Laboratory 2: Simple digital I/O with GPIO and serial communication with USART

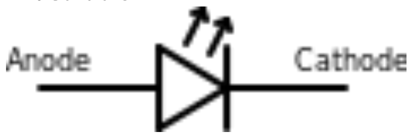
Objectives

1. Understand the concept of Serial Communication
2. Understand the concept of data format
3. Understand General-purpose I/Os (GPIO) and its implementation as peripherals on STM32
4. Understand Universal synchronous asynchronous receiver transmitter (USART) and its implementation as peripherals on STM32
5. Understand the concept of Light Emitted Diode

LED

LED is short for ***Light Emitting Diode***. It is a special kind of diodes (unipolar electronic devices) that will emit photon (light) where there is an electronic current flow in a correct direction (from anode to cathode). Illustration and Illustration show the symbol and the picture of an LED.

Illustration



2: Electronic Svmbol of LED

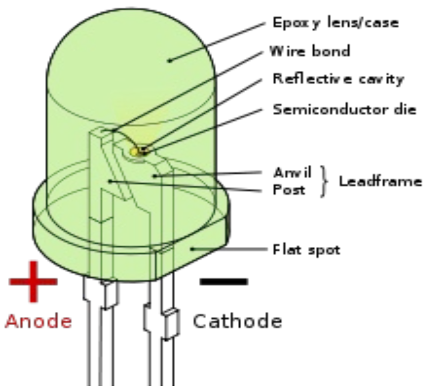


Illustration 1: A picture of LED (a picture from wikipedia.org)

Due to its characteristics, LED is an ideal choice for using as an output of a digital pin. Using LED as a digital output, we can easily observe the logic from the light. There are several ways to connect the LED to an output PIN. Here are two simple methods presented (see Illustration). Depending on the electronic properties of the device, the quality may vary. For most cases, LED1 should provide better quality (brighter).

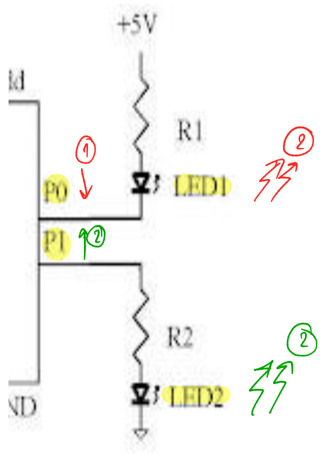


Illustration 3: Two methods for การรบกวน *interfacing LED*

In particular, the LED1 (connected to P0) will emit the light when the output of P0 is LOW. The LED 2 will emit the light when the output of P1 is HIGH. To determine the correct logic, a programmer must understand the schematic of the circuits.

Push-button switch

Push-button switch is, perhaps, the most commonly used input for logic circuits. The idea is to toggle the voltages at the input pin. In Illustration, a method for connecting the switch to a pin is showed. With the pull-up resistor, the micro controller will see the HIGH logic when reading from the input pin. Once a button is pressed, the current will sink to the ground giving the LOW logic to the input pin.

The ease understanding, thinking of the voltage as a level of water. When there is no sink (hole), the water level would be high. However, when we open the sink (hole at the bottom), the water level would go down (low). The same idea can also be used to explain why a pull-up resistor is required. Without the pull-up resistor, it is difficult (if not impossible) to determine the correct level. The pull-up resistor make sure that we will always fill the water to the high level (if there is no sink to drain the water).

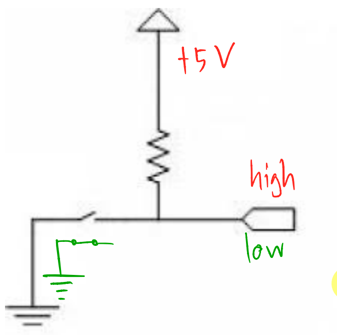


Illustration 4: Symbol of push-button switch with pull-up resistor

For some microcontroller (including our Avr/Arduino), the pull-up resistor is provided internally in the chip. This allows us to avoid connecting more resistor to the circuits.

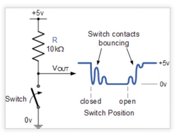
Debounce

Given that a switch is a mechanical device, a mechanical contact between the two surfaces would cause a bounce. (The same idea when dropping a ball to the floor. The ball may bounce few times before stopping at the floor.) The push-button switch also cause a bounce when it is being pressed. From an oscilloscope, the bounce is observed as Illustration. If a program does not carefully avoid the bounce, we may easily observe multiple clicks from only one physical click.

A way to avoid reading multiple clicks from a single click is called ***debounce***. Depending on the size and the mechanical properties of the switch, the bounce may last several milliseconds. There are several ways to avoid bounce. Some methods require extra hardware. However, the naïve software solution is to delay few milliseconds before reading the next input.

Debounce คืออะไร???

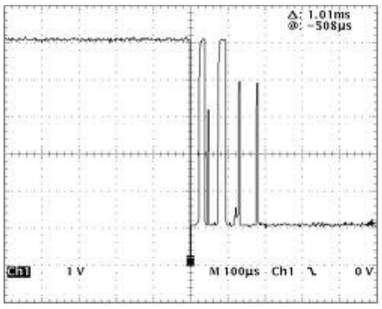
ก่อนจะทำความรู้จัก Debounce นั้นต้องทราบก่อนว่า Bounce คืออะไร
Bounce คือ สัญญาณที่เกิดขึ้นจากการที่การกดปุ่มแต่ละครั้ง เมื่อกดปุ่มหน้าสัมผัสที่ทำการสัมผัสกันนั้นยังไม่แนบสนิทดี จะทำให้เกิดสัญญาณวิ่งขึ้นวิ่งลง หรือ ก็คือ สัญญาณไม่นิ่ง ซึ่งจะทำให้ค่าที่ส่งออกไปนั้นคงาดเคลื่อน จึงต้องมีการทำ Debounce ขึ้นเพื่อป้องกันเหตุการณ์ที่จะเกิดขึ้นนี้



รูปภาพการเกิด Bouncing

จากรูปจะเห็นได้ว่าช่วง Switch Contracts จะเกิดการ Bouncing ที่วงจรเกิดการ เปิด หรือ ปิด ซึ่ง หลักาการทำงานของ Debounce คือ จะทำการนับลูกคลื่นของสัญญาณนั้นไปเรื่อยๆ จนสัญญาณนั้นเกิดการนิ่งแล้วซึ่งก็คือตรงกลางของรูปตัวอย่าง เพื่อค่าที่ส่งนั้นจะเกิดความถูกต้องไม่คลาดเคลื่อน

Illustration



5: bouncina switch

Serial Communication / U(S)ART

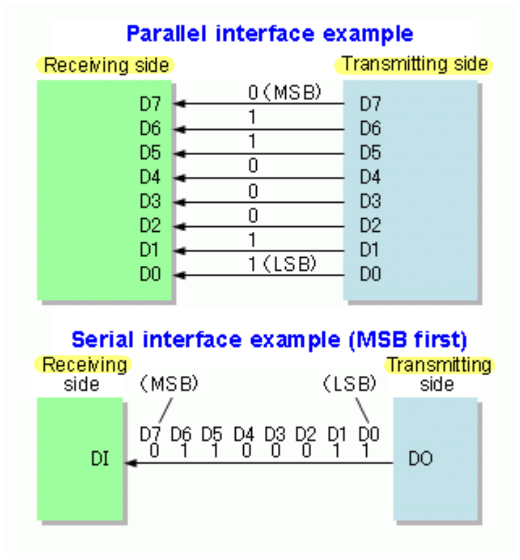


Illustration 6 Parallel versus serial communication.

Serial communication is a method for using a sequence of bits to transfer a byte of data. This is in contrast to parallel communication, where several bits are sent as a whole, on a link with several parallel channels as Illustration.

For universal asynchronous receiver/transmitter (UART) – Universal Synchronous/Asynchronous Receiver/Transmitter (USART) if also supports synchronous operation, we use 8 bit to represent a byte of data. A byte is generally equal to a character. Thus, there are (theoretically) 256 possible characters in a byte. One standard that provides association between a value and a physical character is ASCII (see next section for more details).

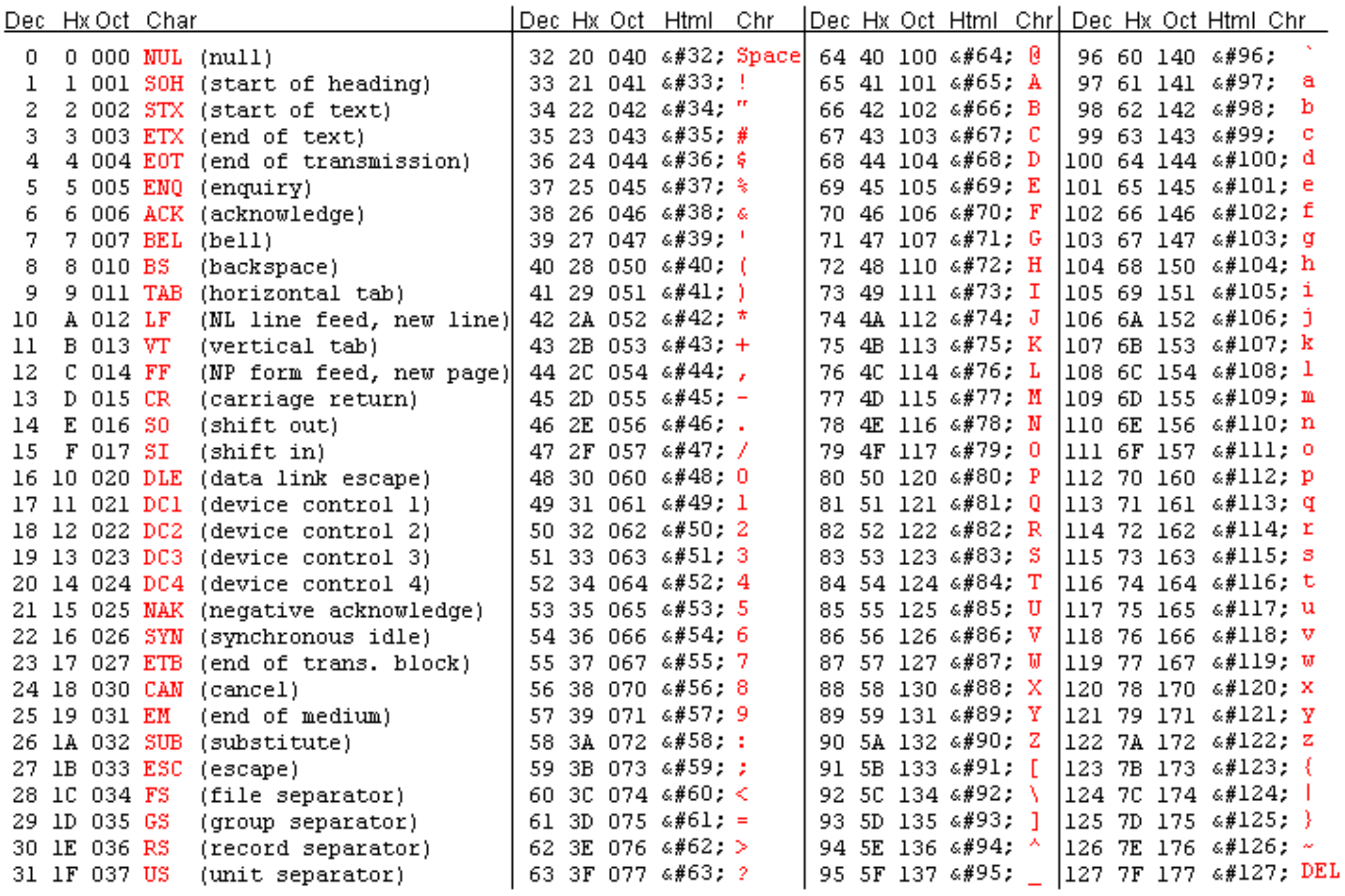
With a serial communication, a communication between two devices can be done with just two wires (one fore sending and one for receiving). This kind of communication is cheaper comparing to parallel communication where 8 wires are need to transmit a byte of data. Thus, to a byte of data from one device to another, we have to send (at least) 8 times (one for each bit). Given that no clock or control signal is involved in the communication, the key critical to the success of such communication is timing. Both devices has to be configured for the same speed of clock (specified as bit per second or bps) and the same protocol in order to observe the right data at the right time. This is the basic of network communication that we used today.

Bit number	1	2	3	4	5	6	7	8	9	10	11	12
	Start bit	**5-9 data bits**									**Stop bit(s)**	
	Start	Data 0	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	Data 8	Stop	

https://en.wikipedia.org/wiki/Universal_asynchronous_receiver/transmitter

ASCII

ASCII is short for the American Standard Code for Information Interchange. It is a character-encoding scheme based on English alphabet, numbers and punctuations. It is nowadays used by several digital devices as a way to representing code for characters. Thus, when type a character 'A', the digital computer system simply record an ASCII code to the memory.



Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040	&#32;	Space	64	40	100	&#64;	@	96	60	140	&#96;	`
1	1	001	SOH (start of heading)	33	21	041	&#33;	!	65	41	101	&#65;	A	97	61	141	&#97;	a
2	2	002	STX (start of text)	34	22	042	&#34;	"	66	42	102	&#66;	B	98	62	142	&#98;	b
3	3	003	ETX (end of text)	35	23	043	&#35;	#	67	43	103	&#67;	C	99	63	143	&#99;	c
4	4	004	EOT (end of transmission)	36	24	044	&#36;	$	68	44	104	&#68;	D	100	64	144	&#100;	d
5	5	005	ENQ (enquiry)	37	25	045	&#37;	%	69	45	105	&#69;	E	101	65	145	&#101;	e
6	6	006	ACK (acknowledge)	38	26	046	&#38;	&	70	46	106	&#70;	F	102	66	146	&#102;	f
7	7	007	BEL (bell)	39	27	047	&#39;	'	71	47	107	&#71;	G	103	67	147	&#103;	g
8	8	010	BS (backspace)	40	28	050	&#40;	(	72	48	110	&#72;	H	104	68	150	&#104;	h
9	9	011	TAB (horizontal tab)	41	29	051	&#41;	)	73	49	111	&#73;	I	105	69	151	&#105;	i
10	A	012	LF (NL line feed, new line)	42	2A	052	&#42;	*	74	4A	112	&#74;	J	106	6A	152	&#106;	j
11	B	013	VT (vertical tab)	43	2B	053	&#43;	+	75	4B	113	&#75;	K	107	6B	153	&#107;	k
12	C	014	FF (NP form feed, new page)	44	2C	054	&#44;	,	76	4C	114	&#76;	L	108	6C	154	&#108;	l
13	D	015	CR (carriage return)	45	2D	055	&#45;	-	77	4D	115	&#77;	M	109	6D	155	&#109;	m
14	E	016	SO (shift out)	46	2E	056	&#46;	.	78	4E	116	&#78;	N	110	6E	156	&#110;	n
15	F	017	SI (shift in)	47	2F	057	&#47;	/	79	4F	117	&#79;	O	111	6F	157	&#111;	o
16	10	020	DLE (data link escape)	48	30	060	&#48;	0	80	50	120	&#80;	P	112	70	160	&#112;	p
17	11	021	DC1 (device control 1)	49	31	061	&#49;	1	81	51	121	&#81;	Q	113	71	161	&#113;	q
18	12	022	DC2 (device control 2)	50	32	062	&#50;	2	82	52	122	&#82;	R	114	72	162	&#114;	r
19	13	023	DC3 (device control 3)	51	33	063	&#51;	3	83	53	123	&#83;	S	115	73	163	&#115;	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	&#84;	T	116	74	164	&#116;	t
21	15	025	NAK (negative acknowledge)	53	35	065	&#53;	5	85	55	125	&#85;	U	117	75	165	&#117;	u
22	16	026	SYN (synchronous idle)	54	36	066	&#54;	6	86	56	126	&#86;	V	118	76	166	&#118;	v
23	17	027	ETB (end of trans. block)	55	37	067	&#55;	7	87	57	127	&#87;	W	119	77	167	&#119;	w
24	18	030	CAN (cancel)	56	38	070	&#56;	8	88	58	130	&#88;	X	120	78	170	&#120;	x
25	19	031	EM (end of medium)	57	39	071	&#57;	9	89	59	131	&#89;	Y	121	79	171	&#121;	y
26	1A	032	SUB (substitute)	58	3A	072	&#58;	:	90	5A	132	&#90;	Z	122	7A	172	&#122;	z
27	1B	033	ESC (escape)	59	3B	073	&#59;	;	91	5B	133	&#91;	[	123	7B	173	&#123;	{
28	1C	034	FS (file separator)	60	3C	074	&#60;	<	92	5C	134	&#92;	\	124	7C	174	&#124;	|
29	1D	035	GS (group separator)	61	3D	075	&#61;	=	93	5D	135	&#93;	]	125	7D	175	&#125;	}
30	1E	036	RS (record separator)	62	3E	076	&#62;	>	94	5E	136	&#94;	^	126	7E	176	&#126;	~
31	1F	037	US (unit separator)	63	3F	077	&#63;	?	95	5F	137	&#95;	_	127	7F	177	&#127;	DEL

http://www.infocellar.com/binary/ascii-ebcdic.htm

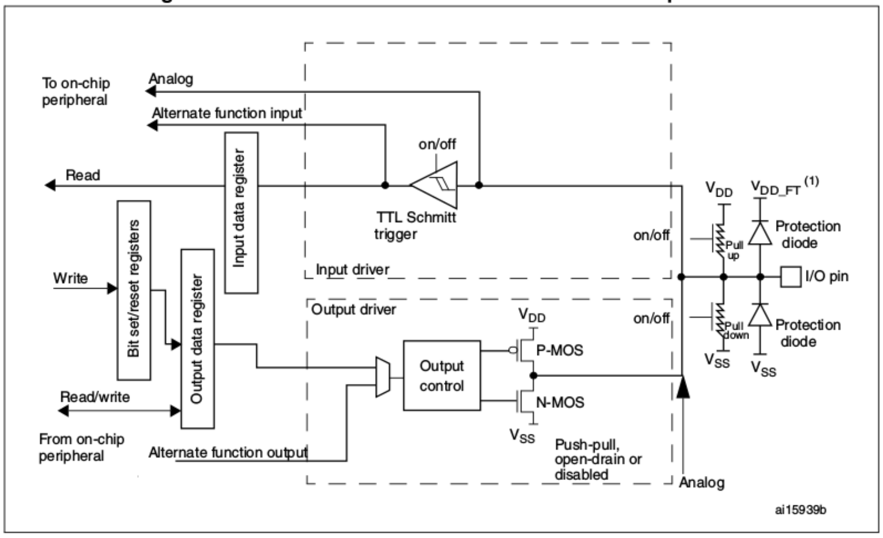
STM32's GPIO functional description

Subject to the specific hardware characteristics of each I/O port listed in the datasheet, each port bit of the general-purpose I/O (GPIO) ports can be individually configured by software in several modes:

- Input floating
- Input pull-up
- Input-pull-down
- Analog
- Output open-drain with pull-up or pull-down capability
- Output push-pull with pull-up or pull-down capability
- Alternate function push-pull with pull-up or pull-down capability
- Alternate function open-drain with pull-up or pull-down capability

Each I/O port bit is freely programmable, however the I/O port registers have to be accessed as 32-bit words, half-words or bytes. The purpose of the GPIOx_BSRR register is to allow atomic read/modify accesses to any of the GPIO registers. In this way, there is no risk of an IRQ occurring between the read and the modify access.

Figure 25. Basic structure of a five-volt tolerant I/O port bit



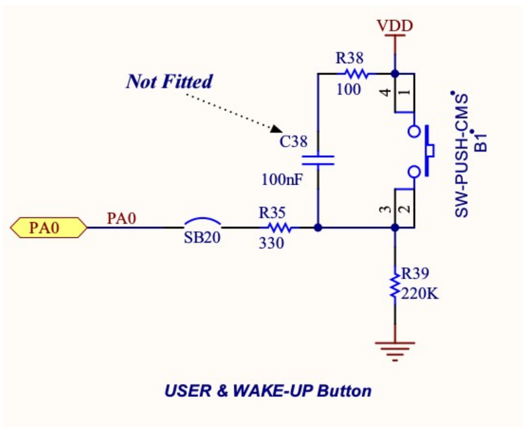
1. V_{DD_FT} is a potential specific to five-volt tolerant I/Os and different from V_{DD}.

Source : **Reference Manuals – RM0090**: STM32F405/415, STM32F407/417, STM32F427/437 and STM32F429/439 advanced ARM®-based 32-bit MCUs

NUCLEO-F411 Basic Input/Output Interface

LEDs

- User LD2: Green LED is a user LED connected to the I/O ***PA5*** of the STM32F411



Pushbuttons

- B1 USER: User and Wake-Up buttons are connected to the I/O PC13 of the STM32F411

Source : **User Manuals – UM1742:** STM32 Nucleo-64 boards User Manual

NUCLEO-F411 UART Communication

✳ The USART2 interface available on PA2 and PA3 of the STM32 microcontroller. You do not need an external USB-Serial adapter in order to communicate between a computer and STM32

Source: **User Manuals – UM1742:** STM32 Nucleo-64 boards User Manual

Lab Exercises

1 Create a new STM32 Project to blink (on/off) with a period of 0.2 sec. for an on board LED เหมือน LAB 1

2 Create a new STM32 Project to blink (on/off) with a period of 0.2 sec. for an external LED

ต่อ PA5 (function Read) กด→ไฟติด กด→ไฟดับ

3 Create a new STM32 Project to toggle an LED (on/off) with pushing USER push-button. (Debouching is required)

Serial Port

4 Create a new STM32 Project to echo back (transmit the receive data) the communication data from UART peripheral (USART2) interface using blocking mode APIs on STM32CUBE library (Look at stm32f4xx_hal_uart.c).

For those of you who are using STM32F4Discovery, you will have to connect to a USB-Serial such as ET-MINI USB-TTL as mention above.

5 Create a new STM32 Project to toggle an LED status (on/off) with commands via serial console. (Type "on" or "off" then press Enter to on or off the LED

Optional Lab:

1. Pair up with your friend, and make one board as a master and another as a slave. The master send a command through UART to the slave board to toggle the LED when a USER push-button.