介面實驗

實驗四

資料擷取系統 TIMER

配合 ADC 使用

班級：機械 2A

學號：108303013

姓名：黃鉦淳

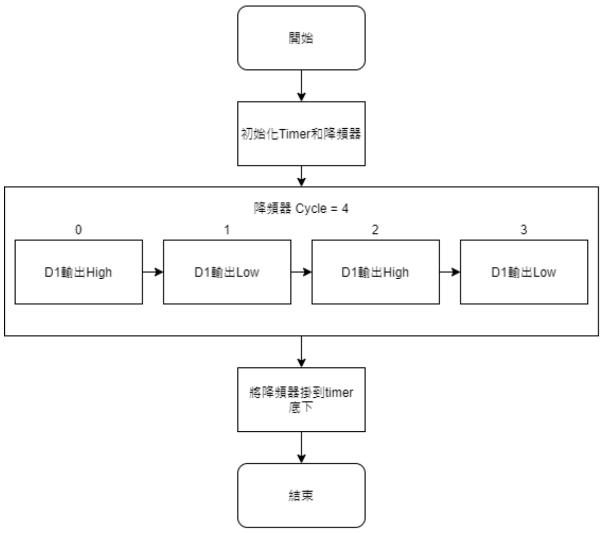
日期：110/7/26

介面實驗工作日誌

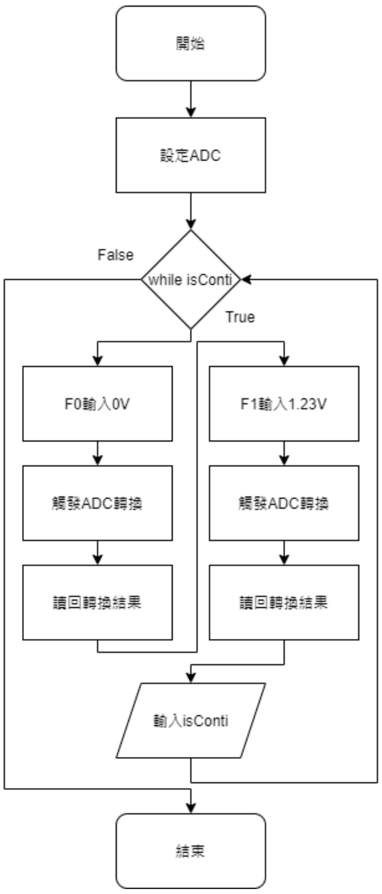
實驗四　　　　　　　　　　　110 年 7 月 26 日

組別		姓名	黃鉦淳	學號	108303013
實驗起始時間	110/7/19			費時	7 天
實驗結束時間	110/7/23				
所遭遇問題	一直不會使用 pipeline 的函式或是包好的函式以及最後的結果跟講義上能做到的有出入				
解決方法	狂問學長				
完成項目及心得：	終於寫完了				
調查	☐是否有看課程講解影片 是否實用？有何建議？		☐是否有看實驗教學影片 是否實用？有何建議？		

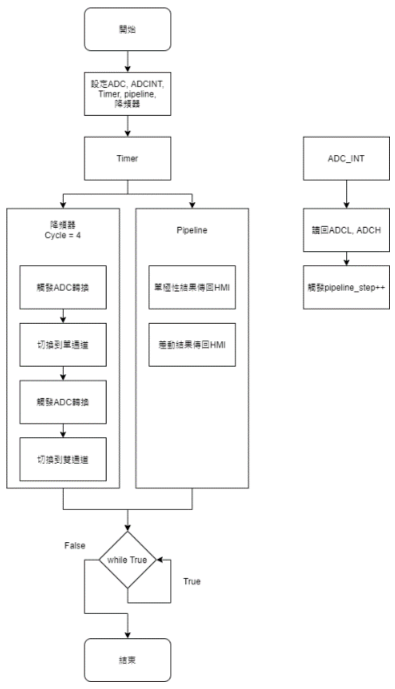
一、流程圖


開始
初始化Timer和降頻器
降頻器 Cycle = 4
0
1
2
3
D1輸出High
D1輸出Low
D1輸出High
D1輸出Low
將降頻器掛到timer底下
結束

實驗一


開始
設定ADC
while isConti
False
True
F0輸入0V
觸發ADC轉換
讀回轉換結果
F1輸入1.23V
觸發ADC轉換
讀回轉換結果
輸入isConti
結束

實驗二



實驗三

二、程式碼

Language : C

實驗一

```
#include "c4mlib.h"
#define N 2

void output_High();
void output_Low();
void timer3_init();

int main()
{
    C4M_DEVICE_set();
    DDRD = 0x01;

    //設定Timer
    TIM3_HW_LAY();
}

    hardware_set(&TIM1_3HWSet_str);

    //設定Timer中斷
    TIMHWINT_LAY(Tim3, 3, 1);

    timer3_init();

    //設定降頻器配置
    uint16_t period[2 * N] = {1000, 2000, 3000, 4000};

    FREQREDU_LAY(redu, 2 * N, 2 * N, &OCR3A, 2, period);

    uint8_t ID[4];

    ID[0] = FreqRedu_reg(&redu, &output_High, NULL, 1, 0);
    ID[1] = FreqRedu_reg(&redu, &output_Low, NULL, 1, 1);

    ID[2] = FreqRedu_reg(&redu, &output_High, NULL, 1, 2);
    ID[3] = FreqRedu_reg(&redu, &output_Low, NULL, 1, 3);

    FreqRedu_en(&redu, ID[0], 1);
    FreqRedu_en(&redu, ID[1], 1);
    FreqRedu_en(&redu, ID[2], 1);
    FreqRedu_en(&redu, ID[3], 1);

    uint8_t TaskID = HWInt_reg(&Tim3, &FreqRedu_step, &redu);
    HWInt_en(&Tim3, TaskID, 1);
    sei();

    while (1)
    {
        ;
    }

    return 0;
}

void timer3_init()
```

```
{
	REGFPT(&TCCR3A, 0x03, 0, 0);
	REGFPT(&TCCR3B, 0x18, 3, 1);
	REGFPT(&TCCR3B, 0x07, 0, 5);
	OCR3A = 188;
	REGFPT(&ETIMSK, 0x10, 4, 1);
}

void output_High()
{
	REGFPT(&PORTD, 0x01, 0, 0x01);
}

void output_Low()
{
	REGFPT(&PORTD, 0x01, 0, 0);
}
```

實驗二

```
#include "c4mlib.h"

#define ADC0_FG_DATA_INT                              \
	{                                                  \
		.DDx0_7 = INPUT,                              \
		.MUXn0_4 = ADC_SINGLE_END_0_X1, \
		.REFSn0_1 = ADC_REF_AREF,             \
		.ADLARn = ADC_RES_10BITS,             \
		.ADPSn0_2 = ADC_CLK_DIV_BY2,       \
		.ADFRn = DISABLE,                          \
		.ADENn = ENABLE,                           \
		.ADIEn = ENABLE,                           \
		.Total = 8                                       \
	}

int main()
{
	C4M_STDIO_init();
```

```
//設定內部2.56V
REGFPT(&ADMUX, 0xC0, 6, 3);

//設定10位元轉換
REGFPT(&ADMUX, 0x20, 5, ADC_RES_10BITS);

//設定非連續或觸發轉換
REGFPT(&ADCSRA, 0x20, 5, DISABLE);

//設定ADC接角 F0~4皆輸入
REGFPT(&DDRF, 0x0f, 0, 0);

//設定輸入訊號來源
REGFPT(&ADMUX, 0x1f, 0, ADC_SINGLE_END_0_X1);

//設定致能ADC
REGFPT(&ADCSRA, 0x80, 7, ENABLE);

//設定工作時脈除頻
REGFPT(&ADCSRA, 0x07, 0, ADC_CLK_DIV_BY2);

int isContinued = 1;
int isDoneConverted;
int data0, data1;

printf("start while-loop\n");

while (isContinued)
{
	printf("Set input voltage 0V\n");
	getchar();
	getchar();
	printf("Start Convert ADC\n");
	//觸發ADC轉換
	REGFPT(&ADCSRA, 0x40, 6, 1);

	do
	{
			_delay_ms(100);
			REGFGT(&ADCSRA, 0x10, 4, &isDoneConverted);
		} while (!isDoneConverted);

		printf("End Convert ADC\n");
		REGGET(&ADCL, 2, &data0);
		printf("Data=%d\n", data0);

		printf("Set input voltage 1.23V\n");
		getchar();
		getchar();
		printf("Start Convert ADC\n");
		//觸發ADC轉換
		REGFPT(&ADCSRA, 0x40, 6, 1);

		do
		{
			_delay_ms(100);
			REGFGT(&ADCSRA, 0x10, 4, &isDoneConverted);
		} while (!isDoneConverted);

		printf("End Convert ADC\n");
		REGGET(&ADCL, 2, &data1);
		printf("Data=%d\n", data1);

		printf("Is continuing detect ADC? (1/0)\n");
		scanf("%d", &isContinued);
	}

	return 0;
}
```

實驗三

```
/*
 * waveGenerator
 * frequency : 10Hz
 * amplitude : 1V
```

```
  * offset : 0.5V
*/

#include "c4mlib.h"

#define N 2

void ADC_single_PostPro_step(void *VoidStr_p);
void ADC_diff_PostPro_step(void *VoidStr_p);

void ADC_init();
void timer3_init();

typedef struct
{
      uint16_t *InData_p;              /*Pointer points to the Input Data Source */
      uint16_t *DataList_p;            /* Pointer points to the buffer array */
      uint8_t DataLength;              /* Length of Datalist */
      uint8_t DataCount;               /*Data Count of the data in list*/
      uint8_t TaskId;                  //The TaskId got after registered
      uint8_t NextTaskNum;               //Number of Next Task
      uint8_t *NextTask_p;             //pointer to the List of TaskId for NextTasks
      volatile uint8_t TrigCount; //Triggered Counter
} ADCPostProStr_t;

#define ADCPOSTPRO_LAY(ADCPostPro_str, ListNum, _NextTaskNum,
InDataAdd) \
      uint16_t ADCPostPro_str##_DataList[ListNum];
\
      uint8_t ADCPostPro_str##_NextTaskList[_NextTaskNum];
\
      ADCPostProStr_t ADCPostPro_str =
\

{                                                                          \
            .InData_p = InDataAdd,
\
                  .DataList_p = ADCPostPro_str##_DataList,
\
                  .DataLength = ListNum,
\
                  .DataCount = 0,
\
                  .TaskId = 0,
\
                  .NextTaskNum = _NextTaskNum,
\
                  .NextTask_p = ADCPostPro_str##_NextTaskList,
\
                  .TrigCount = 0}

int main()
{
      C4M_DEVICE_set();

      //設定 Timer
      TIM3_HW_LAY();
      hardware_set(&TIM1_3HWSet_str);

      //設定 Timer3 中斷
      TIMHWINT_LAY(TIMINT_Str, 3, 1);

      timer3_init();

      //設定 ADC
      ADC_HW_LAY();
      hardware_set(&ADCHWSet_str);

      ADC_init();

      //設定 ADC interrupt
      ADCHWINT_LAY(ADCHWINT_Str, 0, 3);
      hardware_set(&ADCHWINT_Str);

      //設定降頻器配置
```

```
    int period[4];
    for (int i = 0; i < sizeof(period) / sizeof(int); i++)
        period[i] = 33;

    FREQREDU_LAY(FreqRedu_Str, 2 * N, 2 * N, &OCR3A, 2, &period);

    unsigned int ADC_read_single_Data;
    unsigned int ADC_read_diff_Data;

    RT_REG_IO_LAY(ADC_single_result_Str, 0, &ADCL, 2, (uint8_t
*)&ADC_read_single_Data);
    RT_REG_IO_LAY(ADC_diff_result_Str, 1, &ADCL, 2, (uint8_t
*)&ADC_read_diff_Data);

    ADCPOSTPRO_LAY(ADC_single_PostPro_Str, 200, 0,
&ADC_read_single_Data);
    ADCPOSTPRO_LAY(ADC_diff_PostPro_Str, 200, 1,
&ADC_read_diff_Data);

    unsigned char one = 1;
    unsigned char two = 2;
    unsigned char eighteen = 18;

    //切換到單通道 ADC2  and GND
    RT_FLAG_IO_LAY(ADC_single_channel_Str, 0, &ADMUX, 0x1f, 0,
&one);
    //觸發單通道
    RT_FLAG_IO_LAY(ADC0_trigger_convert_Str, 1, &ADCSRA, 0x40, 6,
&one);
    //切換到雙通道 ADC2 -> V+, ADC1 -> V-
    RT_FLAG_IO_LAY(ADC_diff_channel_Str, 2, &ADMUX, 0x1f, 0,
&eighteen);
    //觸發雙通道
    RT_FLAG_IO_LAY(ADC12_trigger_convert_Str, 3, &ADCSRA, 0x40, 6,
&one);

    uint8_t freq_TaskID[4];
    freq_TaskID[0] = FreqRedu_reg(&FreqRedu_Str,
&RealTimeFlagPut_step, &ADC_single_channel_Str, 1, 0);
    freq_TaskID[1] = FreqRedu_reg(&FreqRedu_Str,
&RealTimeFlagPut_step, &ADC0_trigger_convert_Str, 1, 1);
    freq_TaskID[2] = FreqRedu_reg(&FreqRedu_Str,
&RealTimeFlagPut_step, &ADC_diff_channel_Str, 1, 2);
    freq_TaskID[3] = FreqRedu_reg(&FreqRedu_Str,
&RealTimeFlagPut_step, &ADC12_trigger_convert_Str, 1, 3);

    FreqRedu_en(&FreqRedu_Str, freq_TaskID[0], ENABLE);
    FreqRedu_en(&FreqRedu_Str, freq_TaskID[1], ENABLE);
    FreqRedu_en(&FreqRedu_Str, freq_TaskID[2], ENABLE);
    FreqRedu_en(&FreqRedu_Str, freq_TaskID[3], ENABLE);

    //設定 Pipeline
    PIPELINE_LAY(2, 4, 10);

    //單/雙通道資料後處理
    uint8_t pipeline_TaskID[2];
    pipeline_TaskID[0] = Pipeline_reg(&SysPipeline_str,
&ADC_single_PostPro_step, &ADC_single_PostPro_Str, NULL);
    pipeline_TaskID[1] = Pipeline_reg(&SysPipeline_str,
&ADC_diff_PostPro_step, &ADC_diff_PostPro_Str, NULL);

    uint8_t TaskID[4];

    //將降頻器登入進 Timer 中斷
    TaskID[0] = HWInt_reg(&TIMINT_Str, &FreqRedu_step, &FreqRedu_Str);
    HWInt_en(&TIMINT_Str, TaskID[0], ENABLE);

    //將 ADC_single  結果登入進 ADC interrupt
    TaskID[1] = HWInt_reg(&ADCHWINT_Str, &RealTimeRegGet_step,
&ADC_single_result_Str);
    HWInt_en(&ADCHWINT_Str, TaskID[1], ENABLE);

    //將 ADC_diff     結果登入進 ADC interrupt
    TaskID[2] = HWInt_reg(&ADCHWINT_Str, &RealTimeRegGet_step,
&ADC_diff_result_Str);
```

```
	HWInt_en(&ADCHWINT_Str, TaskID[2], ENABLE);

	//將 pipeline 登入進 ADC interrupt
	TaskID[3] = HWInt_reg(&ADCHWINT_Str, &Pipeline_step,
&SysPipeline_str);
	HWInt_en(&ADCHWINT_Str, TaskID[3], ENABLE);

	sei();
	TRIG_NEXT_TASK(0);

	while (1)
	{
		;
	}
	return 0;
}

void ADC_init()
{
	//設定內部 2.56V
	REGFPT(&ADMUX, 0xC0, 6, 3);

	//設定 10 位元轉換靠右
	REGFPT(&ADMUX, 0x20, 5, 0);

	//設定非連續或觸發轉換
	REGFPT(&ADCSRA, 0x20, 5, DISABLE);

	//設定 ADC1:2 F1:2c 皆輸入
	REGFPT(&DDRF, 0x06, 0, 0);

	//設定致能 ADC
	REGFPT(&ADCSRA, 0x80, 7, ENABLE);

	//設定致能 ADC Interrupt
	REGFPT(&ADCSRA, 0x08, 3, ENABLE);

	//設定 ADC 時脈  clk/128
```

```
    REGFPT(&ADCSRA, 0x07, 0, 7);
}

void timer3_init()
{
    //normal mode
    REGFPT(&TCCR3A, 0x03, 0, 0);
    //normal mode
    REGFPT(&TCCR3B, 0x18, 3, 1);
    //設定 timer 時脈 clk/1024
    REGFPT(&TCCR3B, 0x07, 0, 5);

    OCR3A = 269;
    //設定 timer3A 致能
    REGFPT(&ETIMSK, 0x10, 4, 1);

    // freq = 11059200 / (2 * 1024 * (1 + 100)) = 53.46
}

void ADC_single_PostPro_step(void *VoidStr_p)
{
    volatile ADCPostProStr_t *Str_p = (ADCPostProStr_t *)VoidStr_p;

    Str_p->DataList_p[Str_p->DataCount] = *(Str_p->InData_p);

    if ((Str_p->DataCount + 1) == (Str_p->DataLength))
    {
        cli();
        HMI_snput_matrix(HMI_TYPE_UI16, 1, Str_p->DataLength, Str_p-
>DataList_p);
        Str_p->DataCount = 0;
        sei();
    }
    else
        Str_p->DataCount++;

    TRIG_NEXT_TASK(1);
}
```

```
void ADC_diff_PostPro_step(void *VoidStr_p)
{
	volatile ADCPostProStr_t *Str_p = (ADCPostProStr_t *)VoidStr_p;

	Str_p->DataList_p[Str_p->DataCount] = *(Str_p->InData_p);

	if ((Str_p->DataCount + 1) == (Str_p->DataLength))
	{
		cli();
		HMI_snput_matrix(HMI_TYPE_UI16, 1, Str_p->DataLength, Str_p-
>DataList_p);
		Str_p->DataCount = 0;
		sei();
	}
	else
		Str_p->DataCount++;

	TRIG_NEXT_TASK(0);
}
```

Language : Matlab

實驗二

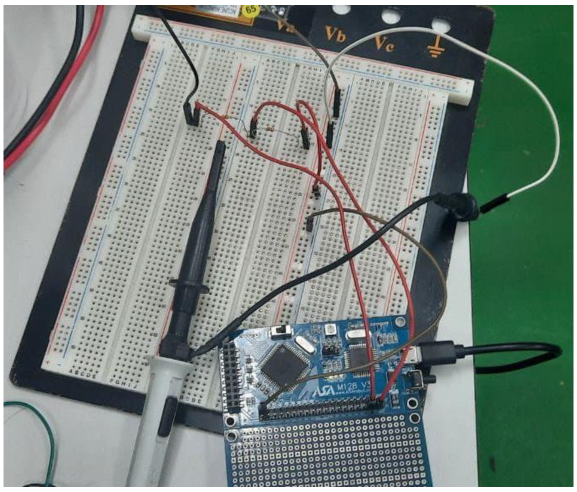
```
clear;clc;close;

[port] = remo_open(8);

[msg] = remo_get_msg(port);
disp(msg);%start while-loop
ans = 1;

while ans
    [msg] = remo_get_msg(port);
    disp(msg);%Set input voltage 0V

    temp = input("pressed any key\n");
    remo_put_msg(port,temp);

    [msg] = remo_get_msg(port);
    disp(msg);%temp=

    [msg] = remo_get_msg(port);
    Code_0V = str2num(msg);
    disp(msg);%Data=

    [msg] = remo_get_msg(port);
    disp(msg);%Set input voltage 1.23V

    temp = input("Press any key\n");
    remo_put_msg(port,temp);

    [msg] = remo_get_msg(port);
    disp(msg);%temp=

    [msg] = remo_get_msg(port);
    Code_1_23V = str2num(msg);
    disp(msg);%Data=

    disp("Offset= "+num2str(Code_0V));
    disp("Gain= "+num2str(1.23/(Code_1_23V-Code_0V)));


    [msg] = remo_get_msg(port);
    disp(msg);%Is continuing detect ADC? (1/0)

    ans = input("Enter:");
    remo_put_msg(port,ans);
    ans = str2num(ans);
end

remo_close(port);
```

實驗三

```
clear;clc;close;

[port] = remo_open(8);
title('freq. 10Hz | V_{pp} 1V | offset 0.5V | sine wave ADC convert');
xlabel('set');
ylabel('Volt[V]');
ylim([0 ,0.6]);

hold on;
[data_single] = remo_snget_matrix(port);
[data_diff] = remo_snget_matrix(port);
data_single_voltage = single(data_single) ./ 1024 .*2.56;
data_diff_voltage = single(data_diff) ./ 512 .*2.56;

plot(data_single_voltage, 'b');
plot(data_diff_voltage, 'g');
legend('single voltage','diff voltage');
hold off;
```

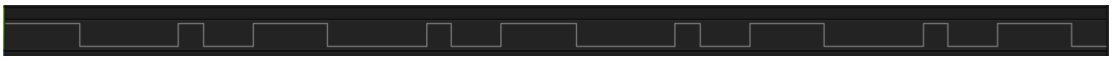
三、實驗數據

1.電路圖

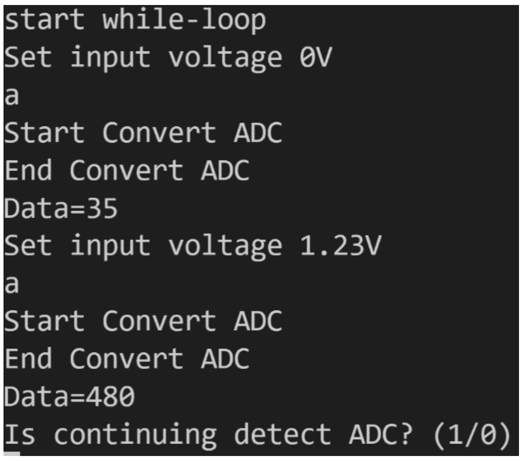


2.實驗結果

實驗一

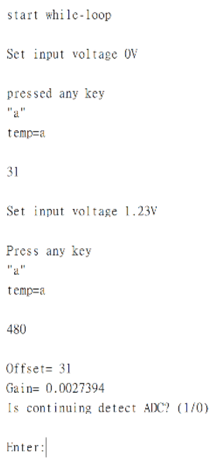


實驗二



```
start while-loop
Set input voltage 0V
a
Start Convert ADC
End Convert ADC
Data=35
Set input voltage 1.23V
a
Start Convert ADC
End Convert ADC
Data=480
Is continuing detect ADC? (1/0)
```

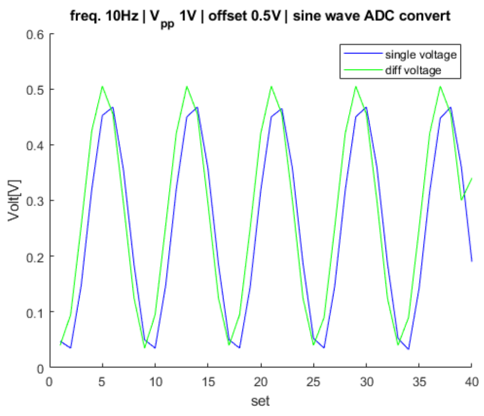
二之一

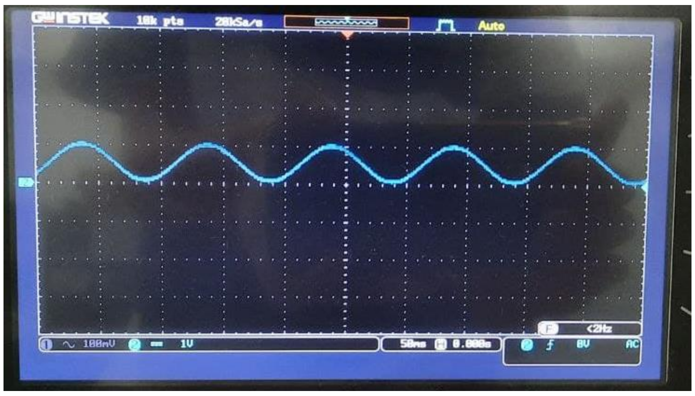


```
start while-loop

Set input voltage 0V

pressed any key
"a"
temp=a

31

Set input voltage 1.23V

Press any key
"a"
temp=a

480

Offset= 31
Gain= 0.0027394
Is continuing detect ADC? (1/0)

Enter:
```

二之二

實驗三





四、實驗問題

1. 本實驗要求，以於計時中斷中來執行ADC開始轉換，再於ADC轉換完成中斷讀回ADC轉換結果，請問若不使用ADC轉換完成中斷。可以如何撰寫相同效果的程式，並比較兩者的優劣。

A.透過讀取ADCSRA,Mask=0x10,Shift=4的暫存器ADIF來判斷轉換是否完成。

優:撰寫上比較輕鬆。

劣:程式碼一旦執行時間過長就有可能會造成讀取到的值不是及時轉換的結果。