市立楊梅高中107學年度優質化計畫

B-3-2子計畫

務實致用特色實作社群

嵌入式微控制器 STM32L053(ARM)

中華民國 108 年 4 月

目 錄

01-點亮一顆 LED 1
02-跑馬燈
03-使用中斷控制 LED16
04-利用 Buzzer 發出聲音 22
05-利用 PWM 驅動 LED
06-驅動三色 LED
07-驅動 WS2812B 彩色燈條 51
08-LCD 顯示器範例 55
STM32L053 Nucleo-64 線路圖

點亮一顆 LED

檔案名稱: 01-Start_Led.doc

首先建立一個目錄作為專案(Project)的目錄。本例設為「Start_Led-1」

開啟 STM32 的 CUBE 程式如下圖一:



本次使用的版本為 4.27.0,如下圖二:



點選 New Project 選項如下圖

STM32CubeMX Untitled	- D ×
	🐵 📘 🔽 🕨 🔀
New Project	STM32
	Cube
Load Project	
Help	
New Project	
MCU Selector Board Selector	
MCU Filters Part Number Search Core Core Check/Uncheck All ARM Cortex-M0 ARM Cortex-M0 ARM Cortex-M3	ST MCU Finder All STM32 & STM8 MCUs in one place
ARM Cortex-M33	Features Block Diagram Datasheet Docs & Resources Buy Start Project
ARM Cortex-M4	Block Diagram is not available
Series 🛞	MCU's List: 1223 items Times Trans Transmission of the second sec
Check/Uncheck All	☆ \$TM32F03 \$TM32F030 Active 0.542 LQFP48 32 kBytes 4 kBytes 39 48 MHz 0.0 0 0 0 0 ∧
STM32F1	x p1re5zer03 p1re3zer03 pcreve 0.037 pUpr/48 04 Ketytes 8 Ketytes 99 48 MH2 0.0 0
STM32F2	Characteristic STM32F03 Active 0.385 TSSOP20 16 kBytes 4 kBytes 15 48 MHz 0.0 <
STM32F3	x p1re52r03 p1re32r03 pctive 0.4/1 µ_UPr32 p2 ktytes 4 ktytes p2 ktytes p3 ktytes
□ S1M32F4 □ STM32F7 ✓	Contractors kithered been being the house here to house here here here here here here here he

如下圖點選 MCU 的 Core, Serial 及 Line:本範例使用 STM32L053R8T6

🧵 New Proje	ect
MCU Selector	Board Selector
-MCU Filter	rs
	י ש ו≣ ⊚
Part N	lumber Search 🛞
Q	~
Core	
Che	eck/Uncheck All ARM Cortex-M0+
Series	
Che	eck/Uncheck All
Line	
	eck/Uncheck All STM32L0x0 Value Line STM32L0x1 STM32L0x2 STM32L0x2

接著再點選 Package(LQFP64)如下圖

Package	8
Check/Uncheck All	
LQFP48	
🔽 LQFP64	
LQFP100	
TFBGA64	
UFBGA64	
UFBGA100	
UFQFPN48 (7x7mm)	

再點選 MCU 的 Mode,

М	CUsL	ist: 6 items		+ I	Display simila	u items										X
*	Pa	art No	Refe	Marketing	Unit Pri	Board	Pack	Flash		10	Freq.	GFX S	Н	М	S	TR
ź	STI 7	M32L053R6	STM32	Active	1.591		LQFP64	32 kBytes	8	51	32 MHz	0.0	0	þ	0	0
2	ז 🔉	M32L053R8	STM32	Active	1.707	NUCLEO-L053R8	QFP64	64 kBytes	8	51	32 MHz	0.0	0	0	0	0
Z	3 STI	M32L063R8	STM32	Active	1.846		LQFP64	64 kBytes	8	51	32 MHz	0.0	0	0	0	0
Z	Z STI	M32L073RB	STM32	Active	1.915		LQFP64	128 kB	2	51	32 MHz	0.0	0	0	0	0
Z	STI	M32L073RZ	STM32	Active	2.059	NUCLEO-L073RZ	LQFP64	192 kB	2	51	32 MHz	0.0	0	0	0	0
Z	ζ STI	M32L083RZ	STM32	Active	2.17		LQFP64	192 kB	2	51	32 MHz	0.0	0	0	0	0
<																>

點選完後再選右上 Start Project



MCU	Js List: 6 items		+	Display simila	ur items										X
*	Part No	Refe	Marketing	Unit Pri	Board	Pack	Flash		10	Freq.	GFX S	Н	М	S	TR
公	STM32L053R6	STM32	Active	1.591		LQFP64	32 kBytes	8	51	32 MHz	0.0	0	D	D	0
	STM32L053R8	STM32	Active	1.707	NUCLEO-L053R8	LQFP64	64 kBytes	8	51	32 MHz	0.0				0
	STM32L063R8	STM32	Active	1.846		LQFP64	64 kBytes	8	51	32 MHz	0.0	0	0	D	0
☆	STM32L073RB	STM32	Active	1.915		LQFP64	128 kB	2	51	32 MHz	0.0	0	0	0	0
☆	STM32L073RZ	STM32	Active	2.059	NUCLEO-L073RZ	LQFP64	192 kB	2	51	32 MHz	0.0	0	0	0	0
	STM32L083RZ	STM32	Active	2.17		LQFP64	192 kB	2	51	32 MHz	0.0	0	0	0	0
<															>

當點選後顯示如下圖,單須確認是否選到正確的晶片 MCU



說明:

- 1. 將滑鼠的指標移至晶片的上方或視窗區域,轉動滾輪放大或所小圖示。
- 2. 按住滑鼠右鍵後,移動滑鼠可以移動 MCU 的位置。

再本範例中欲使用 PA5 的接腳(Pin)作為輸出到 LED 方式:

- 1. 將滑鼠移置 PA5 後按下左鍵。
- 2. 點選倒數第三項左邊的空格「GPIO Output」

板上線路圖參考如下:



gnals Placement 🦃	۵ 🦉	- 🥑 🕂	Find			~	🗹 She	ow User	Label		(19	f	y		\times
on Calculator																
	PC2 PC3 VSSA VDDA PA0 PA1 PA2	PA3 VSS	VDD PA4	PAS DECEMBER 1	PA5 Rest_S1 LDC_IN COMP1 COMP2 COMP1 COMP2 FII SCC_62 SPIO_E SPIO_E SPIO_S SPIO_S SPIO_E SPIO_E	tate 15 _INM _INM CH CH CH CH CH CH CH CH CH CH	2L (5 QF 7	3R81 64	pB1	PB2	PB10	PB11	NSS	VDD	PC8 PC7 PC6 PB15 PB14 PB13 PB12	
	Mcu STM32L053	R6Tx		i L	Package ,QFP64	•				Req Non	uired l ie	Periphe	erals			1

如下圖示



點選左上方的 Project → Setting:以便專案的設定



顯示如下圖,並參考如下填入:

- 1. Project Name
- 2. Project Location
- 3. Tool Chain → MDK-ARM V5
- 4. OK(右下角)

Project Settings	×
Project Code Generator Advanced Settings	^
Project Settings	
Project Name	
Start_Led-1	
Project Location	
J/STM32F相關/STM32L053教學資料/Examples/	Browse
Application Standum	
Basic Do not separate the main()	
Toolchain Folder Location	
J.STM32F相關STM32L053教學資料ExamplesStart_Led-1\	
Toolchain / IDE	
MDK-ARM V5 Generate Under Root	
Linker Settings	
Minimum Heap Size 0x200	
Minimum Stack Size 0x400	
Mcu and Firmware Package	
Mcu Reference	
STM32L053R8Tx	
Firmware Package Name and Version	
STM32Cube FW_L0 V1.10.0	
	×
	Ok

本範例為初次使用自動產生程式碼方式,因此使用其預設值即可。

接下來可以利用 Cube 來產生報告。

點選 Project→Generate Report 如下圖:

這個 Report 是 PDF 檔,可以檢視我們設定是否正確,最好在最後設定完成後檢視。



接後點選 Project→Generate Code 來自動產生預設的程式碼



在產生程式碼後開啟後如下圖:

🔢 J:\STM32F--相關\STM32L053教學資料\Examples\Start_Led-1\MDK-ARM\Start_Led-1.uvprojx - μVision

根據 HAL 驅動說明書找到我們需要的 API, 然後根據說明添加代碼: (根據說明得到使 PA5

輸出高電平的代碼為 HAL_GPIO_WritePin (GPIOA, GPIO_PIN_5, GPIO_PIN_SET);)

25.2.8 HAL_GPIO_WritePin

Function Name	void HAL_GPIO_WritePin (GPIO_TypeDef * GPIOx, uint16_t GPIO_Pin, GPIO_PinState PinState)
Function Description	Sets or clears the selected data port bit.
Parameters	 GPIOx: where x can be (AK) to select the GPIO peripheral for STM32F429X device or x can be (AI) to select the GPIO peripheral for STM32F40XX and STM32F427X devices. GPIO_Pin: specifies the port bit to be written. This parameter can be one of GPIO_PIN_x where x can be (015). PinState: specifies the value to be written to the selected bit. This parameter can be one of the GPIO_PinState enum values: GPIO_PIN_RESET: to clear the port pinGPIO_PIN_SET: to set the port pin
Return values	None
Notes	 This function uses GPIOx_BSRR register to allow atomic read/modify accesses. In this way, there is no risk of an IRQ occurring between the read and the modify access.

輸入程式碼如下:

1. 在 while(1)下方:

/*USER CODE BEGIN 3*/及/* USER CODE END 3 */中間開始輸入,以避免程式重新產生時被覆 蓋。

2. 在程式輸入時會自動出現關鍵字以便選擇。(MDK 5.xx 版後)

```
/* Infinite loop */
         /* USER CODE BEGIN WHILE */
101
102
         while (1)
103 白
104
105
        /* USER CODE END WHILE */
106
107
        /* USER CODE BEGIN 3 */
          HAL_GPIO_WritePin(GPIOA. GPIO_PIN_5, GPIO_PIN_RESET): //Low Active ON(板上LED "LED2"-->Active HIGH
HAL_Delay(1000): //1秒差,1000個 clock
HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET);
108
109
110
111
          HAL_Delay(1000);
112
113
        /* USER CODE END 3 */
 14
```

(程式碼會附在後面)

程式碼:

HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET); HAL_Delay(1000); HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET); HAL_Delay(1000);

注意:

- 我們編寫程式時,需將程式碼放在/*USER CODE BEGIN N */與 /*USER CODE END N */內, 以免在 CUBE 產生程式碼時被覆蓋。
- 2. 按下 Reset 按鈕開始執行程式,可以看到 LED2 以每秒時間閃爍一次。
- 3. 亦可事先在 Keil 中設定燒錄完成後自動執行程式。

方式:在 MDK 的 Option for Target...(仙女棒) 按鈕點選。

🖁 J:\STM32F相關\STM32L053教學資料\Examples\Start_Led-1\MDK-ARM\Start_Led-1.uvprojx - μVisi								
File Edit View Project Flash Debug	Peripherals Tools SVCS Window Help							
n 💕 🛃 🥔 👗 🖻 🛍 🔊 (°	← → 陀 🎨 🔥 限 💷 🕼 //絵 💆 PFOUT							
🧇 🔛 🕮 🥔 🔜 🔤 Start_Led-1	V 💦 🛔 🗟 🗇 🏟							
Project 🛛 📮 🔀	main.c 🔆 Options for Target							
□ ♣ Project: Start_Led-1	132 Configure target options AHB and AP							
🖻 ᇶ Start_Led-1	133 - */							
Application/User	134 RCC_OscInitStruct.OscillatorType = 1							
main c	135 RCC_OscInitStruct.MSIState = RCC_							
	136 RCC_OscInitStruct.MSICalibrationVa							
	137 RCC_OscInitStruct.MSIClockRange =							
🖃 📄 stm32l0xx_hal_msp.c	138 RCC_OscInitStruct.PLL.PLLState = R							
🕮 🛅 Drivers/STM32L0vy HAL Driv	139 if (HAL RCC OscConfig(& BCC Os							

點選 Option for Target...(仙女棒)後即可開啟另一視窗,如下:

接者點選 Debug 後,注意在紅框中需選設在 ST-Link Debugger 或相容的 Tool。

確認後再點選右邊的	settings	選項	0
-----------	----------	----	---

Options for Target 'Start_Led-1'	×					
Device Target Output Listing User C/C++ A	sm Linker Debug Utilities					
C Use Simulator <u>with restrictions</u> Settings	• Use: ST-Link Debugger Settings					
✓ Load Application at Startup ✓ Run to main() Initialization File:	Load Application at Startup Run to main() Initialization File:					
Restore Debug Session Settings	Restore Debug Session Settings					
CPU DLL: Parameter: SARMCM3.DLL -REMAP	Driver DLL: Parameter: SARMCM3.DLL					
Dialog DLL: Parameter: DARMCM1.DLL pCM0+	Dialog DLL: Parameter: TARMCM1.DLL -pCM0+					
Manage Component Viewer Description Files						
OK Ca	ncel Defaults Help					

點選上示的選項後,會出現如下示的視窗。

接著點選紅框中的 Flash Download 選項,接著點選下面紅框中的 Reset and Run 選項,這樣會在程式下載完後自動執行,不用再按 Reset 鍵。

bug Trace Flash Download Download Function Carase Full Ch Erase Full Ch Carase Sector Do not Erase	ip ⊽ Program s ⊽ Verify e ⊽ Reset a	n nd Run	1 for Algorithm	Size: Ox	0800	
Programming Algorithm Description STM32L0 64KB Flash	Device Size 64k	Device Type On-chip Flash	Address F 08000000H - (Range 0800FFFFH		
		S	tart:	Size:		
	A	i dd Ren	nove			

跑馬燈

檔案名稱: 02-ledx8-shift.doc

本例可參考前例的練習,說明方式如下:功能要求使8顆 LED 由右至左依序點亮,重複執行。 如下圖:

程式簡單的方式如下:

HAL_GPIO_WritePin() → 寫入一個值到輸出的腳位。HAL 是指使用 HAL Library 庫。
GPIOC → 是指 ARM 的輸出入腳位群 "C",本例即 PCO~PC7。
GPIO_PIN_0 → 是指腳位 0,其餘相同。
GPIO_PIN_SET → 是指 Logic High (1)。
GPIO_PIN_RESET →是指 Logic Low (0)。
HAL Delay(100) → 使用 HAL Library 的 Delay 函數。括號內的數值是 ms 單位。

將以下的程式碼複製於主程式 main.c

/* Infinite loop */

/* USER CODE BEGIN WHILE */内的 while 函數內。

CODE :如下

/* LED Shift 1 --> Code */

HAL_GPIO_WritePin(GPIOC, GPIO_PIN_0, GPIO_PIN_SET);

HAL_Delay(100);

HAL_GPIO_WritePin(GPIOC, GPIO_PIN_1, GPIO_PIN_SET); HAL_Delay(100);

HAL_GPIO_WritePin(GPIOC, GPIO_PIN_2, GPIO_PIN_SET); HAL_Delay(100);

HAL_GPIO_WritePin(GPIOC, GPIO_PIN_3, GPIO_PIN_SET); HAL_Delay(100);

HAL_GPIO_WritePin(GPIOC, GPIO_PIN_4, GPIO_PIN_SET); HAL_Delay(100);

HAL_GPIO_WritePin(GPIOC, GPIO_PIN_5, GPIO_PIN_SET); HAL_Delay(100);

HAL_GPIO_WritePin(GPIOC, GPIO_PIN_6, GPIO_PIN_SET); HAL_Delay(100);

HAL_GPIO_WritePin(GPIOC, GPIO_PIN_7, GPIO_PIN_SET); HAL_Delay(100);

HAL_GPIO_WritePin(GPIOC, GPIO_PIN_0, GPIO_PIN_RESET); HAL_Delay(100);

HAL_GPIO_WritePin(GPIOC, GPIO_PIN_1, GPIO_PIN_RESET); HAL_Delay(100);

HAL_GPIO_WritePin(GPIOC, GPIO_PIN_2, GPIO_PIN_RESET); HAL_Delay(100);

HAL_GPIO_WritePin(GPIOC, GPIO_PIN_3, GPIO_PIN_RESET); HAL_Delay(100);

HAL_GPIO_WritePin(GPIOC, GPIO_PIN_4, GPIO_PIN_RESET); HAL_Delay(100);

HAL_GPIO_WritePin(GPIOC, GPIO_PIN_5, GPIO_PIN_RESET); HAL_Delay(100);

HAL_GPIO_WritePin(GPIOC, GPIO_PIN_6, GPIO_PIN_RESET); HAL_Delay(100);

HAL_GPIO_WritePin(GPIOC, GPIO_PIN_7, GPIO_PIN_RESET); HAL_Delay(100);

使用中斷控制 LED

檔案名稱: 03-使用中斷控制 LED

一、API 說明

HAL 庫一共包含如下 6 個 IO 操作函數:

- 1、讀取某個 Pin(接腳)的電位狀態: HAL_GPIO_ReadPin()
- 2、寫入某個 Pin(接腳)的電位狀態: HAL GPIO WritePin()
- 3、翻轉某個 Pin(接腳)的電位狀態: HAL_GPIO_TogglePin()
- 4、鎖定某個 Pin(接腳)的配置狀態(直到下次重定): HAL_GPIO_LockPin()
- 5、外部中斷服務函數: HAL_GPIO_EXTI_IRQHandler()

6、外部中斷回呼函數: HAL_GPIO_EXTI_Callback()

具體使用方法參見 User Manual-STM32L0 HAL and Low Layer drivers。

2.11.2 GPIOs

GPIO HAL APIs are the following:

- HAL_GPIO_Init() / HAL_GPIO_DeInit()
- HAL_GPIO_ReadPin() / HAL_GPIO_WritePin()
- HAL_GPIO_TogglePin ().

46/1466

DocID026232 Rev 6

เธิญเรเติเร.

- 5. To get the level of a pin configured in input mode use HAL_GPIO_ReadPin().
- 6. To set/reset the level of a pin configured in output mode use HAL_GPIO_WritePin()/HAL_GPIO_TogglePin().
- 7. To lock pin configuration until next reset use HAL_GPIO_LockPin().
- During and just after reset, the alternate functions are not active and the GPIO pins are configured in input floating mode (except JTAG pins).

二、GPIO 使用示例

示例功能:使用按键(PC13)翻轉 LED(PA5)電位狀態。

1、使用 STM32CubeMX 配置好 Pin(接腳)功能以及巢狀中斷控制器:

配置 LED 燈 Pin(接腳)為輸出模式

	5	<u>e</u>									
	PA	PAG	PA7	PC ²	PC	PBC	PB 1	PB2	PB 1		
					2			11.0	ا <mark>ت</mark>		
	ନ୍ଧ	(2	1)-P/	45: ²							
I	ĕ	 GPIO_Output (no mode) 									
	ξl	<(Ctrl+	Click	> To	Sho	w Alt	ernat	te(s)		

將按鍵 Pin(接腳)掛接到外部中斷 13 號線上

點開 GPIO 設定按鈕

選擇 GPIO 模式為上升緣觸發的外部中斷模式

(何謂上升縁觸發、何謂外部中斷,以及其他模式,請自行參考其他有關資料。)

💿 Pin Config	uration						×
GPIO							
Search Signal Search (Cr	ls					Show only M	odified Pins
Pin Name	Signal on	GPIO outp	GPIO mode	GPIO Pull	Maximum o	User Label	Modified
PA5	n/a	Low	Output Pus	No pull-up	Low		
PC13-ANTI_TAMP	n/a	n/a	External I	No pull-up	n/a		
PC13-ANTI_TAN	MP Configuration	n :	-				
GPIO mode		External	l Interrupt Mo	de with Rising	edge trigger d	etection	~
GPIO Pull-up	/Pull-down	External External External External	Interrupt Mod Interrupt Mod Interrupt Mod Event Mode wi	le with Rising o le with Falling le with Rising/H th Rising edge	edge trigger de edge trigger d Falling edge tr trigger detect	tection letection rigger detection	PIO_MODE_IT_
User Label		External External	Event Mode wi Event Mode wi	th Falling edge th Rising/Falli	e trigger detec ing edge trigge	r detection	
Group By	IP				Apply	Ok	Cancel

點選巢狀中斷控制器(NVIC)設定按鈕

勾選置能外部中斷 4~15, 並且生成相應初始化程式碼

۲	NVIC Configuration				×
✓ NVIC ✓ Code generation					
			_ Sort	by Premption Priority and Sub Pr	rority
Search (Crtl+F)	•		_ Sho	w only enabled interrupts	
Interrupt Table		Enabled		Preemption Priority	
Non maskable interrupt		~		0	~
Hard fault interrupt		~		0	
System service call via SWI instruction		~		0	
Pendable request for system service		~		0	
Time base: System tick timer		\checkmark		0	
PVD interrupt through EXTI line 16				0	
Flash and EEPROM global interrupt				0	
RCC and CRS global interrupt				0	
EXTI line 4 to 15 interrupts		 ✓ 		0	

0	NVIC Configuration		×
🖋 NVIC 🗹 Code generation			
Enabled interrupt table	Select for init sequence ordering	🖌 Generate IRQ handler	
Non maskable interrupt		✓	
Hard fault interrupt		✓	
System service call via SWI instruction		✓	
Pendable request for system service		✓	
Time base: System tick timer		✓	
EXTI line 4 to 15 interrupts	✓	✓	
Interrupt unmasking ordering table (interrupt init co	de is moved after all the peripheral init code)		~
Rank Inter	ant name		
1 EVTI	line 4 to 15 interrupts		
I BAII	me 4 to 15 menubs		- ^

2、在生成的工程中的對應位置定義外部中斷回 Call 函數:

```
/* USER CODE BEGIN 4 */
void HAL_GPIO_EXTI_Callback (uint16_t GPIO_Pin)
{
    if(GPIO_Pin == GPIO_PIN_13)
        HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5);
}
```

/* USER CODE END 4 */

3、编譯、下載程式至開發板,並且按下 RESET 按鈕。

三、程式碼分析

1、在開發板的開機檔案 startup_stm32l053xx.s 中將 EXTI15_10_IRQHandler 函數註冊為外部中斷 10~15 號線的中斷服務函數,當外部中斷 10~15 號線產生外部中斷時由硬體調用 EXTI15_10_IRQHandler 函數,中斷當前運行的程式,CPU 開始執行中斷服務函數內的程式,執行完之後繼續運行中斷前的程式;

2、因為 STM32L053 的硬體結構決定了外部中斷 10~15 號線共用一個中斷向量,因此只能 註冊一個中斷服務函數,而 HAL 框架使用 HAL_GPIO_EXTI_IRQHandler()函數和 HAL_GPIO_EXTI_Callback()函數使使用者看來每個外部中斷線都擁有自己的中斷服務函數 (後面會講解這兩個函數);

3、在 stm32l0xx_it.c 檔中定義了 EXTI15_10_IRQHandler 函數,該函式呼叫了 HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_13);語句;

4、HAL_GPIO_EXTI_IRQHandler()在 stm32l0xx_hal_gpio.c 檔中定義了,該函數判斷外部中斷是由哪一號外部中斷線產生的,並且清除中斷掛起寄存器中的對應位,然後調用外部中斷回呼函數 HAL_GPIO_EXTI_Callback(),並將產生外部中斷的中斷線作為參數傳遞給外部中斷回呼函數 HAL_GPIO_EXTI_Callback();

5、而上面的所有工作都由 STM32CubeMX 幫我們做好了,我們只需要在 main.c 檔中重定 義 HAL_GPIO_EXTI_Callback()函數就行了;

6、因為所有外部中斷都會調用 HAL_GPIO_EXTI_Callback()函數,所以我們需要在

HAL_GPIO_EXTI_Callback()函數內部根據輸入的 GPIO_Pin 參數判斷是哪一號外部中斷線的 產生了外部中斷,然後根據不同的外部中斷執行不同的程式碼;

7、因此我們使用語句 if(GPIO_Pin == GPIO_PIN_13)判斷該外部中斷是否是由外部中斷 13 號線產生的,然後執行 HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5); 語句翻轉 LED 燈的電 位狀態;

提示:我們不應該在中斷服務函數或者中斷回 Call 函數內執行過多的程式碼,這樣做是為了 最大程度地減少中斷嵌套。有關中斷嵌套和中斷優先順序的內容請自行參考相關資料,中斷優 先順序(不僅限於外部中斷)可以在 STM32CubeMX 中的 NVIC Configuration 中設置。

利用 Buzzer 發出聲音

檔案名稱: 04-Buzzer.doc

本例將利用 PC13 的 GPIO PIN 接一個按鈕(板子上),使得 Buzzer 發聲。 PC13 的接腳為輸入,因此須利用內部的 Pull up 電阻將電位提升,以便使得此按鈕為 Low Active(Low 動作)。

首先建立一個目錄作為專案(Project)的目錄。本例設為「3_Buzzer1」。

開啟 STM32 的 CUBE 程式如下圖一:

本次使用的版本為 4.27.0,如下圖二:

點選 New Project 選項如下圖

MCU Filters Part Number Search ST MCU Finder All STM322 & STM8 Core St MCUsine & All ARM Cortex-M0 Stat ARM Cortex-M0 Stat ARM Cortex-M1 Stat ARM Cortex-M1 Stat Stries Notice Check/Uncheck All Detached Doc: & Resources Buy Stat MCUs List: 1223 items Display simular items MCUs List: 1223 items Display simular items Part No Reference Marketing Unit Price for Boa. Pack Fiash RM 10 Freq. GFX S H M S STM32P0 STM32P03 Active DG57 L.QP748 St Habytes	CU Filters													
Core Check/Uncheck All Check/Uncheck All Striss Check/Uncheck All Block Diagram Detasheet Docs & Resources Buy Start P. Series Check/Uncheck All Display similar items MCU's List 1223 items Display similar items * Part No Reference Marketing Unit Price for Boa Plack Flash RAM IO Freq. GFX S H M S Striss * Part No Reference Marketing Unit Price for Boa Plack Flash RAM IO Freq. GFX S H M S * Part No Reference Marketing Unit Price for Boa Plack Flash RAM IO Freq. GFX S H M S * Part No Reference Marketing Unit Price for Boa Plack Flash RAM IO Freq. GFX S H M S * Part No Reference Marketing Unit Price for Boa Pl	Part Number Search			May Market	Asy an 1 million MA32 boords	0 0 1 1 sold 577		A	ST N		Fil & S	nde	er 18	
ARM Cortex-M4 Block Diagram is not available MCU's List: 1223 items * Part No Reference MCU's List: 1223 items * Part No Reference MCU's List: 1223 items * Part No Reference MCU's List: 1223 items * STM32F03 STM32F03 Active 0.542 LOFF48 122 kBytes 4 kBytes 19 # STM32F03 STM32F03 Active 0.657 LOFF48 10 KBytes 10 KBytes # STM32F03 STM32F03 Active 0.657	Core (Check/Uncheck All ARM Cortex-M0 ARM Cortex-M0+ ARM Cortex-M3 ARM Cortex-M3	Features	Elock D	A CONTRACTOR OF A CONTRACTOR O	Datasheet		Docs & Reg	JUDGES		Buy		57	Start Pr	piect
ARM Cortex:M7		1000000	Bloc	k Diagram is not av	ailable		Dots to reca			200			000111	Jeer
Check/Uncheck All * Part No Reference Marketing Unit Price for Boa Pack Flash RAM IO Freq. GFX S H M S Check/Uncheck All STM32F03 STM32F03 Active 0.542 LQFP48 32 kBytes 4 kBytes 39 48 MHz 0.0 0	AKM Cortex-M4													
Child Markan ☆ \$TM32F03 \$TM32F03 Active 0.542 LQFF48 32 kBytes 4 kBytes 39 48 MHz 0.0 0	ARM Cortex-M4	MCUs List: 1223 items	4	- Display similar items			1.1				1			[
L 🖓 🖓 🖓 🖓 🖓 🖓 🖓 🖓 🖓 🖓 🖓 🖓 🖓	ARM Cortex-M4 ARM Cortex-M7 Sories	MCUs List: 1223 items * Part No Reference	nce Marketing	Display similar items Unit Price for	Boa	Pack Fla	ash RAM	10	Freq.	GFX S	Н	М	S	[2 TR
	AKM Cortex-M4 ARM Cortex-M7 Series	MCUs List: 1223 items * Part No Referent ☆ STM32F03 STM32F0	nce Marketing 130 Active	 Display similar items Unit Price for 0.542 	Boa I	Pack Fla ,QFP48 32 1	ish RAM Bytes 4 kBytes	IO : 39	Freq. 48 MHz (GFX S 0.0	н 0	M 0	S 0	[2 TR 0
□ STM32F1 10 LQFP48 256 kB 32 kBytes 37 48 MHz 0.0 0 0 0	ARM Cortex:M4 ARM Cortex:M7 Series	MCUs List: 1223 items MCUs List: 1223 items * Part Ño Referen ☆ STM32P03 STM32P ☆ STM32P03 STM32P	nce Marketing 030 Active 130 Active	Display similar items Unit Price for 0.542 0.657	Boa I L	Pack Fla QFP48 32 1 QFP48 64 1	ish RAM Bytes 4 kBytes Bytes 8 kBytes	IO : 39 : 39	Freq. 48 MHz (48 MHz (GFX S 0.0 0.0	H 0 0	M 0 0	S 0 0	TR
□ STM32F2 □ ☆ STM32F030 & Ctive 0.385	ARM Cortex:M4 ARM Cortex:M7 Series Check/Uncheck All STM32P0 STM32F1	MCUs List: 1223 items * Part Ño Referen ☆ \$TM32P03\$TM32P ☆ \$TM32P73\$TM32P ☆ \$TM32P73\$TM32P ☆ \$TM32P73\$TM32P	nce Marketing D30 Active D30 Active D30 Active	 Display similar items Unit Price for 0.542 0.657 1.0 	Boa L	Pack Fla .QFP48 321 .QFP48 641 .QFP48 256	ash RAM Bytes 4 kBytes Bytes 8 kBytes kB 32 kByte	IO : 39 : 39 : 39 :: 37	Freq. 48 MHz (48 MHz (48 MHz (GFX S 0.0 0.0 0.0	H 0 0	M 0 0	S 0 0	2 TR 3
STM 32F2 2 / KBytes /	ARM Cortex-M4 ARM Cortex-M7 Series Check/Uncheck All STM32F0 STM32F1 STM32F2	MCU's List: 1223 items Part No Reference STM32P03 STM32P STM32P03 STM32P STM32P03 STM32P STM32P03 STM32P	nce Marketing 130 Active 130 Active 130 Active 130 Active	Display similar items Unit Price for 0.542 0.657 1.0 0.385 0.455	Boa L L L L I I I	Pack Fla .QFP48 32 1 .QFP48 64 1 .QFP48 256 .SSOP20 16 1	ish RAM Bytes 4 kBytes Bytes 8 kBytes kB 32 kByte Bytes 4 kBytes	IO 39 39 39 37 15	Freq. 48 MHz (48 MHz (48 MHz (48 MHz (GFX S 0.0 0.0 0.0 0.0	H 0 0 0 0	M 0 0 0	S 0 0 0	TR 0 0
STRUE	ARM Cortex-M4 ARM Cortex-M7 Series Check/Uncheck All STM32P0 STM32P1 STM32P2 STM32P3	MCUs List: 1223 items * Part Ño Referent ☆ STM32F03, STM32F0 ☆ STM32F03, STM32F0	nce Marketing 130 Active 130 Active 130 Active 130 Active 130 Active	Display similar items Unit Price for 0.542 0.657 1.0 0.385 0.471	Boa L L L L L L L L	Pack Fla .QFP48 32 1 .QFP48 64 1 .QFP48 256 ISSOP20 16 1 .QFP32 32 1	ish RAM Bytes 4 kBytes Bytes 8 kBytes kB 32 kByte Bytes 4 kBytes Bytes 4 kBytes	IO 39 39 37 15 25	Freq. 48 MHz (48 MHz (48 MHz (48 MHz (48 MHz (GFX S 0.0 0.0 0.0 0.0 0.0	H 0 0 0 0 0	M 0 0 0 0	S 0 0 0 0	[2 TR 0 0 0

如下圖點選 MCU 的 Core, Serial 及 Line:本範例使用 STM32L053R8T6

😻 New Project	
MCU Selector Board Selector	
MCU Filters	^
⊠ ⊍ ≔ ⊚	
Part Number Search 🛞	
۹	
Core	
Check/Uncheck All	
Series 🔊	
Series	
Series (Scheck/Uncheck All	
Series ⊗ Check/Uncheck All STM32L0 Line ⊗	
Series ⊗ Check/Uncheck All STM32L0 Line ⊗ Check/Uncheck All	
Series Check/Uncheck All ✓ STM32L0 Line Check/Uncheck All ☐ STM32L0x0 Value Line	
Series Check/Uncheck All ☑ STM32L0 Line Check/Uncheck All □ STM32L0x0 Value Line □ STM32L0x1 □ STM32L0x1 □ STM32L0x1	
Series	

接著再點選 Package(LQFP64)如下圖

再點選 MCU 的 Mode,

MC	Us List: 6 items		+ I	Display simila	u items										X
*	Part No	Refe	Marketing	Unit Pri	Board	Pack	Flash		10	Freq.	GFX S	Н	М	S	TR
ŵ	STM32L053R6	STM32	Active	1.591		LQFP64	32 kBytes	8	51	32 MHz	0.0	0	0	0	0
	STM32L053R8	STM32	Active	1.707	NUCLEO-L053R8	QFP64	64 kBytes	8	51	32 MHz	0.0	0	0	0	0
ŝ	STM32L063R8	STM32	Active	1.846		LQFP64	64 kBytes	8	51	32 MHz	0.0	0	0	0	0
ŝ	STM32L073RB	STM32	Active	1.915		LQFP64	128 kB	2	51	32 MHz	0.0	0	0	0	0
ŝ	STM32L073RZ	STM32	Active	2.059	NUCLEO-L073RZ	LQFP64	192 kB	2	51	32 MHz	0.0	0	0	0	0
ŝ	STM32L083RZ	STM32	Active	2.17		LQFP64	192 kB	2	51	32 MHz	0.0	0	0	0	0
<					-		-								>

點選完後再選右上 Start Project

Visite Unit Price for 10kU (US\$): 1.707 Visite for 10kU (US\$): 1.707 Board: NUCLEO-L053R8 LQFP64 The ultra-low-power STM32L053x6/8 microcontrollers incorporate the connectivity power of the universal serial bus (USB 2.0 crystal-less) with the high-performance Arm® Cortex®-M0+ 32-bit RISC core operating at a 32 MHz frequency, a memory protection unit (MPU), high-speed embedded memories (up to 64 Kbytes of Flash program memory, 2 Kbytes of data EEPROM and 8 Kbytes of RAM) plus an extensive range of enhanced I/Os and peripherals. The STM32L053x6/8 devices provide high power efficiency for a wide range of performance. It is achieved with a Image: Peatures Block Diagram Datasheet Docs & Resources Buy Start Project VIS List: 6 items Display similar items Part No Refe Marketing Unit Pri Board Pack F132L053R6 \$TM32 Active 1.591 LQFP64 \$12 kBytes 8 51 STM32L053R8 \$TM32 Active 1.707 NUCLEO-L053R8 LQFP64 STM32L053R8 \$TM32 Active 1.915 LQFP64 128 kB 51 22 MHz 0.0 0 0 <	Image: STM3210 Image: Active Product is in mass production Unit Price for 10kU (US\$): 1.707 Image: Board: NUCLEO-L053R8 Image: LQFP64 Image: Stm121 Description Image: Stm121 Description Image: Stm121 Description Image: Stm121 Description Image: LQFP64 Image: Stm121 Description Image: Stm121 Description Image: Stm121 Description Image: Stm121 Description Image: LQFP64 Image: Stm121 Description Image: Stm1221 Description<	bus m	LQFP64 versal serial a 32 MHz lash prograr is and bieved with	LQFP64 e universal sei ng at a 32 MHz s of Flash prog ed I/Os and	for 10kU (US\$) : 1.707 <u>JCLEO-L053R8</u> e the connectivity power of the unive M0+ 32-bit RISC core operating at a ed memories (up to 64 Kbytes of File an extensive range of enhanced I/Os	nit Price for 10 bard: <u>NUCLE(</u> borporate the bortex®-M0+ 3 mbedded m) plus an ext	duction ocontrollers ir mance Arm® I), high-speed Kbytes of RA	ACTIVE Active Product is in mass prod STM32L053x6/8 micro with the high-perform protection unit (MPU) data EEPROM and 8	The ultra-low-power S (USB 2.0 crystal-less) frequency, a memory memory, 2 Kbytes of peripherals.
Active Product is in mass production Board: NUCLEO-L053R8 LOFP64 The ultra-low-power STM32L053x6/8 microcontrollers incorporate the connectivity power of the universal serial bus (USB 2.0 crystal-less) with the high-performance Arm® Cortex®-M0+ 32-bit RISC core operating at a 32 MHz frequency, a memory protection unit (MPU), high-speed embedded memories (up to 64 Kbytes of Flash program memory, 2 Kbytes of data EEPROM and 8 Kbytes of RAM) plus an extensive range of enhanced I/Os and peripherals. The STM32L053x6/8 devices provide high power efficiency for a wide range of performance. It is achieved with a Image: State S	The ultra-low-power STM32L053x6/8 microcontrollers incorporate the connectivity power of the universal serial (USB 2.0 crystal-less) with the high-performance Arm® Cortex®-M0+ 32-bit RISC core operating at a 32 MHz frequency, a memory protection unit (MPU), high-speed embedded memories (up to 64 Kbytes of Flash program memory, 2 Kbytes of data EEPROM and 8 Kbytes of RAM) plus an extensive range of enhanced I/Os and peripherals.	bus m	LQFP64 versal serial a 32 MHz lash prograr s and bieved with	LQFP64 e universal sen ng at a 32 MHz s of Flash prog ed I/Os and	UCLEO-L053R8 e the connectivity power of the universe M0+ 32-bit RISC core operating at a ed memories (up to 64 Kbytes of Fla an extensive range of enhanced I/Os	oard: <u>NUCLE(</u> orporate the ortex®-M0+ : mbedded m) plus an ext	duction ocontrollers ir mance Arm® I), high-speed Kbytes of RA	Product is in mass prod STM32L053x6/8 micro with the high-perform protection unit (MPU) data EEPROM and 8	The ultra-low-power S (USB 2.0 crystal-less) frequency, a memory memory, 2 Kbytes of peripherals.
The ultra-low-power STM32L053x6/8 microcontrollers incorporate the connectivity power of the universal serial bus (USB 2.0 crystal-less) with the high-performance Arm® Cortex®-M0+ 32-bit RISC core operating at a 32 MHz frequency, a memory protection unit (MPU), high-speed embedded memories (up to 64 Kbytes of Flash program memory, 2 Kbytes of data EEPROM and 8 Kbytes of RAM) plus an extensive range of enhanced I/Os and peripherals. The STM32L053x6/8 devices provide high power efficiency for a wide range of performance. It is achieved with a Image: State Projection Image: State Projection Image: State	The ultra-low-power STM32L053x6/8 microcontrollers incorporate the connectivity power of the universal serial (USB 2.0 crystal-less) with the high-performance Arm® Cortex®-M0+ 32-bit RISC core operating at a 32 MHz frequency, a memory protection unit (MPU), high-speed embedded memories (up to 64 Kbytes of Flash program memory, 2 Kbytes of data EEPROM and 8 Kbytes of RAM) plus an extensive range of enhanced I/Os and peripherals.	bus m	versal serial a 32 MHz lash prograr s and bieved with	e universal sei ng at a 32 MHz s of Flash prog ed I/Os and	e the connectivity power of the universe M0+ 32-bit RISC core operating at a ed memories (up to 64 Kbytes of Fla an extensive range of enhanced I/Os	orporate the ortex®-M0+ : mbedded m) plus an ex	ocontrollers ir mance Arm® I), high-speed Kbytes of RA	STM32L053x6/8 micro) with the high-perform / protection unit (MPU) data EEPROM and 8	The ultra-low-power S (USB 2.0 crystal-less) frequency, a memory memory, 2 Kbytes of peripherals.
Us List: 6 items Part No Refe Marketing Unit Pri Board Pack Flash IO Freq. GFX S H M S YSTM32L053R6 STM32 Active 1.591 LQFP64 32 kBytes 8 51 32 MHz 0.0 0	Features Block Diagram Datasheet Docs & Resources Buy Sta	a art Project	Sta	t is achieved w	Docs & Resources Buy	v for a wide	Dower efficie	B devices provide high	The STM32L053x6/8
Part No Refe Marketing Unit Pri Board Pack Flash IO Freq. GFX S H M S STM32L053R6 STM32 Active 1.591 LQFP64 32 kBytes 8 51 32 MHz 0.0 0 <th>CUs List: 6 items</th> <th>[</th> <th></th> <th></th> <th>,</th> <th></th> <th>ar items</th> <th>,-</th> <th>Js List: 6 items</th>	CUs List: 6 items	[,		ar items	,-	Js List: 6 items
STM32L053R6 STM32 Active 1.591 LQFP64 32 kBytes 8 51 32 MHz 0.0 0<	Part No Refe., Marketing., Unit Pri., Board Pack., Flash IO Freq. GFX S., H., M.,	S 1	. H M	FX S H I	ck Flash IO Freq. GFX S	Pack	Board	Marketing Unit Pri	Part No Refe
STM32L053R8 STM32 Active 1.707 NUCLEO-L053R8 LQFP64 64 kBytes 851 32 MHz 0.0 0	STM32L053R6 STM32 Active 1.591 LQFP64 32 kBytes 851 32 MHz 0.0 0 0	0 0	0 0	0 0 0	7P64 32 kBytes 8 51 32 MHz 0.0	LQFP64		Active 1.591	STM32L053R6 STM32
STM32L063R8 STM32 Active 1.846 LQFP64 64 kBytes 8 51 32 MHz 0.0 0	STM32L053R8 STM32 Active 1.707 NUCLEO-L053R8 LQFP64 64 kBytes 8 51 32 MHz 0.0 0 0	0 0	0 0	0 0 0	P64 64 kBytes 8 51 32 MHz 0.0	LQFP64	NUCLEO-L053E	Active 1.707	STM32L053R8 STM32
STM32L073RB STM32 Active 1.915 LQFP64 128 kB 2 51 32 MHz 0.0 0 0 0 STM32L073RZ STM32 Active 2.059 NUCLEO-L073RZ LQFP64 192 kB 2 51 32 MHz 0.0 0 </td <td>2 STM32L063R8 STM32Active 1.846 LQFP64 64 kBytes 851 32 MHz 0.0 0 0</td> <td>0 0</td> <td>0 0</td> <td><u>o o</u> o</td> <td>P64 64 kBytes 8 51 32 MHz 0.0</td> <td>LQFP64</td> <td></td> <td>. Active 1.846</td> <td>STM32L063R8 STM32</td>	2 STM32L063R8 STM32Active 1.846 LQFP64 64 kBytes 851 32 MHz 0.0 0 0	0 0	0 0	<u>o o</u> o	P64 64 kBytes 8 51 32 MHz 0.0	LQFP64		. Active 1.846	STM32L063R8 STM32
STM32L073RZ STM32 Active 2.059 NUCLEO-L073RZ LQFP64 192 kB 251 32 MHz 0.0 0 0 0	STM32L073RB STM32Active 1.915 LQFP64 128 kB 251 32 MHz 0.0 0 0	0 0	0 0	0 0 0	7P64 128 kB 2 51 32 MHz 0.0	LQFP64		Active 1.915	STM32L073RB STM32
	7 STM32L073RZ STM32 Active 2.059 NUCLEO-L073RZ LQFP64 192 kB 2 51 32 MHz 0.0 0 0	0 0	0 0	0 0 0	P64 192 kB 2 51 32 MHz 0.0	LQFP64	NUCLEO-L073	. Active 2.059	STM32L073RZ STM32

>

<

當點選後顯示如下圖,單須確認是否選到正確的晶片 MCU

設定 PAO, 說明如下:

- 3. 將滑鼠的指標移至晶片的上方或視窗區域,轉動滾輪放大或所小圖示。
- 4. 按住滑鼠右鍵後,移動滑鼠可以移動 MCU 的位置。

再本範例中欲使用 PAO 的接腳(Pin)作為輸出到 LED 方式:

- 3. 將滑鼠移置 PAO 後按下左鍵。
- 4. 點選倒數第三項左邊的空格「GPIO Output」

設定 PC13, 說明如下:

將滑鼠的指標移至晶片的上方或視窗區域,轉動滾輪放大或所小圖示。 按住滑鼠右鍵後,移動滑鼠可以移動 MCU 的位置。

再本範例中欲使用 PC13 的接腳(Pin)作為輸入方式: 將滑鼠移置 PC13 後按下左鍵。 點選倒數第三項左邊的空格「GPIO Input」

點選左上方的 Project → Setting:以便專案的設定

顯示如下圖,並參考如下填入:

- 5. Project Name
- 6. Project Location
- 7. Tool Chain \rightarrow MDK-ARM V5
- 8. OK(右下角)

Project Settings	×
Project Code Generator Advanced Settings	_
Project Settings Project Name Buzzer1 Project Location C:STM32L053教學資料\Examples Application Structure	
Basic Do not generate the main()	
Toolchain Folder Location C:%TM32L053教學資料\Examples\3_Buzzer1-\ Toolchain / IDE MDK-ARM V5 Linker Settings Minimum Heap Size 0x200 Minimum Stack Size 0x400	
Mcu and Firmware Package Mcu Reference	
STM32L053R8Tx	
Firmware Package Name and Version	-
Ok	<u>1</u>

接下來要設定 PC13 為 PULL High 輸入,如下圖選擇「Configuration」,「GPIO」,

STM32CubeMX Buzzer1.ioc: STM3	32L053R8Tx				
File Project Window Help					
🗛 🐸 🐘 🖯 🖏 💊 🖉	+- 🤊 🧈 🦊				💿 🛛 🔽 🔽 💽
Pinout Clock Configuration Configuratio	n Power Consumption Calculator				
Configuration					
Additional Softwar					
FATES					
User-defined			Middl	ewares	
Enabled					
-Peripherals					
CRC					
Activated	Multimedia	Connectivity	Analog	System	Control
⊨- 💽 IWDG	Pluitimetia	Connectivity	Allalog	System	Control
Activated					
🖨 💿 RNG				DMA 🛄	
Activated					
				GPIO ->>	
Timebase Source SysTick					
🖨 💿 TIM6					
Activated					
🔲 One Pulse Mode				RCC 🔧	
📗 📩 🝙 աարգ			J		

在 PC13 的位置,並點選 GPIO 「Pull-up」。

1	💿 Pin Config	juration								×	
	I GPIO										┝
	Search Signals Search (Crtl-	4F))			Show only	Modified Pins	F
	Pin Name	Signal on Pin	GPIO outpu	GPIO mode	GPIO Pull-u	Maxin	1000 O	Fast Mode	User Label	Modified	h
Π	PAO	n/a	Low	Output Push	. No pull-up an	Low		n/a			
	PC13	n/a	n/a	Input mode	Pull-up	n/a		n/a		V	Г
U											L
											ŀ

設定完成後,點選「OK」即可。

接下來可以利用 Cube 來產生報告。

點選 Project→Generate Report 如下圖:

這個 Report 是 PDF 檔,可以檢視我們設定是否正確,最好在最後設定完成後檢視。

本範例為初次使用自動產生程式碼方式,因此使用其預設值即可。 點選左上方的 Project → Generate Code:以產生程式碼。

1		🖲 S	тмз	2CubeMX Buzzer1.ioc: STM	2L053R8Tx						
		File	Proje	ect Pinout Window Help							
		Г,	-	Select additional software of	omponents	Ctrl+E	ne	ent 🤊	ه 🛽	- (🧭 🔶 F
		•	4	Generate Code		Ctrl+Shift+	+G				
		Pinc	2	Generate Report		Ctrl+R	or	r			
		Con	्र	Settings		Alt+P		<mark>g</mark> v e	20 <mark></mark> r. v	e 0.4	212
	Η	7	+	FATES		L.		<u> </u>	8 8 8 8	20202	888
			÷	FREERTOS		GPIO_Input	PC13/	0			
			∲ ~()	TOUCHSENSING		1	PC1				
			÷	USB_DEVICE		1	PC1				
		⊨ … F	eripl	herals			PH0				
			🕂 🕀	ADC	=	ľ	NRST				_

在產生程式碼後開啟後並開啟專案檔後如下圖:

根據 HAL 驅動說明書找到我們需要的 API,然後根據說明添加代碼:(根據說明得到使 PA5 輸出高電位的代碼為 HAL_GPIO_WritePin (GPIOA, GPIO_PIN_5, GPIO_PIN_SET);)

25.2.8 HAL_GPIO_WritePin

Function Name	void HAL_GPIO_WritePin (GPIO_TypeDef * GPIOx, uint16_t GPIO_Pin, GPIO_PinState PinState)
Function Description	Sets or clears the selected data port bit.
Parameters	 GPIOx: where x can be (AK) to select the GPIO peripheral for STM32F429X device or x can be (AI) to select the GPIO peripheral for STM32F40XX and STM32F427X devices. GPIO_Pin: specifies the port bit to be written. This parameter can be one of GPIO_PIN_x where x can be (015). PinState: specifies the value to be written to the selected bit. This parameter can be one of the GPIO_PinState enum values: GPIO_PIN_RESET: to clear the port pinGPIO_PIN_SET: to set the port pin
Return values	None
Notes	 This function uses GPIOx_BSRR register to allow atomic read/modify accesses. In this way, there is no risk of an IRQ occurring between the read and the modify access.

根據 HAL 驅動說明書找到我們需要的 API,然後根據說明添加代碼:(根據說明得到使 PC13 輸出高電位的代碼為 HAL_GPIO_ReadPin (GPIOC, GPIO_PIN_13);)

 __PPP_CLK_SLEEP_ENABLE/__PPP_CLK_SLEEP_DISABLE to peripheral clock during low power (Sleep) mode.

2.11.2 GPIOs

GPIO HAL APIs are the following:

- HAL_GPIO_Init() / HAL_GPIO_DeInit()
- HAL_GPIO_ReadPin() / HAL_GPIO_WritePin()
- HAL_GPIO_TogglePin ().

40/1400 D00D020232 Rev 0

輸入程式碼如下:

- 在 while(1)下方:
 /*USER CODE BEGIN 3*/及/* USER CODE END 3 */中間開始輸入,以避免程式重新產生時被覆蓋。
- 4. 在程式輸入時會自動出現關鍵字以便選擇。(MDK 5.xx 版後)

```
程式碼如下:
while (1)
  {
  /* USER CODE END WHILE */
  /* USER CODE BEGIN 3 */
        Key_in = HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_13);
        if(Key_in==0){
        //for(int i=0;i<T;i++) //輸出反相次數,決定音長
      //{
                 HAL_GPIO_WritePin(GPIOA, GPIO_PIN_0, GPIO_PIN_RESET);
                 HAL Delay(2);
                 HAL_GPIO_WritePin(GPIOA, GPIO_PIN_0, GPIO_PIN_SET);
                 HAL_Delay(3);
            //}
     }
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_0, GPIO_PIN_SET);
        }
```

注意:

- 我們編寫程式時,需將程式碼放在/*USER CODE BEGIN N */與 /*USER CODE END N */內, 以免在 CUBE 產生程式碼時被覆蓋。
- 5. 按下 Reset 按鈕開始執行程式,可以聽到 PAO 的 Buzzer 聲音。
- 6. 亦可事先在 Keil 中設定燒錄完成後自動執行程式。

方式:在 MDK 的 Option for Target...(仙女棒) 按鈕點選。

點選 Option for Target...(仙女棒)後即可開啟另一視窗,如下: 接者點選 Debug 後,注意在紅框中需選設在 ST-Link Debugger 或相容的 Tool。 確認後再點選右邊的 settings 選項。

-	→ 🥐 ឌ ឌ ឌ ឌ 🕸 🕸 🛊 🚎 🚛 🚲 🙆	• العالي المراجع
mai	Options for Target 'Buzzer1'	
10	Device Target Output Listing User C/C++ A	sm Inker Debug Utilities
$11 \\ 11$	C Use Simulator with restrictions Settings	O Use: ST-Link Debugger
11	Limit Speed to Real-Time	
11	✓ Load Application at Startup ✓ Run to main()	Load Application at Startup Run to main()
11	Initialization File:	Initialization File:
11	Edit	Edit
11	Restore Debug Session Settings	Restore Debug Session Settings
12	Breakpoints Toolbox	Breakpoints Toolbox
12	Watch Windows & Performance Analyzer	Watch Windows
12		
12	CPU DLL: Parameter:	Driver DLL: Parameter:
12	SARMCM3.DLL -REMAP	SARMCM3.DLL
12	Dialog DLL: Parameter:	Dialog DLL: Parameter:
12	DARMCMT.DLL J-pCMU+	TARMCM1.DLL pCM0+
13		
13	Manage Component Vi	ewer Description Files
	ОК Са	ncel Defaults Help

點選上示的選項後,會出現如下示的視窗。

接著點選紅框中的 Flash Download 選項,接著點選下面紅框中的 Reset and Run 選項,這樣會在程式下載完後自動執行,不用再按 Reset 鍵。

Cortex-M Target Driver Setup	- 53					×	
Debug Trace Flash Download							
Download Function C Erase Full Chip C Erase Sectors C Do not Erase	o I ⊽ Program I ⊽ Verify I ⊽ Reset an	d Run	/l for Algorithm	Size: 0x08	00		
Programming Algorithm							
Description	Device Size	Device Type	Address Rai	nge			
Start: Size:							
	Ac	id Ren	nove				
				確定	取消	套用(A)	

利用 PWM 驅動 LED

檔案名稱: 05-pwm_led.doc

本例練習利用 PWM 的方式,設定 LED 產生漸明漸減的方式,本例會用到 HAL 庫及 Timer 計時。 由於 STM 對於 RCC 有其相對的 GPIO,因此,PH0,PH1 綁訂在 PA0 ~PA3。本例僅用一顆 LED, 所以設定在 PA0 作為輸出。

參考下圖

先利用 STM 公司提供的 CUBE,本例使用的版本為 4.27.0(如圖),如果開啟舊檔時可能會有版本差異問題時,先選擇移植的選項,有部分較新的專案需升級到版本 5 以上。

接下來將 PHO 點選為 RCC_OSC_IN,將 PH1 點選為 RCC_OSC_OUT,注意不用點選 PAO。

點開在 CUBE 左邊的選單 RCC,檢查 High Speed Clock(HSE)

接著要選輸出的 PIN,因此先點開 Channel1 並下拉到 PWM Generation CH1 即可。

						Ŧ
-💓 TIM2						
-Slave Mod	e Di	sable				¥
Trigger So	urce	Disa	ble			¥
-Clock Sou	rce	Disa	ble			V
Channell PWM Generation CH1					~	
-Channel2	hannel2 Disable				Y	
Channel3	Disal	ble				¥
Channel4	Disal	ble				¥
Combined	Char	nnels	Disable	!		\vee
Use ETR a	ıs Cle	aring	Source	Disable		¥
🔲 XOR :	activa	tion				
🗌 One P	ulse l	Mode				

點選 Clock Configuration:

參考下圖,將1的MUX 點選 HSE,將2的MUX 點選 PLLCLK,將2的頻率設為最大32。

接下來的設定不一定要照順序,但為了講解方便起見,會依序由上而下,由左而右。

現進入設定的選項模式,由於ARM的站存器眾多,且不易了解,應用時容易出錯,因此STM 公司將其繁雜的設定工作交由第三方開發較為簡便設定,雖然簡便,仍有些設定的關係稍微了 解即可,且日後STM公司也不再出CMSIS庫及標準暫存器庫了,但過去的仍然可相容使用。 以下,參考下圖:

Multimedia	Connectivity	Analog	System	Control	Security
			DMA 📥+	тім2 🦂	
			gpio 🛶		
			NVIC		
			RCC 🔧 🧹		

點選 GPIO 項目進入設定,再選擇 RCC 確認一下即可,不做點選。

	۲			Pin Conf	iguration				×
	RCC 🗹 TI	M2							
	Search Signals Search (Crtl+F)							Show only Mc	dified Pins
	Pin Name	Signal on Pin	GPIO outp	GPIO mode	GPIO Pull	Maximum	Fast Mode	User Label	Modified
	PHO-OSC_IN	RCC_OSC_IN	n∕a	n/a	n/a	n/a	n/a		
	PH1-OSC_OUT	RCC_OSC_OUT	u/a	n/a	n/a	n/a	n/a		
1									

再選擇 TIM2 確認一下為 PAO 即可,不做點選項。

۲				Pir	n Configura	tion			×	
✓ RCC ✓ TIM2										
Search Signal: Search (Crth	s +F)							Show only I	Modified Pins	
Pin Name	Signal on Pin	GPIO ou	tpu	GPIO mode	GPIO Pull-u	Maximum o	Fast Mode	User Label	Modified	
PAO	TIM2_CH1	n/a		Alternate Fun	No pull-up an	Low	n/a			

參考下圖 點選 Enable 項

٥	NVIC Configuration			×
VIC 🗸 Code generation				
		🔄 Sort by	Premption Priority and Sub Pr	ority
Search (Crtl+F)	•	Show c	only enabled interrupts	
Interrupt Table		Enabled	Preemption Priority	
Non maskable interrupt		\checkmark	0	
Hard fault interrupt		~	0	
System service call via SWI instruction		~	0	
Pendable request for system service		~	0	
Time base: System tick timer		~	0	
PVD interrupt through EXTI line 16			0	
Flash and EEPROM global interrupt			0	
RCC and CRS global interrupt				
TIM2 global interrupt		✓	0	
	T	IM2_IRQn		

接著確認 RCC Parameters(這是內建的,所以不用調整)。

I RCC (Configuration	×
🚽 Parameter Settings 🧹 User Constants 🚽	NVIC Settings 🞻 GPIO Settings	
Configure the below parameters :		
Search : Search (Crtl+F)	* ☆	
🖃 System Parameters		
VDD voltage (V)	3.3 ♥	
Buffer Cache	Enabled	
Prefetch	Disabled	
Preread	Enabled	
Flash Latency(WS)	1 WS (2 CPU cycle)	
RCC Parameters		
HSI Calibration Value	16	
MSI Calibration Value	0	
HSE Startup Timout Value (ms)	100	
LSE Startup Timout Value (ms)	5000	
Power Parameters		
Power Regulator Voltage Scale	Power Regulator Voltage Scale 1	

接著設定 TIM2 的 Counter 值,如 prescaler ...等參數值。 這個項目很重要,須確實輸入。

© TIM	2 Configuration	×				
🖋 Parameter Settings 🚽 User Constants 🚽 NVIC Settings 🚽 DMA Settings 🚽 GPIO Settings						
Configure the below parameters :						
Search : Search (CrtI+F)						
😑 Counter Settings						
Prescaler (PSC - 16 bits value)	420-1					
Counter Mode	Up					
Counter Period (AutoReload Register - 16 bits	value) 2000-1					
Internal Clock Division (CKD)	Division by 4					
🖃 Trigger Output (TRGO) Parameters						
Master/Slave Mode (MSM bit)	Disable (Trigger input effect not delayed)					
Trigger Event Selection	Reset (UG bit from TIMx_EGR)					
😑 PWM Generation Channel 1						
Mode	PWM mode 1					
Pulse (16 bits value)	0					
Fast Mode	Disable					
CH Polarity	High					

Project 設定項目再確認,如下圖:

Project Settings
Project Code Generator Advanced Settings
Project Settings
Project Name
pwm_led
Project Location
D:%TM32L053
Application Structure
Basic \checkmark Do not generate the main()
Toolchain Folder Location
D.S. TAISZEDSS Qwin_eux
Toolchain / IDE
MDK-ARM V5
Linker Settings
Minimum Heap Size
Minimum Stack Size
Mcu and Firmware Package
Mcu Reference
STM32L053R8Tx
Firmware Package Name and Version
STM32Cube FW_L0 V1.10.0
Ok

以上設定完後,可選擇 Project --> Generate Code 來產生程式碼。

至此,STM的CUBE工作已完成,接下來僅需鑽寫主程式即可。 注意:將程式碼填入USERCODE區內,且不要將其原有的註解刪除,以避免程式碼在自動產 生時被刪除。

Main 主程式內的變數宣告,如下圖:

程式 While 迴圈,如下圖:

驅動三色 LED

檔案名稱: 06-RGB_LED.doc

本例是利用晶片內部的計時器及相對應的輸出腳位推動 LED。STM 的本顆晶片有特定相對的輸出腳位,有別於其他廠家的晶片,如 Cypress, Nuvoton 的部分晶片有提供內建的 CPLD 可重新 規劃輸出輸入腳位。

首先參考「RGB_LED-參考設計.pdf」先建立 Cube 專案檔,依照下列步驟:

選擇欲使用的晶片可參考其他範例建立 Cube 專案檔。 在建立專案檔後依下列方式設定

首先在 PHO 及 PH1 點選 RCC_OSC_IN 及 RCC_OSC_OUT。以選擇即設定內部 Clock。

在 Pinout 選項內的 TIM2,應將 Channel1 ~ Channel1 依序選擇 PWM Generation CH1 ~ PWM Generation CH3,將利用這三個 Pin 作為驅動 LED 用,此時可看到 PAO ~ PA2 三個腳位已被自動設定。

接著在「Clock configuration」選項內點選 clock 的來源為 PLLCLK,再將 HCLK 的頻率設定為 最大 (即 32MHz)。

在 Configuration →GPIO 選項內觀察 TIM2 選項內的 PA0~PA2。如果有需要可再設定其他功能, 在本例中均不另做設定。

♥ Pin Configuration ✓ RCC ✓ TIM2								
Search Signals Search (Crtl+F) Show only Modified Pins								
Pin Name	Signal on Pin	GPIO outpu	GPIO mode	GPIO Pull-u	Maximum o	Fast Mode	User Label	Modified
PAO	TIM2_CH1	n/a	Alternate Fun	No pull-up an	Low	n/a		
PA1	TIM2_CH2	n/a	Alternate Fun	No pull-up an	Low	n/a		
PA2	TIM2_CH3	n/a	Alternate Fun	No pull-up an	Low	n/a		
2 01		~						
Select Pins from table to configure them. Multiple selection is Allowed.								
🔲 Group By	Peripherals					Apply	Ok	Cancel

接著在 Configuration→Parameter Setting 內的 Counter settings ,將 Prescacler 及 Counter period 設定值如下圖。

TIM2 Configuration					
💞 Parameter Settings 🗹 User Constants 🗹 NVIC Settings 🗹 DMA Settings 🗹 GPIO Settings					
Configure the below parameters :					
Search : Search (CrtI+F)					
Counter Settings		-			
Prescaler (PSC - 16 bits value)	1384-1				
Counter Mode	Up				
Counter Period (AutoReload Register - 16 bits valu.	255-1				
Internal Clock Division (CKD)	No Division				
🖃 Trigger Output (TRGO) Parameters					
Master/Slave Mode (MSM bit)	Disable (Trigger input effect not delayed)				
Trigger Event Selection	Reset (UG bit from TIMx_EGR)	=			
😑 PWM Generation Channel 1					
Mode	PWM mode 1				
Pulse (16 bits value)	0				
Fast Mode	Disable				
CH Polarity	High				
😑 PWM Generation Channel 2					
Mode	PWM mode 1				
Pulse (16 bits value)	0				
Fast Mode	Disable				
CH Polarity	High				
😑 PWM Generation Channel 3		Ŧ			
Restore Default	Apply Ok Cance	1			

在 main.c 主程式中加入以下程式碼:

100	
107	/* USER CODE BEGIN 2 */
108	HAL_TIM_PWM_Start (&htim2, TIM_CHANNEL_1);
109	HAL_TIM_PWM_Start (&htim2, TIM_CHANNEL_2);
110	HAL_TIM_PWM_Start (&htim2, TIM_CHANNEL_3);
111	<pre>//HAL_TIM_PWM_Start (&htim2, TIM_CHANNEL_4);</pre>
112	/* USER CODE END 2 */
113	

119 120 E	while (1)
120 6	/* USER CODE END WHILE */
122	/* USER CODE BEGIN 3 */
124 125 126	rgb_set (255,0,0); // only red HAL_Delay (1000);
127 128 129	rgb_set (0,255,0); // only blue HAL_Delay (1000);
131 132	rgb_set (0,0,255); // only green HAL_Delay (1000);
135 134 135	rgb_set (255,255,0); HAL_Delay (1000);
130 137 138	rgb_set (0,255,255); HAL_Delay (1000);
139 140 141	1gb_set (255,0,255); HAL_Delay (1000);
142 143 144	rgb_set (192,192,192); HAL_Delay (1000);
145 146 147	- } /* USER CODE END 3 */

驅動 WS2812B 彩色燈條

檔案名稱: 07-驅動 WS2812B 彩色燈條.doc

本例是利用 PWM 方式送出訊號來驅動 WS2812B 的串珠燈條,詳細及控制方式可參考 WS2812B 的規格書。 燈條的接線:紅色線接+5V,白色線接 GND,綠色線接到 PAO。 開始時 LD2(版上的 LED)接腳是 PA5,會一直閃爍,當按下 PC13(板子上藍色按鈕)時,所有動 作會停止,PC13 在程式定義是 B1,放開按鈕後燈條則繼續進行閃爍。

本例我使用的燈條依公式 15 顆 LED。

先依序點選 Pinout→TIM2→Channel1→PWM Generation CH1 如圖,在 Clock Source→Internal Clock。

PA0 腳位會呈現綠色,表示已經選上了,並與 TIMER2 形成了相依。 接著將 PA5 的腳位設為輸出,並改腳位名稱(Lable)為 LD2。

在 Clock configuration 設定內, Clock 來源選擇 HSI, 並選擇 System Clock 選為 PLLCLK, 並將 HCLK 用手動方式設為最大值(32MHz)。

請將以下程式碼載入主程式中。

```
/* USER CODE BEGIN 0 */
#define H VAL 26
#define L VAL 14
#define N LEDS 15
                                 //LED ???, ?????15?
#define BITS PER LED (3*8)
#define BIT_BUF_SIZE (N_LEDS * BITS_PER_LED)
uint16 t ws2812BitBuf[BIT BUF SIZE + 1]; //DMA transfer needs one byte more with ZERO,
because it will be output after DMA transfer has finished
void ws2812 set color(int led, uint8 t r, uint8 t g, uint8 t b)
{
  if (led >= N LEDS) return;
  int i = led * BITS PER LED;
  uint8 t mask;
  mask = 0x80;
  while(mask) {
    ws2812BitBuf[i] = (mask & g)?H_VAL:L_VAL;
    mask >>= 1;
    i++;
  }
  mask = 0x80;
```

```
while(mask) {
ws2812BitBuf[i] = (mask & r)?H_VAL:L_VAL;
mask >>= 1;
```

```
i++;
  }
  mask = 0x80;
  while(mask) {
    ws2812BitBuf[i] = (mask & b)?H VAL:L VAL;
    mask >>= 1;
    i++;
  }
}
/* USER CODE END 0 */
/* USER CODE BEGIN 2 */
//memset(ws2812BitBuf, 0, sizeof(ws2812BitBuf));
  for (int led = 0; led < N LEDS; led++) ws2812 set color(led, 2, 2, 2);
  HAL_TIM_PWM_Start_DMA(&htim2, TIM_CHANNEL_1, (uint32_t*)ws2812BitBuf,
sizeof(ws2812BitBuf) / sizeof(ws2812BitBuf[0]));
  HAL_Delay(10);
  /* USER CODE END 2 */
/* Infinite loop */
 /* USER CODE BEGIN WHILE */
  while (1)
  {
  /* USER CODE END WHILE */
  /* USER CODE BEGIN 3 */
static int cursorled = 0;
    static uint8_t r = 0;
    static uint8 t g = 0;
    static uint8_t b = 0;
    static uint8 t state = 0;
    if (GPIO PIN SET == HAL GPIO ReadPin(B1 GPIO Port, B1 Pin)) {
      for (int led = 0; led < N_LEDS; led++) ws2812_set_color(led, r, g, b);</pre>
      cursorled++;
      if (N LEDS <= cursorled) {
        cursorled = 0;
      }
      switch (state) {
        case 0:
          r++;
          if (g) g--;
          b = 0;
          if (r == 255) {
            state++;
```

```
}
           break;
         case 1:
           if (r) r--;
           g = 0;
           b++;
           if (b == 255) {
              state++;
           }
           break;
         case 2:
           r = 0;
           g++;
           if (b) b--;
           if (g == 255) {
              state++;
           }
           break;
         default:
           state = 0;
           break;
       }
       ws2812_set_color(cursorled, 255, 255, 255);
       HAL_TIM_PWM_Stop_DMA(&htim2, TIM_CHANNEL_1);
       HAL_TIM_PWM_Start_DMA(&htim2, TIM_CHANNEL_1, (uint32_t*)ws2812BitBuf,
sizeof(ws2812BitBuf) / sizeof(ws2812BitBuf[0]));
       HAL_GPIO_TogglePin(LD2_GPIO_Port, LD2_Pin);
       HAL Delay(50);
//
        HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_SET);
    }
    else {
       HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_RESET);
    }
  }
  /* USER CODE END 3 */
```

}

LCD 顯示器範例

專案名稱:08-LCDx1A2(HAL)

在本例中,先假設已經十分孰悉 CUBE 專案的建置,及設定的方式。 本範例中以 GPIO Pin 作為驅動 LCD 的方式,僅需簡單的設定即可,關於 LCD 的規範可參考 LCD 1602 的規格書。

如下圖所示,並參考 Pins configuration 設定各個 Pin 並加入該 Pin 的 Lable。

3. Pins Configuration

	Pin Number LQFP64	Pin Name (function after reset)	Pin Type	Alternate Function(s)	Label
	1	VLCD	Power		
	7	NRST	Reset		
	8	PC0 *	I/O	GPIO_Output	DB0
	9	PC1 *	I/O	GPIO_Output	DB1
	10	PC2 *	I/O	GPIO_Output	DB2
	11	PC3 *	I/O	GPIO_Output	DB3
	12	VSSA	Power		
	13	VDDA	Power		
	18	VSS	Power		
Æ	19	VDD	Power		
	24	PC4 *	I/O	GPIO_Output	DB4
	25	PC5 *	I/O	GPIO_Output	DB5

20	PC9 "	1/0	GPIO_Output	DBC
31	VSS	Power		
82	VDD	Pewer		
37	PC6 *	I/O	GPIO_Output	DB6
38	PC7 *	I/O	GPIO_Output	DB7
39	PC8 *	I/O	GPIO_Output	RS
40	PC9 *	I/O	GPIO Output	RW
47	VSS	Power		
40	VDD USD	Power		
51	PC10 *	I/O	GPIO_Output	EN
60	BOOT0	Boot		
63	VSS	Power		
64	VDD	Power		

参考以下接腳方式連接 LCD 及 STM32L053 的接腳。

Source Code:(可將以下的原始碼複製到如下的位置下)

/* USER CODE BEGIN PV */ /* Private variables ------*/ void SystemClock_Config(void); static void MX_GPIO_Init(void); 註;以上不用複製

//****宣告副程式
void LCD_Data(uint8_t dat); //傳送資料到 LCD
void LCD_Cmd(uint8_t Cmd); //傳送命令到 LCD
void LCD_init(void); //LCD 的啟始程式
//#define Data GPIOC->ODR //資料 BUS 輸出

繼續如下圖,將 Source Code 複製到重確的位置。

```
97 int main (void)
 98
    ₽{
       /* USER CODE BEGIN 1 */
99
100
         uint8 t i;    //資料計數
        HAL_Init();
101
        SystemClock Config();
102
103
        MX GPIO Init();
104
        LCD init();
                     //重置及清除LCD
105
        //LCD Cmd(0x0F);//0000 1111
          //bit2:D=1, 顯示幕ON
106
          //bit1:C=1,顯示游標
107
108
          //bit0:B=1,游標閃爍
109
        //LCD Cmd(0x04); //0000 0100,
          //bit1:I/D=0,游標左移反向顯示
110
111
112
                        //游標由第一行第5個字開始顯示
        LCD Cmd(0x85);
        for(i='0'; i<='9';i++) //字元a~j
113
114
    Ē
         Ł
115
          LCD Data(i); //字元送到LCD顯示
          HAL Delay(300);//延時,慢速逐一顯示
116
117
           }
118
119
        LCD_Cmd(0xC5); //游標由第二行第5個字開始顯示
        for(i='A'; i<= 'J';i++)//LCD顯示字元A~J
120
121
    E
         Ł
122
             LCD Data(i); //字元送到LCD顯示
123
            HAL_Delay(100);//延時, 慢速逐一顯示
124
           ł
       /* USER CODE END 1 */
125
```

/* USER CODE BEGIN 1 */

uint8_t i; //資料計數 HAL_Init(); SystemClock_Config(); MX_GPIO_Init(); LCD_init(); //重置及清除 LCD //LCD_Cmd(0x0F);//0000 1111 //bit2:D=1,顯示幕 ON //bit1:C=1,顯示游標 //bit0:B=1,游標閃爍 //LCD_Cmd(0x04); //0000 0100, //bit1:I/D=0,游標左移反向顯示

LCD_Cmd(0x85); //游標由第一行第 5 個字開始顯示 for(i='0'; i<='9';i++) //字元 a~j

```
{
    LCD Data(i); //字元送到 LCD 顯示
    HAL Delay(300);//延時, 慢速逐一顯示
     }
  LCD Cmd(0xC5); //游標由第二行第5 個字開始顯示
  for(i='A'; i<= 'J';i++)//LCD 顯示字元 A~J
   {
        LCD_Data(i); //字元送到 LCD 顯示
      HAL Delay(100);//延時,慢速逐一顯示
     }
 /* USER CODE END 1 */
/* USER CODE BEGIN 4 */
                            ****
₽/**********
 *函數名稱: LCD Data
 *功能描述: 傳送資料到文字型LCD
 *輸入參數:dat
            void LCD_Data(uint8_t dat) //傳送資料到LCD
₽{
    //uint8 t dly=2;
      //Data=dat;
                             //資料送到BUS
           HAL_GPIO_WritePin(GPIOC, dat , GPIO_PIN_SET);
           HAL_GPIO_WritePin(GPIOC, ~dat, GPIO_PIN_RESET);
    //RS_1; RW_0; EN_1; //資料寫入到LCD內
       HAL GPIO WritePin (GPIOC, GPIO PIN 8, GPIO PIN SET);
                                                             //RS --> 1
       HAL_GPIO_WritePin(GPIOC, GPIO_PIN_9, GPIO_PIN_RESET);
                                                             //RW --> 0
       HAL GPIO WritePin (GPIOC, GPIO PIN 10, GPIO PIN SET);
                                                              //EN --> 1
    //while(dly--);
       HAL Delay(2);
      //EN 0;
                       //禁能LCD
       HAL_GPIO_WritePin(GPIOC, GPIO_PIN_10, GPIO_PIN_RESET);
                                                          //EN --> 0
      HAL Delay(1); //LCD等待寫入完成
L }
```

/* USER CODE BEGIN 4 */

```
//Data=dat;
                                //資料送到 BUS
            HAL_GPIO_WritePin(GPIOC, dat, GPIO_PIN_SET);
            HAL GPIO WritePin(GPIOC, ~dat, GPIO PIN RESET);
    //RS_1; RW_0; EN_1; //資料寫入到 LCD 內
        HAL_GPIO_WritePin(GPIOC, GPIO_PIN_8, GPIO_PIN_SET);
                                                                   //RS --> 1
        HAL_GPIO_WritePin(GPIOC, GPIO_PIN_9, GPIO_PIN_RESET);
                                                                   //RW --> 0
        HAL GPIO_WritePin(GPIOC, GPIO_PIN_10, GPIO_PIN_SET);
                                                                   //EN --> 1
   //while(dly--);
        HAL_Delay(2);
                          //禁能 LCD
     //EN 0;
        HAL GPIO WritePin(GPIOC, GPIO PIN 10, GPIO PIN RESET); //EN --> 0
                    //LCD 等待寫入完成
      HAL Delay(1);
}
                              *函數名稱: LCD Cmd
  *功能描述: 傳送命令到文字型LCD
 *輸入參數:Cmd
                   void LCD_Cmd(uint8_t Cmd) //傳送命令到LCD
₽{
                                //命令送到BUS
      //Data=Cmd;
             HAL_GPIO_WritePin(GPIOC, Cmd , GPIO_PIN_SET);
HAL_GPIO_WritePin(GPIOC, ~Cmd, GPIO_PIN_RESET);
      //RS 0; RW 0; EN 1; //命令寫入到LCD內
         HAL_GPIO_WritePin(GPIOC, GPIO_PIN_8, GPIO_PIN_RESET);
                                                                      //RS --> 0
         HAL_GPIO_WritePin(GPIOC, GPIO_PIN_9, GPIO_PIN_RESET);
                                                                     //RW --> 0
         HAL GPIO WritePin(GPIOC, GPIO PIN 10, GPIO PIN SET);
                                                                      //EN --> 1
     HAL Delay(1);
      //EN 0;
                         //禁能LCD
         HAL GPIO WritePin(GPIOC, GPIO PIN 10, GPIO PIN RESET); //EN --> 0
     HAL Delay(1);
```

//Data=Cmd;

//命令送到 BUS

```
HAL_GPIO_WritePin(GPIOC, Cmd , GPIO_PIN_SET);
HAL_GPIO_WritePin(GPIOC, ~Cmd, GPIO_PIN_RESET);
//RS_0; RW_0; EN_1; //命令寫入到 LCD 內
HAL_GPIO_WritePin(GPIOC, GPIO_PIN_8, GPIO_PIN_RESET); //RS --> 0
HAL_GPIO_WritePin(GPIOC, GPIO_PIN_9, GPIO_PIN_RESET); //RW --> 0
HAL_GPIO_WritePin(GPIOC, GPIO_PIN_10, GPIO_PIN_SET); //EN --> 1
HAL_Delay(1);
//EN_0; //禁能 LCD
HAL_GPIO_WritePin(GPIOC, GPIO_PIN_10, GPIO_PIN_RESET); //EN --> 0
```

HAL_Delay(1);

}

```
******
288
   -/***********
     *函數名稱: LCD init
289
290
     *功能描述: 啟始化文字型LCD
                  291
292 void LCD_init(void) //LCD的啟始程式
293 📮 🗧
       LCD_Cmd(0x38); //0011 1000,8bit傳輸,顯示2行,5*7字型
294
295
       LCD_Cmd(0x38); //bit4:DL=1,8bit傳輸,
296
       LCD_Cmd(0x38); //bit3:N=1, 顯示2行
297
                                  //bit2:F=0,5*7字型
298
       LCD Cmd(0x0c); //0000 1100,顯示幕ON,不顯示游標,游標不閃爍
299
                                 //bit2:D=1,顯示幕ON
300
                                 //bit1:C=0,不顯示游標
301
                                   //bit0:B=0,游標不閃爍
302
       LCD_Cmd(0x06); //0000 0110,//顯示完游標右移,游標移位禁能
303
                                 //bit1:I/D=1,顯示完游標右移,
304
                                 //bit0:s=0,游標移位禁能
305
       LCD Cmd(0x01); //清除顯示幕
306
       LCD Cmd(0x02); //游標回原位
307
    Ll
308
   /* USER CODE END 4 */
/**********
```

*函數名稱: LCD_init

*功能描述: 啟始化文字型 LCD

void LCD_init(void) //LCD 的啟始程式

{

LCD_Cmd(0x38); //0011 1000,8bit 傳輸,顯示 2 行,5*7 字型 LCD_Cmd(0x38); //bit4:DL=1,8bit 傳輸, LCD_Cmd(0x38); //bit3:N=1,顯示 2 行 //bit2:F=0,5*7 字型

LCD_Cmd(0x0c); //0000 1100,顯示幕 ON,不顯示游標,游標不閃爍 //bit2:D=1,顯示幕 ON //bit1:C=0,不顯示游標 //bit0:B=0,游標不閃爍 LCD_Cmd(0x06); //0000 0110,//顯示完游標右移,游標移位禁能 //bit1:I/D=1,顯示完游標右移, //bit0:S=0,游標移位禁能

LCD_Cmd(0x01); //清除顯示幕 LCD_Cmd(0x02); //游標回原位

}

/* USER CODE END 4 */

注意:以上均在 User Code 4 之内,要輸入的 Source Code。