

# Micro-Controller Experiment

Week3

Teacher: 廖裕評 Yu-Ping Liao

TA: 陳大荃 Da-chuan Chen, 陳恩妮 En-ni Chen

# Class Rules

1. No drink besides water.
2. Bring a laptop and breadboard if needed.
3. Ask us TAs to sign and borrow development boards. Do not sign or ask others to sign for you without TAs' permission.
4. Arriving 10 minutes after the bell rings will be regarded as absent.
5. If you damage any borrowed equipment, you have to pay for it.

# Homework Rules

1. Includes: A. Class content, B. Class exercise, C. Homework (screenshot or video)
2. Editing software: MS PowerPoint
3. File format: PDF
4. Filename: "date\_group\_studentID\_name.pdf", like "0916\_第1組\_11028XXX\_陳OO.pdf"
5. The homework deadline is 23:59 of the day before the next class. If you are late, then your grade will be deducted.

# Contact

If you encounter any problems with this class, please get in touch with us with the following E-mails:

1. Teacher, Prof. Yu-Ping Liao 廖裕評 : [lyp@cycu.org.tw](mailto:lyp@cycu.org.tw)
2. TA, Da-chuan Chen 陳大荃 : [dachuan516@gmail.com](mailto:dachuan516@gmail.com)
3. TA, En-ni Chen 陳恩妮 : [anna7125867@gmail.com](mailto:anna7125867@gmail.com)

Or visit 篤信 Lab353 for further questions.

# Outline of the Week

1. Last Week Homework W2-3.
2. ADC Introduction.
3. ADC Project.
4. Homework 3-1.
5. Homework 3-2 Bonus Question.

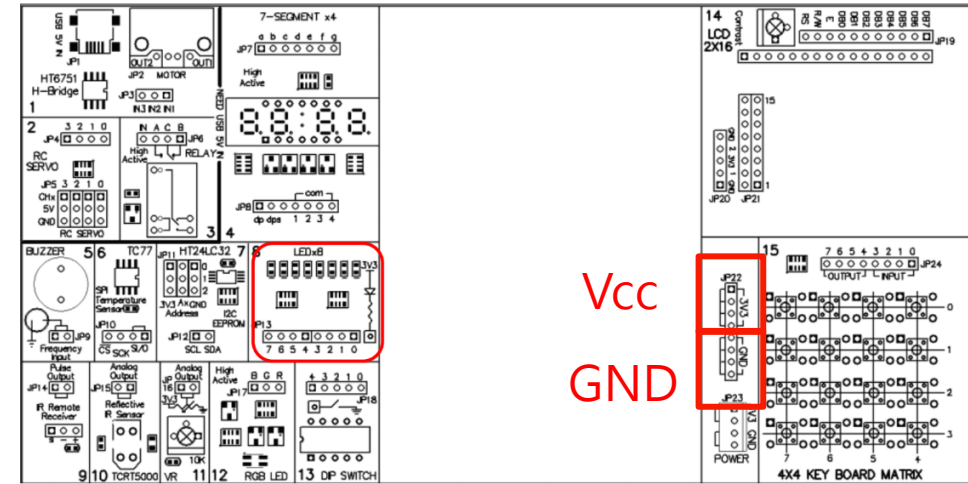
# Last Week Homework W2-3.

[https://github.com/CYCU-AIoT-System-  
Lab/Microcontroller-  
Experiment/blob/main/w2/GPIO-InputOutput-  
Experiment\\_Steps.md](https://github.com/CYCU-AIoT-System-Lab/Microcontroller-Experiment/blob/main/w2/GPIO-InputOutput-Experiment_Steps.md)

# Control LED

- Objective: Make LED lights flash using the delay function.
- Hint:

1. Use PC14、PC15、PB0、PB1、PB2、PB3、PB4、PB5 attach to JP13[0:7].
2. Add Delay function, modify the main function.
3. Draw the wiring diagram and explain how to turn the LEDs on.



# Open Clock

```
101 void CKCU_Configuration(void)
102 {
103     CKCU_PeripClockConfig_TypeDef CKCUClock = {{0}};
104     CKCUClock.Bit.PB = 1;
105     CKCUClock.Bit.PC = 1;
106     CKCUClock.Bit.AFIO = 1;
107     CKCU_PeripClockConfig(CKCUClock, ENABLE);
108 }
```

Clock of Port B and Port C is enable



# Edit GPIO\_OUT\_Configuration

```
140 void GPIO_OUT_Configuration(void)
141 {
142     /* Configure LED1, LED2 pins as output function
143     /* Configure AFIO mode of output pins
144     AFIO_GPxConfig(HTCFG_OUTPUT_LED1_ID, HTCFG_OUTPUT_LED1_AFIO_PIN, AFIO_FUN_GPIO);
145     AFIO_GPxConfig(HTCFG_OUTPUT_LED2_ID, HTCFG_OUTPUT_LED2_AFIO_PIN, AFIO_FUN_GPIO);
146
147     /* Configure GPIO direction of output pins
148     GPIO_DirectionConfig(HTCFG_LED1, HTCFG_OUTPUT_LED1_GPIO_PIN, GPIO_DIR_OUT);
149     GPIO_DirectionConfig(HTCFG_LED2, HTCFG_OUTPUT_LED2_GPIO_PIN, GPIO_DIR_OUT);
150 }
151
```

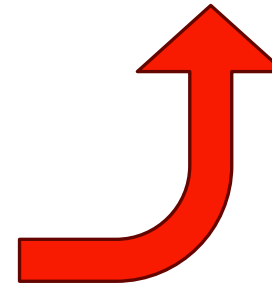
F12



```
172 #if defined(USE_HT32F52352_SK)
173 #define HTCFG_OUTPUT_LED1_ID
174 #define HTCFG_OUTPUT_LED2_ID
175 #define HTCFG_INPUT_WAKE_ID
176 #define HTCFG_INPUT_KEY1_ID
177
178 #define HTCFG_OUTPUT_LED1_CLK(CK) (CK.Bit.PC)
179 #define HTCFG_OUTPUT_LED2_CLK(CK) (CK.Bit.PC)
180 #define HTCFG_INPUT_WAKE_CLK(CK) (CK.Bit.PB)
181 #define HTCFG_INPUT_KEY1_CLK(CK) (CK.Bit.PD)
182
183 #define HTCFG_LED1 (HT_GPIOC)
184 #define HTCFG_LED2 (HT_GPIOC)
185 #define HTCFG_WAKE (HT_GPIOB)
186 #define HTCFG_KEY1 (HT_GPIOD)
187
188 #define HTCFG_OUTPUT_LED1_AFIO_PIN (AFIO_PIN_14)
189 #define HTCFG_OUTPUT_LED2_AFIO_PIN (AFIO_PIN_15)
190 #define HTCFG_INPUT_WAKE_AFIO_PIN (AFIO_PIN_12)
```

```
137 void GPIO_OUT_Configuration(void)
138 {
139     /* Configure LED1, LED2 pins as output function
140     /* Configure AFIO mode of output pins
141     AFIO_GPxConfig(GPIO_PC, AFIO_PIN_14, AFIO_FUN_GPIO);
142     AFIO_GPxConfig(HTCFG_OUTPUT_LED2_ID, HTCFG_OUTPUT_LED2_AFIO_PIN, AFIO_FUN_GPIO);
143
144     /* Configure GPIO direction of output pins
145     GPIO_DirectionConfig(HTCFG_LED1, HTCFG_OUTPUT_LED1_GPIO_PIN, GPIO_DIR_OUT);
146     GPIO_DirectionConfig(HTCFG_LED2, HTCFG_OUTPUT_LED2_GPIO_PIN, GPIO_DIR_OUT);
147 }
148
```

Copy/Past



# Edit GPIO\_MainRoutine

```
153 void GPIO_MainRoutine(void)
154 {
155     /* Read WAKEUP and then output to LED1
156     GPIO_WriteOutBits(HTCFG_LED1, HTCFG_OUTPUT_LED1_GPIO_PIN, SET);
157     _Delay(1000);
158     /* Read KEY1 and then output to LED2
159     GPIO_WriteOutBits(HTCFG_LED2, HTCFG_OUTPUT_LED2_GPIO_PIN, RESET);
160     _Delay(1000);
161 }
```

F12

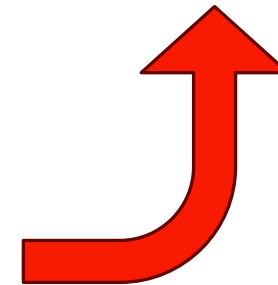


```
178 #define HTCFG_OUTPUT_LED1_CLK(CK) (CK.Bit.PC)
179 #define HTCFG_OUTPUT_LED2_CLK(CK) (CK.Bit.PC)
180 #define HTCFG_INPUT_WAKE_CLK(CK) (CK.Bit.PB)
181 #define HTCFG_INPUT_KEY1_CLK(CK) (CK.Bit.PD)
182
183 #define HTCFG_LED1 (HT_GPIOC)
184 #define HTCFG_LED2 (HT_GPIOC)
185 #define HTCFG_WAKE (HT_GPIOB)
186 #define HTCFG_KEY1 (HT_GPIOD)
187
188 #define HTCFG_OUTPUT_LED1_AFIO_PIN (AFIO_PIN_14)
189 #define HTCFG_OUTPUT_LED2_AFIO_PIN (AFIO_PIN_15)
190 #define HTCFG_INPUT_WAKE_AFIO_PIN (AFIO_PIN_12)
191 #define HTCFG_INPUT_KEY1_AFIO_PIN (AFIO_PIN_1)
192
193 #define HTCFG_OUTPUT_LED1_GPIO_PIN (GPIO_PIN_14)
194 #define HTCFG_OUTPUT_LED2_GPIO_PIN (GPIO_PIN_15)
195 #define HTCFG_INPUT_WAKE_GPIO_PIN (GPIO_PIN_12)
```

```
153 void GPIO_MainRoutine(void)
154 {
155     /* Read WAKEUP and then output to LED1
156     GPIO_WriteOutBits(HT_GPIOC, GPIO_PIN_14, 0);
157     _Delay(1000);
158     /* Read KEY1 and then output to LED2
159     GPIO_WriteOutBits(HTCFG_LED2, HTCFG_OUTPUT_LED2_GPIO_PIN, RESET);
160     _Delay(1000);
161 }
```

output is  
high(1)/low(0)

Copy/Past



# GPIO\_MainRoutine

- GPIO\_ReadInBit ( Port PA ~ PD 、 PIN number )

Read in the input data from a specified pin.

- GPIO\_WriteOutBits ( Port PA ~ PD 、 PIN number , 0 or 1 )

Write out the output data with a specified pin.

# Delay function

Add Delay function

```
165 static void _Delay(vu32 count)
166 {
167     vu32 i;
168     for(i=0; i<5000*count; i++)
169     {
170     }
171 }
```

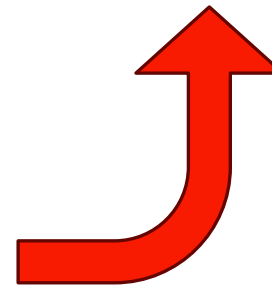


Function Declaration at the Beginning

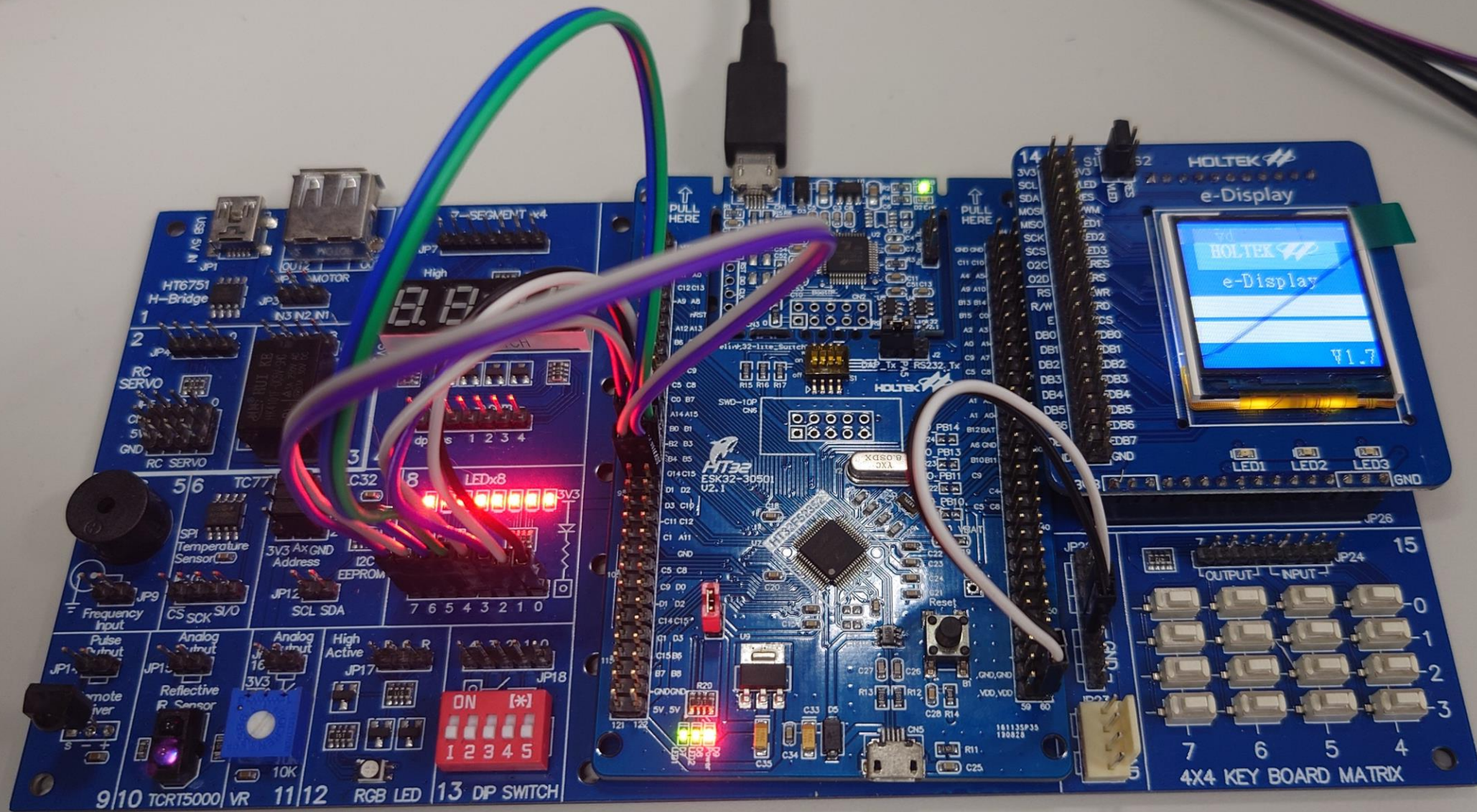
```
45 /* Private function prototypes ---
46 void CKCU_Configuration(void);
47 void GPIO_IN_Configuration(void);
48 void GPIO_OUT_Configuration(void);
49 void GPIO_MainRoutine(void);
50 static void _Delay(vu32 count);
```

Modify GPIO\_MainRoutine

```
156 void GPIO_MainRoutine(void)
157 {
158     /* Read WAKEUP and then output to LED1
159     GPIO_WriteOutBits(HTCFG_LED1, HTCFG_OUTPUT_LED1_GPIO_PIN, SET);
160     _Delay(1000);
161     /* Read KEY1 and then output to LED2
162     GPIO_WriteOutBits(HTCFG_LED2, HTCFG_OUTPUT_LED2_GPIO_PIN, RESET);
163     _Delay(1000);
164 }
```







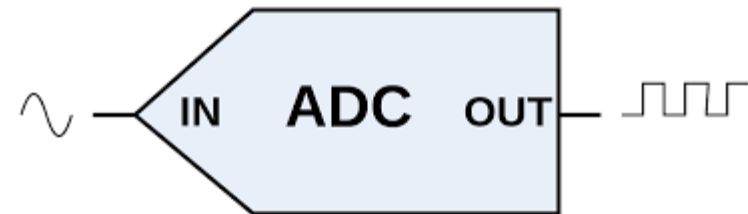
The background features a series of concentric circles in light gray, some solid and some dashed, creating a ripple effect. A large red speech bubble is centered on the page, containing the text "ADC Introduction".

# ADC Introduction

# What is ADC?

- 類比數位轉換器 ( 英語：Analog-to-digital converter, ADC, A/D 或 A to D ) 是用於將連續的類比訊號轉換為離散的數位訊號。
- Analog-to-digital converter plays a crucial role in enabling digital systems to interface with the analog world by converting continuous analog signals into discrete digital values.

□ The example of continuous signal conversion



# ADC specifications

- Sampling rate: How often should we do the sampling?
- Resolution: How many storage bits can we use?
- ADC Conversion Equation:

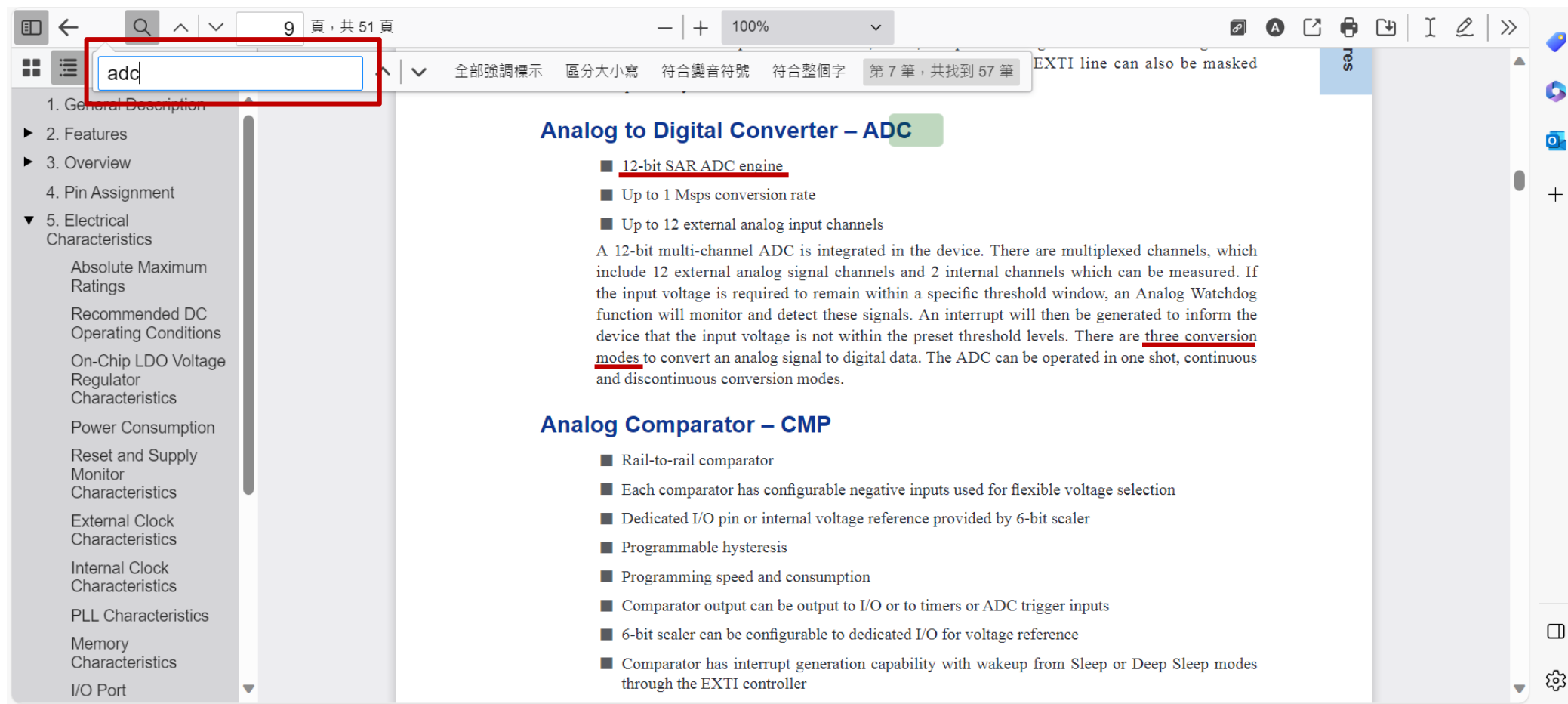
$$\text{ConvertedVoltage} = \text{ConvertedValue} \times \frac{V_{ref}}{2^n - 1}$$

- Vref: reference voltage, usually between  $V_{DD}$  and  $V_{SS}$
- n: number of bits



# Datasheet

Ctrl+f: search for "ADC"



9 頁, 共 51 頁

add

全部強調標示 區分大小寫 符合變音符號 符合整個字 第 7 筆, 共找到 57 筆

EXTI line can also be masked

## Analog to Digital Converter – ADC

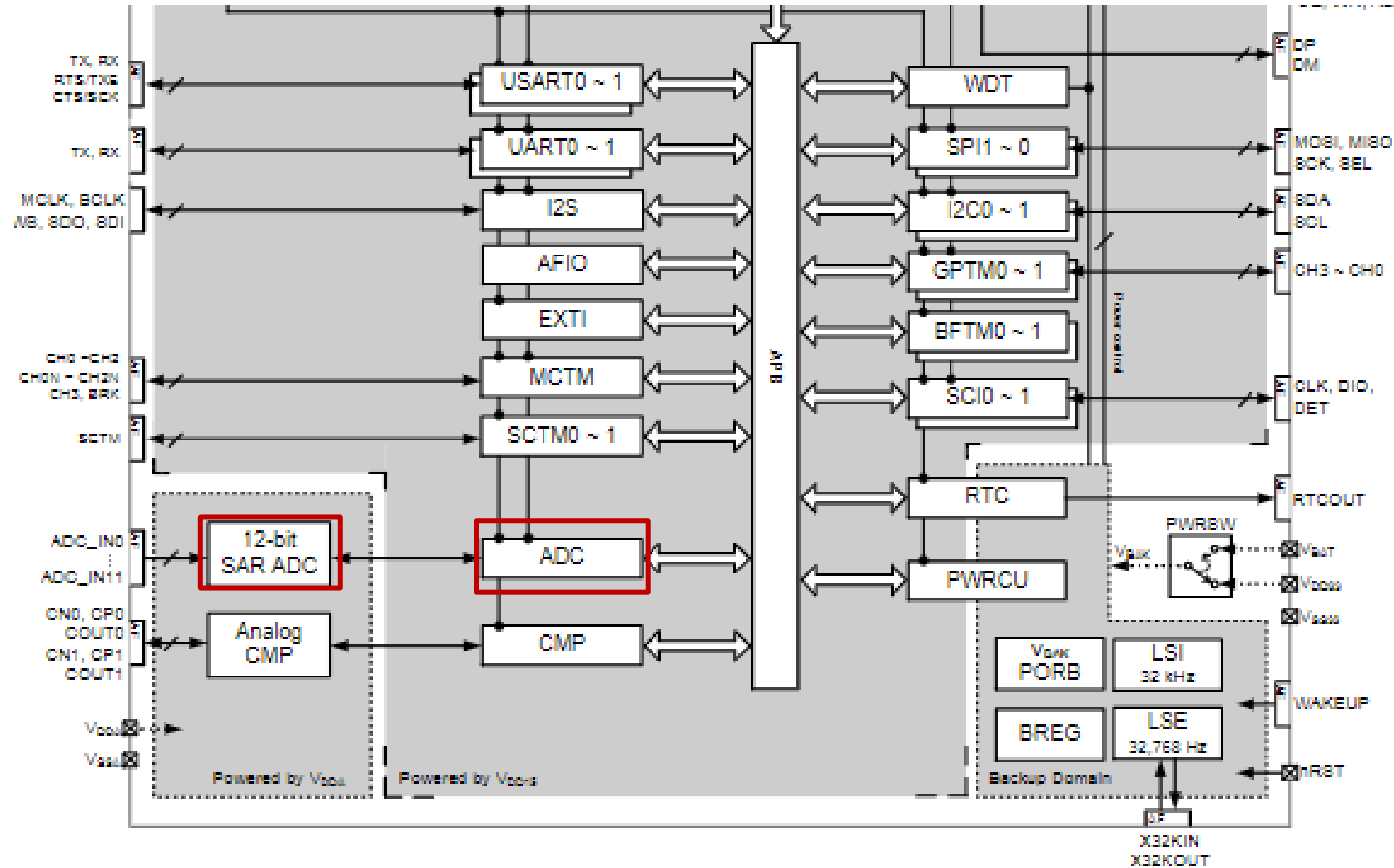
- 12-bit SAR ADC engine
- Up to 1 Msps conversion rate
- Up to 12 external analog input channels

A 12-bit multi-channel ADC is integrated in the device. There are multiplexed channels, which include 12 external analog signal channels and 2 internal channels which can be measured. If the input voltage is required to remain within a specific threshold window, an Analog Watchdog function will monitor and detect these signals. An interrupt will then be generated to inform the device that the input voltage is not within the preset threshold levels. There are three conversion modes to convert an analog signal to digital data. The ADC can be operated in one shot, continuous and discontinuous conversion modes.

## Analog Comparator – CMP

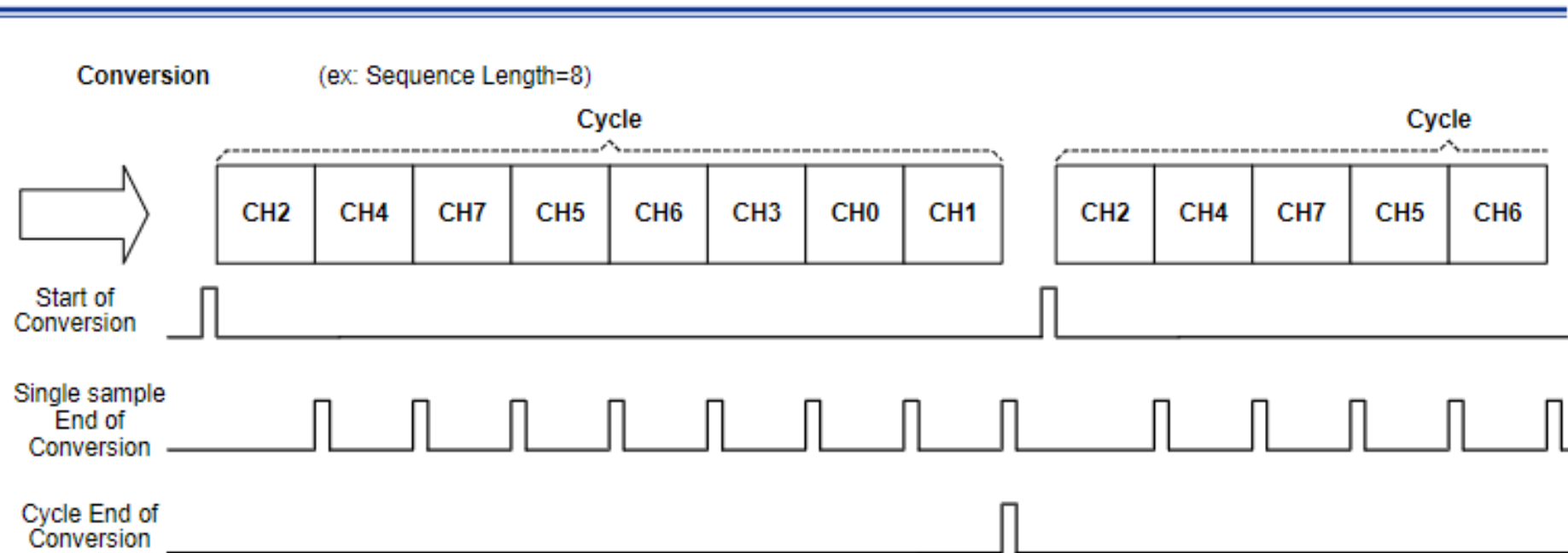
- Rail-to-rail comparator
- Each comparator has configurable negative inputs used for flexible voltage selection
- Dedicated I/O pin or internal voltage reference provided by 6-bit scaler
- Programmable hysteresis
- Programming speed and consumption
- Comparator output can be output to I/O or to timers or ADC trigger inputs
- 6-bit scaler can be configurable to dedicated I/O for voltage reference
- Comparator has interrupt generation capability with wakeup from Sleep or Deep Sleep modes through the EXTI controller

# Block Diagram



# Three conversion modes

- One shot conversion mode



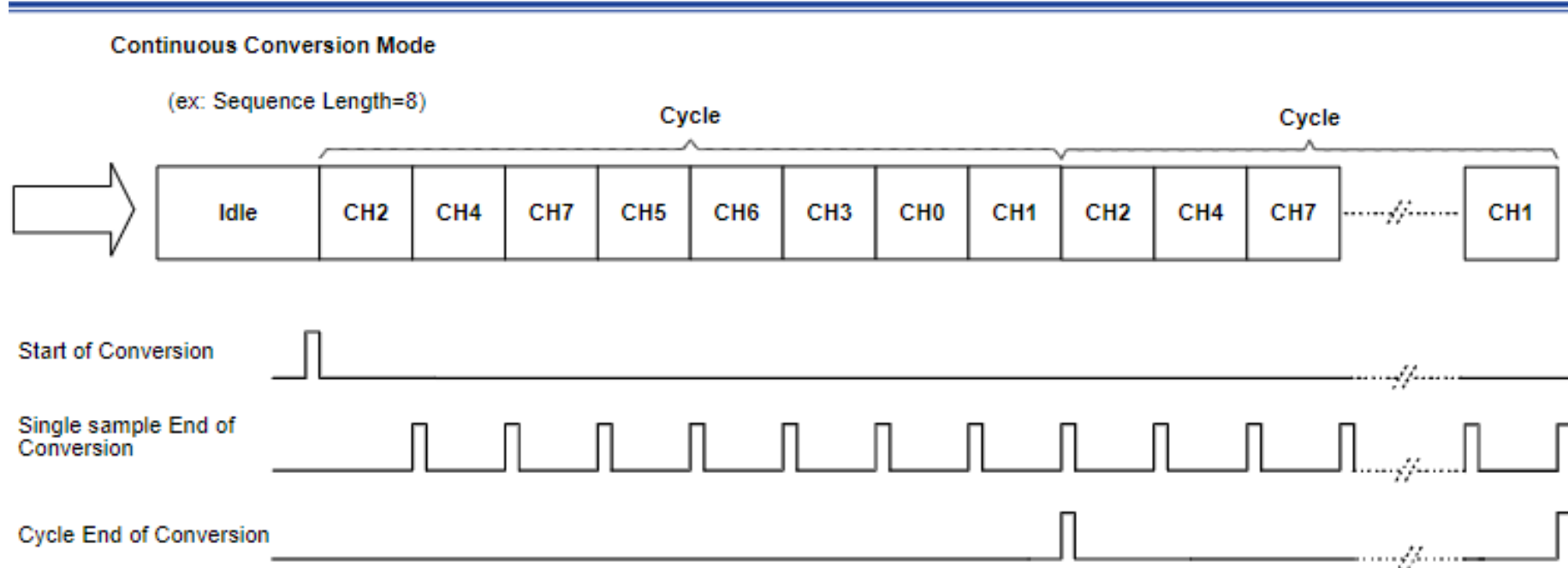
**Figure 28. One Shot Conversion Mode**

---

---

# Three conversion modes

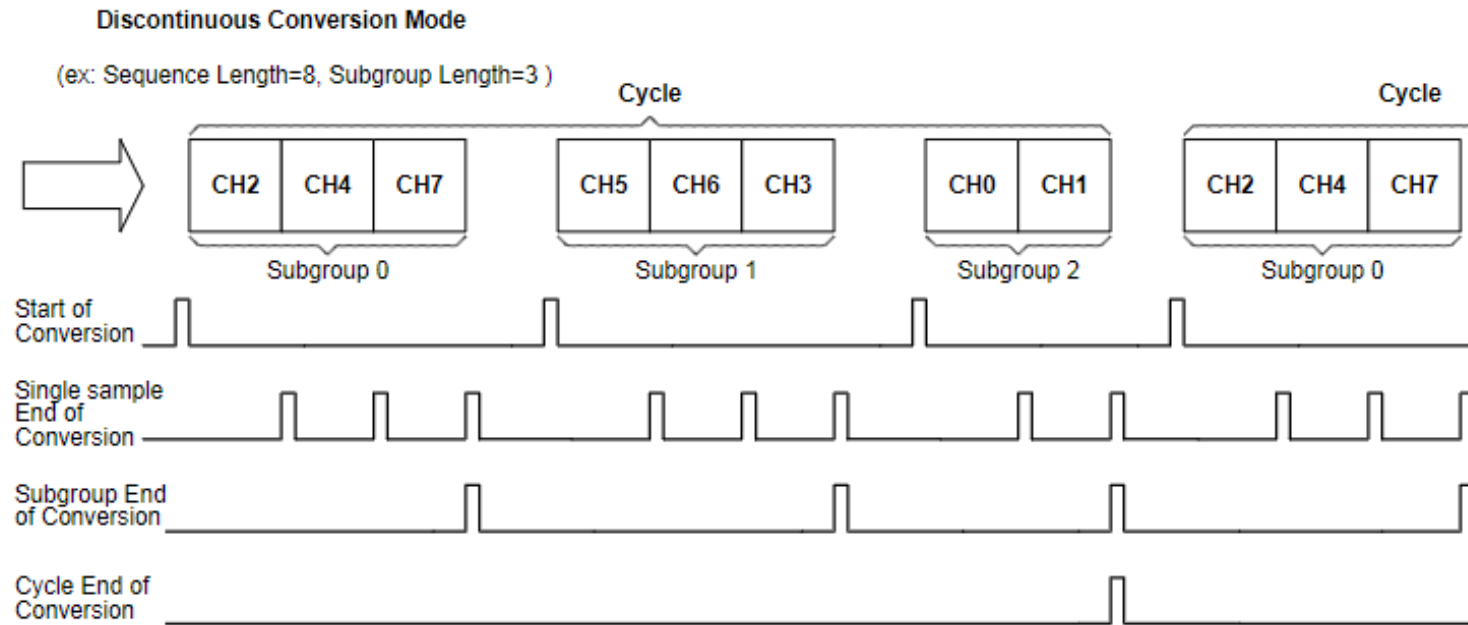
- Continuous conversion mode



**Figure 29. Continuous Conversion Mode**

# Three conversion modes

- Discontinuous conversion mode



**Figure 30. Discontinuous Conversion Mode**

A red speech bubble with a tail pointing downwards, containing the text "ADC Project". The background features a light gray pattern of concentric circles and curved lines.

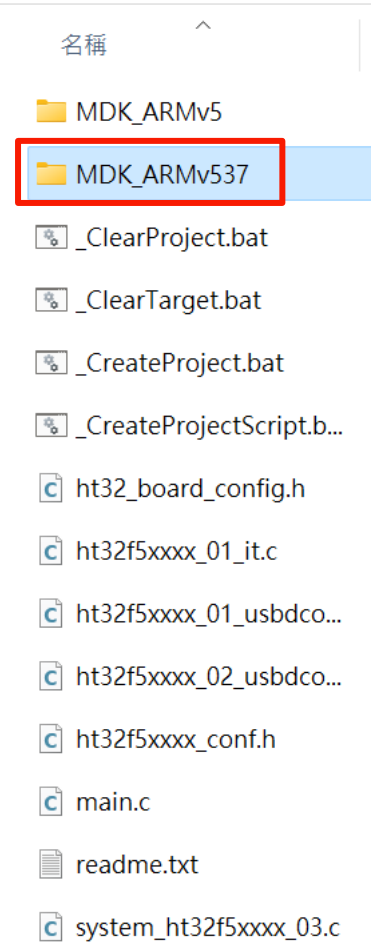
ADC Project

# 1. Execute "\_CreatProject"

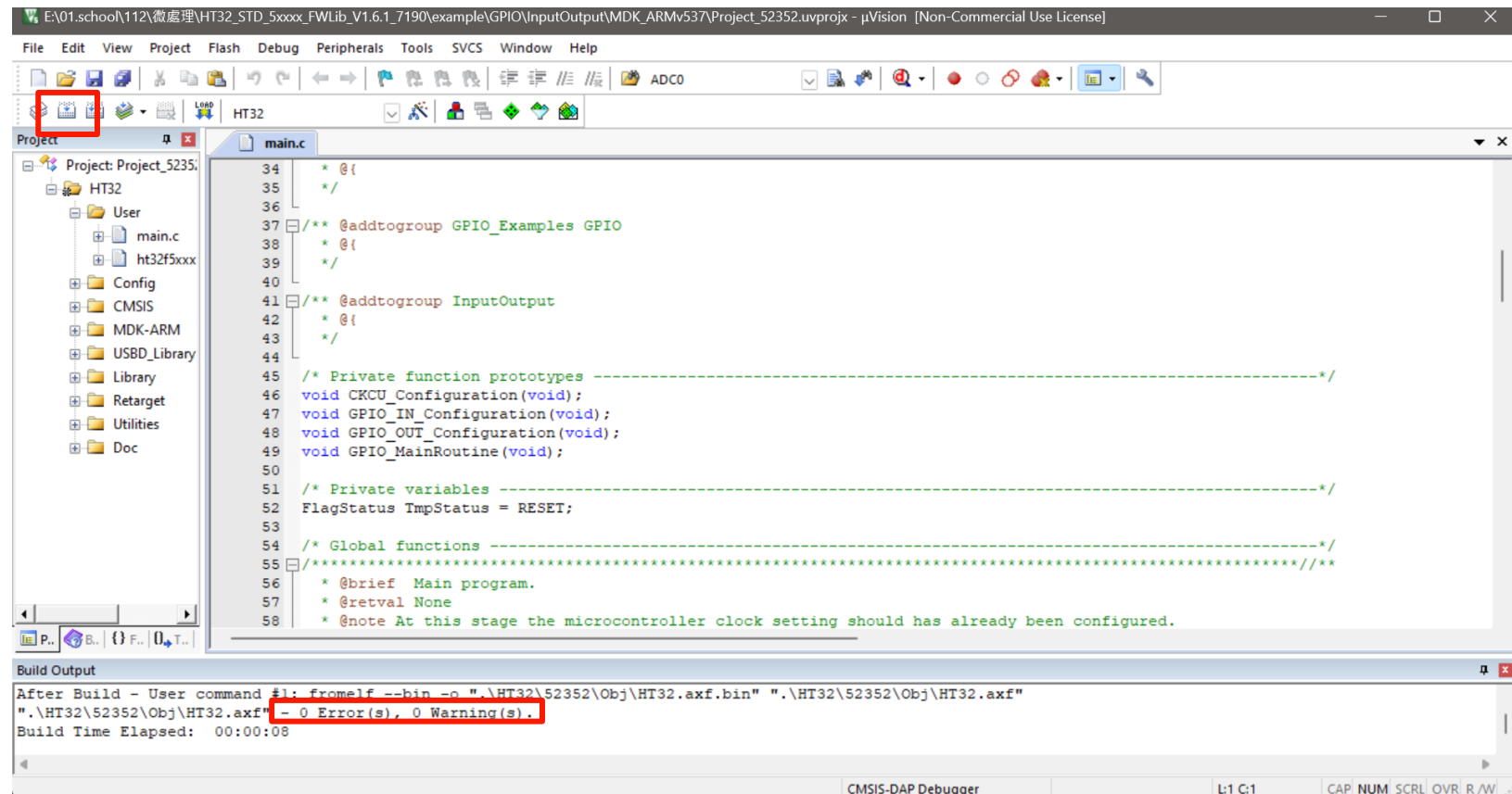
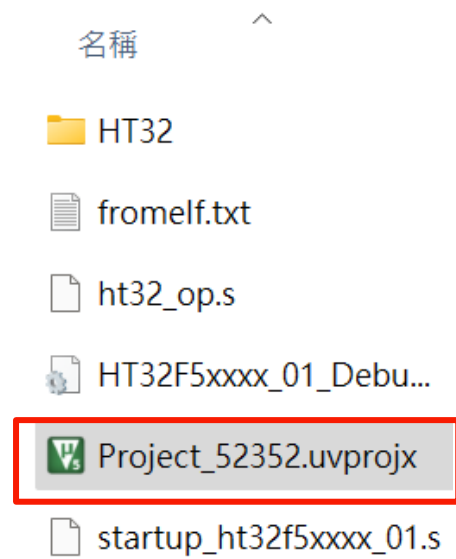
1. Go to "~ /HT32\_STD\_5xxxx\_FWLib\_V1.5.1\_7084/example/ADC/Continuous\_Pontentiometer".
2. Double click "\_CreateProject.bat".

## 2. Launch project

1 > HT32\_M0p\_V20230621 > Fir



! > 微處理 > HT32\_STD\_5xxx





# main

```
58 int main(void)
59 {
60     RETARGET_Configuration();
61
62     ADC_Configuration();
63
64     /* Enable ADC
65     ADC_Cmd(HTCFG_ADC_PORT, ENABLE);
66
67     /* Software trigger to start ADC conversion
68     ADC_SoftwareStartConvCmd(HTCFG_ADC_PORT, ENABLE);
69
70     while (1)
71     {
72         if (gADC_SingleEndOfConversion)
73         {
74             printf("\rPotentiometer level is %04d", (int)gPotentiometerLevel);
75         }
76     }
77 }
```

# ADC\_Configuration

```
83 void ADC_Configuration(void)
84 {
85     /* Enable peripheral clock
86     CKCU_PeripClockConfig_TypeDef CKCUClock = {{ 0 }};
87     CKCUClock.Bit.AFIO = 1;
88     CKCUClock.Bit.HTCFG_ADC_IPN = 1;
89     CKCU_PeripClockConfig(CKCUClock, ENABLE);
90 }
```

Open the clock that will be used

# Before edit

```
92  /* Configure AFIO mode as ADC function */
93  AFIO_GPxConfig(HTCFG_VR_GPIO_ID, HTCFG_VR_AFIO_PIN, HTCFG_ADC_AFIO_MODE);
94
95  { /* ADC related settings */
96      /* CK_ADC frequency is set to (CK_AHB / 64) */
97      CKCU_SetADCnPrescaler(HTCFG_ADC_CKCU_ADCPRE, CKCU_ADCPRE_DIV64);
98
99      /* Continuous mode, sequence length = 1 */
100     ADC-RegularGroupConfig(HTCFG_ADC_PORT, CONTINUOUS_MODE, 1, 0);
101
102     /* ADC conversion time = (Sampling time + Latency) / CK_ADC = (1.5 + ADST + 12.5) / CK_ADC */
103     /* Set ADST = 0, sampling time = 1.5 + ADST */
104     #if (LIBCFG_ADC_SAMPLE_TIME_BY_CH)
105         // The sampling time is set by the last parameter of the function "ADC-RegularChannelConfig()".
106     #else
107         ADC_SamplingTimeConfig(HTCFG_ADC_PORT, 0);
108     #endif
109
110     /* Set ADC conversion sequence as channel n */
111     ADC-RegularChannelConfig(HTCFG_ADC_PORT, HTCFG_VR_ADC_CH, 0, 0);
112
113     /* Set software trigger as ADC trigger source */
114     ADC-RegularTrigConfig(HTCFG_ADC_PORT, ADC_TRIG_SOFTWARE);
115 }
116
117 /* Enable ADC single end of conversion interrupt */
118 ADC_IntConfig(HTCFG_ADC_PORT, ADC_INT_SINGLE_EOC, ENABLE);
119
120 /* Enable the ADC interrupts */
121 NVIC_EnableIRQ(HTCFG_ADC_IRQn);
122 }
```

# After edit

```
93 AFIO_GPxConfig(GPIO_PA, AFIO_PIN_6, HTCFG_ADC_AFIO_MODE);
94
95 { /* ADC related settings
96  /* CK ADC frequency is set to (CK_AHB / 64)
97  CKCU_SetADCPrescaler(CKCU_ADCPRE_ADC0, CKCU_ADCPRE_DIV64);
98
99  /* Continuous mode, sequence length = 1
100  ADC_RegularGroupConfig(HT_ADC0, CONTINUOUS_MODE, 1, 0);
101
102  /* ADC conversion time = (Sampling time + Latency) / CK_ADC = (1.5 + ADST + 12.5) / CK_ADC
103  /* Set ADST = 0, sampling time = 1.5 + ADST
104  #if (LIBCFG_ADC_SAMPLE_TIME_BY_CH)
105  // The sampling time is set by the last parameter of the function "ADC_RegularChannelConfig()"
106  #else
107  ADC_SamplingTimeConfig(HT_ADC0, 0);
108  #endif
109
110  /* Set ADC conversion sequence as channel n
111  ADC_RegularChannelConfig(HT_ADC0, ADC_CH_6, 0, 0);
112
113  /* Set software trigger as ADC trigger source
114  ADC_RegularTrigConfig(HT_ADC0, ADC_TRIG_SOFTWARE);
115  }
116
117  /* Enable ADC single end of conversion interrupt
118  ADC_IntConfig(HTCFG_ADC_PORT, ADC_INT_SINGLE_EOC, ENABLE);
119
120  /* Enable the ADC interrupts
121  NVIC_EnableIRQ(HTCFG_ADC_IRQn);
122 }
```

Set AFIO mode for the pins

Select the ADC sampling frequency

Set ADC mode :  
1. Which ADC channel?  
2. Which mode?  
3. How many conversion times ?

Which register stores the read data?

Software triggering

Configure ADC interrupt mode

# main

```
58 int main(void)
59 {
60     RETARGET_Configuration();
61
62     ADC_Configuration();
63
64     /* Enable ADC
65     ADC_Cmd(HTCFG_ADC_PORT, ENABLE);
66
67     /* Software trigger to start ADC conversion
68     ADC_SoftwareStartConvCmd(HTCFG_ADC_PORT, ENABLE);
69
70     while (1)
71     {
72         if (gADC_SingleEndOfConversion)
73         {
74             printf("\rPotentiometer level is %04d", (int)gPotentiometerLevel);
75         }
76     }
77 }
```



# Retarget\_Configuration

```
118 void RETARGET_Configuration(void)
119 {
120 #ifdef RETARGET_IS_UART
121 /* !!! NOTICE !!!
122 Notice that the local variable (structure) did not have an initial value.
123 Please confirm that there are no missing members in the parameter settings below in this function.
124 */
125 USART_InitTypeDef USART_InitStructure;
126 #ifdef RETARGET_UxART_BAUDRATE
127 USART_InitStructure.USART_BaudRate = RETARGET_UxART_BAUDRATE;
128 #else
129 USART_InitStructure.USART_BaudRate = 115200;
130 #endif
131 USART_InitStructure.USART_WordLength = USART_WORDLENGTH_8B;
132 USART_InitStructure.USART_StopBits = USART_STOPBITS_1;
133 USART_InitStructure.USART_Parity = USART_PARITY_NO;
134 USART_InitStructure.USART_Mode = USART_MODE_NORMAL;
```

← BaudRate: 115200





# Download Tera Term


<https://github.com/TeraTermProject/osdn-download/releases>

## Tera Term 4.106 Latest

Source code is not available.

### ▼ Assets 4

 <a href="#">teraterm-4.106.exe</a>	12.2 MB	Jul 12
 <a href="#">teraterm-4.106.zip</a>	8.63 MB	Jul 12
 <a href="#">Source code (zip)</a>		Jul 12
 <a href="#">Source code (tar.gz)</a>		Jul 12

 8 8 people reacted

# Tera Term

<https://github.com/TeraTermProject/osdn-download/releases>





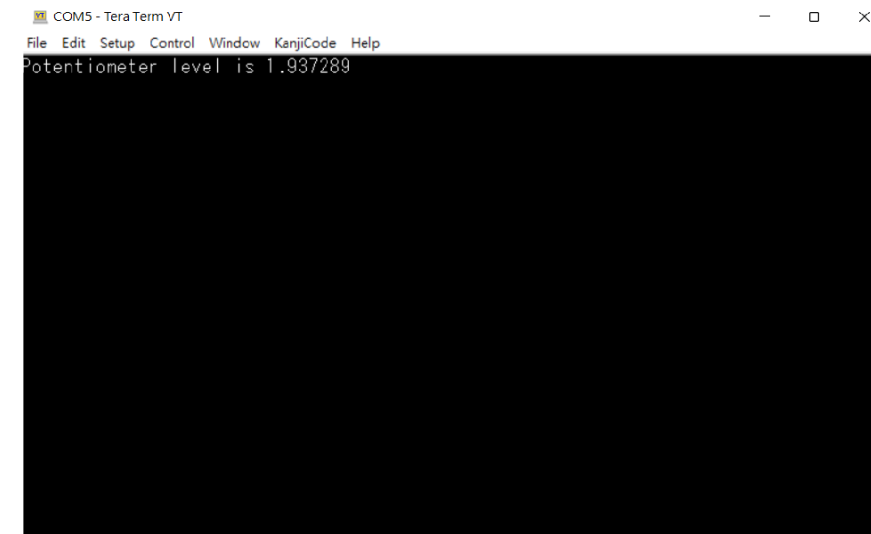
# Homework W3-1.

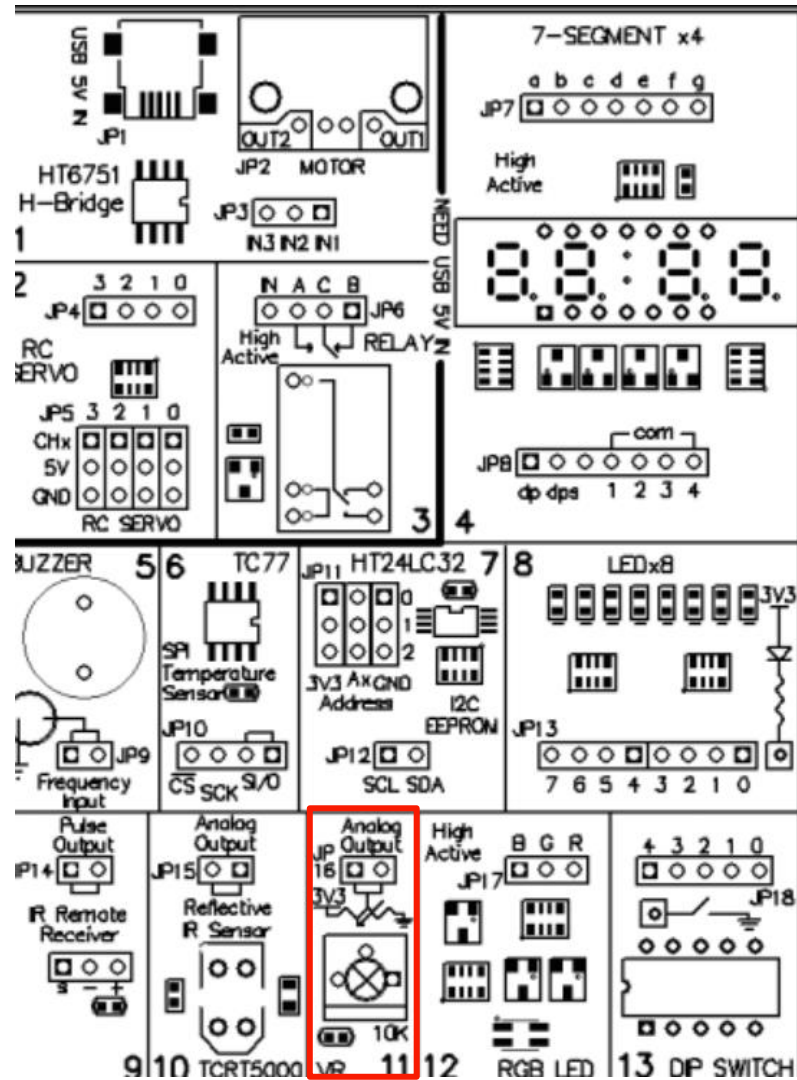
[https://github.com/CYCU-AIoT-System-  
Lab/Microcontroller-  
Experiment/blob/main/w3/ADC-  
Continuous\\_Potentiometer-Experiment\\_Steps.md](https://github.com/CYCU-AIoT-System-Lab/Microcontroller-Experiment/blob/main/w3/ADC-Continuous_Potentiometer-Experiment_Steps.md)

# Use potentiometer to display measured voltage.

- Objective: Connect pin **A6** to a variable resistor, rotate the variable resistor, and observe the change in voltage value.
- Hint: Look for the formula in page 10

```
58 int main(void)
59 {
60     RETARGET_Configuration();
61
62     ADC_Configuration();
63
64     /* Enable ADC
65     ADC_Cmd(HTCFG_ADC_PORT, ENABLE);
66
67     /* Software trigger to start ADC conversion
68     ADC_SoftwareStartConvCmd(HTCFG_ADC_PORT, ENABLE);
69
70     while (1)
71     {
72         if (gADC_SingleEndOfConversion)
73         {
74             printf("\rPotentiometer level is %04d", (int)gPotentiometerLevel);
75         }
76     }
77 }
78
```





Output

VDD

GND





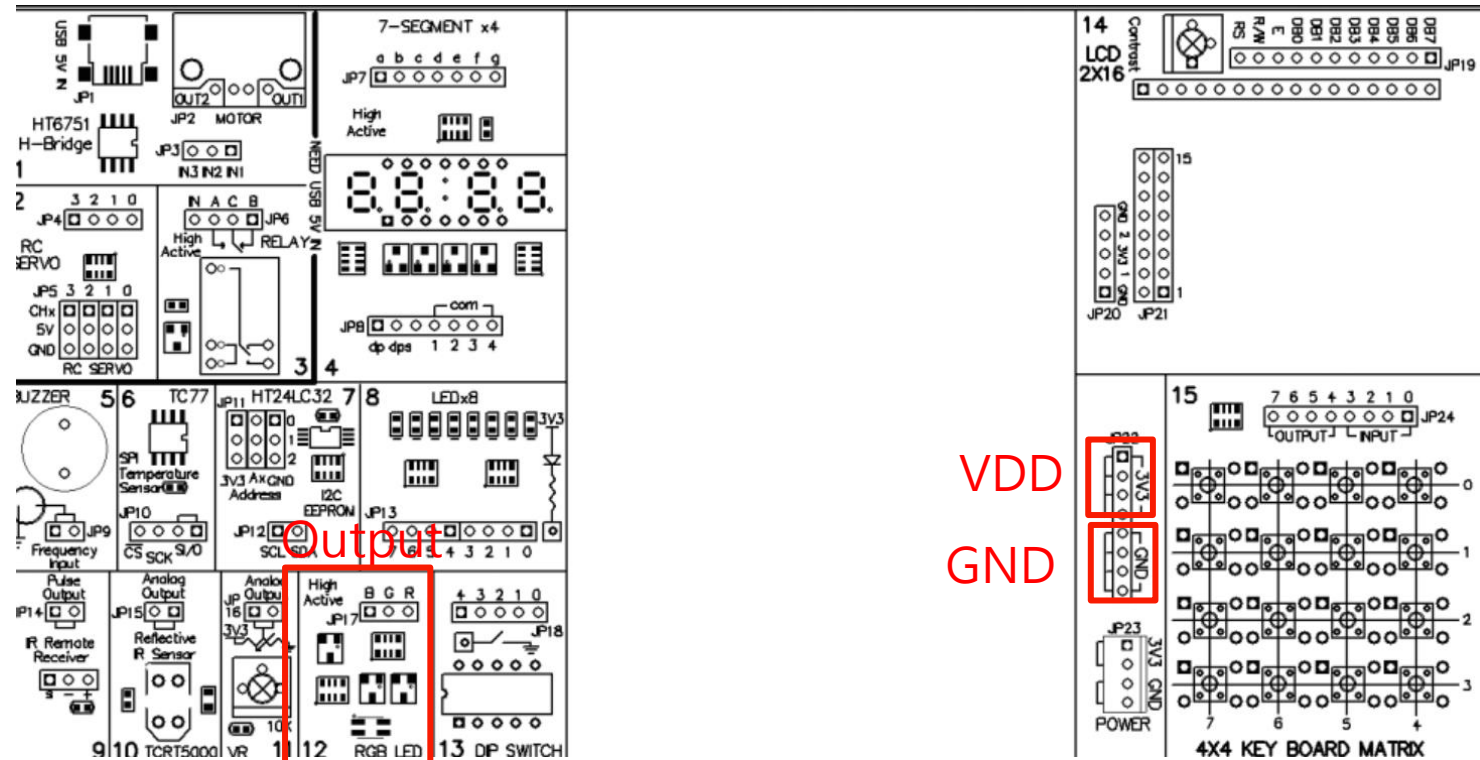
The background features a series of concentric circles in light gray, some solid and some dashed, creating a ripple effect. In the center, there is a red speech bubble with a white outline and a small tail pointing downwards.

# Homework W3-2 Bonus Question

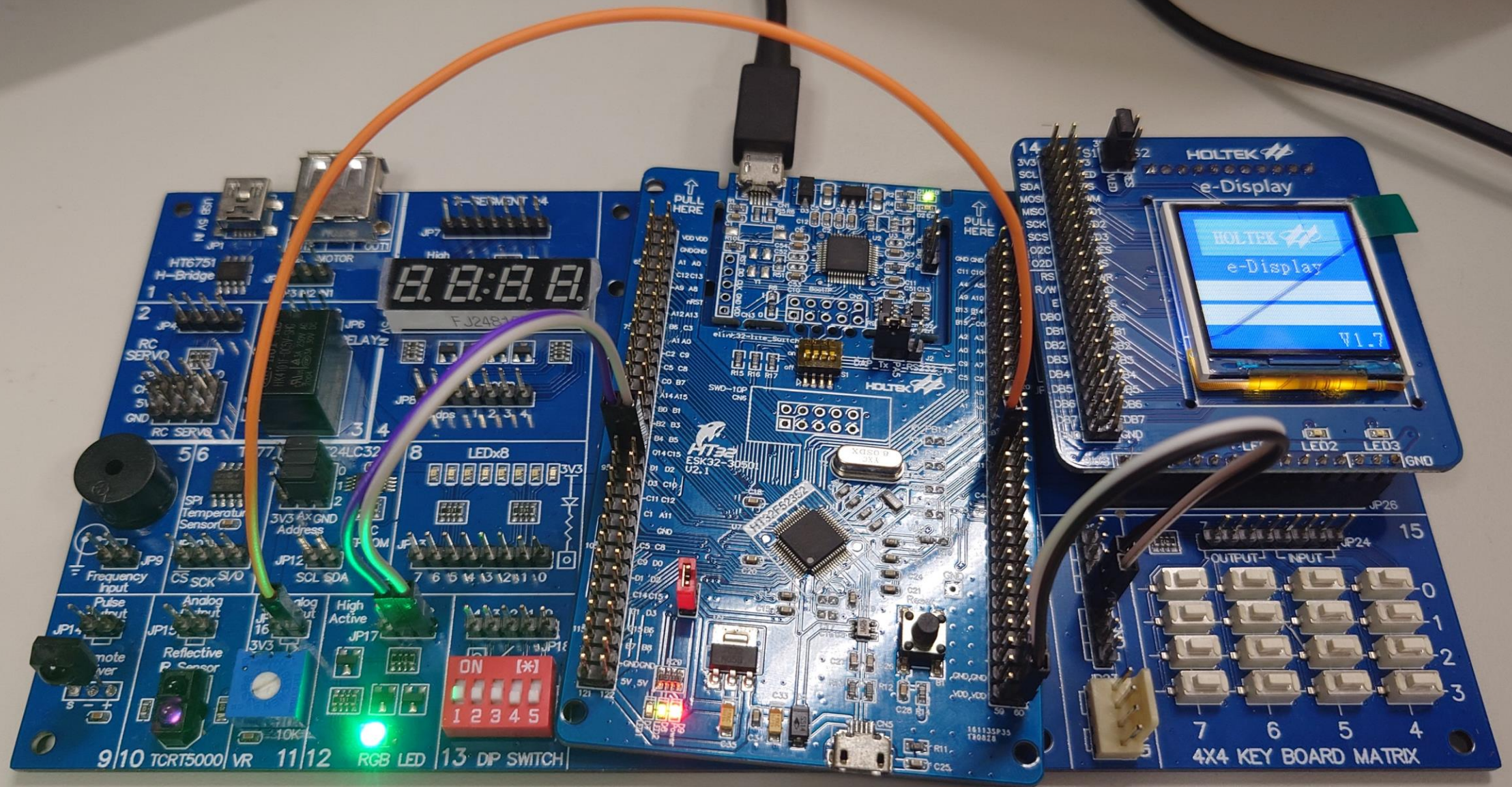


# Use potentiometer control RGB LED

- Objective: If measured value is larger than 2000, turn red light of RGB LED on; if it is smaller than 2000, turn green light on.
- Hint:
  - Output pins: PC14、PC15
  - Input pins: **PA6**









Class  
Dismissed