Lab7 STM32 Interrupt and Exception
實驗七 STM32 Interrupt and Exception

1. Lab objectives 實驗目的

- Understand how to set stm32 SysTick timer.
- Understand the setting and principle of NVIC and External.
- 瞭解 STM32 SysTick timer 設定
- 瞭解 STM32 NVIC 和 External 設定和原理

2. Theory 實驗原理

Please refer to 009-MCSL-CounterTimer and 010-MCSL-Interrupt lecture slide.

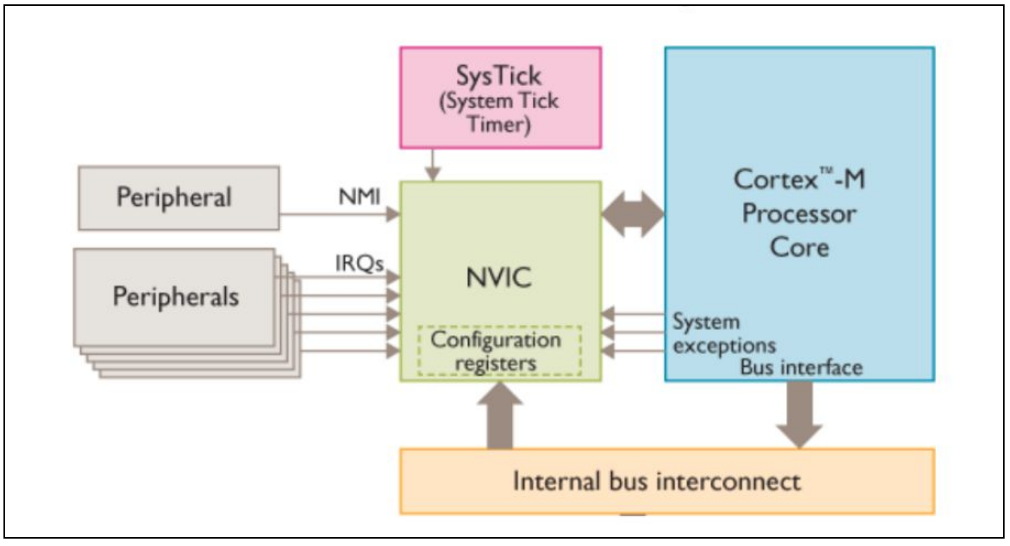
請參考上課 009-MCSL-CounterTimer 及 010-MCSL-Interrupt 講義。

Nested vectored interrupt controller(NVIC):

All interrupts including the core exceptions are managed by the NVIC before handling. When exceptions occur, the signal line connected to the NVIC will rise. Then, NVIC prioritizes exceptions and decides whether to handle them immediately, suspend or even ignore (mask) the exception. When a nested exception occurs, it can also decide whether to suspend the current handler to the new handler。

在處理之前，所有中斷（包括核心異常）都由 NVIC 管理。 發生異常時，連接到 NVIC 的信號線將被拉高。接著，NVIC 對異常做優先排序，並決定要立即處理它們、擱置，還是忽略（屏蔽）。當發生巢狀異常時，它還可以決定是否將當前處理程序暫停並跳到新的處理程序上。

Reference: *PM0214 Programming manual Sec4.3 NVIC registers*



Vector Table and Interrupt Handler:

When an interrupt occurs, the processor may switch the corresponding register, and jump to the "interrupt handler" according to the vector table. Well, setting up the table and handler is a very annoying task. Thankfully, the IDE has already done this for us (*startup/startup_stm32.s Line100~500*). You can check the source code and refer to *PM0214 Programming manual Sec2.3* to understand its configuration in detail.

發生中斷時，處理器可能會切換相應的暫存器，並根據中斷向量表跳轉到“中斷處理程序”。 設置向量表和處理程序是一項非常煩人的工作。值得慶幸的是，IDE 已經替我們完成了這項工作 (startup/startup_stm32.s Line 100~500) 。您可以查看原始碼及參閱 *PM0214 開發手冊 Sec2.3* 來了解他的詳細配置。

13.3 Interrupt and exception vectors

The grey rows in *Table 57* describe the vectors without specific position.

Table 57. STM32L4x5/STM32L4x6 vector table

Position	Priority	Type of priority	Acronym	Description	Address
-	-	-	-	Reserved	0x0000 0000
-	-3	fixed	Reset	Reset	0x0000 0004
-	-2	fixed	NMI	Non maskable interrupt. The RCC Clock Security System (CSS) is linked to the NMI vector.	0x0000 0008
-	-1	fixed	HardFault	All classes of fault	0x0000 000C
-	0	settable	MemManage	Memory management	0x0000 0010
-	1	settable	BusFault	Pre-fetch fault, memory access fault	0x0000 0014
-	2	settable	UsageFault	Undefined instruction or illegal state	0x0000 0018
-	-	-	-	Reserved	0x0000 001C - 0x0000 0028
-	3	settable	SVCall	System service call via SWI instruction	0x0000 002C
-	4	settable	Debug	Monitor	0x0000 0030
-	-	-	-	Reserved	0x0000 0034
-	5	settable	PendSV	Pendable request for system service	0x0000 0038
-	6	settable	SysTick	System tick timer	0x0000 003C
0	7	settable	WWDG	Window Watchdog interrupt	0x0000 0040
1	8	settable	PVD_PVM	PVD/PVM1/PVM2/PVM3/PVM4 through EXTI lines 16/35/36/37/38 interrupts	0x0000 0044
2	9	settable	RTC_TAMP_STAMP /CSS_LSE	RTC Tamper or TimeStamp /CSS on LSE through EXTI line 19 interrupts	0x0000 0048
3	10	settable	RTC_WKUP	RTC Wakeup timer through EXTI line 20 interrupt	0x0000 004C
4	11	settable	FLASH	Flash global interrupt	0x0000 0050
5	12	settable	RCC	RCC global interrupt	0x0000 0054
6	13	settable	EXTI0	EXTI Line0 interrupt	0x0000 0058
7	14	settable	EXTI1	EXTI Line1 interrupt	0x0000 005C
8	15	settable	EXTI2	EXTI Line2 interrupt	0x0000 0060
9	16	settable	EXTI3	EXTI Line3 interrupt	0x0000 0064

In the following labs, you will learn how to enable the interrupt and overwrite the interrupt handler.

在以下實驗中，您將學習如何啟用中斷及如何覆寫中斷處理程序。

3. Steps 實驗步驟

3.1. SysTick timer interrupt

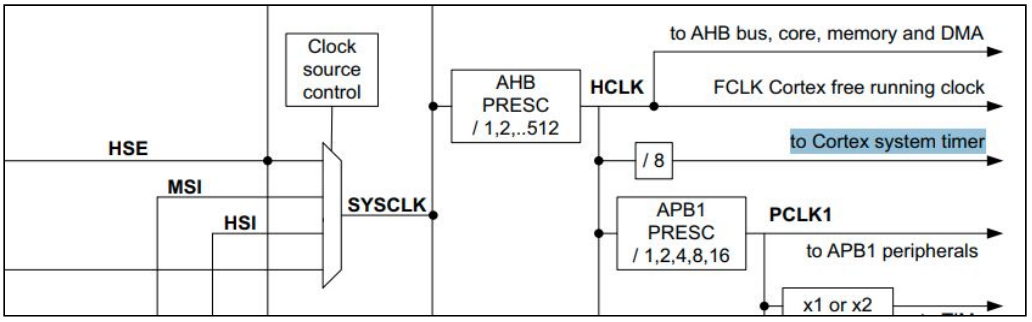


Fig1. SysTcik Timer and clock tree

Requirement:

Please set HSI as the clock source and set the corresponding registers so that the SysTick Timer (STK) generates an "Exception Request" every 3 seconds. Next, write an exception handler which changes the LED state when exceptions occur. If you set up correctly, the LED should be dark for 3 seconds and light for 3 seconds repeatedly.

請設置 HSI 作為時鐘源並設置相應的暫存器，使 SysTick Timer (STK) 每 3 秒生成一個 “Exception request”。接著，編寫異常處理程序，它會在每次發生異常時更改 LED 的狀態。如果設置正確，LED 應該暗 3 秒鐘，然後亮 3 秒鐘重複的變換。

main.c

```
void GPIO_init() {}
void SystemClock_Config() {}
void SysTick_config() {}
void SysTick_Handler() {}

int main(){
	GPIO_init();
	SystemClock_Config();
	SysTick_config();
	while(1);
}
```

Note:

To set up the SysTick registers please refer to PM0214 Programming manual Sec 4.5.

設定 SysTcik 暫存器請參考PM0214 Programming manual Sec 4.5.

count down / 24-bit reload value

when the processor is halted for debugging the counter does not decrement

HSI16 : 16MHz ÷2 → HCLK : 8MHz ÷8 → systick : 1 MHz

1秒 ÷ $\left(\frac{1}{1\times10^{6}}\right) = 10^{6}$

我 10^{6} = 1秒 ∋ 3×10^{6} = 3秒

3.2. Multiple External Interrupt

Requirement:

Set the input pins of the keypad as the external interrupt source. When buttons are all released, an LED will keep lighting. When any button is pressed, an interrupt is triggered, which causes the LED to flash the same times as the number pressed on the keypad (off and on for 0.5 seconds each is one flash). After the flash, the LED will return to a lighting state.

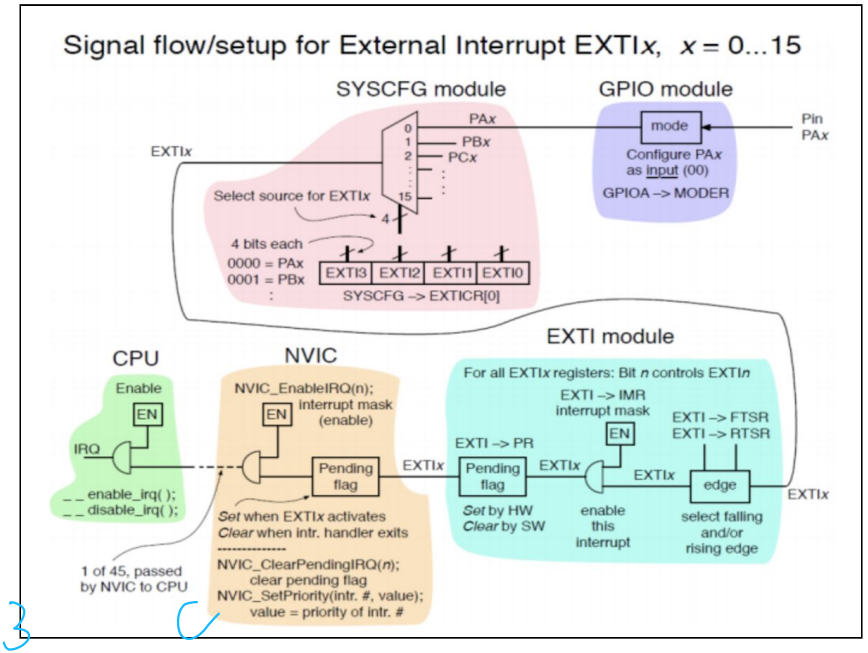
設定 keypad 的 input 腳為外部中斷源。當按鍵均釋放時，一顆 LED 會保持恆亮。當按下按鍵時會觸發中斷，使 LED 產生與 keypad 按下數字相同次數的閃爍 (亮、暗各保持 0.5秒為一次閃爍)，閃爍結束後 LED 回到恆亮的狀態。

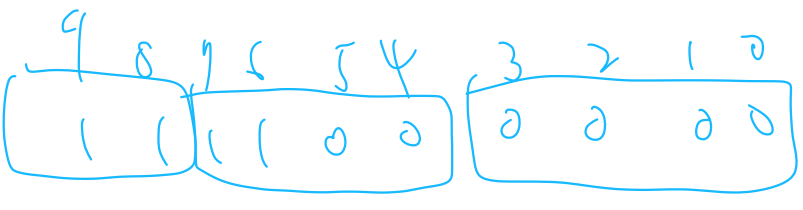
	X0	X1	X2	X3
Y0	1	2	3	10
Y1	4	5	6	11
Y2	7	8	9	12
Y3	15	0	14	13

main.c

```
void init_GPIO() {}
void EXTI_config() {}
void NVIC_config() {}
void EXTIx_IRQHandler() {}
int main() {
    NVIC_config();
    EXTI_config();
    init_GPIO();
}
```

Hint:





It has several steps to enable the interrupts coming from GPIO pins. First, you must set GPIO to Input mode. Next, select the EXTI source and set up the interrupt mask. Third, enable the interrupt pending of NVIC, and finally, overwrite the interrupt handler.

它有幾個步驟來啟用來自 GPIO 引腳的中斷。首先，必須將 GPIO 設置為 Input 模式。接下來，選擇 EXTI 源並設置中斷屏蔽。 第三，啟用 NVIC 的中斷緩存，最後覆蓋中斷處理程序。

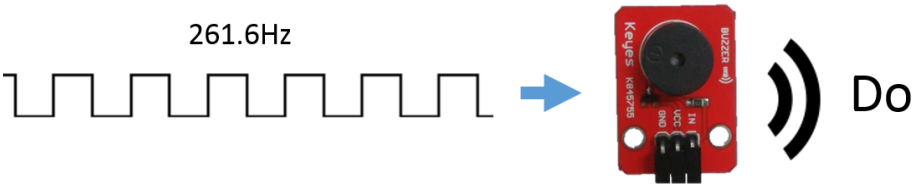
If you skip any steps, your processor will not be able to receive interrupt signals. Please check the I/O register panel frequently to ensure that each module has been set correctly

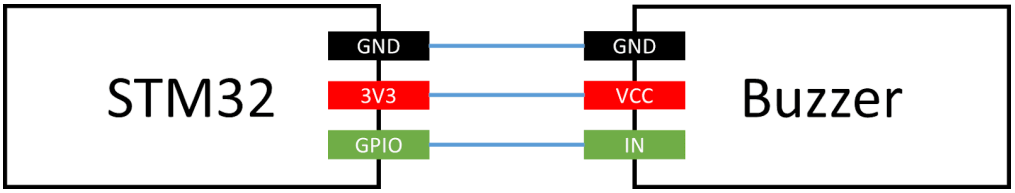
如果您跳過任何步驟，則您的處理器將無法接收中斷信號。 請頻繁查看 I/O 暫存器面板，已確保每個模組都有被正確的設置

3.3. Alarm Clock

BUZZER 蜂鳴器

The buzzer is divided into the active (self-excited) buzzer and the passive (excited) buzzer. The active buzzer directly designs the drive circuit into the buzzer, so it is only necessary to provide a DC voltage to make a sound, but the disadvantage is that the frequency of the sound cannot be changed. The external buzzer needs to provide an oscillating waveform to make a sound, and the frequency of the sound is the frequency of the input wave. Our LAB is using a passive buzzer.
蜂鳴器分為有源（自激式）蜂鳴器和無源（他激式）蜂鳴器。有源蜂鳴器將驅動電路直接設計到蜂鳴器中，因此只需提供直流電壓就可以發出聲音，但其缺點是聲音的頻率無法更改。無源蜂鳴器外部需提供震盪波形才會發出聲音，其聲音的頻率就是輸入波的頻率。我們這次 LAB 使用的是無源蜂鳴器。





The buzzer's VCC is connected to 3.3V, GND is connected to GND, and IN is connected to the GPIO pin.
蜂鳴器的VCC接3.3V、GND接GND、IN接GPIO腳位。

Requirement:

Please use the SysTick timer, user button, and buzzer to design a simple alarm clock. Users can press the numbers on the keypad to decide how many seconds need to wait before the clock starts to alarm. After the clock starts to alarm, users can then click the user button to stop the buzzer, and the alarm clock will keep silent until the next input from the keypad.

請使用 SysTick 計時器、使用者按鈕和蜂鳴器設計一個簡單的鬧鐘。使用者可以按壓鍵盤上的數字來決定時鐘開始響之前需要等待幾秒鐘。時鐘開始響後，使用者可以點擊使用者按鈕停止蜂鳴器，並且鬧鐘將保持靜音，直到來自鍵盤的下一次輸入。

Note:

We assume that only one number on the keypad will be pressed at the same time and any inputs while the clock counting is not allowed.

我們假設同時只會按下鍵盤上的一個數字，並且不允許在計時器倒數時進行任何輸入。

	X0	X1	X2	X3
Y0	1	2	3	10
Y1	4	5	6	11
Y2	7	8	9	12
Y3	15	0	14	13