Lab8 UART and ADC

1.實驗目的

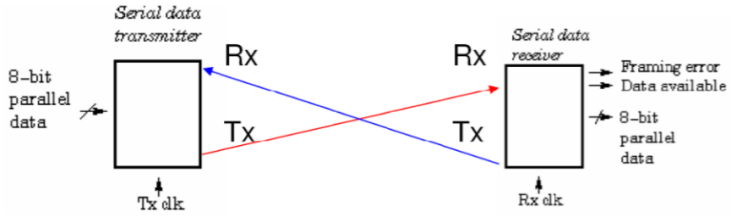
Understand the use of UART.
Understand the use of ADC.

2.實驗原理

- USART

線有沒有接對，Tx 是要接到 Rx 的 (你講話別人用耳朵聽)

- UART interface has two signals
  - Rx – Receive
  - Tx – Transmit



USART Baud Rate

- The baud rate for the receiver and transmitter (Rx and Tx) are both set to the same value as programmed in the USART_BRR register.
- In case of oversampling by 16, the equation is: Baud = $\frac{f_{CK}}{USARTDIV}$
- In case of oversampling by 8, the equation is: Baud = $\frac{2 \times f_{CK}}{USARTDIV}$
- When OVER8 = 0, BRR = USARTDIV
- When OVER8 = 1
  - BRR[2:0] = USARTDIV[3:0] shifted 1 bit to the right.
  - BRR[3] must be kept cleared.
  - BRR[15:4] = USARTDIV[15:4]

- BRR = f_{clock} / Baud rate of uart (USART1:115200Hz

http://en.radzio.dxp.pl/stm32vldiscovery/lesson8,communication,with,usart,basics.html

```
GPIO_Init();
USART???_Init();
UART_Transmit((uint8_t*)"???", 3);
```

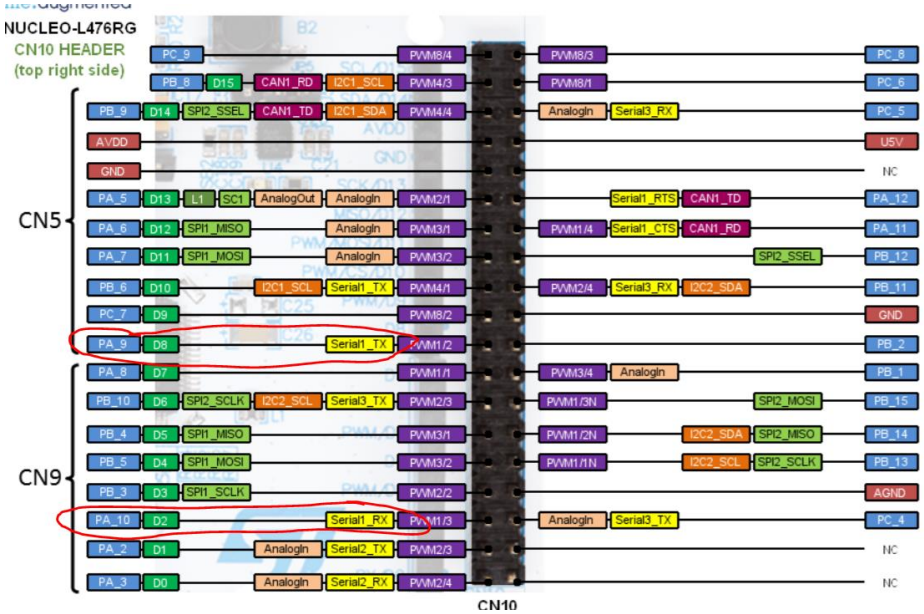
```
void GPIO_Init(void) {
    RCC->AHB2ENR |= (RCC_AHB2ENR_GPIOAEN | RCC_AHB2ENR_GPIOBEN | RCC_AHB2ENR_GPIOCEN);

    // UART
    TM_GPIO_Init(???, ?, TM_GPIO_Mode_AF, TM_GPIO_OType_PP, TM_GPIO_PuPd_NOPULL, TM_GPIO_Speed_Low);
    TM_GPIO_Init(???, ?, TM_GPIO_Mode_AF, TM_GPIO_OType_PP, TM_GPIO_PuPd_NOPULL, TM_GPIO_Speed_Low);
    GPIOB->AFR[0] = (GPIOB->AFR[0] & ~(0xFF000000)) | 0x77000000; // AF7 for pin

    // BUTTON
    TM_GPIO_Init(GPIOC, 13, TM_GPIO_Mode_IN, TM_GPIO_OType_PP, TM_GPIO_PuPd_NOPULL, TM_GPIO_Speed_Medium);
}
```

TM_GPIO_Mode_AF: Alternative function mode. 非 general input 或 output，而是其他已規定的用途時使用。

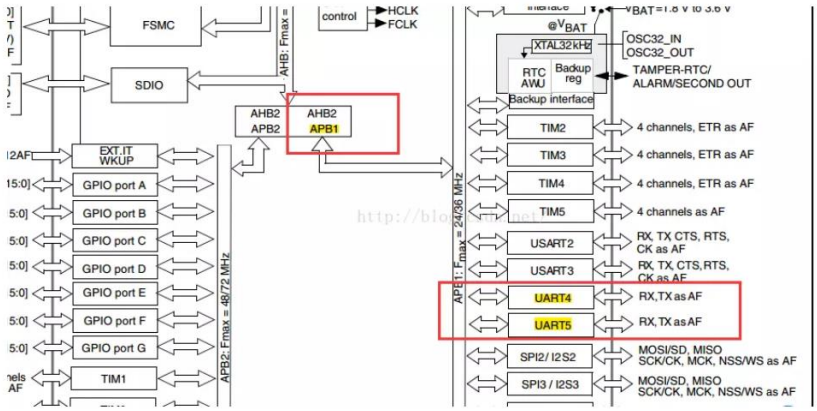
USART 編號怎麼看:



RX:PA9, uart1 input (receive)，接 Uart Tx

TX:PA10, uart1 output (transmission) 接 Uart Rx

如果用其他 port 看表依此類推。



https://www.jianshu.com/p/880362a41016

(以下各參數詳細用途見 datasheet)

USART_CR1、USART_CR2、USART_CR3、USART_TDR

40.8.1 Control register 1 (USART_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

IE

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	M1	EOBIE	RTOIE	DEAT[4:0]					DEDT[4:0]				
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

About TE:

The TE bit must be set before writing the data to be transmitted to the LPUART_TDR.

The TE bit should not be reset during transmission of data. Resetting the TE bit during the transmission will corrupt the data on the TX pin as the baud rate counters will get frozen. The current data being transmitted will be lost.

40.8.2 Control register 2 (USART_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD[7:4]				ADD[3:0]				RTOEN	ABRMOD[1:0]		ABREN	MSBFIRST	DATAINV	TXINV	RXINV
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWAP	LINEN	STOP[1:0]		CLKEN	CPOL	CPHA	LBCL	Res.	LBDIE	LBDL	ADDM7	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw				

40.8.3 Control register 3 (USART_CR3)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCBGTIE	UCESM	WUFIE	WUS1	WUS0	SCARCNT2	SCARCNT1	SCARCNT0	Res.
							rw	rw	rw	rw	rw	rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEP	DEM	DDRE	OVRDIS	ONEBIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

40.8.11 Transmit data register (USART_TDR)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDR[8:0]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **TDR[8:0]**: Transmit data value

Contains the data character to be transmitted.

The TDR register provides the parallel interface between the internal bus and the output shift register (see *Figure 421*).

When transmitting with the parity enabled (PCE bit set to 1 in the USART_CR1 register), the value written in the MSB (bit 7 or bit 8 depending on the data length) has no effect because it is replaced by the parity.

Note: This register must be written only when TXE=1.

40.8.10 Receive data register (USART_RDR)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDR[8:0]								
							r	r	r	r	r	r	r	r	r

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **RDR[8:0]**: Receive data value

Contains the received data character.

The RDR register provides the parallel interface between the input shift register and the internal bus (see *Figure 421*).

When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

USART_ISR: TXE & TC & RXNE

40.8.8 Interrupt and status register (USART_ISR)

Address offset: 0x1C

Reset value: 0x0200 00C0

T R

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	TCBGT	Res.	Res.	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY
						r			r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABRF	ABRE	Res.	EOBF	RTOF	CTS	CTSIF	LBDF	TXE	TC	RXNE	IDLE	ORE	NF	FE	PE
r	r		r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 7 **TXE**: Transmit data register empty

This bit is set by hardware when the content of the USART_TDR register has been transferred into the shift register. It is cleared by a write to the USART_TDR register.

The TXE flag can also be cleared by writing 1 to the TXFRQ in the USART_RQR register, in order to discard the data (only in Smartcard T=0 mode, in case of transmission failure).

An interrupt is generated if the TXEIE bit =1 in the USART_CR1 register.

0: data is not transferred to the shift register

1: data is transferred to the shift register)

Note: This bit is used during single buffer transmission.

Bit 6 **TC**: Transmission complete

This bit is set by hardware if the transmission of a frame containing data is complete and if TXE is set. An interrupt is generated if TCIE=1 in the USART_CR1 register. It is cleared by software, writing 1 to the TCCF in the USART_ICR register or by a write to the USART_TDR register.

An interrupt is generated if TCIE=1 in the USART_CR1 register.

0: Transmission is not complete

1: Transmission is complete

Note: If TE bit is reset and no transmission is on going, the TC bit will be set immediately.

Bit 5 **RXNE**: Read data register not empty

This bit is set by hardware when the content of the RDR shift register has been transferred to the USART_RDR register. It is cleared by a read to the USART_RDR register. The RXNE flag can also be cleared by writing 1 to the RXFRQ in the USART_RQR register.

An interrupt is generated if RXNEIE=1 in the USART_CR1 register.

0: data is not received

1: Received data is ready to be read.

Summary: Character transmission procedure

Transmission Procedure

- Program the M bits in USART_CR1 to define the word length.
- Select the desired baud rate using the USART_BRR register.
- Program the number of stop bits in USART_CR2.
- Enable the USART
- Write the data to send in the USART_TDR register
- After writing the last data into the USART_TDR register, wait until TC=1. This indicates that the transmission of the last frame is complete

Reception Procedure

- Program the M bits in USART_CR1 to define the word length.
- Select the desired baud rate using the baud rate register USART_BRR
- Program the number of stop bits in USART_CR2.
- Enable the USART by writing the UE bit in USART_CR1 register to 1.
- Set the RE bit USART_CR1. This enables the receiver which begins searching for a start bit.
- When a character is received, the RXNE bit is set to indicate that the content of the shift register is transferred to the RDR.

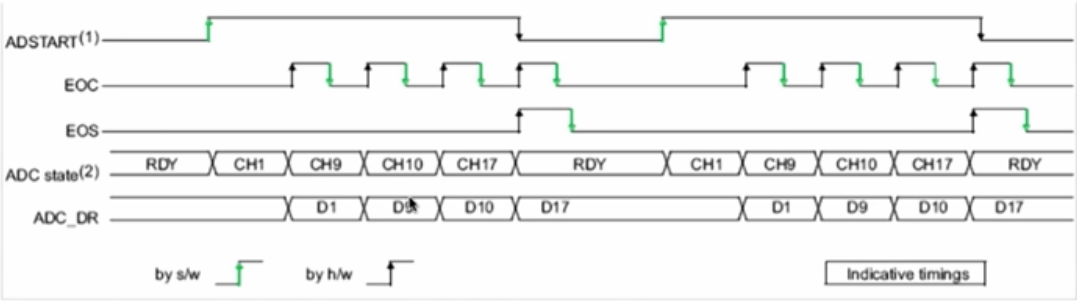
單一 byte：等 TXE

全部資料：等 TC

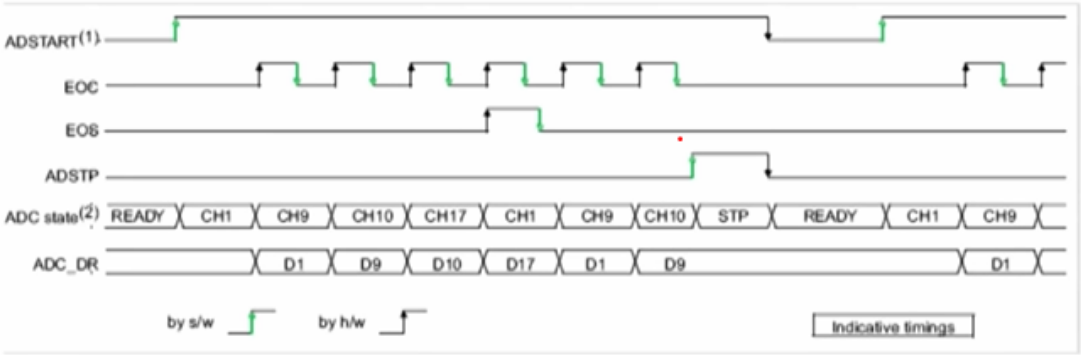
- ADC

Register	Bits	Name	Function
ADCx_ISR	3	EOS	End of regular sequence flag
	2	EOC	End of conversion flag
	0	ADRDY	ADC ready
ADCx_CR	31	ADCAL	Start ADC calibration
	30	ADCALDIF	Differential mode for calibration
	29:28	ADVREGEN[1:0]	ADC voltage regulator enable (Reset value: '01' : disabled)
	2	ADSTART	ADC start of regular conversion
	1	ADDIS	ADC disable command
	0	ADEN	ADC enable control
ADCx_CFGR	13	CONT	Single / continuous conversion mode for regular conversions
	5	ALIGN	Right/Left data alignment
	4:3	RES[1:0]	Data resolution (0:12, 1:10, 2:8, 3:6)
ADCx_SMPR1:2	29:0	SMPx[2:0]	Channel x sampling time selection (ADC clock cycles: 0:1.5; 1:2.5, 2:4.5, 3:7.5, 4:19.5; 5:61.5; 6:181.5; 7:601.5)
ADCx_SQR1:4		SQx[4:0]	x conversion in regular sequence (channel to be converted)
		L[3:0]	Regular channel sequence length (0: 1 conversions, 1: 2 conversions,...)
ADCx_DR	15:0	RDATA[15:0	Regular Data converted
ADCx_CALFACT	6:0	CALFACT_S[6:0]	Calibration factor

單一性轉換



連續性轉換



只在 ADSTP 時停下。

CCIPR：設定 ADC clock source

6.4.28 Peripherals independent clock configuration register (RCC_CCIPR)

Address: 0x88

Reset value: 0x0000 0000

Access: no wait states, word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DFSDM1SEL	SWPMI1SEL	ADCSEL[1:0]		CLK48SEL[1:0]		SAI2SEL[1:0]		SAI1SEL[1:0]		LPTIM2SEL[1:0]		LPTIM1SEL[1:0		I2C3SEL[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I2C2SEL[1:0]		I2C1SEL[1:0]		LPUART1SEL[1:0]		UART5SEL[1:0]		UART4SEL[1:0]		USART3SEL[1:0]		USART2SEL[1:0]		USART1SEL[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 29:28 **ADCSEL[1:0]**: ADCs clock source selection

These bits are set and cleared by software to select the clock source used by the ADC interface.

00: No clock selected

01: PLLSAI1 “R” clock (PLLADC1CLK) selected as ADCs clock

10: PLLSAI2 “R” clock (PLLADC2CLK) selected as ADCs clock

11: System clock selected as ADCs clock

ADC_ISR：回報狀態

18.6.1 ADC interrupt and status register (ADC_ISR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	JQOVF	AWD3	AWD2	AWD1	JEOS	JEOC	OVR	EOS	EOC	EOSMP	ADRDY
					rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bit 2 **EOC**: End of conversion flag

This bit is set by hardware at the end of each regular conversion of a channel when a new data is available in the ADCx_DR register. It is cleared by software writing 1 to it or by reading the ADCx_DR register

0: Regular channel conversion not complete (or the flag event was already acknowledged and cleared by software)

1: Regular channel conversion complete

ADC_CR

18.6.3 ADC control register (ADC_CR)

Address offset: 0x08

Reset value: 0x2000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADCAL	ADCALDIF	DEEPPWD	ADVREGEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rs	rw	rw	rw												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JADSTP	ADSTP	JADSTART	ADSTART	ADDIS	ADEN
										rs	rs	rs	rs	rs	rs

ADC_CFGR

18.6.4 ADC configuration register (ADC_CFGR)

Address offset: 0x0C

Reset value: 0x8000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
JQDIS	AWD1CH[4:0]					JAUTO	JAWD1EN	AWD1EN	AWD1SGL	JQM	JDISCEN	DISCNUM[2:0]			DISCEN
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	AUTDLY	CONT	OVRMOD	EXTEN[1:0]		EXTSEL3	EXTSEL2	EXTSEL1	EXTSEL0	ALIGN	RES[1:0]		DFSDMCFG	DMACFG	DMAEN
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

ADC_CFGR2

18.6.5 ADC configuration register 2 (ADC_CFGR2)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	ROVSM	TROVS	OVSS[3:0]				OVSR[2:0]			JOVSE	ROVSE
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

SQR1

18.6.11 ADC regular sequence register 1 (ADC_SQR1)

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	SQ4[4:0]					Res.	SQ3[4:0]					Res.	SQ2[4]
			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ2[3:0]				Res.	SQ1[4:0]					Res.	Res.	L[3:0]			
rw	rw	rw	rw		rw	rw	rw	rw	rw			rw	rw	rw	rw

SMPR1

Channel-wise programmable sampling time (SMPR1, SMPR2)

Before starting a conversion, the ADC must establish a direct connection between the voltage source under measurement and the embedded sampling capacitor of the ADC. This sampling time must be enough for the input voltage source to charge the embedded capacitor to the input voltage level.

Each channel can be sampled with a different sampling time which is programmable using the SMP[2:0] bits in the ADCx_SMPR1 and ADCx_SMPR2 registers. It is therefore possible to select among the following sampling time values:

- SMP = 000: 2.5 ADC clock cycles
- SMP = 001: 6.5 ADC clock cycles
- SMP = 010: 12.5 ADC clock cycles
- SMP = 011: 24.5 ADC clock cycles
- SMP = 100: 47.5 ADC clock cycles
- SMP = 101: 92.5 ADC clock cycles
- SMP = 110: 247.5 ADC clock cycles
- SMP = 111: 640.5 ADC clock cycles

The total conversion time is calculated as follows:

T_{CONV} = Sampling time + 12.5 ADC clock cycles

Example:

With F_{ADC_CLK} = 80 MHz and a sampling time of 2.5 ADC clock cycles:

T_{CONV} = (2.5 + 12.5) ADC clock cycles = 15 ADC clock cycles = 187.5 ns (for fast channels)

要看 Table16 去設定。

Ex：PC1 可以當 ADC1 or 2 or 3 的 channel2 output。

若設定 L[1:0]=0 (Regular channel sequence length=1)且使用 ADC3：

則要去 ADC3->SQR1 把 SQ1[4:0]設為 2。

ADC3->SMPR1 的 SMP2 設你要的 sampling time。

Calibration 校準

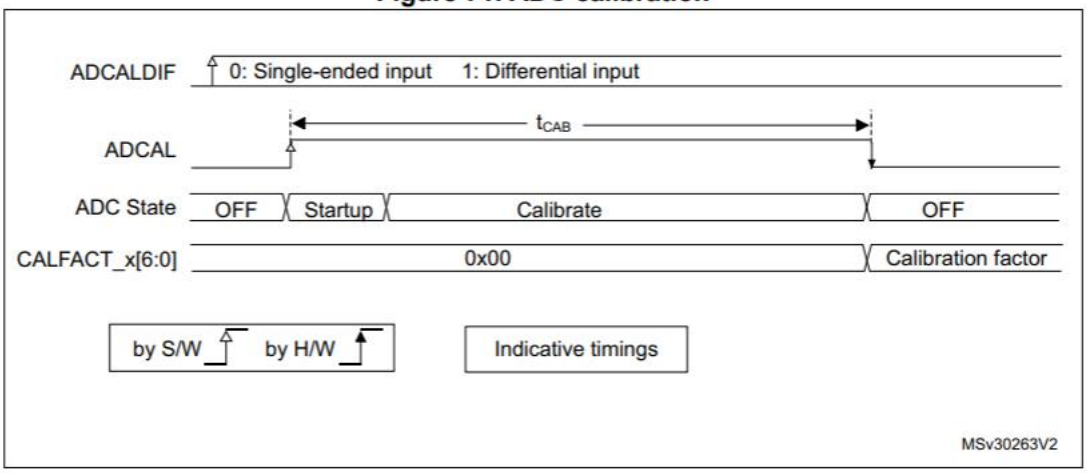
ADC 正式運作前必須做 calibration，以避免電容的狀態不是預期而造成誤差

Software procedure to calibrate the ADC

Software procedure to calibrate the ADC

1. Ensure DEEPPWD=0, ADVREGEN=1 and that ADC voltage regulator startup time has elapsed.
2. Ensure that ADEN=0.
3. Select the input mode for this calibration by setting ADCALDIF=0 (single-ended input) or ADCALDIF=1 (differential input).
4. Set ADCAL=1.
5. Wait until ADCAL=0.
6. The calibration factor can be read from ADCx_CALFACT register.

Figure 71. ADC calibration



小心 ADVREGEN (voltage regulater)

Bit 28 **ADVREGEN**: ADC voltage regulator enable

This bits is set by software to enable the ADC voltage regulator.
Before performing any operation such as launching a calibration or enabling the ADC, the ADC voltage regulator must first be enabled and the software must wait for the regulator start-up time.
0: ADC Voltage regulator disabled
1: ADC Voltage regulator enabled.
For more details about the ADC voltage regulator enable and disable sequences, refer to *Section 18.4.6: ADC Deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN)*.

The software can program this bit field only when the ADC is disabled (ADCAL=0, JADSTART=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).

ADC_CR_DEEPPWD = 0, too.

18.4.6 ADC Deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN)

By default, the ADC is in Deep-power-down mode where its supply is internally switched off to reduce the leakage currents (the reset state of bit DEEPPWD is 1 in the ADCx_CR register).

To start ADC operations, it is first needed to exit Deep-power-down mode by setting bit DEEPPWD=0.

Then, it is mandatory to enable the ADC internal voltage regulator by setting the bit ADVREGEN=1 into ADCx_CR register. The software must wait for the startup time of the ADC voltage regulator ($T_{ADCVREG\_STUP}$) before launching a calibration or enabling the ADC. This delay must be implemented by software.

For the startup time of the ADC voltage regulator, refer to device datasheet for $T_{ADCVREG\_STUP}$ parameter.

After ADC operations are complete, the ADC can be disabled (ADEN=0). It is possible to save power by also disabling the ADC voltage regulator. This is done by writing bit ADVREGEN=0

Then, to save more power by reducing the leakage currents, it is also possible to re-enter in ADC Deep-power-down mode by setting bit DEEPPWD=1 into ADCx_CR register. This is particularly interesting before entering STOP mode.

Delay for ADC voltage regulator startup time in STM32L4 :

#define ADC_STAB_DELAY_US ((uint32_t) 10)

也就是 10us

特別是 ADCAL，需在 ADEN=1 and ADSTART=0 and JADSTART=0 時做

Bit 31 **ADCAL**: ADC calibration

This bit is set by software to start the calibration of the ADC. Program first the bit ADCALDIF to determine if this calibration applies for single-ended or differential inputs mode.
It is cleared by hardware after calibration is complete.
0: Calibration complete
1: Write 1 to calibrate the ADC. Read at 1 means that a calibration in progress.

Note: The software is allowed to launch a calibration by setting ADCAL only when ADEN=0.
The software is allowed to update the calibration factor by writing ADCx_CALFACT only when ADEN=1 and ADSTART=0 and JADSTART=0 (ADC enabled and no conversion is ongoing)

Finally, enable ADC

Software procedure to enable the ADC

1. Clear the ADRDY bit in the ADCx_ISR register by writing ‘1’.
2. Set ADEN=1.
3. Wait until ADRDY=1 (ADRDY is set after the ADC startup time). This can be done using the associated interrupt (setting ADRDYIE=1).
4. Clear the ADRDY bit in the ADCx_ISR register by writing ‘1’ (optional).

然後 DR 就會放轉換出的電壓值。

ADC _DR：ADC1 Converted Value (Result)

18.6.15 ADC regular Data Register (ADC_DR)

Address offset: 0x40

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDATA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **RDATA[15:0]**: Regular Data converted

These bits are read-only. They contain the conversion result from the last converted regular channel. The data are left- or right-aligned as described in *Section 18.4.26: Data management*.

Procedure

- To start ADC operations, it is first needed to exit deep-power-down mode by setting bit DEEPPWD=0.
- It is mandatory to enable the ADC internal voltage regulator by setting the bit ADVREGEN=1 into ADCx_CR register.
- Once DEEPPWD=0 and ADVREGEN=1, the ADC can be enabled
  - Clear the ADRDY bit in the ADC_ISR register by writing ‘1’.
  - Set ADEN=1.
  - Regular conversion can then start by setting ADSTART=1

USART interrupts

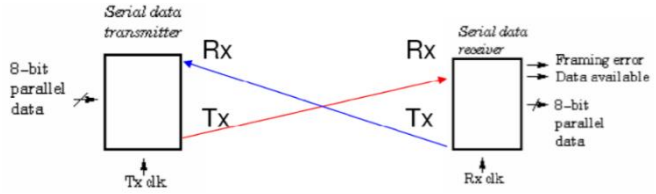
3.實驗步驟

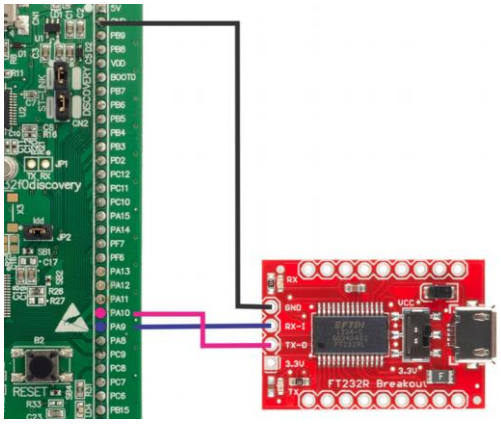
3.1. Lab8.1 : Hello World! (30%)

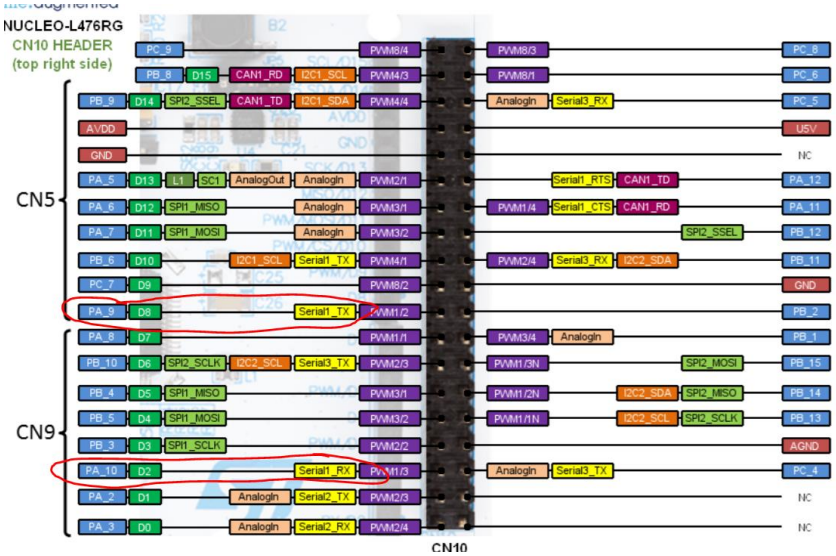
在按下板子上藍色按鈕時 (PC13)，請利用 UART 將 "Hello World!" 字串傳送到電腦。並且可以在電腦端的 serial monitor (e.g. putty, pietty, MobaXterm ...) 顯示出來。

- init 需要使用到的 GPIO。
- 瞭解 UART 的暫存器以及使用方式
- 利用 UART 的傳出 (TX) 來實作此功能

把 Rx, Dx 插在相對的 port。

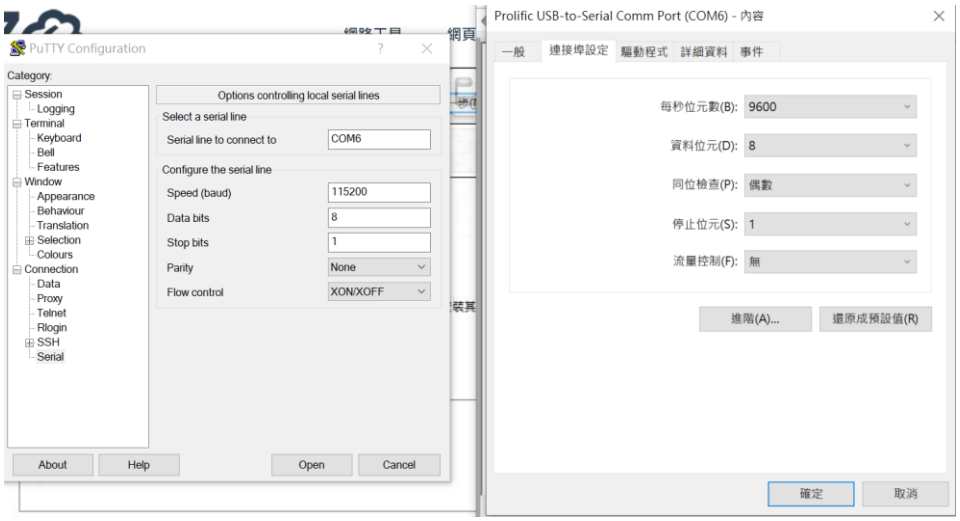







```
int UART_Transmit(uint8_t *arr, uint32_t size) {
for (int i = 0; i < ???; i++) {
while (!IS_UART_TRANS_READY);
USART???->TDR = ???;
}
while (!IS_UART_TRANS_DONE);
return ???;
}
```

- 其他軟體的設定注意：



裝置管理員和 putty 的 serial 設定，和你的 code 設定要完全一樣(上面是錯的)。

- UART init

CR1: 大部分的設定

```
SET_REG(USART1->CR1, USART_CR1_M, USART_CR1_M0);//0
SET_REG(USART1->CR1, USART_CR1_PS, 0);//0: Even par
SET_REG(USART1->CR1, USART_CR1_PCE, USART_CR1_PCE);
SET_REG(USART1->CR1, USART_CR1_RE, 0);//0: Receiver
SET_REG(USART1->CR1, USART_CR1_OVER8, 0);//0: Overs
```

CR2: stop bit

```
SET_REG(USART1->CR2, USART_CR2_STOP, 0);
```

CR3: Flow control

(RTS/CTS/1-bit 我們全關掉)

```
SET_REG(USART1->CR3, USART_CR3_RTSE, 0);//0
SET_REG(USART1->CR3, USART_CR3_CTSE, 0);//0
SET_REG(USART1->CR3, USART_CR3_ONEBIT, 0);//0
RRR
```

問題：

- 沒顯示東西

PA9:TX 接 RX, PA10:RX 接 TX (綠接 10 白接 9)
TDR:資料正常
USART_ISR_TC:有在運作
ECHO (RX 自接 TX):正常 (不是 COM6 問題)
GPIOA mode (!)

8.4.10 GPIO alternate function high register (GPIOx_AFRH) (x = A to I)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFSEL15[3:0]				AFSEL14[3:0]				AFSEL13[3:0]				AFSEL12[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFSEL11[3:0]				AFSEL10[3:0]				AFSEL9[3:0]				AFSEL8[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

註：For each GPIO pins 0 through 7 are set in AFR[0] which is called GPIOx_AFRL. For GPIO pins 8 through 15 set the corresponding 4 bits for the desired pin in AFR[1] which is called GPIOx_AFRH

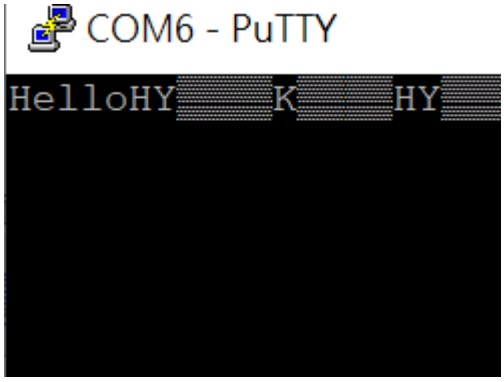
PA9,PA10 的 UART 的 Alternative Function 是 AF7，在 AFRH 內

Port A: Alternate Functions (STM32L4)

Pin	AF0	AF1	AF2	AF3	AF4	AF5	AF6	AF7
	SYS_AF	TIM1/ TIM2/ TIM5/ TIM8/ LPTIM1	TIM1/ TIM2/ TIM3/ TIM4/ TIM5	TIM8	I2C1/ I2C2/ I2C3	SPI1/ SPI2	SPI3/ DFSDM	USART1/ USART2/ USART3
PA0		TIM2_CH1	TIM5_CH1	TIM8_ETR	-	-	-	USART2_CTS
PA1	-	TIM2_CH2	TIM5_CH2	-	-	-	-	USART2_RTS_DE
PA2	-	TIM2_CH3	TIM5_CH3	-	-	-	-	USART2_TX
PA3	-	TIM2_CH4	TIM5_CH4	-	-	-	-	USART2_RX
PA4	-	-	-	-	-	SPI1_NSS	SPI3_NSS	USART2_CK
PA5	-	TIM2_CH1	TIM2_ETR	TIM8_CH1N	-	SPI1_SCK	-	-
PA6	-	TIM1_BKIN	TIM3_CH1	TIM8_BKIN	-	SPI1_MISO	-	USART3_CTS
PA7	-	TIM1_CH1N	TIM3_CH2	TIM8_CH1N	-	SPI1_MOSI	-	-
PA8	MCO	TIM1_CH1	-	-	-	-	-	USART1_CK
PA9	-	TIM1_CH2	-	-	-	-	-	USART1_TX
PA10	-	TIM1_CH3	-	-	-	-	-	USART1_RX
PA11	-	TIM1_CH4	TIM1_BKIN2	-	-	-	-	USART1_CTS
PA12	-	TIM1_ETR	-	-	-	-	-	USART1_RTS_DE
PA13	JTMS/ SWDIO	IR_OUT	-	-	-	-	-	-
PA14	JTCK/ SWCLK	-	-	-	-	-	-	-
PA15	JTDI	TIM2_CH1	TIM2_ETR	-	-	SPI1_NSS	SPI3_NSS	-

AF8	AF9	AF10	AF11	AF12	AF13	AF14
UART4/ UART5/ LPUART1	CAN1/ TSC	OTG_FS/ QUADSPI	LCD	SDMMC1/ COMP1/ COMP2/ FMC/ SWPMI1	SAI1/ SAI2	TIM2/ TIM1/ TIM16/ TIM17/ LPTIM2
UART4_TX	-	-	-	-	SAI1_EXTCLK	TIM2_ETR
UART4_RX	-	-	LCD_SEG0	-	-	TIM15_CH1N
-	-	-	LCD_SEG1	-	SAI2_EXTCLK	TIM15_CH1
-	-	-	LCD_SEG2	-	-	TIM15_CH2
-	-	-	-	-	SAI1_FS_B	LPTIM2_OUT
-	-	-	-	-	-	LPTIM2_ETR
-	-	QUADSPI_BK1_IO3	LCD_SEG3	TIM1_BKIN_COMP2	TIM8_BKIN_COMP2	TIM16_CH1
-	-	QUADSPI_BK1_IO2	LCD_SEG4	-	-	TIM17_CH1
-	-	OTG_FS_SOF	LCD_COM0	-	-	LPTIM2_OUT
-	-	-	LCD_COM1	-	-	TIM15_BKIN
-	-	OTG_FS_ID	LCD_COM2	-	-	TIM17_BKIN
-	CAN1_RX	OTG_FS_DM	-	TIM1_BKIN2_COMP1	-	-
-	CAN1_TX	OTG_FS_DP	-	-	-	-
-	-	OTG_FS_NOE	-	-	-	-
-	-	-	-	-	-	-
UART4_RTS_DE	TSC_G3_IO1	-	LCD_SEG17	-	SAI2_FS_B	-

- 全亂碼



解法；

1. 忘記等 TDR 傳完！
2. 資料長度沒對齊 (PCE 有設定的話要變 8+1=9bit data)

<設斷點一次次執行時發現不會有錯時有可能發生>

- 只有開頭亂碼

解法；

1. TE 要在 UE 後開 (還沒管好嘴巴不要亂說話)

<設斷點在執行 Transmit 前，發現就偷傳了那一個亂碼>

8.2 光敏電阻讀值

請利用板子上提供的 ADC (Analog-to-Digital Converter) 利用 Interrupt 將光敏電阻的值以 12-bit 的解析度讀出，並且每按一次按鈕 (PC13) 時利用 UART 輸出數值。

- 開啟 ADC 並且初始化其設定
- 每次按下按鈕利用 UART 傳輸光敏電阻值出去

ADC 是歸 GPIOC 管的，同時記得要選 ADC 的 Clock source(CCIPR)

```
    //ADC1: one of ADC's instances
RCC->AHB2ENR |= RCC_AHB2ENR_GPIOCEN;
RCC->AHB2ENR |= RCC_AHB2ENR_ADCEN; //
//ADC peripheral clock
RCC->CCIPR |= (RCC_CCIPR_ADCSEL_1|RCC_CCIPR_ADCSEL_0); //as system clock
```

- ADC init

我用 pc2 當 analog output，所以要開 ADC1 (or 2 or 3) 的 channel 3

同時 pc2 要開 analog mode。

還要記得 GPIOC 設定 ASCR (Analog Switch Control Register)，與 PC1 連在一起

```
    //PC2: analog mode
    GPIOC->MODER |= GPIO_MODER_MODER2_1;
    GPIOC->MODER |= GPIO_MODER_MODER2_0;
    GPIOC->OTYPER  &=~ GPIO_OTYPER_OT2; //PP
    GPIOC->PUPDR  &=~ GPIO_PUPDR_PUPD2_1;
    GPIOC->PUPDR  &=~ GPIO_PUPDR_PUPD2_0;//00 no
    GPIOC->OSPEEDR &=~ GPIO_OSPEEDR_OSPEED2_1;
    GPIOC->OSPEEDR &=~ GPIO_OSPEEDR_OSPEED2_0;//00 low
    GPIOC->ASCR |= GPIO_ASCR_ASC2;
}

 //PC2 is the output of ADC"1" regular channel"3" conf
 //(See Production data Table 16. STM32L476xx pin defi
 ADC1->SQR1 &= ~ADC_SQR1_L; //0000: Regular channel se
 // 1 conversion => sq1
 ADC1->SQR1 &=~ ADC_SQR1_SQ1_3;
 ADC1->SQR1 &=~ ADC_SQR1_SQ1_3;
 ADC1->SQR1 &=~ ADC_SQR1_SQ1_2;
 ADC1->SQR1 |= ADC_SQR1_SQ1_1;
 ADC1->SQR1 |= ADC_SQR1_SQ1_0;//0011 = channel"3"

 ADC1->SMPR1 |= (ADC_SMPR1_SMP3_1 |ADC_SMPR1_SMP3_0);
```

問題：

讀不到值通常是因為忘記開 CCIPR

只讀到相同的值通常是因為沒有設 ASCR，或是 SQRn 和 SMPRn 沒設好

記得查表檢查你的 sampling port 有沒有支援 ADC analog。

只讀到相同的值另一個原因是沒設 Overrun mode：

ADC overrun (OVR, OVRMOD)

The overrun flag (OVR) notifies of a buffer overrun event, when the regular converted data was not read (by the CPU or the DMA) before new converted data became available.

The OVR flag is set if the EOC flag is still 1 at the time when a new conversion completes. An interrupt can be generated if bit OVRIE=1.

When an overrun condition occurs, the ADC is still operating and can continue to convert unless the software decides to stop and reset the sequence by setting bit ADSTP=1.

OVR flag is cleared by software by writing 1 to it.

It is possible to configure if data is preserved or overwritten when an overrun event occurs by programming the control bit OVRMOD:

- OVRMOD=0: The overrun event preserves the data register from being overrun: the old data is maintained and the new conversion is discarded and lost. If OVR remains at 1, any further conversions will occur but the result data will be also discarded.
- OVRMOD=1: The data register is overwritten with the last conversion result and the previous unread data is lost. If OVR remains at 1, any further conversions will operate normally and the ADCx_DR register will always contain the latest converted data.

Overrun mode 讓 DR 會一直更新，即使沒有去 read DR 內的值！

8.3 Simple Shell
在板子上利用 UART 實作 Simple Shell，並且擁有三個指令
1) showid：顯示你的學號
2) light：以每 0.5 秒更新並顯示光敏電阻的值 (lab10.2)、按 'q' 回到原本的 shell 模式
3) led {onloff}：led on 將會把 PA5 的 LED 打開、led off 則會關閉
在電腦端連接 USB 到 UART 後，任何電腦輸入的字元必須透過板子的 UART RX 收到並且在利用 TX 傳回給電腦達到 echo 的效果。電腦端跑 serial monitor 去使用該 shell.

- 用 UART Tx 做 Terminal

因為要立即反應 Terminal 每一個字(收到字元就輸出，不然看不到你打了什麼)，所以要用 interrupt (NVIC, IRQHandler)。

```
void NVIC_config()
{
    NVIC_EnableIRQ(EXTI15_10_IRQn);
    NVIC_EnableIRQ(USART1_IRQn);
}
```

```
void USART1_IRQHandler(void){
```

- Tx 只要一收到一個字元就會放進 TDR，一旦好了就會跳 interrupt，觸發 handler

```
//recv interrupt enable bit
SET_REG(USART1->CR1, USART_CR1_TXEIE, 0);//0
SET_REG(USART1->CR1, USART_CR1_TCIE, 0);//0
SET_REG(USART1->CR1, USART_CR1_RXNEIE, USART_CR1_RXNEIE);//1
```

(只有打字才觸發，不然每次 uart transmit 都會 interrupt)

- 收到字元就輸出，並且記錄現在指令多長了。

```
        else{
            UART_Transmit(&c, 1);
            cmd[len++] = c;
        }
```

- 之後就是一些控制字串的部分，再依照字串種類挑功能執行(從之前 lab 拿)

```
else if(~TIM5->CR1 & TIM_CR1_CEN == TIM_CR1_CEN ){
    if (c == 13){ //\n
        UART_Transmit("\r\n", 4);
        //remind :\0
        cmd[len]= 0;
        int cmp=1;
        int none=1;
        cmp = strcmp(cmd, "showid");
        if (cmp == 0) {
            mode = 1;
            none = 0;
        }
        cmp = strcmp(cmd, "light");
        if (cmp == 0) {
            mode = 2;
            none = 0;
        }
        cmp = strcmp(cmd, "led on");
        if (cmp == 0) {
            mode = 3;
            none = 0;
        }
        cmp = strcmp(cmd, "led off");
        if (cmp == 0) {
            mode = 4;
            none = 0;
        }
```

例如：

1. 吃到 enter：要 transmit \r\n <註：長度=4>，刷新指令長度，並放結束字元\0 (ascii 0)，不然會影響下一個字元的讀取
2. 吃到 backspace：要把目前暫存指令的長度-2。(backspace + 上一個字元)
   (註：直接輸出 backspace 可以刪掉上個字元！)
3. 在 light mode(也就是 TIM5 enable 時)，要屏蔽掉除了 q 以外的字元…等 (略)

4.參考資料

¤ 跨平台資料型別:

uint8_t = 8 bits (char)

uint32_t = 32 bits (int)

https://blog.csdn.net/Mary19920410/article/details/71518130

¤ 顯示裝置是用哪一個 com port

From the Device Manager, select View - Show Hidden Devices. Now when you expand the (PORTS) COM ports section you will see all of the COM ports listed there.

¤ USB PL2303HX 在 Win8 Win8.1 Win10 無法使用的解決辦法

https://www.ez2o.com/Blog/Post/PL2303HX-Win8-Win8.1-Win10

成大資工有一個詳盡的圖，介紹傳送流程

http://wiki.csie.ncku.edu.tw/embedded/USART#usart-block-diagram

new line issue：\r\n vs only \n

https://seacatcry.pixnet.net/blog/post/13732061-%E3%80%90%E8%BD%89%E8%B2%BC%E3%80%91%5Cr%5Cn%E5%92%8C%5Cn%E7%9A%84%E5%B7%AE%E7%95%B0

http://dinownote.blogspot.com/2015/11/stm32-3-usart.html

&=~: 設成 0

|=: 設成 1

Errors running builder 'CDT Builder' when trying to build

https://stackoverflow.com/questions/11377765/errors-running-builder-cdt-builder-when-trying-to-build-opencv-for-android-sam

Interrupt event table

Table 116. ADC interrupts per each ADC

Interrupt event	Event flag	Enable control bit
ADC ready	ADRDY	ADRDYIE
End of conversion of a regular group	EOC	EOCIE
End of sequence of conversions of a regular group	EOS	EOSIE
End of conversion of a injected group	JEOC	JEOCIE
End of sequence of conversions of an injected group	JEOS	JEOSIE
Analog watchdog 1 status bit is set	AWD1	AWD1IE
Analog watchdog 2 status bit is set	AWD2	AWD2IE
Analog watchdog 3 status bit is set	AWD3	AWD3IE
End of sampling phase	EOSMP	EOSMPIE
Overrun	OVR	OVRIE
Injected context queue overflows	JQOVF	JQOVFIE

Why ADRDY or ADCAL doesn’t work ?

https://stackoverflow.com/questions/54856078/stm32l476-adc-not-ready

How to return an array in C

https://www.tutorialspoint.com/cprogramming/c_return_arrays_from_function.htm

ADC value (DR) not changing, or flickering around certain value

https://electronics.stackexchange.com/questions/416809/stm32l476-discovery-adc-output-is-not-changing-even-though-the-input-voltage-is

還有這半個學期的一切。