Short Report Mini project FRDM-K64F ต่อเชื่อมกับโมดูล Multi-functional shield

รายวิชา EL-314: Microprocessor Laboratory

จัดทำโดย :

1. พงศภัค สุวรรณดี รหัสนักศึกษา 1630902953

2. วรากร บัวชุม รหัสนักศึกษา 1630903100

อาจารย์ที่ปรึกษา :

ผศ.ดร.วิศาล พัฒน์ชู(Asst.Prof.Dr. Wisarn Patchoo)

อ.สุรเชษฐ์ โทวราภา (Aj. Surachad Tohvarapa)

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยกรุงเทพ

หลักในการเลือก **Clock Gate**

Clock Gate Register

12.2.12 System Clock Gating Control Register 5 (SIM_SCGC5)

Address: 4004_7000h base + 1038h offset = 4004_8038h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0													1	0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0		PORTE	PORTD	PORTC	PORTB	PORTA	1	0	0	0	0		1		LPTMR
W																
Reset	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1	0

set port ที่ต้องการใช้เป็น 1 ซึ่งใน project มีการใช้ port A,B,C,D

หลักในการ **Set GPIO**

ให้ดูว่าขาที่เราอยากให้ทำประเภทที่เป็น Input/output ว่าอยู่ขาที่เท่าไหร่ และ set ค่าเป็น 0x100

11.5.1 Pin Control Register n (PORTx_PCR*n*)

Address: Base address + 0h offset + (4d × i), where i=0d to 31d

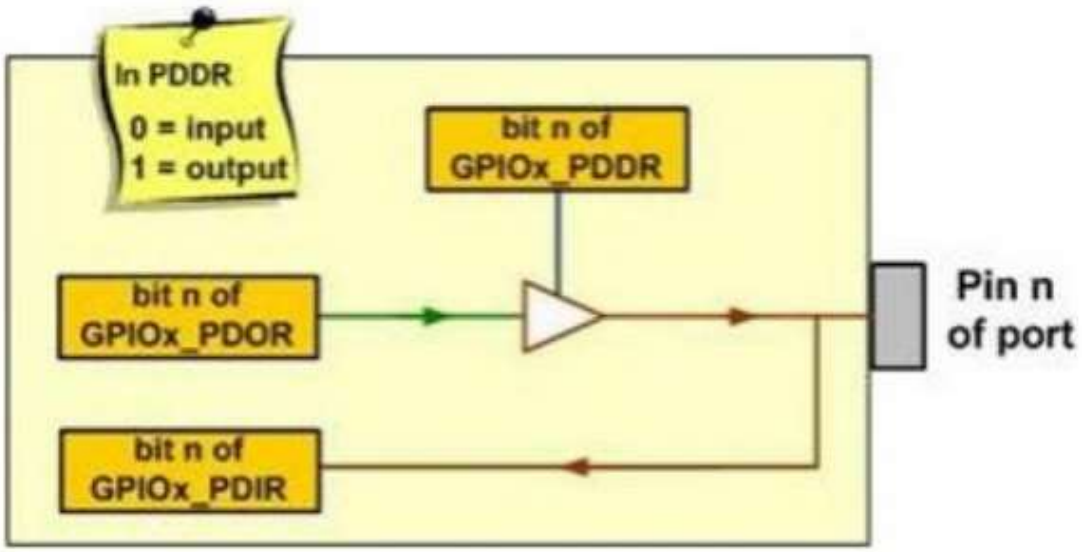
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0							ISF	0				IRQC			
W								w1c								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LK	0				MUX			0	DSE	ODE	PFE	0	SRE	PE	PS
W																
Reset	0	0	0	0	0	*	*	*	0	*	0	*	0	*	*	*

* Notes:

หลักในการ **Set PDDR**

PDDR เป็นการกำหนดให้ขานั้นทำหน้าที่เป็น Input หรือ Output โดยถ้าอยากให้เป็น input ให้เป็น 0 อยากให้เป็น Output ให้เป็น 1



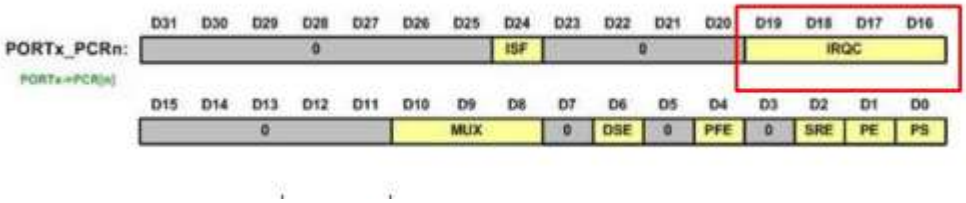
หลักในการ **Set Interrupt**

เปิด/ปิดการใช้งานของ Global Interrupt

```
__disable_irq(); /* Disable interrupt Globally */

__enable_irq(); /* Enable interrupt Globally */
```

เปิด/ปิดการใช้งาน Interrupt ที่ขาของ MCU ที่ต้องการผ่านบิต D19 - D16 ของรีจิสเตอร์ PORTx_PCRn



-ที่ตรงนี้สามารถ Set ให้เป็น rising edge หรือ falling edge ได้

เปิด/ปิดการใช้งาน Interrupt ที่ต้องการที่โมดูล NVIC

หากต้องการเปิด/ปิด Interrupt ของพอร์ต GPIOB ให้ไปกำหนดที่รีจิสเตอร์ NVIC- >ISER1 ในบิตตำแหน่งที่ 60 % 32 = 28 โดย 60 คือเลข IRQ

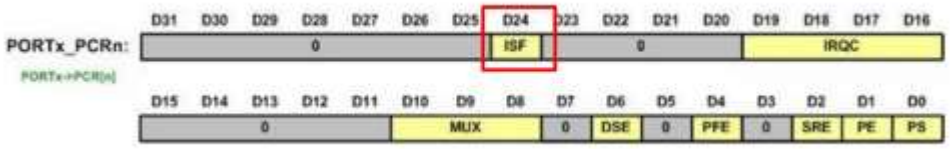
IRQและเลข 1 หลังISER ก็สามารถดูได้จากตาราง

ตารางที่ 6.3 Interrupt Vector Table, IRQ ของ MCU ตระกูล K64 ของบริษัท NXP (ต่อ)

[ตาราง 3-5 NXP K64 Series Reference Manual]

Address	Vector	IRQ[1]	NVIC non-IPR register number [2]	NVIC IPR register number [3]	Source module	Source description
0x0000_00D0	52	36	1	9	UART2	Single interrupt vector for UART error sources
0x0000_00D4	53	37	1	9	UART3	Single interrupt vector for UART status sources
0x0000_00D8	54	38	1	9	UART3	Single interrupt vector for UART error sources
0x0000_00DC	55	39	1	9	ADC0	—
0x0000_00E0	56	40	1	10	CMP0	—
0x0000_00E4	57	41	1	10	CMP1	—
0x0000_00E8	58	42	1	10	FTM0	Single interrupt vector for all sources
0x0000_00EC	59	43	1	10	FTM1	Single interrupt vector for all sources
0x0000_00F0	60	44	1	11	FTM2	Single interrupt vector for all sources
0x0000_00F4	61	45	1	11	CMT	—
0x0000_00F8	62	46	1	11	RTC	Alarm interrupt
0x0000_00FC	63	47	1	11	RTC	Seconds interrupt
0x0000_0100	64	48	1	12	PIT	Channel 0
0x0000_0104	65	49	1	12	PIT	Channel 1
0x0000_0108	66	50	1	12	PIT	Channel 2
0x0000_010C	67	51	1	12	PIT	Channel 3
0x0000_0110	68	52	1	13	PDB	—
0x0000_0114	69	53	1	13	USB OTG	—
0x0000_0118	70	54	1	13	USB Charger Detect	—
0x0000_011C	71	55	1	13	—	—
0x0000_0120	72	56	1	14	DAC0	—
0x0000_0124	73	57	1	14	MCG	—
0x0000_0128	74	58	1	14	Low Power Timer	—
0x0000_012C	75	59	1	14	Port control module	Pin detect (Port A)
0x0000_0130	76	60	1	15	Port control module	Pin detect (Port B)
0x0000_0134	77	61	1	15	Port control module	Pin detect (Port C)
0x0000_0138	78	62	1	15	Port control module	Pin detect (Port D)
0x0000_013C	79	63	1	15	Port control module	Pin detect (Port E)
0x0000_0140	80	64	2	16	Software	Software interrupt[4]
0x0000_0144	81	65	2	16	SPI2	Single interrupt vector for all sources
0x0000_0148	82	66	2	16	UART4	Single interrupt vector for UART status sources
0x0000_014C	83	67	2	16	UART4	Single interrupt vector for UART error sources

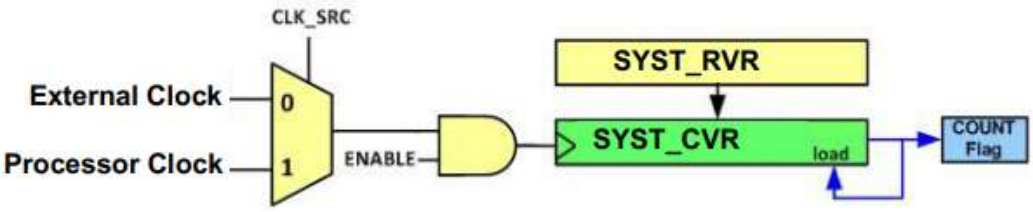
เมื่อเกิด Interrupt ขึ้น ณ ขาที่ n ของพอร์ต GPIOx ค่าบิต ISF ของ PORTx_PCRn จะถูกเซตให้เป็น 1



ตราบใดที่ ISFR ถูกเซ็ตเป็น1 โปรแกรม ISR ของ Interrupt สำหรับพอร์ต GPIOx จะถูก เรียกใช้งานตลอดเวลา ดังนั้นต้องทำการเขียนโดยการเขียน ลอจิก 1 ลงไปที่งรีจิสเตอร์ PORTx_ISFR ซึ่งจะส่งผลให้บิต ISF รีจิสเตอร์ PORTx_PCRn ถูกเคลียร์

หลักในการ Systick Timer

ไทม์เมอร์SysTick ประกอบด้วยวงจรนับลง (Down Counter) ขนาด 24 บิต ซึ่งถูกขับด้วยแหล่งกำเนิดสัญญาณนาฬิกาหรือวงจรออสซิสเลเตอร์ภายใน ซึ่งจะทำการนับลงเริ่มตั้งแต่ค่ามากที่สุดจนกระทั่งถึง 0 โดยเมื่อนับได้ค่า 0 แล้ว ที่สัญญาณนาฬิกาถัดไป จะเกิดการ underflow ขึ้น ทำให้"COUNT" flag เปลี่ยนจาก 0 เป็น 1 เมื่อ COUNT flag มีค่าเท่ากับ 1 ไทม์เมอร์SysTick จะทำการโหลดค่าเริ่มต้นเพื่อเริ่มการนับครั้งใหม่ หลังจากนั้น COUNT flag จะถูกเคลียร์ให้เป็น 0 แล้วกระบวนการนับลงจะเกิดขึ้นเหมือนเดิมอีกครั้งซ้ำๆไปเรื่อยๆ ค่าเริ่มต้นสำหรับการนับ สามารถกำหนดได้โดยมีค่าอยู่ระหว่าง 0x000000 และ 0xFFFFFF



รูปที่ 8.1

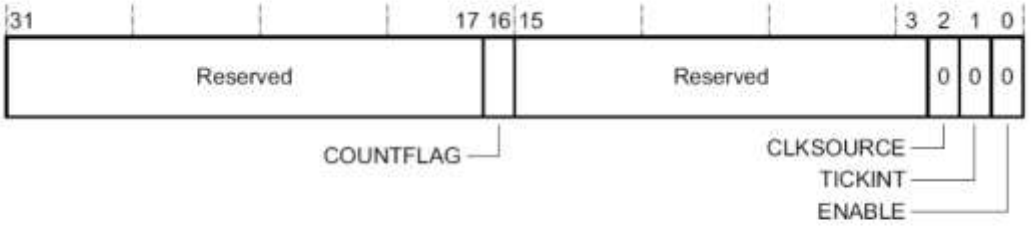
ไทม์เมอร์Systick จะถูกใช้ในการสร้างการหน่วงเวลา1 ms ดังนั้นค่าของรีจิสเตอร์SYST_RVRจะถูกกำหนดให้มีค่าเท่ากับ

$$\frac{20,480,000}{\text{SYST\_RVR}} = \frac{1}{0.001} \quad \Rightarrow \quad \text{SYST\_RVR} = 20,480,000 \times 0.001 = 20,480 = 0 \times 5000_{16}$$

ในส่วนของ Systick->Val ใช้อ่านค่าที่นับได้ณเวลานั้น

8.1.1 รีจิสเตอร์ Systick Control and Status (SYST_CSR)

รีจิสเตอร์นี้จะอยู่ที่ตำแหน่งหน่วยความจำ 0xE000E010 รีจิสเตอร์ SYST_CSR จะถูกใช้ในการกำหนดพารามิเตอร์สำหรับการทำงานของไทม์เมอร์ SysTick ผ่านบิตควบคุมดังรูปที่ 8.2

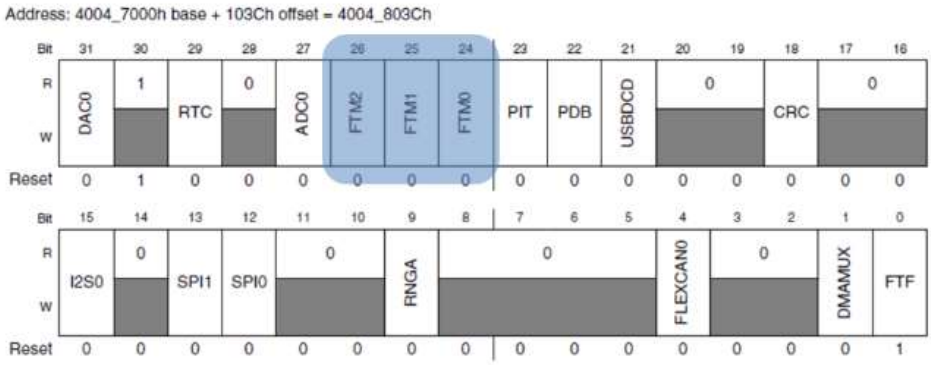


รูปที่ 8.2

หลักในการ **FTM Timer**

1. เปิดใช้งานแหล่งกำเนิดสัญญาณนาฬิกาของ FTMn

ก่อนหน้าที่จะใช้งาน FTM เราจะต้องทำการเปิดการใช้งานแหล่งกำเนินสัญญาณนาฬิกาของ FTM ก่อน ซึ่งสามารถทำได้ผ่านรีจิสเตอร์ SIM_SCGC6 ที่บิต D26 – D24 ขึ้นอยู่กับ FTM ที่เลือก ดังแสดงในรูปที่ 9.2 โดยกำหนดเปิดการใช้งานด้วยการเซตให้บิตดังกล่าวมีค่าเท่ากับ 1



(a)

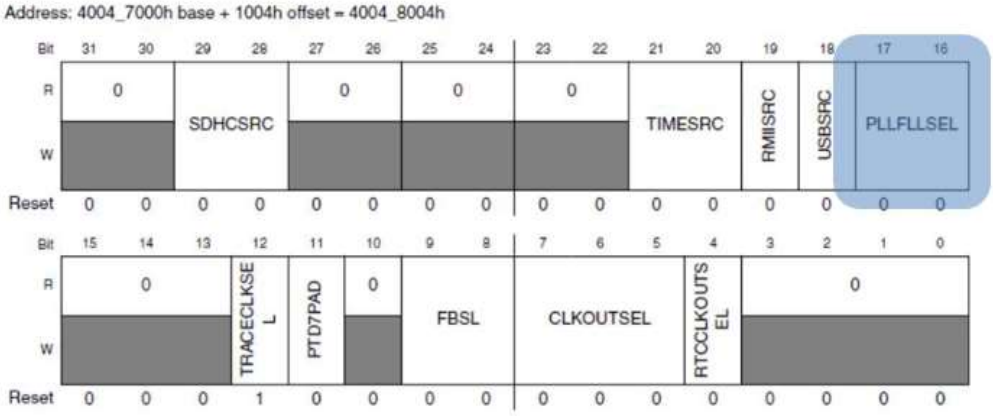
26 FTM2	FTM2 Clock Gate Control This bit controls the clock gate to the FTM2 module. 0 Clock disabled 1 Clock enabled
25 FTM1	FTM1 Clock Gate Control This bit controls the clock gate to the FTM1 module. 0 Clock disabled 1 Clock enabled
24 FTM0	FTM0 Clock Gate Control This bit controls the clock gate to the FTM0 module. 0 Clock disabled 1 Clock enabled

(b)

รูปที่ 9.2 บิต D26 - D24 ที่รีจิสเตอร์ SIM_SCGC6 สำหรับเปิดใช้งานแหล่งกำเนิดสัญญาณนาฬิกาของ FTM

2. เลือกแหล่งกำเนิดสัญญาณนาฬิกาสำหรับ FTM

แหล่งกำเนิดสัญญาณนาฬิกาที่จะใช้สำหรับของ FTM สามารถเลือกได้ผ่านบิต D17 - D16 ของรีจิสเตอร์ SIM_SOPT2



(a)

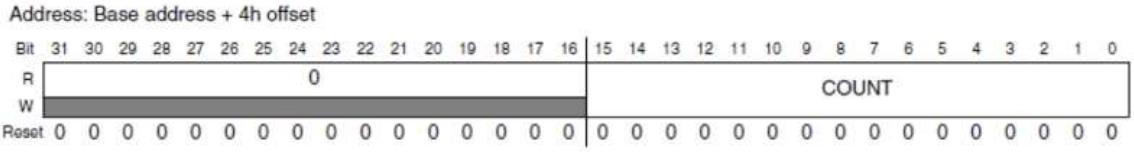
17–16 PLLFLLSEL	PLL/FLL clock select Selects the high frequency clock for various peripheral clocking options. 00 MCGFLLCLK clock 01 MCGPLLCLK clock 10 Reserved 11 IRC48 MHz clock

(b)

รูปที่ 9.3 บิต D17 – D16 ที่รีจิสเตอร์ SIM_SOPT2 สำหรับเลือกแหล่งกำเนิดสัญญาณนาฬิกาของ FTM

3. กำหนดค่าเริ่มต้นในการนับสำหรับวงจรนับของ FTM

FTM แต่ละชุด ทำงานโดยอาศัยวงจรนับขนาด 16 บิต ค่าเริ่มต้นของวงจรนับนี้ จะกำหนดผ่านค่าของรีจิสเตอร์ FTMn_CNTIN เมื่อ FTM เริ่มทำงาน วงจรนับจะเริ่มนับจากค่าเริ่มต้น โดยค่าที่นับได้จะเก็บอยู่ที่รีจิสเตอร์ FTMn_CNT สังเกตว่า 16 บิตแรกของ FTMn_CNT และ FTMn_CNTIN เท่านั้นที่จะถูกใช้งาน



รูปที่ 9.4 บิต D15 - D0 ของรีจิสเตอร์ FTMn_CNT และ FTMn_CNTIN

4. กำหนดค่าสิ้นสุดของการนับสำหรับวงจรนับของ FTM ที่รีจิสเตอร์ FTMn_MOD

ในกรณีวงจรนับ FTM ถูกกำหนดให้นับขึ้น (หรือนับลง) เมื่อค่าที่รีจิสเตอร์ FTMn_CNT เพิ่มขึ้น (หรือลดลง) เรื่อยๆจากค่าเริ่มต้นจนมีค่าเท่ากับค่าในรีจิสเตอร์ FTMn_MOD แฟล็ก overflow (TOF) ที่รีจิสเตอร์ FTMn_SC (จะกล่าวต่อไป) จะถูกเซ็ตเป็น 1 หลังจากนั้น รีจิสเตอร์ FTMn_CNT จะถูกเซ็ตเท่ากับค่าเริ่มต้นโดยอ่านค่ามาจากรีจิสเตอร์ FTMn_CNTIN ดังที่อธิบายไปก่อนหน้านี้ แล้วการนับก็จะดำเนินต่อไปในลักษณะเดียวกัน

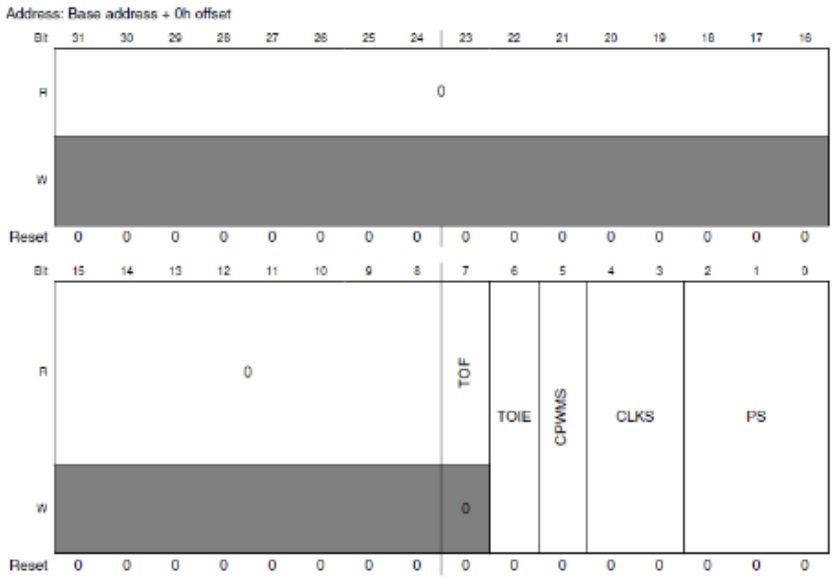
Address: Base address + 8h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																MOD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

รูปที่ 9.5 บิต D15 - D0 ของรีจิสเตอร์ FTMn_MOD

5. กำหนดพารามิเตอร์ต่างๆที่รีจิสเตอร์ FTMn Status Control (FTMn_SC)

FTM แต่ละชุด จะมีรีจิสเตอร์ที่ใช้ตรวจสอบสถานะและควบคุม เรียกว่า FTMn_SC แสดงดังรูปที่ โดยที่ในขั้นตอนของการเริ่มต้นการใช้งานของ FTM จะต้องปิดการใช้งานวงจรนับของ FTM ก่อน เนื่องจาก การปรับเปลี่ยนค่าพารามิเตอร์ต่างๆของ FTM อาจจะส่งผลให้มีการผิดพลาดในการทำงานของวงจรนับของ FTM ได้ เราสามารถปิด/เปิด การทำงานวงจรนับของ FTM ได้โดยเซตที่บิต D4 และ D3 (CLKS) ของ FTMn_SC



รูปที่ 9.7 บิตพารามิเตอร์ของรีจิสเตอร์ FTMn_SC

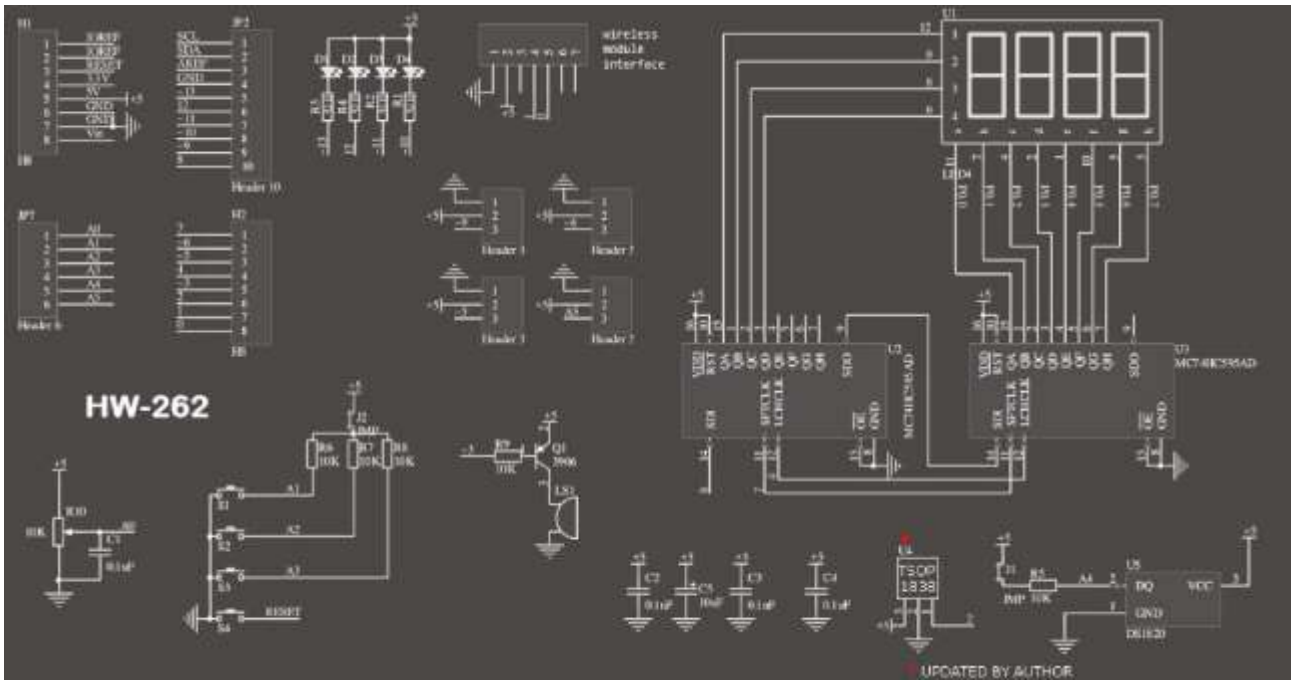
- บิต D7 -- Timer Overflow Flag (TOF) มีค่าเป็น 1 เมื่อ ค่าของ FTMn_CNT มีค่าเท่ากับ FTMn_MOD (ดูรูปที่ 9.1) หากเราต้องการเคลียร์ TOF สามารถทำได้โดยอ่านค่าของรีจิสเตอร์ FTMn_SC แล้วตามด้วยการเขียนค่า 0 ไปที่บิต D7
- บิต D6 -- Timer Overflow Interrupt Enable (TOIE) สำหรับปิด/เปิดการใช้งานของอินเทอร์รัปท์ของไทม์เมอร์ FTM ซึ่งจะเกิดขึ้นเมื่อ TOF = 1
- บิต D5 -- Center-Aligned PWM Select (CPWMS) ใช้สำหรับการกำหนดให้วงจรนับของ FTM ทำการนับขึ้น (CPWMS = 0) หรือ นับลง (CPWMS = 1)
- บิต D4 – D3 -- Clock Source Selection (CLKS) ใช้สำหรับเลือกแหล่งกำเนิดสัญญาณนาฬิกาสำหรับ FTM (ดูรูปที่ 9.1)
    - 00 No clock selected. This in effect disables the FTM counter.
    - 01 System clock
    - 10 Fixed frequency clock
    - 11 External clock
- บิต D2 – D0 -- Prescale Factor Selection (PS) ใช้ในการกำหนดค่า Prescaler ที่ใช้ลดความถี่ของสัญญาณนาฬิกาจากแหล่งกำเนิด เพื่อใช้สำหรับวงจรนับของ FTM (ดูรูปที่ 9.1)
    - 000 Divide by 1
    - 001 Divide by 2
    - 010 Divide by 4
    - 011 Divide by 8
    - 100 Divide by 16
    - 101 Divide by 32
    - 110 Divide by 64
    - 111 Divide by 128

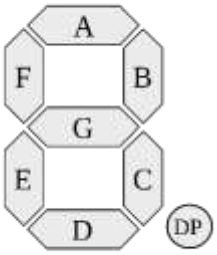
ทั้งนี้ สำหรับการพัฒนาโปรแกรมด้วย Keil MDK และ MCU NXP K64 บนบอร์ดพัฒนา FRDM-K64 แหล่งกำเนิดสัญญาณนาฬิกาโดยปริยายของคอร์จะอยู่ที่ 20.48 MHz (ดูจากไฟล์ system_MK64F12.c) ดังนั้นวงจรนับของ FTM0 จะ overflow ที่ความถี่เท่ากับ

$$\frac{20,480,000}{2^{16}} = \frac{20,480,000}{65,536} = 312.5 \quad \text{Hz.}$$

โดยที่ค่า 2^{16} ในสมการมาจากจำนวนครั้งของการนับทั้งหมดที่เป็นไปได้ของวงจรนับ นั่นคือ เริ่มจากค่าเริ่มต้นเท่ากับ 0 (FTM0_CNTIN = 0) นับจนกระทั่งถึง 0xFFFF (FMT0_MOD = 0xFFFF) ดังนั้น หากเราต้องการให้ GREEN-LED ดับเป็นระยะเวลา 1 วินาที และ ติดเป็นระยะเวลา 1 วินาที เราจะต้องสร้าง For-Loop ในโปรแกรมเพิ่มเติมขึ้นมา เพื่อให้เมื่อสิ้นสุด For-Loop เวลาจะผ่านไป 1 วินาที

หลักในการใช้ **7 segment**



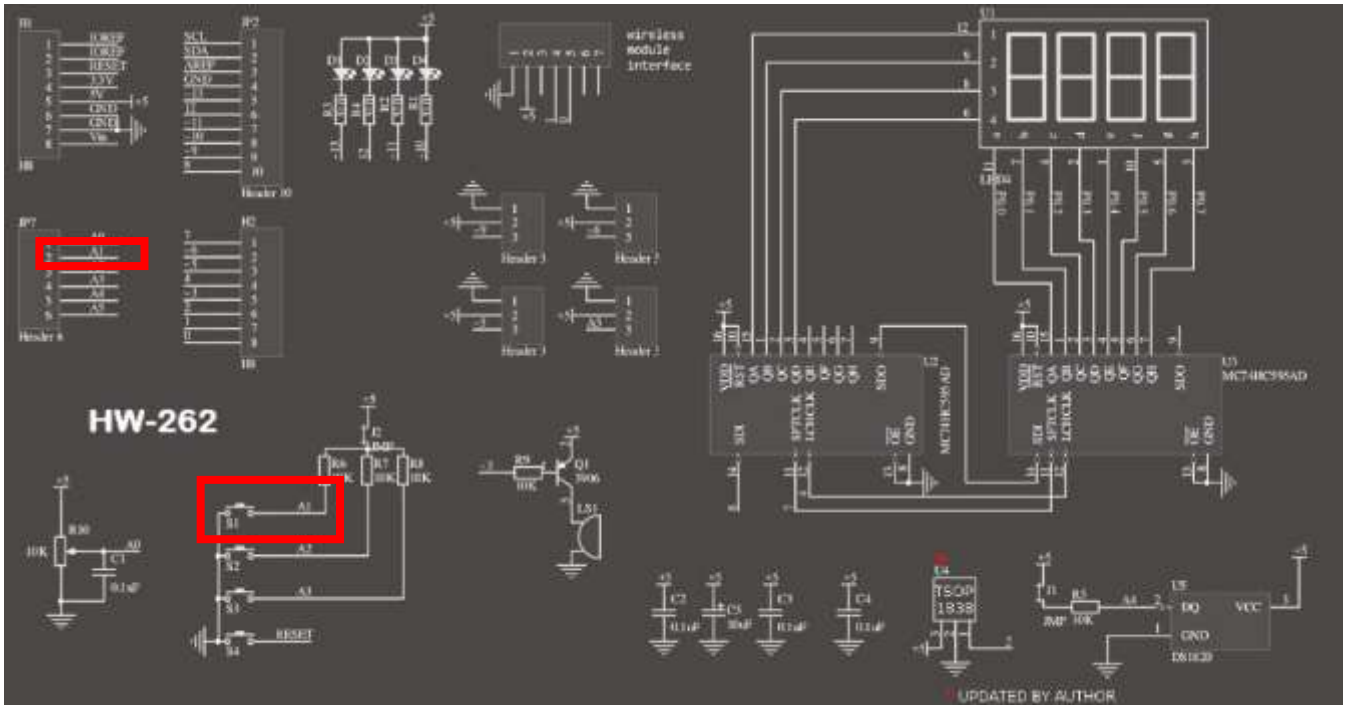


การใช้งานจะต้องส่งบิตไปที่ ic ตัวที่สองก่อน โดยส่งบิต h g f e d c b a ตามลำดับ โดยอยากให้ไฟขีดไหนติดให้ส่งบิต 0 เพราะเป็น Active low หลังจากนั้นส่งบิต h g f e d c b a ไปอีกครั้งให้ ic ตัวแรก เพื่อกำหนดหลักว่าจะให้แสดงหลักไหน 1 2 3 หรือ 4 โดยอยากให้หลักไหนติดให้ส่งบิต 1 เพราะเป็น Active high

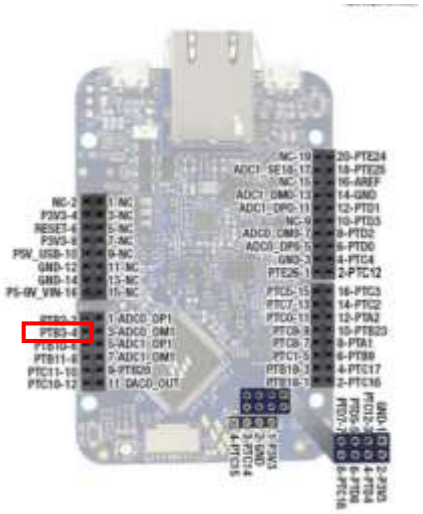
และในการส่งแต่ละบิต จะต้องเปิดปิด SFTCLK ด้วย หลังจากนั้นให้แสดงด้วยการเปิด LCHCLK

หลักในการใช้เลือกใช้ port และ pin ให้ตรงกับ Multifunctional Shield

Schematic Diagram ของ Multifunctional Shield



ดูว่าสิ่งที่เราจะใช้อยู่ที่ขาไหน เช่น S1 เชื่อมต่อกับขา 2 ของซ๊อกเก็ต JP7 และไปดูที่ตัวบอร์ด



ดูเปรียบเทียบกันว่าอยู่ที่ช่องไหน และเราจะรู้ port และ pin ที่เราต้องตั้งค่า

Source Code

```
#include "MK64F12.h"
void delayMs(int n);
int time = 0;
int q;
int zero[8] = {1,1,0,0,0,0,0,0};
int numsegment[5][8] =
{{0,1,0,0,0,0,0,0},{0,1,1,1,1,0,0,1},{0,0,1,0,0,1,0,0},{0,0,1,1,0,0,0,0},{0,0,0,1,1,0,0,1}}; //
dp,g,f,e,d,c,b,a and 0,1,2,3,4 (value)
int seenum[4][8] = {{0,0,0,0,1,0,0,0},{0,0,0,0,0,1,0,0},{0,0,0,0,0,0,1,0},{0,0,0,0,0,0,0,1}};
//position ----,000-,00-0,0-00,-000
void segment(int time);
int main (void){
	SIM->SCGC5 |= (1 << SIM_SCGC5_PORTB_SHIFT) |  (1<<
SIM_SCGC5_PORTD_SHIFT )| (1<< SIM_SCGC5_PORTA_SHIFT ) | (1<<
SIM_SCGC5_PORTC_SHIFT );
	//set GPIO
	//switch
	PORTB -> PCR[3] = 0x100 ;  //S1
	PORTB -> PCR[10] = 0x100 ; //S2
	PORTB -> PCR[11] = 0x100 ; //S3
	//led
	PORTD -> PCR[0] = 0x100 ; //D4
	PORTD -> PCR[1] = 0x100 ; //D1
	PORTD -> PCR[2] = 0x100 ; //D3
	PORTD -> PCR[3] = 0x100 ; //D2
	//buzzer
	PORTA -> PCR[1] = 0x100 ;
	// 7 segment
	PORTB -> PCR[23] = 0x100 ; //LCHCLK
	PORTC -> PCR[3] = 0x100 ; //SFTCLK
	PORTC -> PCR[12] = 0x100 ; //SDI
	//output 7 segment
	PTC -> PDDR |= 0x01008;
	PTB -> PDDR |= 0x0800000;
	//input sw
	PTB -> PDDR &= ~0xC08;
	//output led
	PTD -> PDDR |= 0x0F;

	//sys timmer
	SysTick->LOAD = 204800 - 1; //1 ms
	SysTick->CTRL = 5; // enable counter, - no interrupt,- use system clock  bit0-2
	//FTM Timmer
	SIM->SCGC6 |= 0x01000000;  //enable clock gate to FTM0 bit24 set FTM1-25
FTM2-26 (selection)
```

```
	SIM->SOPT2 |= 0 ; // enable MCGFLLCLK clock bit16-17
	FTM0->CNTIN = 0x00; //start 0
	FTM0->MOD = 0xFFFF; //end 65536 time = 3.2 ms
	FTM0->SC = 0x08; // bit 3 system clock

	PTC -> PDOR |= 0x01008;
	PTB -> PDOR |= 0x0800000;

	//interup switch
	__disable_irq(); //disable all irq
	PORTB->PCR[3] &= ~0xF0000; // clear interupt section bit16-19 set 0000
	PORTB->PCR[3] |= 0xA0000;  // enable falling edge bit16-19 set 1010
	PORTB->PCR[10] &= ~0xF0000;
	PORTB->PCR[10] |= 0xA0000;
	PORTB->PCR[11] &= ~0xF0000;
	PORTB->PCR[11] |= 0xA0000;
	NVIC->ISER[1] |= 0x10000000; //enable NVIC-portB 60 mod 32 = bit28
	__enable_irq();  //enable all irq
	//set all led = 1
	PTD -> PDOR |= 0x0F;


	while(1){
				if (time == 4){
					PTA->PDOR = (SysTick->VAL)>>16;   // buzzer in
systick
					segment(time);
					delayMs(1);
					time = time -1;		//time - 1
				}
				else if (time == 3){
					PTD -> PDOR |= 0x02;
					PTA->PDOR = (SysTick->VAL)>>14;
					segment(time);
					delayMs(1);
					time = time -1;
				}
				else if (time == 2){
					PTD -> PDOR |= 0x08;
					PTA->PDOR = (SysTick->VAL)>>12;
					segment(time);
					delayMs(1);
					time = time -1;
				}
				else if (time == 1){
					PTD -> PDOR |= 0x04;
					PTA->PDOR = (SysTick->VAL)>>10;
```

```
                                        segment(time);
                                        delayMs(1);
                                        PTA->PDOR = (SysTick->VAL)>>8;
                                        time = time - 1;
                                }
                                else if(time <= 0){
                                PTA -> PDDR &= ~0x02;
                                PTD -> PDOR |= 0x0F;

                                segment(time);
                                }
        }

        }
void PORTB_IRQHandler(void) {
        if ((PTB -> PDIR & 0x08)==0){   //press S1
                        PTD -> PDOR |= 0x0F;   //close all led
                        PTD -> PDOR &= ~0x0F;  //led = 4
                        PTA -> PDDR = 0x02;
                        time = 4;
                        PORTB->ISFR = 0x008; //clear ISF portB pin3
                }
        if ((PTB -> PDIR & 0x400)==0){ //press S2
                        PTD -> PDOR |= 0x0F;   //close all led
                        PTD -> PDOR &= ~0x0D;  //led = 3
                        PTA -> PDDR = 0x02;
                        time = 3;
                        PORTB->ISFR = 0x400; //clear ISF portB pin10
                }
        if ((PTB -> PDIR & 0x800)==0){ //press S3
                        PTD -> PDOR |= 0x0F;  //close all led
                        PTD -> PDOR &= ~0x05; //led = 2
                        PTA -> PDDR = 0x02;
                        time = 2;
                        PORTB->ISFR = 0x800; //clear ISF portB pin11
                }
}

void delayMs(int n) {
        int i;
        for(i = 0; i < (320*n); i++) { //320 = 1 sec
        while((FTM0->SC & 0x80) == 0) { //FTM0->sc = 1 if CNTIN = Mod,TOF=1
                segment(time);
        }
        FTM0->SC &= ~(0x80); //clear TOF

        }
```

```
}

void segment(int time){
        for (int i= 0 ; i < 4;i++){
                PTB -> PCOR = 0x0800000; //LCHCLK = 0
                if(i != 3){
                        for (q= 0 ; q < 8;q++){                 //bit data
                                PTC -> PDOR = (zero[q]) << 12; //SDI
                                PTC -> PSOR = 0x08; //SFTCLK
                                PTC -> PCOR = 0x08;
                        }
                }else{
                        for (q= 0 ; q < 8;q++){                 //bit data
                                PTC -> PDOR = (numsegment[time][q]) << 12; //SDI
                                PTC -> PSOR = 0x08; //SFTCLK
                                PTC -> PCOR = 0x08;
                        }
                }
                        for (q = 0; q < 8; q++){                //bit position
                                PTC -> PDOR = (seenum[i][q]) << 12; //SDI
                                PTC -> PSOR = 0x08; //SFTCLK
                                PTC -> PCOR = 0x08;
                        }
                        PTB -> PSOR = 0x0800000;  //LCHCLK = 1
                }

}
```

Link online GDB

https://onlinegdb.com/Cf10oYru5I

การทำงานของโค้ด

- เมื่อกดSwitch 1 ลำโพงจะดัง สุ่มความดังเบาจากค่าsystickที่นับได้ ไฟLED ติดจากบนสุดลงล่างสุดและดับ 7segment โชว์เลข 4.000และนับถอยหลังจนเหลือ0
- เมื่อกดSwitch 2 ลำโพงจะดัง สุ่มความดังเบาจากค่าsystickที่นับได้ ไฟLED ติดจากดวงที่2(นับจากบนลงล่าง)จนถึงดวงที่4และดับ 7segment โชว์เลข 3.000และนับถอยหลังจนเหลือ0
- เมื่อกดSwitch 3 ลำโพงจะดัง สุ่มความดังเบาจากค่าsystickที่นับได้ ไฟLED ติดจากดวงที่3(นับจากบนลงล่าง)จนถึงดวงที่4และดับ 7segment โชว์เลข 2.000และนับถอยหลังจนเหลือ0
- **แต่ละปุ่มสามารถinterruptกันได้**