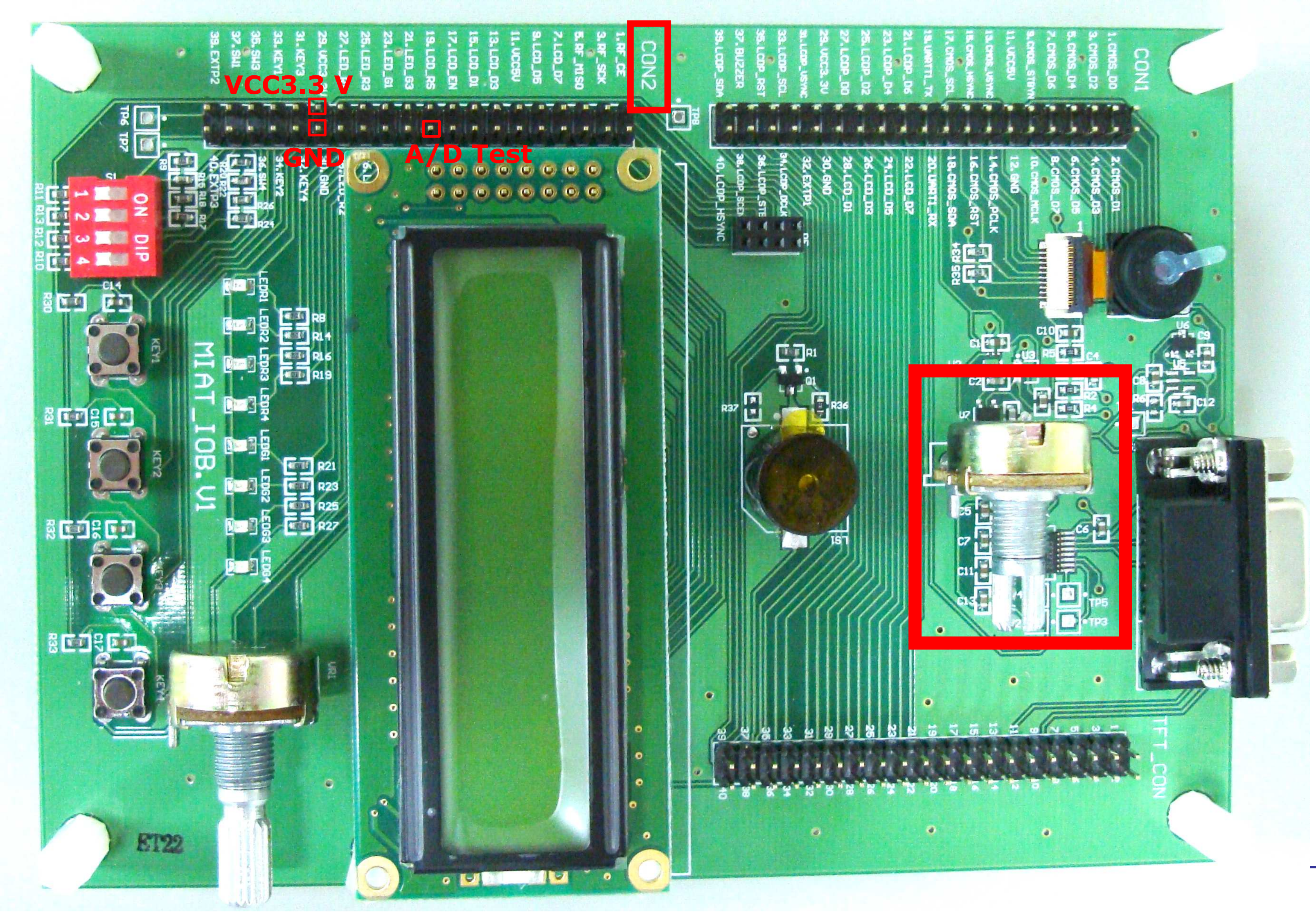
DMA1 request mapping00

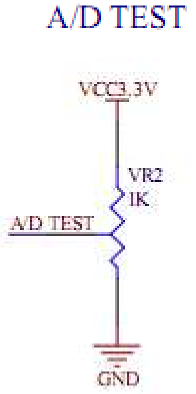


硬體電路配置

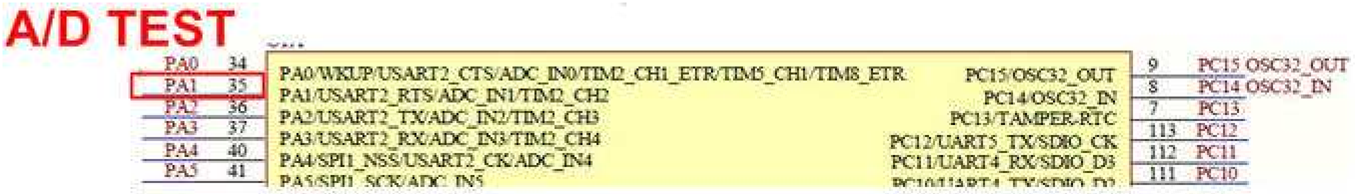
- □A/D Test 電路配置
- □Text LCD 電路配置
- □硬體接線示意圖

A/D Test 電路配置(1)

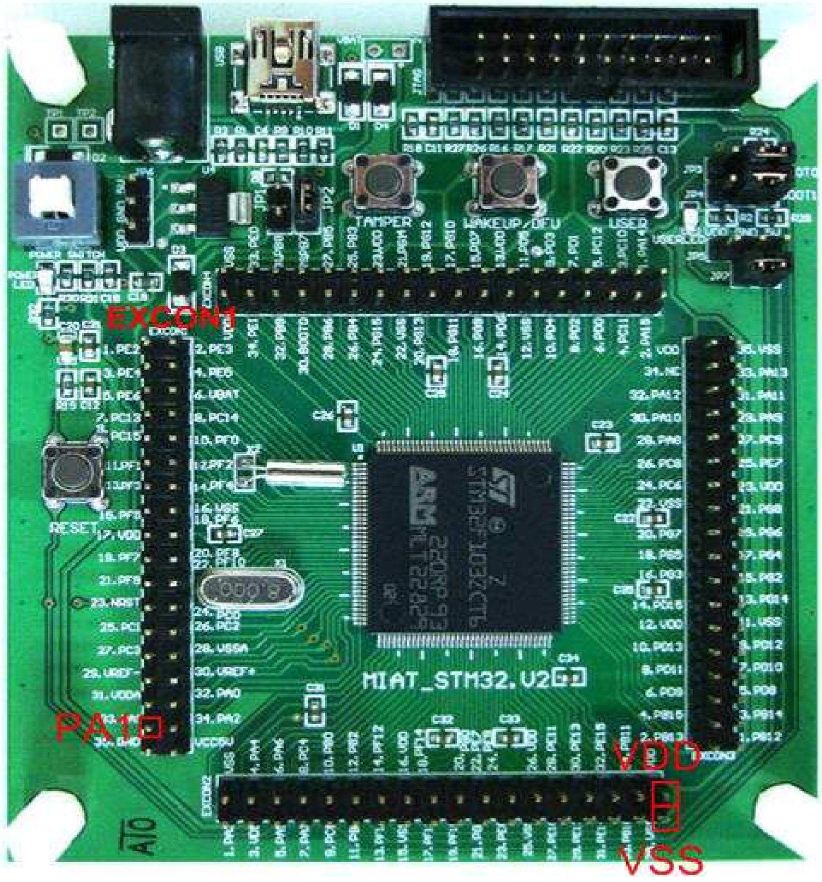




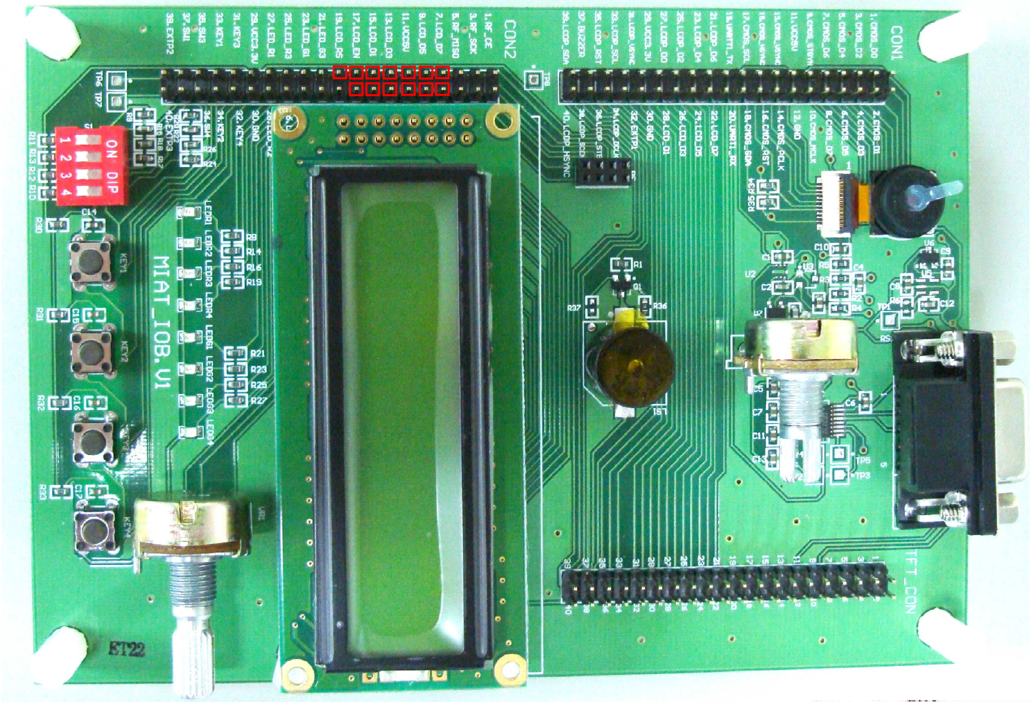
A/D Test 電路配置*(2)*



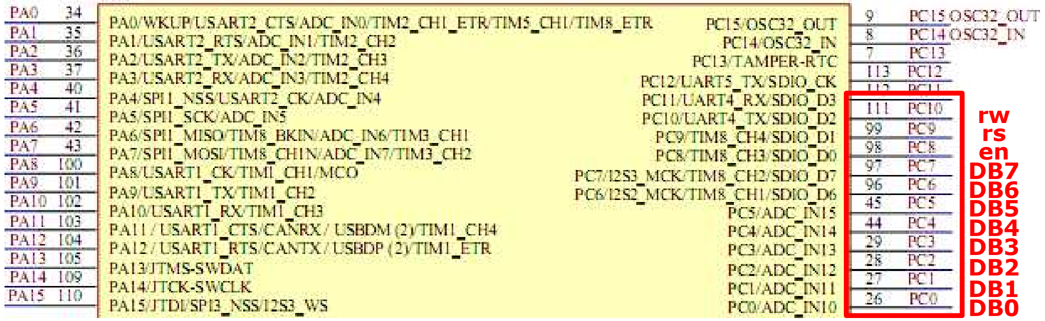
子版腳位名稱	子版腳位編號	母版腳位名稱	母版腳位編號
A/D Test	CON2.20	PA1	CON1.33
VCC3.3V	CON2.29	VDD	CON2.36
GND	CON2.30	VSS	CON2.35



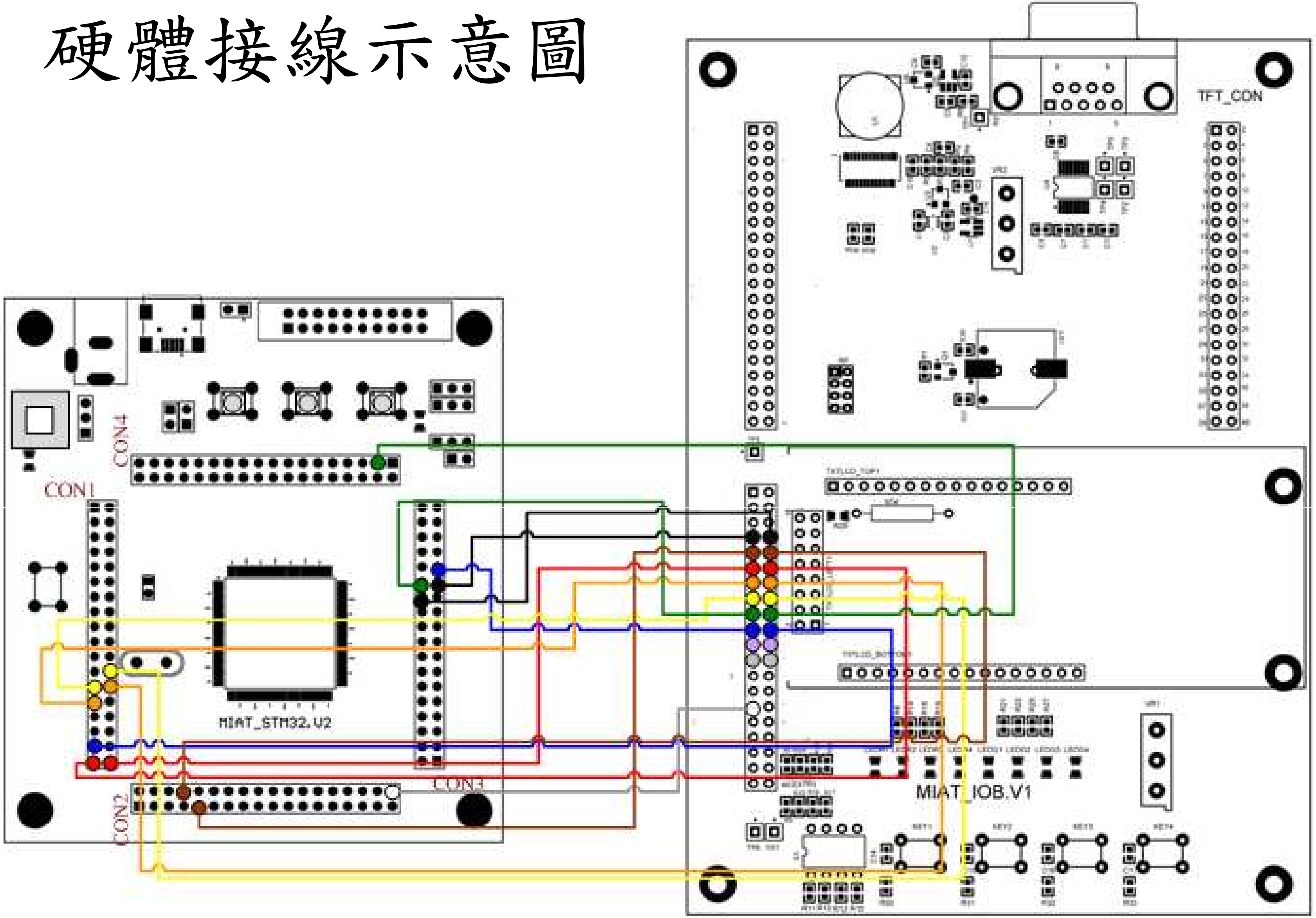
Text LCD 電路配置



子版腳位名稱	子版腳位編號	母版腳位名稱	母版腳位編號
VCC5V	CON2.11	VCC5V	CON1.36
GND	CON2.12	GND	CON1.35
LCD_D0	CON2.16	PC0	CON1.24
LCD_D1	CON2.15	PC1	CON1.25
LCD_D2	CON2.14	PC2	CON1.26
LCD_D3	CON2.13	PC3	CON1.27
LCD_D4	CON2.8	PC4	CON2.8
LCD_D5	CON2.7	PC5	CON2.9
LCD_D6	CON2.6	PC6	CON3.24
LCD_D7	CON2.5	PC7	CON3.25
LCD_EN	CON2.15	PC8	CON3.26
LCD_RS	CON2.17	PC9	CON3.27
LCD_R/W	CON2.18	PC10	CON4.3


PA0 34 PA0/WKUP/USART2_CTS/ADC_IN0/TIM2_CH1_ETR/TIM5_CH1/TIM8_ETR
PA1 35 PA1/USART2_RTS/ADC_IN1/TIM2_CH2
PA2 36 PA2/USART2_TX/ADC_IN2/TIM2_CH3
PA3 37 PA3/USART2_RX/ADC_IN3/TIM2_CH4
PA4 40 PA4/SPI1_NSS/USART2_CK/ADC_IN4
PA5 41 PA5/SPI1_SCK/ADC_IN5
PA6 42 PA6/SPI1_MISO/TIM8_BKIN/ADC_IN6/TIM3_CH1
PA7 43 PA7/SPI1_MOSI/TIM8_CH1N/ADC_IN7/TIM3_CH2
PA8 100 PA8/USART1_CK/TIM1_CH1/MCO
PA9 101 PA9/USART1_TX/TIM1_CH2
PA10 102 PA10/USART1_RX/TIM1_CH3
PA11 103 PA11 / USART1_CTS/CANRX / USBDM (2)/TIM1_CH4
PA12 104 PA12 / USART1_RTS/CANTX / USBDP (2)/TIM1_ETR
PA13 105 PA13/JTMS-SWDAT
PA14 109 PA14/JTCK-SWCLK
PA15 110 PA15/JTDI/SPI3_NSS/I2S3_WS
PC15/OSC32_OUT 9 PC15 OSC32_OUT
PC14/OSC32_IN 8 PC14 OSC32_IN
PC13/TAMPER-RTC 7 PC13
PC12/UART5_TX/SDIO_CK 113 PC12
PC11/UART4_RX/SDIO_D3 112 PC11
PC10/UART4_TX/SDIO_D2 111 PC10 rw
PC9/TIM8_CH4/SDIO_D1 99 PC9 rs
PC8/TIM8_CH3/SDIO_D0 98 PC8 en
PC7/I2S3_MCK/TIM8_CH2/SDIO_D7 97 PC7 DB7
PC6/I2S2_MCK/TIM8_CH1/SDIO_D6 96 PC6 DB6
PC5/ADC_IN15 45 PC5 DB5
PC4/ADC_IN14 44 PC4 DB4
PC3/ADC_IN13 29 PC3 DB3
PC2/ADC_IN12 28 PC2 DB2
PC1/ADC_IN11 27 PC1 DB1
PC0/ADC_IN10 26 PC0 DB0

硬體接線示意圖



實驗步驟

□ 軟體設置

□ 原始碼檔案瀏覽

□ 編譯燒錄程式並觀察結果

檔案目錄結構

<目錄>/檔案	說明
	<..\ADC2TextLCD\>
<project>	單元實驗Project目錄
<source>	程式碼目錄
<include>	引入檔目錄
<library>	函式庫目錄
<image>	燒錄配置檔目錄
	<..\ADC2TextLCD\image>
Lab.dfu	燒錄配置檔
	<..\ADC2TextLCD\source>
hw_config.c	硬體配置程式
lcd_func.c	Text LCD 控制程式

編譯燒錄程式並觀察結果

- □ 完成系統硬體設置之工作後，依 MIAT STM32 user manual (ch3, ch4)之操作指示，將編譯後的hex檔轉換為dfu
- □ 透過USB 燒錄dfu檔
- □ 旋轉VR2觀察電壓變化後A/D轉換後之結果

實作重點提示

- 觀察部份原始碼檔案，藉以了解程式架構
- 載入執行 Lab.dfu ，操作 VR2(可變電阻) 並觀察 Text LCD 是否有相對映之數字顯示

實際操作

操作時間~***(30min)***

習作及參考資料

- 習作

  Exe_1 : 改變Text LCD可顯示十進位輸出。

  Exe_2 : 將溫度感測轉換結果顯示於Text LCD第二列

- 參考資料

  [1] MIAT_STM32_user_manual_V1.00.pdf

  [2] STM32F10xxx reference manual.pdf

  [3] STM32F103XX firmware library.pdf

  [4] LMC-SSC2D16-01 Serial USER MANUAL(Text LCD datasheet)

Q & A

WU-YANG

Technology Co., Ltd.