ELEC 3300 – Tutorial for LAB5

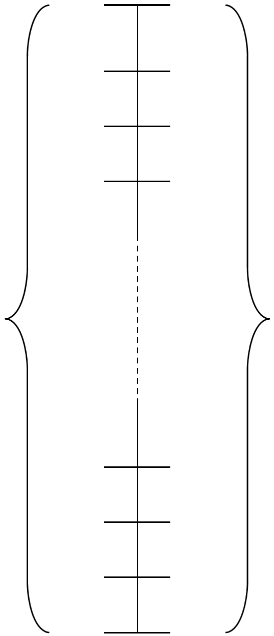
Department of Electronic and Computer Engineering
HKUST
by WU Chi Hang 

Terminology

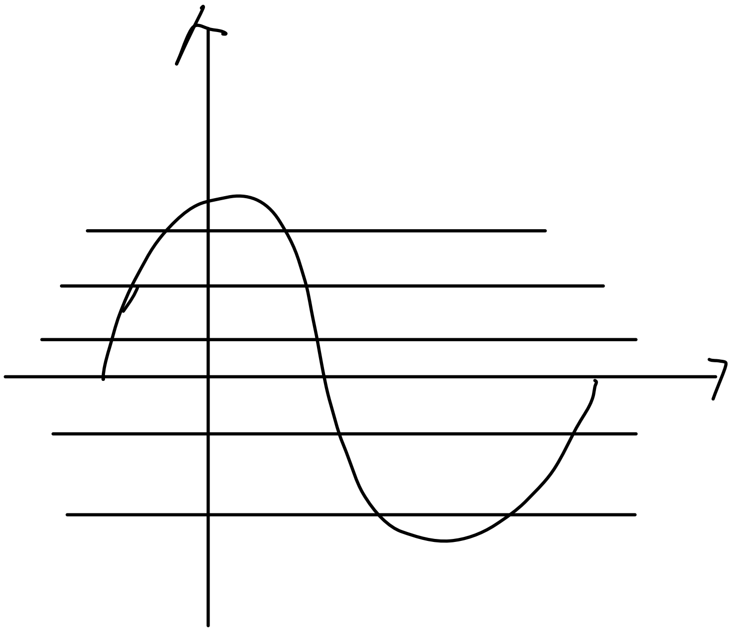
ADC : analog to digital conversion

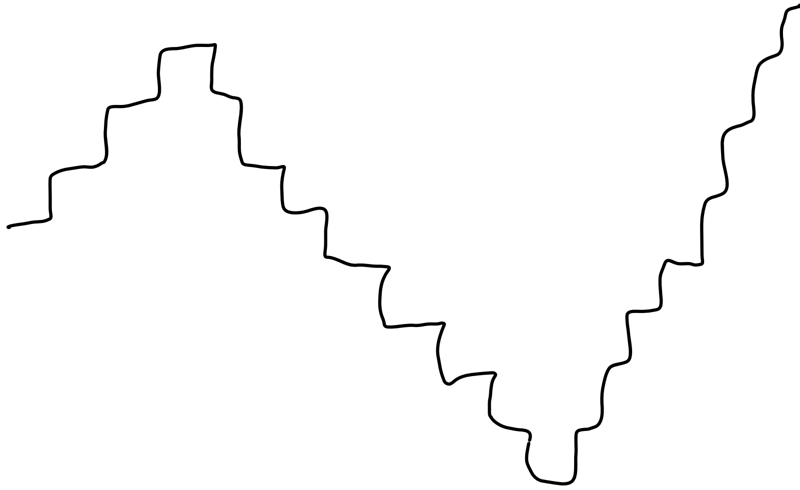
Analogue $\longrightarrow$ Digital

The whole analogue range that you want to chop into



How many levels depends on the number of bits used.





Nyquist rate
20kHZ * 2
= 40kHZ

L - 16bits   R - 16bits

sampling rate = 44.1kHZ

why not higher?

because human hearing range 20 - 20kHz

An N-bit Linear DAC/ADC

$2^N - 1$

$\frac{V_{max}}{2^n} \times 2^N - 1$

So can never output $\Rightarrow V_{max}$

Analogue

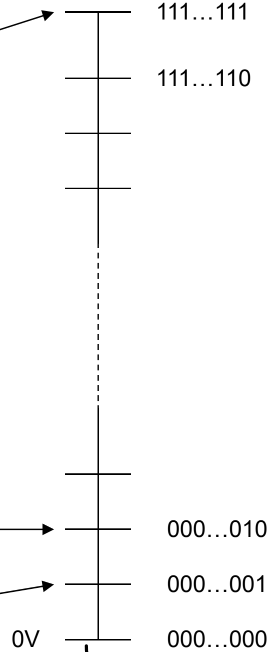
e.g. $0 - V_{max}$

Digital

N bits

$\frac{V_{max}}{2^n}$

$V_{out} = ?$



How many levels are there ?

can be "chopped" into?

2^n levels

$$V_{out} = \frac{V_{max}}{2^N} \times 2$$

$$V_{out} = \frac{V_{max}}{2^N} \times 1$$

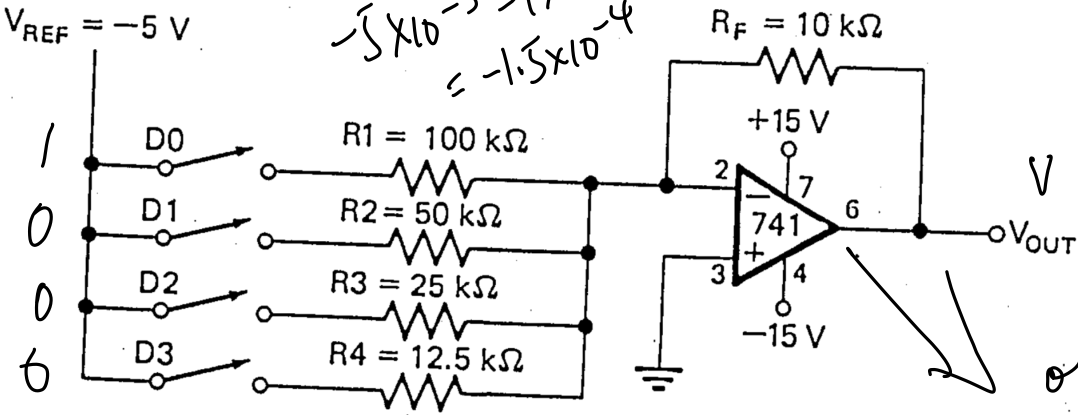
0 already occupy 1 bit

Digital to Analogue Converter (DAC)

DAC: need setting time

Digital to Analog

$-5\times10^{-5} - 1\times10^{-4}$
$= -1.5\times10^{-4}$



1
0
0
0

2V or $\frac{1}{2}$V

V

operation amplifier

FIGURE 10-14 Simple 4-bit D/A converter.

takes time: setting time

1
1
0
0

$D_0 + D_1 = 3V$

negative feedback summing amplifier

Terminology

- Resolution
  - Number of bits in the binary word used in the sample.
- Full Scale Output Voltage (for DAC)
  - Maximum output voltage of the D/A convertor. (Always 1 LSB below the stated value)
- Setting Time (for DAC)
  - When you change a binary word applied to the input of a converter, the output will change to appropriate new value. (Time the output to takes to get within ± ½ LSB
- Conversion Time (for ADC)
  - Time that convertor takes to produce a valid output binary code for an applied input voltage.

8bit after

challenge (comparator → Binary search 8 comparison I get my result.)

ADC → successive-approximation

Analogue to Digital Converter (ADC) 8 bit

comparator

$V_{max} \times 30 / 256$

128

corresponding voltage

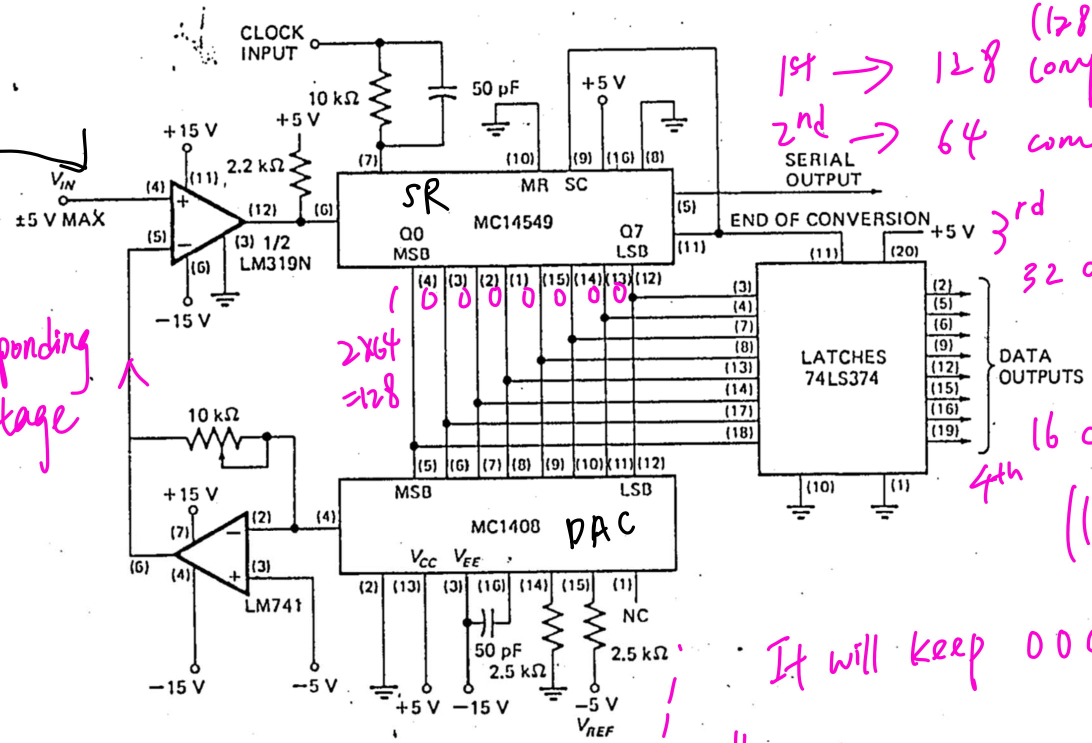


FIGURE 10-20 Successive-approximation A/D converter circuit.

conversion process

1st → 128 compares with 30 (128 > 30)

2nd → 64 compares with 30 (64 > 30)

3rd 32 compares with 30 (32 > 30)

4th 16 compares with 30 (16 < 30)

It will keep 00010000

5th 00011000 24 < 30

6th 00011100 28 < 30

8th 00011111 → 9th 00011110 | 7th → 00011110
31 > 30 30 = 30 30 = 30

Analogue to Digital Convertor (ADC)

- In STM32 there are three ADCs inside. Each ADC is 12-bit. It has up to 18 multiplexed channels allowing it measure signals from 16 external and two internal sources.
- A/D conversion of the various channels can be performed in single, continuous, scan or discontinuous mode.
- The result of the ADC is stored in a left-aligned or right-aligned 16-bit data register.
- The conversion time is around 1μs (refer to page 207 of the reference manual)

$T = \frac{1}{f} \Rightarrow f = \frac{1}{1\mu s}$

can be used as a CD ripper

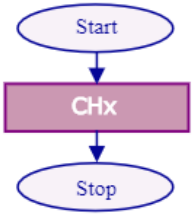
$f_{eq} = 1MHz > 44.1kHz$

About Conversion Mode

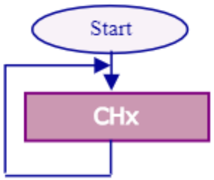
- A/D conversion of the various channels can be performed in single, continuous, scan or discontinuous mode.
  - Single conversion mode
    - ADC does one conversion, after finished conversion, ADC stopped.
  - Continuous conversion mode
    - ADC starts another conversion as soon as it finishes one.
  - Scan mode
    - This mode is used to scan a group of analog channels. A single conversion is performed for each channel of the group. After each end of conversion the next channel of the group is converted automatically.
    - When using scan mode, DMA bit must be set and the direct memory access controller is used to transfer the converted data of regular group channels to SRAM after each update of the ADC_DR register.
- Please refer to Reference Manual Section 11.3 for details.

About Conversion Mode

Single Channel
Single Conversion Mode

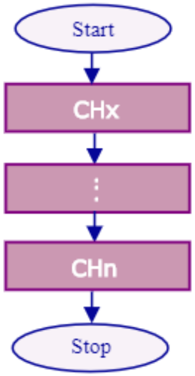


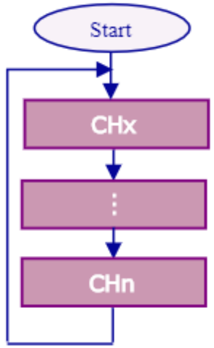
Single Channel
Continuous Conversion Mode



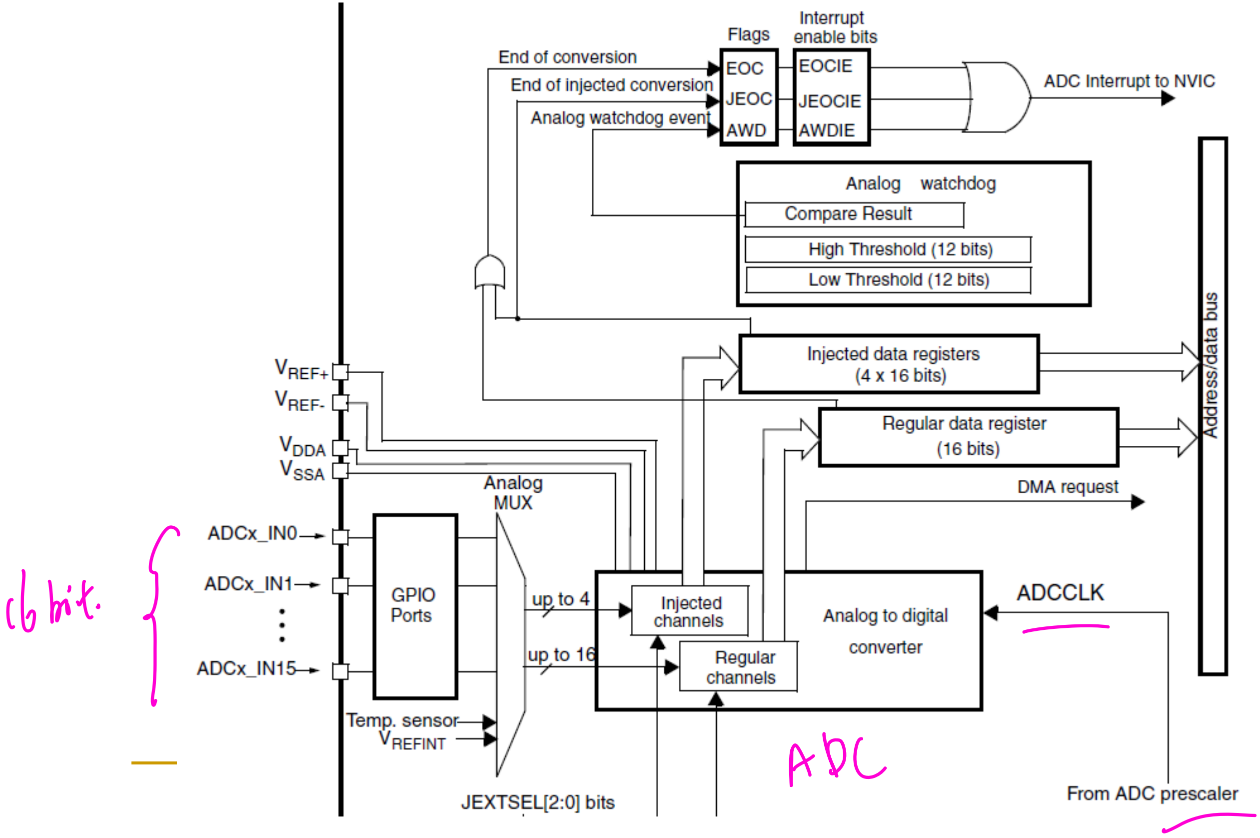
Scan mode

Multi-channel Single Conversion Mode	Multi-channel Continuous Conversion Mode

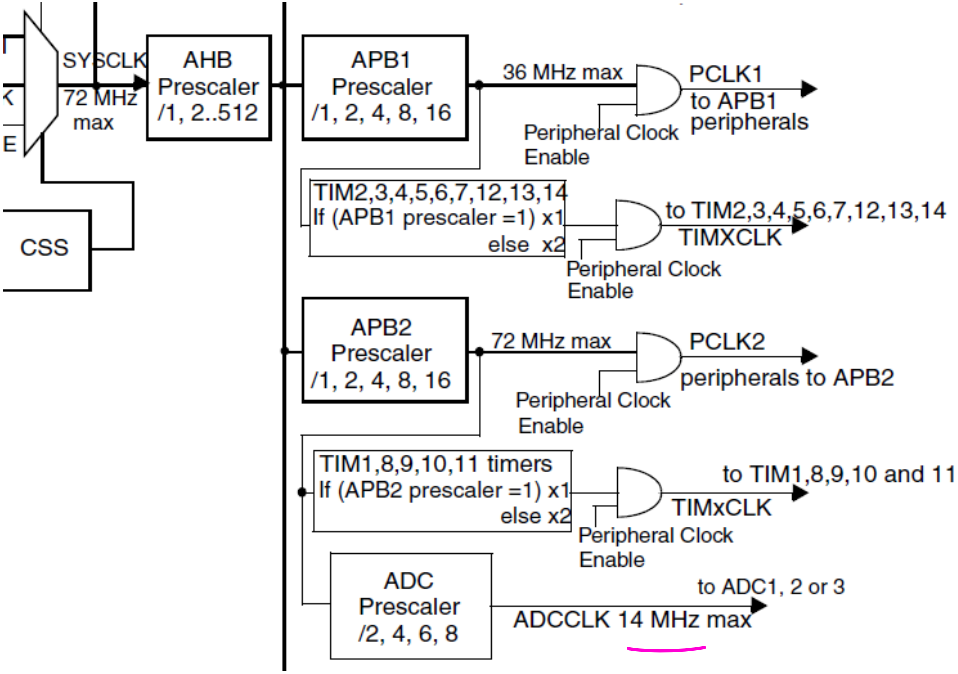




Block Diagram (Ref. Man. P. 208)



Clock Tree



Where is the ADCCLK source ? APB1/APB2 ?
What is the Max value of ADCCLK ?

Note on ADCCLK

Refer to Page 207 of the Reference Manual, it said

- ADC conversion time:
  - STM32F103xx performance line devices: 1 μs at 56 MHz (1.17 μs at 72 MHz)
- Question
  - Why the SYSCLK faster, but conversion time is longer ?

Ans:
ADC Prescaler

Refer to last page.

- ADCCLK originates from APB2
- The Prescaler of ADCCLK can only be /2, 4, 6, 8
- If APB2 = 72MHz, what the max ADCCLK can be ? 72MHz / 6 = 12MHZ
- If APB2 = 56MHz, what the max ADCCLK can be ? 56MHz / 6 = 14MHz

Total Conversion Time

Refer to Section 11.6 of the Reference Manual.

- Each channel can be sampled with a different sample time.
- The total conversion time is calculated as follows:
  - Tconv = Sampling time + 12.5 cycles → fixed (time during conversion)
- The sampling time can be set via the SMP[2:0] bits in the ADC_SMPR1 and ADC_SMPR2 registers.
- Example:
- With an ADCCLK = 14 MHz and a sampling time of 1.5 cycles:
  - Tconv = 1.5 + 12.5 = 14 cycles = 1 μs

can be programmed

Sampling Time

11.12.4 ADC sample time register 1 (ADC_SMPR1)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								SMP17[2:0]			SMP16[2:0]			SMP15[2:1]	
								rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMP 15_0	SMP14[2:0]			SMP13[2:0]			SMP12[2:0]			SMP11[2:0]			SMP10[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:0 **SMPx[2:0]:** Channel x Sample time selection

These bits are written by software to select the sample time individually for each channel. During sample cycles channel selection bits must remain unchanged.

000: 1.5 cycles
001: 7.5 cycles
010: 13.5 cycles
011: 28.5 cycles
100: 41.5 cycles
101: 55.5 cycles
110: 71.5 cycles
111: 239.5 cycles

1.5+12.5=14 cycles

$$\frac{14}{14\text{MHz}} = 1\,\mu s$$

Note: ADC1 analog Channel16 and Channel 17 are internally connected to the temperature sensor and to V_{REFINT}, respectively.

ADC2 analog input Channel16 and Channel17 are internally connected to V_{SS}.

ADC3 analog inputs Channel14, Channel15, Channel16 and Channel17 are connected to V_{SS}.

About Data Alignment

- The data alignment means how the ADC data is being put into the register.

Figure 27. Right alignment of data

Injected group

SEXT	SEXT	SEXT	SEXT	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Regular group

0	0	0	0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Figure 28. Left alignment of data

Injected group

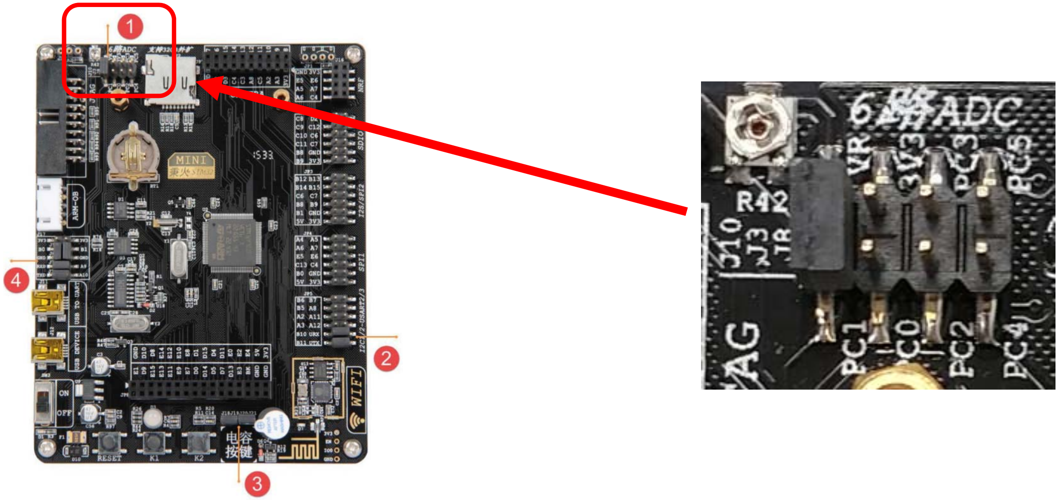
SEXT	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0

Regular group

D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0	0

ADC in MINI-V3

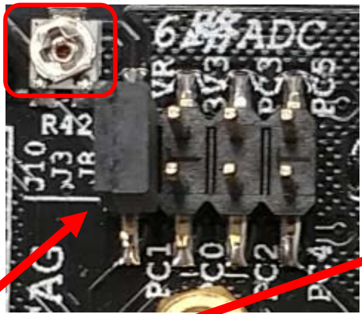
- In MINI-V3, there are different ADC channels located at position 1 of the figure below



ADC in MINI-V3

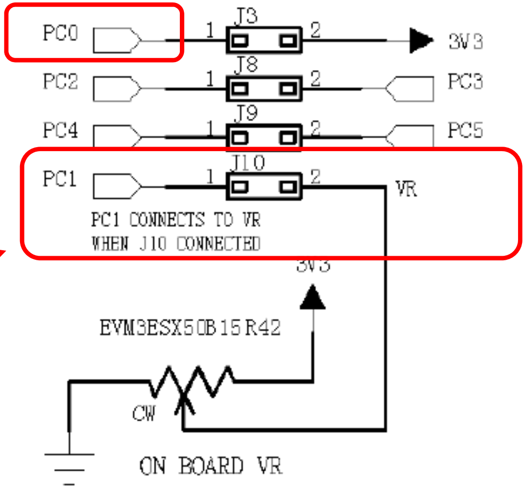
- The corresponding schematic is also shown here, note that it is not pin to pin correspondence to the board.

ON BOARD VR



As you see there is a jumper at J10 connected to PC1 and VR

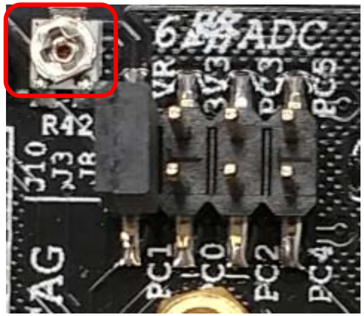
Note: PC0 has a 4.7kΩ to 3V3



ADC in MINI-V3

- The on board VR is not good to use and easy to be damaged. As a result, we will use an outside circuit to connect the analogue input to one of the channels

ON BOARD VR



- Actually there are many ADC channels (not limited to the one shown above) in STM32 that you can use.

ADC in STM32

- Refer to STM32 Datasheet,
  - ADC1 and ADC2 have 16 channels (from IN0 to IN15) to use
  - ADC3 have 13 channels (from IN0 to IN8, IN10 to IN13)

	IN0	IN1	IN2	IN3	IN4	IN5	IN6	IN7	IN8	IN9	IN10	IN11	IN12	IN13	IN14	IN15
ADC1																
ADC2																
ADC3										N/A					N/A	N/A

- In Datasheet
  - ADC123_IN10 → means Channel 10 shared by ADC1 ADC2 and ADC3
  - ADC12_IN5 → means Channel 5 shared by ADC1 and ADC2 only
  - ADC3_IN8 → means Channel 8 for ADC3 only
- Question : If one shared channel means one I/O pin, if we use all the channels available, how many I/O pins will be used by ADC ?

Ans: 21

ADC in STM32

20654091

- In this LAB, we will use ADC1 only, you need to use the channel according to the last digit of your student ID

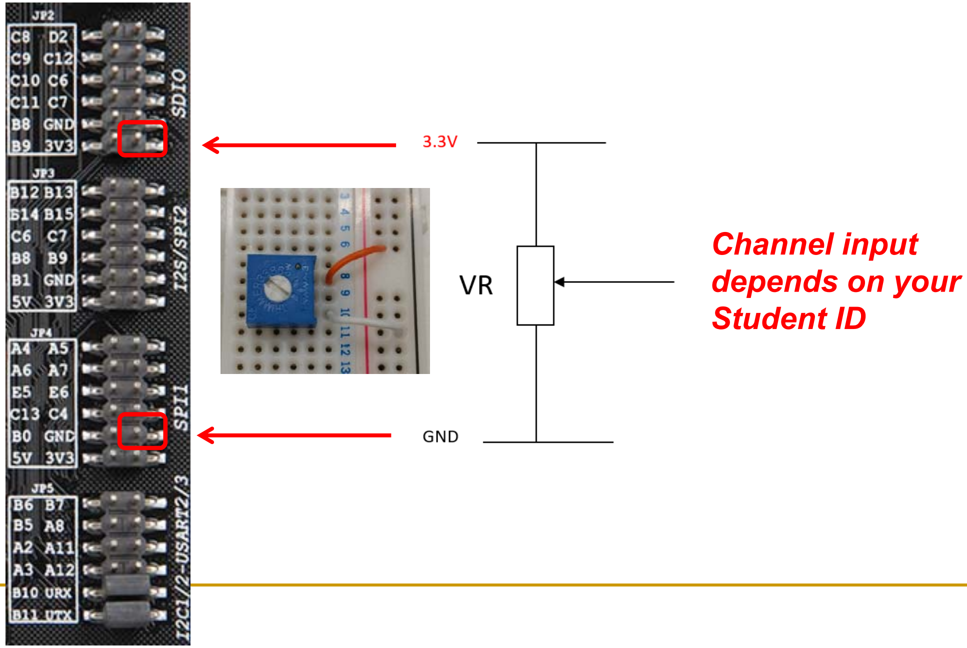
Last digit of your student ID	0	1	2	3	4	5	6	7	8	9
Channel number	12	13	2	3	4	5	6	7	14	15
Corresponding I/O pin in STM32 ?										

- Check the corresponding I/O pin from datasheet then check the location of that pin on MINI V3 board

1 = ADC123_IN13 => PC3

Using VR as input

- Using your breadboard, assemble the following circuit, this would be used for your Task 1 and Task 2.

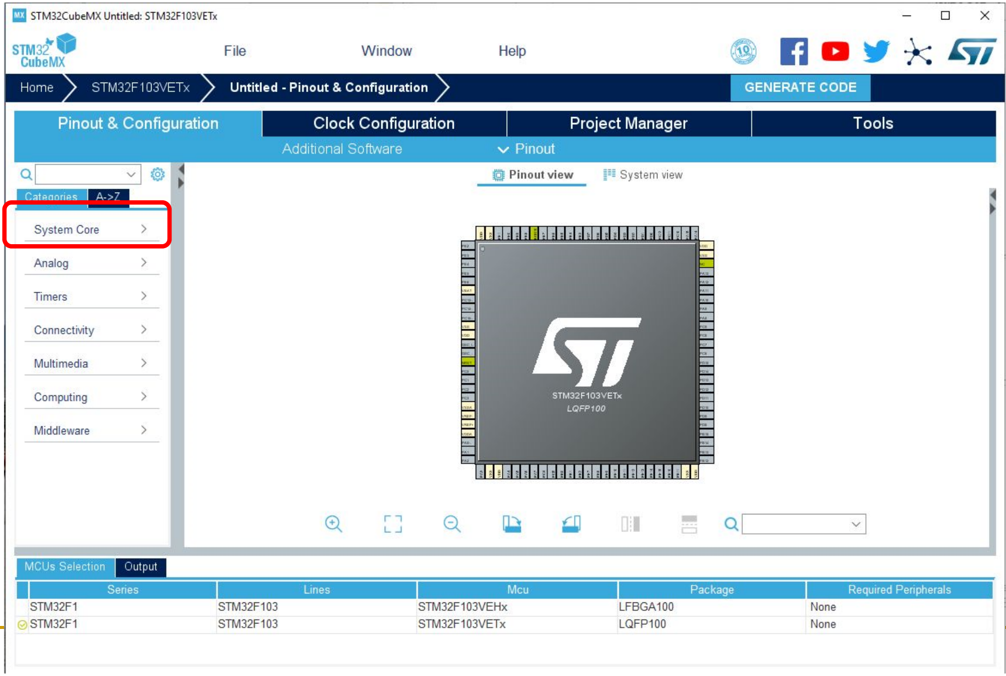


Configuration of LCD

- In this LAB, we need to use the LCD to display the value.
- Please refer to the Tutorial for CubeMX and Tutorial for LAB3 to create a project that allows you to use the LCD Display.
- Or you may start your LAB5 by using the LAB3 as a starting point.

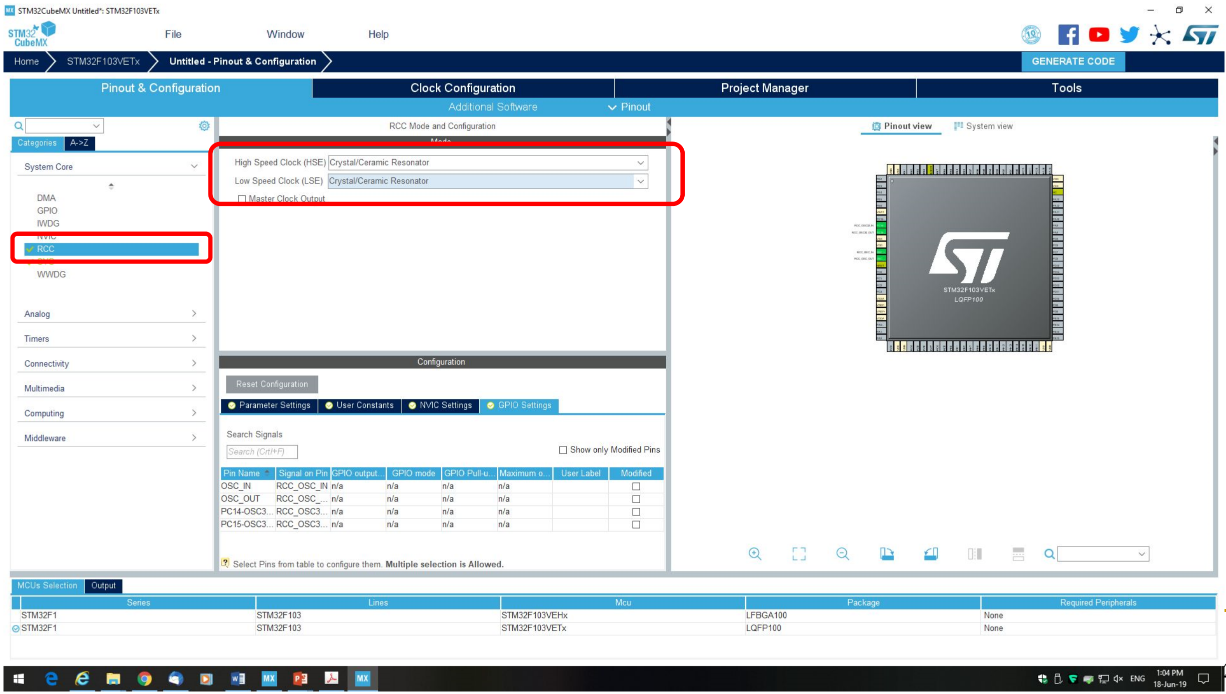
Set Clock

- You will go to this screen, first we need to set the clock, Expand System Core



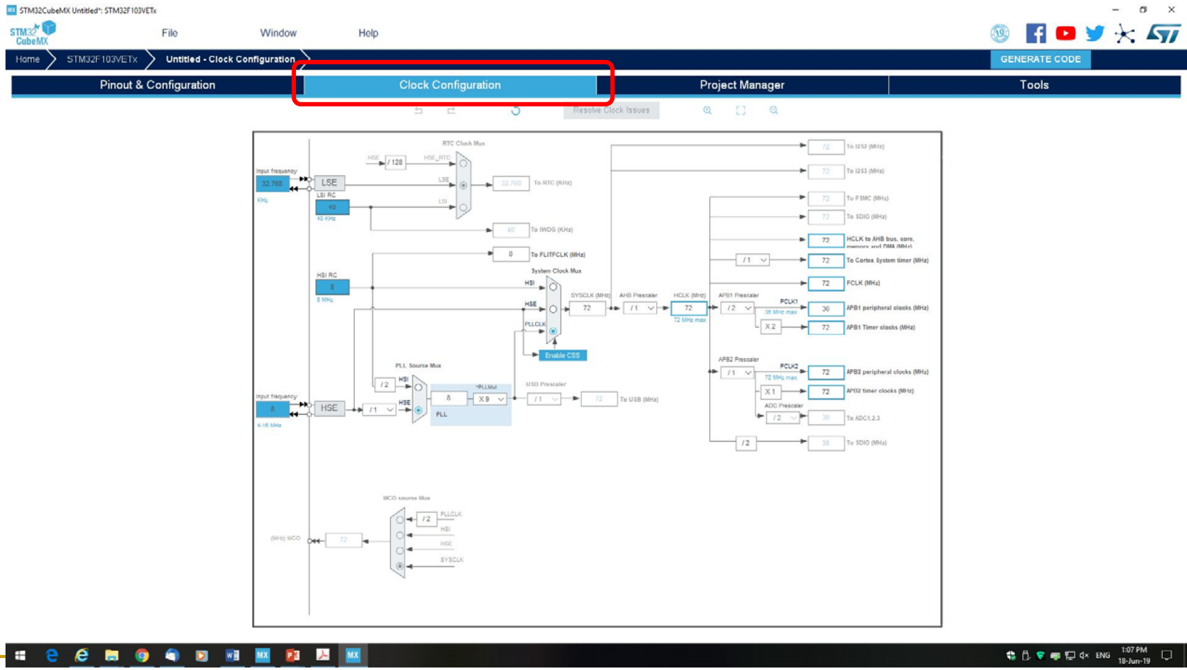
Change Clock to Crystal

- Click RCC, enable the High Speed Clock and Low Speed Clock to
  - Crystal/Creamic Resonator

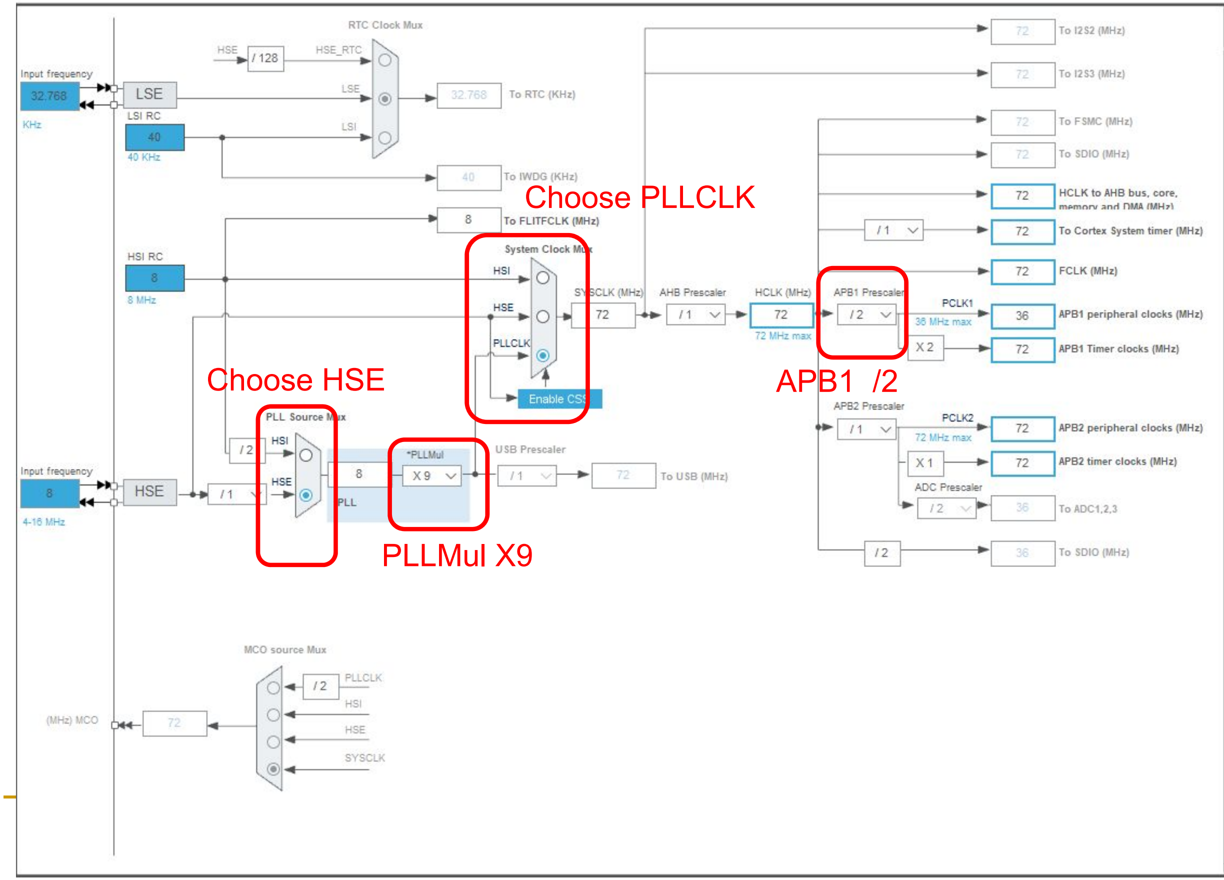


Clock Configuration

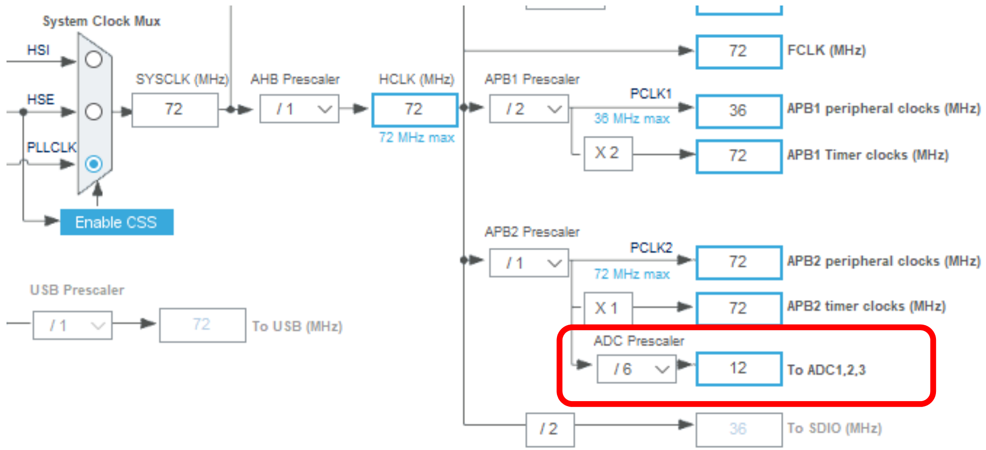
- Go to Clock Configuration



Clock Configuration



Setting the clock of ADC in CubeMX

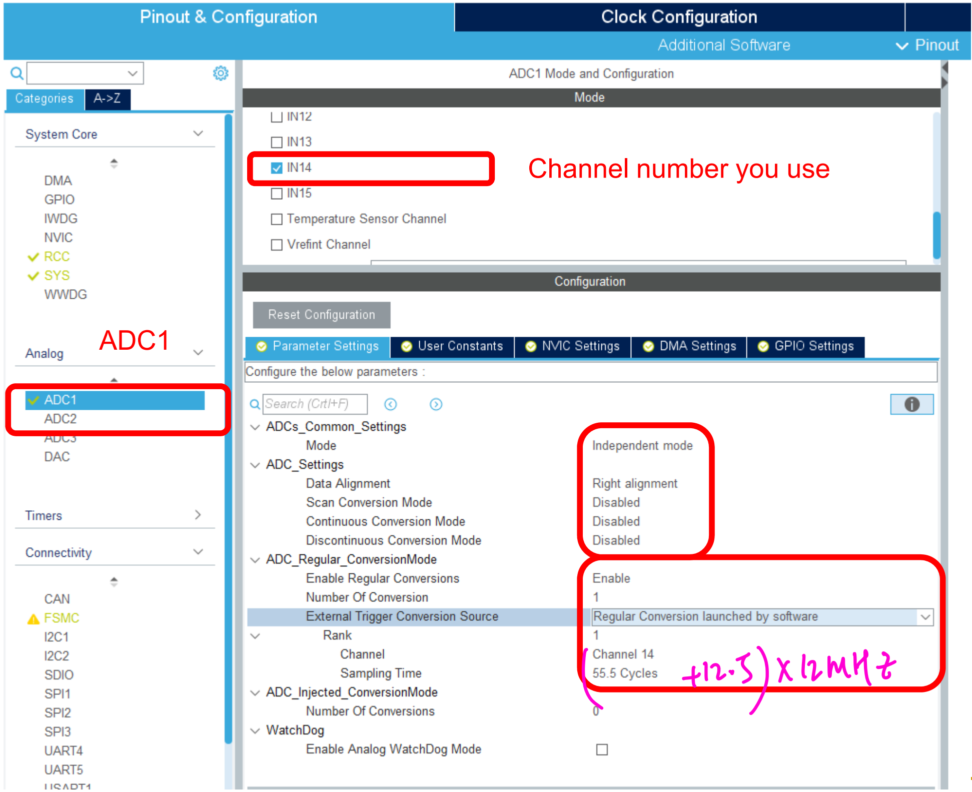


- Is it the fastest setting of ADCCLK?
- How can you change it to fastest ?

No

56MHz

Configuration of ADC

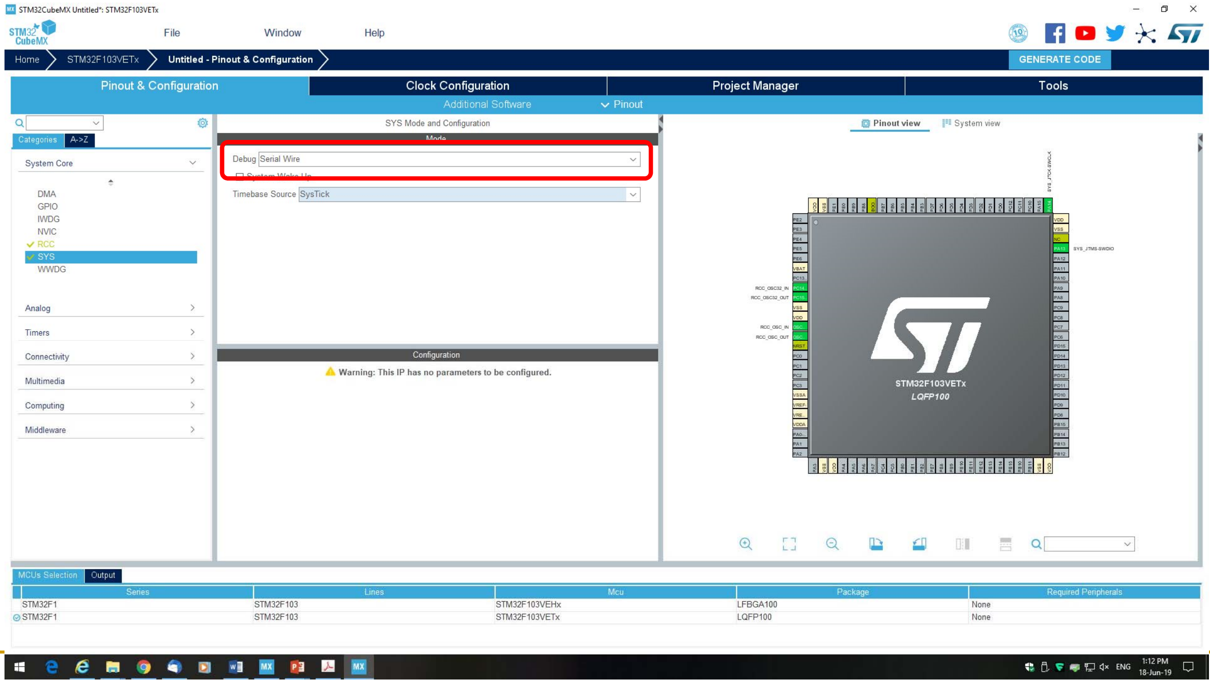


Mode and Alignment Selection

Conversion Selection

Communicate with Debugger

- Go to Pinout & Configuration, in SYS, Choose Serial Wire for Debug



About Calibration

- The ADC has an built-in self calibration mode. Calibration significantly reduces accuracy errors due to internal capacitor bank variations.
- During calibration, an error-correction code (digital word) is calculated for each capacitor, and during all subsequent conversions, the error contribution of each capacitor is removed using this code.
- Once calibration is over, the CAL bit is reset by hardware and normal conversion can be performed. It is recommended to calibrate the ADC once at power-on. The calibration codes are stored in the ADC_DR as soon as the calibration phase ends.

About Calibration

- The datasheet suggested to perform a self calibration at the startup, so, you may want to add the following line before the while(1) loop of your code.

```
HAL_ADCEx_Calibration_Start(&hadc1);
```

Depends on which ADC you use

How to start a conversion and get the result ?

- The ADC conversion can be initiated by

```
HAL_ADC_Start(&hadc1);
```

- Please note that there is a time needed for the conversion, you can use the following line to poll the end of conversion.

```
HAL_ADC_PollForConversion(&hadc1, 1000);
```

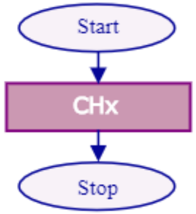
Timeout in milliseconds

- You can read the conversion result by calling the procedure

```
uint32_t HAL_ADC_GetValue(&hadc1);
```

Task 1 – Single Conversion

- You need to write a program to display an ADC conversion result of the external VR in both decimal and hex. The converted value will be updated when K2 is pressed.
- You are required to use Single Conversion Mode of ADC to finish Task 1
- You need to show to your TA your main.c for verifying the mode used



Task 1 – Display the Result

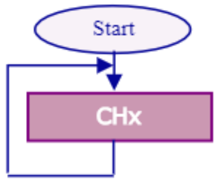
- You need to think about how to decompose the value to display. The following functions and table might help you. Details refer back to lcd.c in the lcd.zip

```
void LCD_DrawChar(uint16_t usC, uint16_t usP, uint8_t cChar);
void LCD_DrawString(uint16_t usC, uint16_t usP, uint8_t * pStr);
```

- You can also use stdlib function like sprintf() to help you.

Task 2 – Continuous Conversion

- Change your program such that the LCD will be able to update the ADC result for the external VR at a certain period without pressing K2.
- You are required to use Continuous Conversion Mode of ADC to finish Task 2.
- You need to show to your TA your main.c for verifying the mode used



Task 1 and 2 – Note

The result you displayed should be

1. From _______ to _______ for DEC
2. From _______ to _______ for HEX
3. 1 and 2 should be SAME value but

DIFFERENT REPRESENTATION ONLY

DEC	HEX	Correct ?
0	000	
10	0a0	
0004	000	
1234	4d2	
6007	258	
7129	2c8	

If there exists (∃) a result violating any of above 3 conditions, it means your program have BUG.

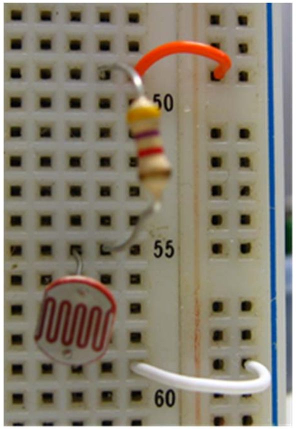
Please think out why ….and FIX it.

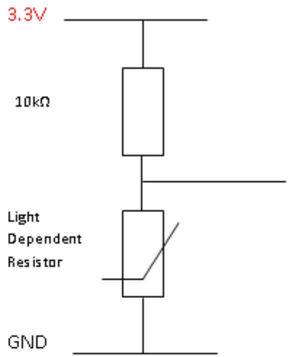
Connecting to LDR

- Light Dependent Resistor is a resistor that the resistance will depends on the light intensity. In this part, you need to use the LDR as an analogue input to the system.

Using LDR as input

- Replace your VR with a fixed resistor and a LDR





Channel input depends on your Student ID

Task 3: Using LDR as input

- Following the steps listed on the LAB Sheet, fill the corresponding data.
- You might need to try to swap the LDR and the resistor.

Task 4: Light Intensity System

- Using your knowledge from Task 3 and LAB2, together with the RGB LED on MINI V3 Development Board, implement a five-level Light Intensity System such that..

	Light Intensity				
	Very Dark	Dark	Medium	Bright	Very Bright
RGB LED COLOR	WHITE	RED	GREEN	BLUE	OFF

- You can use either Figure 1 or Figure 2 on Page 2 of the LAB sheet to implement the system, as long as it follows above requirement.
- You are free to choose the boundary for the system, but you need to clearly show the TA the five different levels according to the above requirement during the demo.

END