TECNOLÓGICO DE MONTERREY®

Subject 05
PARALLEL PORTS (GPIO)

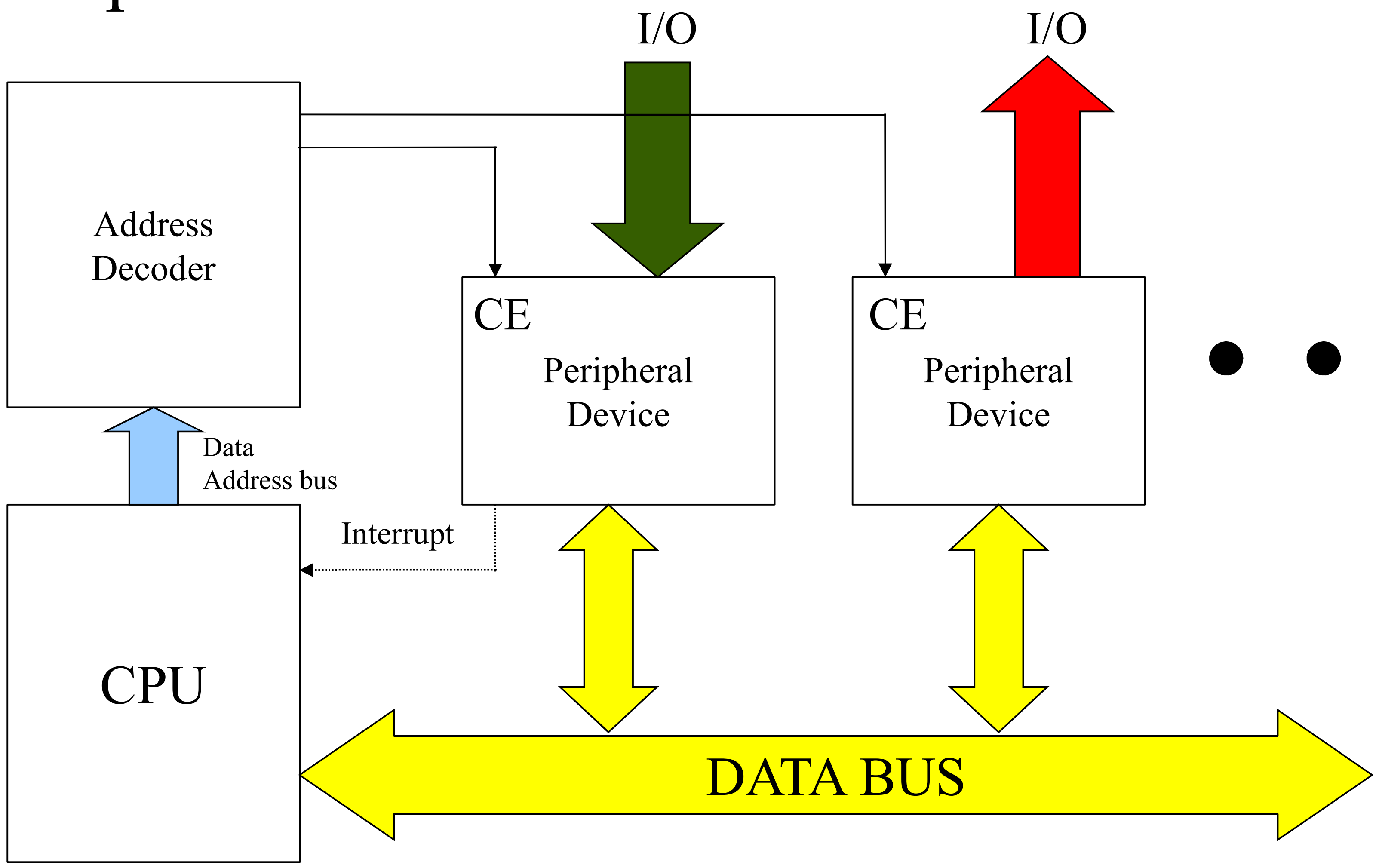
Parallel Ports (GPIO)

- Parallel port also know as general purpose IO or GPIO
- Enable the CPU to interact with the outside world using logical level bit per bit our as a group
- They can be used as inputs and/or outputs
- As outputs we can turn on and off things
- As inputs we can sense the state of a logic value
- In the PIC18 the IO or GPIO functionality can be multiplexed with other alternate function
- The GPIO is another peripheral of the MCU

Peripherals

- Embedded devices are designed to interact with the environment
- The CPU cannot do this directly given the electrical and logic differences on their interfaces
- The interaction is made trough what is called a "peripheral devices "
- The peripheral devices pass the information from the world to the CPU as if they where a memory or a register.
- So they are like a virtual window to the world that is presented to the CPU as any other memory location

Peripherals



CPU synchronization

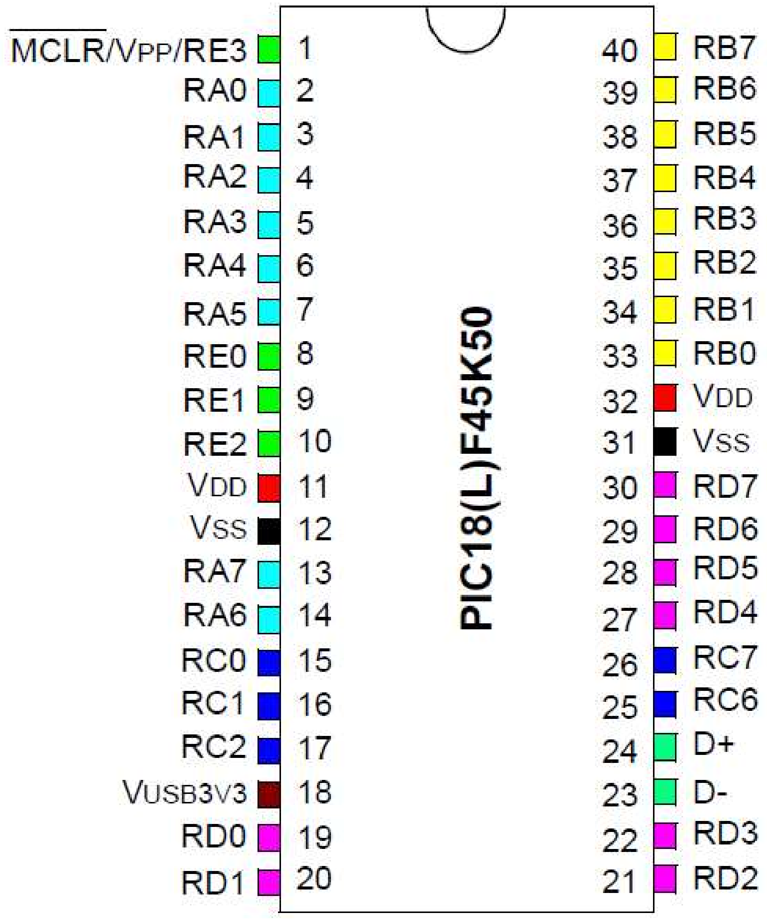
- To read the data that represents an input, the CPU must be sure that the peripheral has a new valid value. There are two methods to do this.
  - "Polling": The CPU check the state of a bit "of flag" or a group of them that are mapped in a memory location, this is done voluntarily the use of an specific instruction
  - "Interrupt", The program counter (PC) "jumps" to an special address associated that that particular peripheral. In that address the user programs code that process the event. This is a very efficient way used in time critical situations.

CPU synchronization

- For the output peripherals, the same methods are employed. In this case for some peripherals this is used to check if the peripheral can accept new data to transfer to world

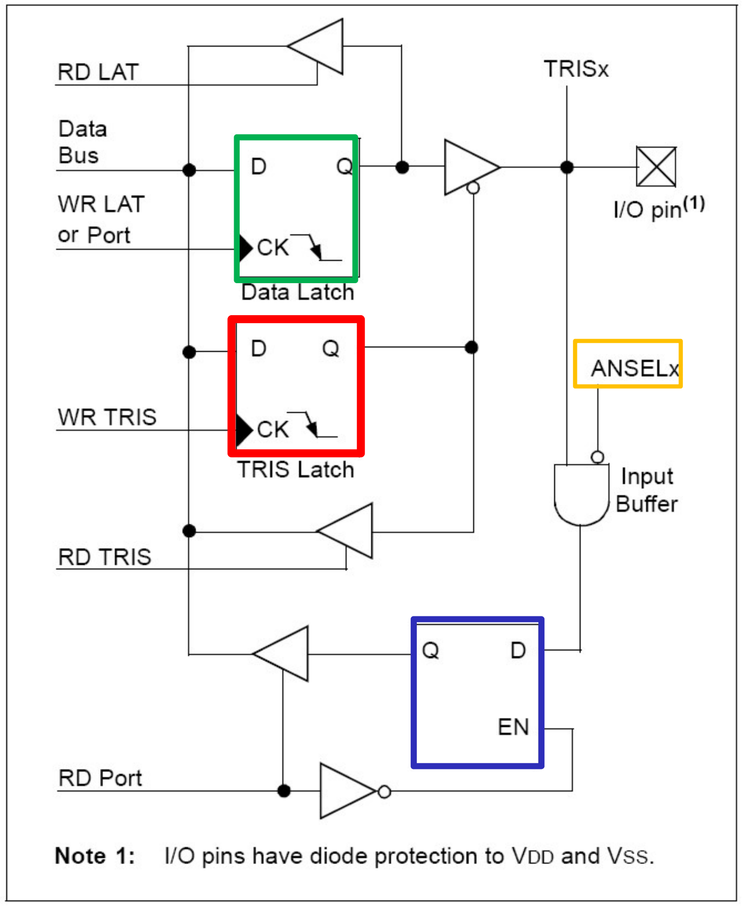
- The PIC18 can use both methods

Parallel ports on the PIC18F45K50



- A 8bits RA0:7
- B 8bits RB0:7
- C 5bits RC0:2,7,6
- D 8bits RD7:RD0
- E 4bits RE3:RE0

Ports



- Ports have four associated registers
  - TRISx defines if is input or output
  - PORTx reads the logic level at the IO pin
  - LATx is a “latch” fix the value to the output
  - ANSELx selects if analog or digital
  - x = A, B,C,D,E

(1) Does not consider other peripherals

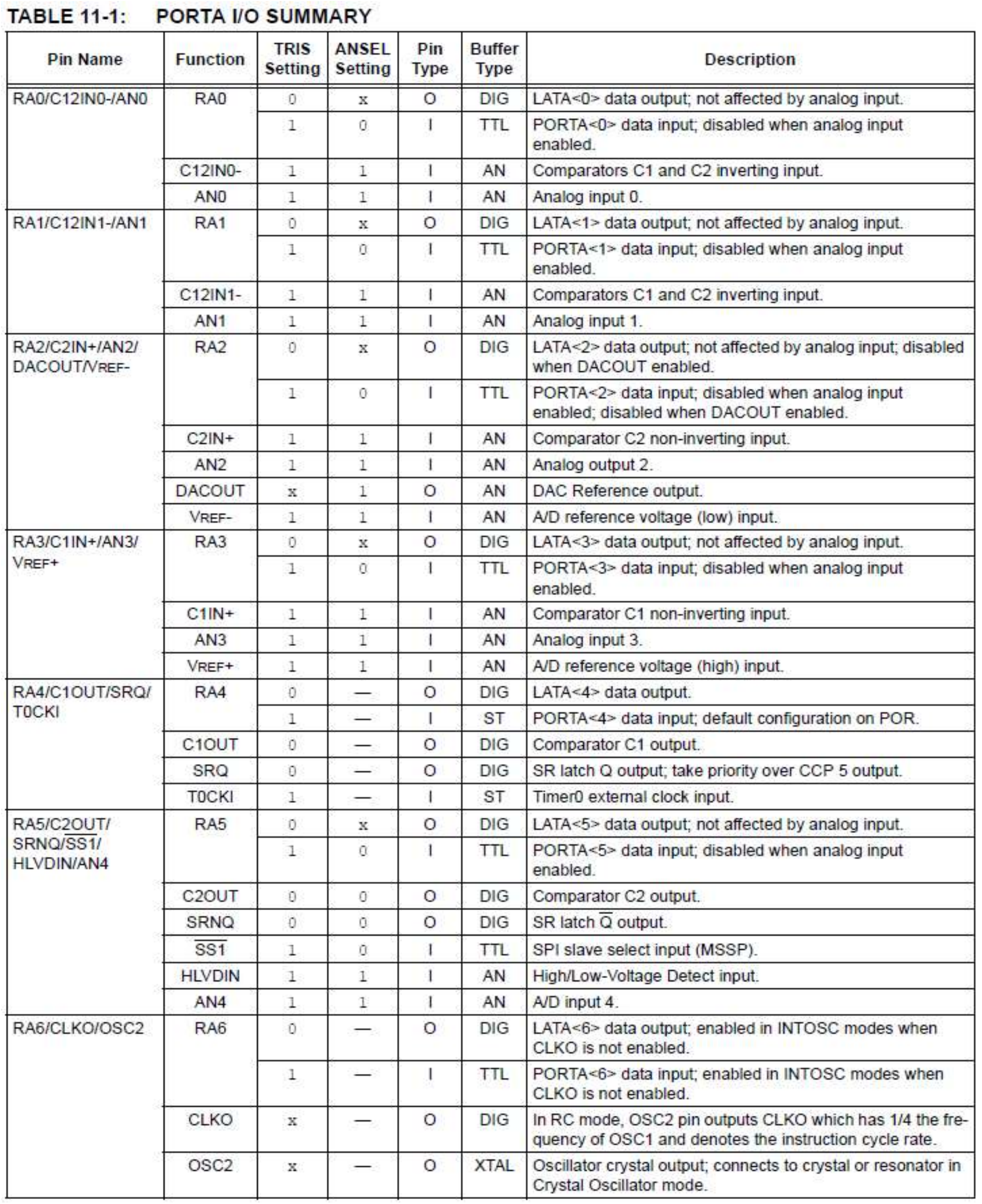


TABLE 11-1: PORTA I/O SUMMARY

Pin Name	Function	TRIS Setting	ANSEL Setting	Pin Type	Buffer Type	Description
RA0/C12IN0-/AN0	RA0	0	x	O	DIG	LATA<0> data output; not affected by analog input.
		1	0	I	TTL	PORTA<0> data input; disabled when analog input enabled.
	C12IN0-	1	1	I	AN	Comparators C1 and C2 inverting input.
	AN0	1	1	I	AN	Analog input 0.
RA1/C12IN1-/AN1	RA1	0	x	O	DIG	LATA<1> data output; not affected by analog input.
		1	0	I	TTL	PORTA<1> data input; disabled when analog input enabled.
	C12IN1-	1	1	I	AN	Comparators C1 and C2 inverting input.
	AN1	1	1	I	AN	Analog input 1.
RA2/C2IN+/AN2/ DACOUT/VREF-	RA2	0	x	O	DIG	LATA<2> data output; not affected by analog input; disabled when DACOUT enabled.
		1	0	I	TTL	PORTA<2> data input; disabled when analog input enabled; disabled when DACOUT enabled.
	C2IN+	1	1	I	AN	Comparator C2 non-inverting input.
	AN2	1	1	I	AN	Analog output 2.
	DACOUT	x	1	O	AN	DAC Reference output.
	VREF-	1	1	I	AN	A/D reference voltage (low) input.
RA3/C1IN+/AN3/ VREF+	RA3	0	x	O	DIG	LATA<3> data output; not affected by analog input.
		1	0	I	TTL	PORTA<3> data input; disabled when analog input enabled.
	C1IN+	1	1	I	AN	Comparator C1 non-inverting input.
	AN3	1	1	I	AN	Analog input 3.
	VREF+	1	1	I	AN	A/D reference voltage (high) input.
RA4/C1OUT/SRQ/ T0CKI	RA4	0	—	O	DIG	LATA<4> data output.
		1	—	I	ST	PORTA<4> data input; default configuration on POR.
	C1OUT	0	—	O	DIG	Comparator C1 output.
	SRQ	0	—	O	DIG	SR latch Q output; take priority over CCP 5 output.
	T0CKI	1	—	I	ST	Timer0 external clock input.
RA5/C2OUT/ SRNQ/$\overline{SS1}$/ HLVDIN/AN4	RA5	0	x	O	DIG	LATA<5> data output; not affected by analog input.
		1	0	I	TTL	PORTA<5> data input; disabled when analog input enabled.
	C2OUT	0	0	O	DIG	Comparator C2 output.
	SRNQ	0	0	O	DIG	SR latch $\overline{Q}$ output.
	$\overline{SS1}$	1	0	I	TTL	SPI slave select input (MSSP).
	HLVDIN	1	1	I	AN	High/Low-Voltage Detect input.
	AN4	1	1	I	AN	A/D input 4.
RA6/CLKO/OSC2	RA6	0	—	O	DIG	LATA<6> data output; enabled in INTOSC modes when CLKO is not enabled.
		1	—	I	TTL	PORTA<6> data input; enabled in INTOSC modes when CLKO is not enabled.
	CLKO	x	—	O	DIG	In RC mode, OSC2 pin outputs CLKO which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate.
	OSC2	x	—	O	XTAL	Oscillator crystal output; connects to crystal or resonator in Crystal Oscillator mode.

TABLE 10-1: PORTA I/O SUMMARY

Pin Name	Function	TRIS Setting	ANSEL Setting	Pin Type	Buffer Type	Description
RA0/C12IN0-/AN0	RA0	0	0	O	DIG	LATA<0> data output; not affected by analog input.
		1	0	I	TTL	PORTA<0> data input; disabled when analog input enabled.
	C12IN0-	1	1	I	AN	Comparators C1 and C2 inverting input.
	AN0	1	1	I	AN	Analog input 0.
RA1/C12IN1-/AN1	RA1	0	0	O	DIG	LATA<1> data output; not affected by analog input.
		1	0	I	TTL	PORTA<1> data input; disabled when analog input enabled.
	C12IN1-	1	1	I	AN	Comparators C1 and C2 inverting input.
	AN1	1	1	I	AN	Analog input 1.
RA2/C2IN+/AN2/ DACOUT/VREF-	RA2	0	0	O	DIG	LATA<2> data output; not affected by analog input; disabled when DACOUT enabled.
		1	0	I	TTL	PORTA<2> data input; disabled when analog input enabled; disabled when DACOUT enabled.
	C2IN+	1	1	I	AN	Comparator C2 non-inverting input.
	AN2	1	1	I	AN	Analog output 2.
	DACOUT	x	1	O	AN	DAC Reference output.
	VREF-	1	1	I	AN	A/D reference voltage (low) input.
RA3/C1IN+/AN3/ VREF+	RA3	0		O	DIG	LATA<3> data output; not affected by analog input.
		1	0	I	TTL	PORTA<3> data input; disabled when analog input enabled.
	C1IN+	1	1	I	AN	Comparator C1 non-inverting input.
	AN3	1	1	I	AN	Analog input 3.
	VREF+	1	1	I	AN	A/D reference voltage (high) input.
RA4/CCP5/ C1OUT/SRQ/ T0CKI	RA4	0	—	O	DIG	LATA<4> data output.
		1	—	I	ST	PORTA<4> data input; default configuration on POR.
	CCP5	0	—	O	DIG	CCP5 Compare output/PWM output, takes priority over RA4 output.
		1	—	I	ST	Capture 5 input/Compare 5 output/ PWM 5 output.
	C1OUT	0	—	O	DIG	Comparator C1 output.
	SRQ	0	—	O	DIG	SR latch Q output; take priority over CCP 5 output.
	T0CKI	1	—	I	ST	Timer0 external clock input.
RA5/C2OUT/ SRNQ/$\overline{SS1}$/ HLVDIN/AN4	RA5	0	0	O	DIG	LATA<5> data output; not affected by analog input.
		1	0	I	TTL	PORTA<5> data input; disabled when analog input enabled.
	C2OUT	0	0	O	DIG	Comparator C2 output.
	SRNQ	0	0	O	DIG	SR latch $\overline{Q}$ output.
	$\overline{SS1}$	1	0	I	TTL	SPI slave select input (MSSP1).
	HLVDIN	1	1	I	AN	High/Low-Voltage Detect input.
	AN4	1	1	I	AN	A/D input 4.

Port A cont..

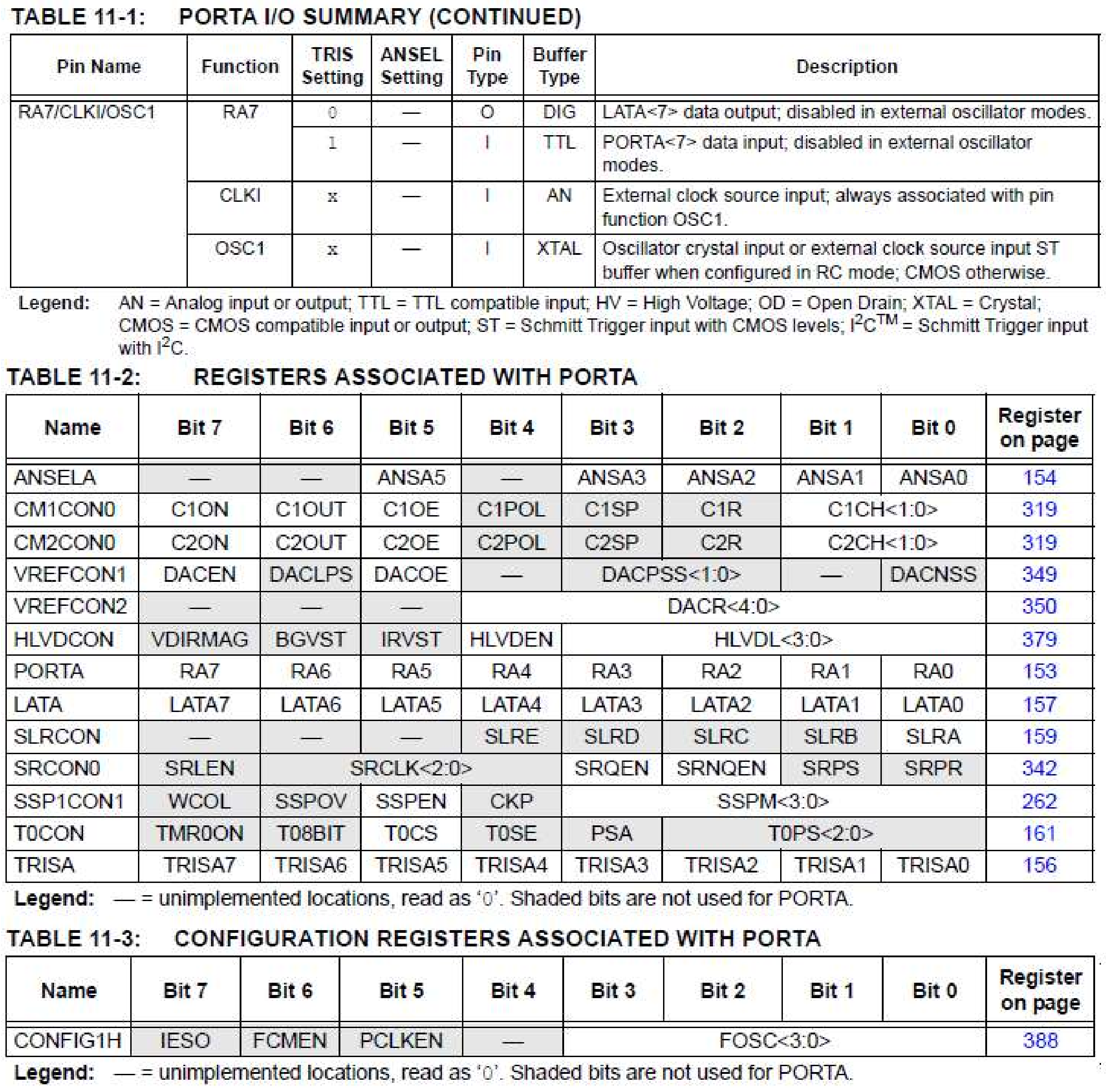


TABLE 11-1: PORTA I/O SUMMARY (CONTINUED)

Pin Name	Function	TRIS Setting	ANSEL Setting	Pin Type	Buffer Type	Description
RA7/CLKI/OSC1	RA7	0	—	O	DIG	LATA<7> data output; disabled in external oscillator modes.
		1	—	I	TTL	PORTA<7> data input; disabled in external oscillator modes.
	CLKI	x	—	I	AN	External clock source input; always associated with pin function OSC1.
	OSC1	x	—	I	XTAL	Oscillator crystal input or external clock source input ST buffer when configured in RC mode; CMOS otherwise.

Legend: AN = Analog input or output; TTL = TTL compatible input; HV = High Voltage; OD = Open Drain; XTAL = Crystal; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; I²C™ = Schmitt Trigger input with I²C.

TABLE 11-2: REGISTERS ASSOCIATED WITH PORTA

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on page
ANSELA	—	—	ANSA5	—	ANSA3	ANSA2	ANSA1	ANSA0	154
CM1CON0	C1ON	C1OUT	C1OE	C1POL	C1SP	C1R	C1CH<1:0>		319
CM2CON0	C2ON	C2OUT	C2OE	C2POL	C2SP	C2R	C2CH<1:0>		319
VREFCON1	DACEN	DACLPS	DACOE	—	DACPSS<1:0>		—	DACNSS	349
VREFCON2	—	—	—	DACR<4:0>					350
HLVDCON	VDIRMAG	BGVST	IRVST	HLVDEN	HLVDL<3:0>				379
PORTA	RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0	153
LATA	LATA7	LATA6	LATA5	LATA4	LATA3	LATA2	LATA1	LATA0	157
SLRCON	—	—	—	SLRE	SLRD	SLRC	SLRB	SLRA	159
SRCON0	SRLEN	SRCLK<2:0>			SRQEN	SRNQEN	SRPS	SRPR	342
SSP1CON1	WCOL	SSPOV	SSPEN	CKP	SSPM<3:0>				262
T0CON	TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS<2:0>			161
TRISA	TRISA7	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	156

Legend: — = unimplemented locations, read as ‘0’. Shaded bits are not used for PORTA.

TABLE 11-3: CONFIGURATION REGISTERS ASSOCIATED WITH PORTA

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on page
CONFIG1H	IESO	FCMEN	PCLKEN	—	FOSC<3:0>				388

Legend: — = unimplemented locations, read as ‘0’. Shaded bits are not used for PORTA.

To see the rest of the Ports, review the data sheet

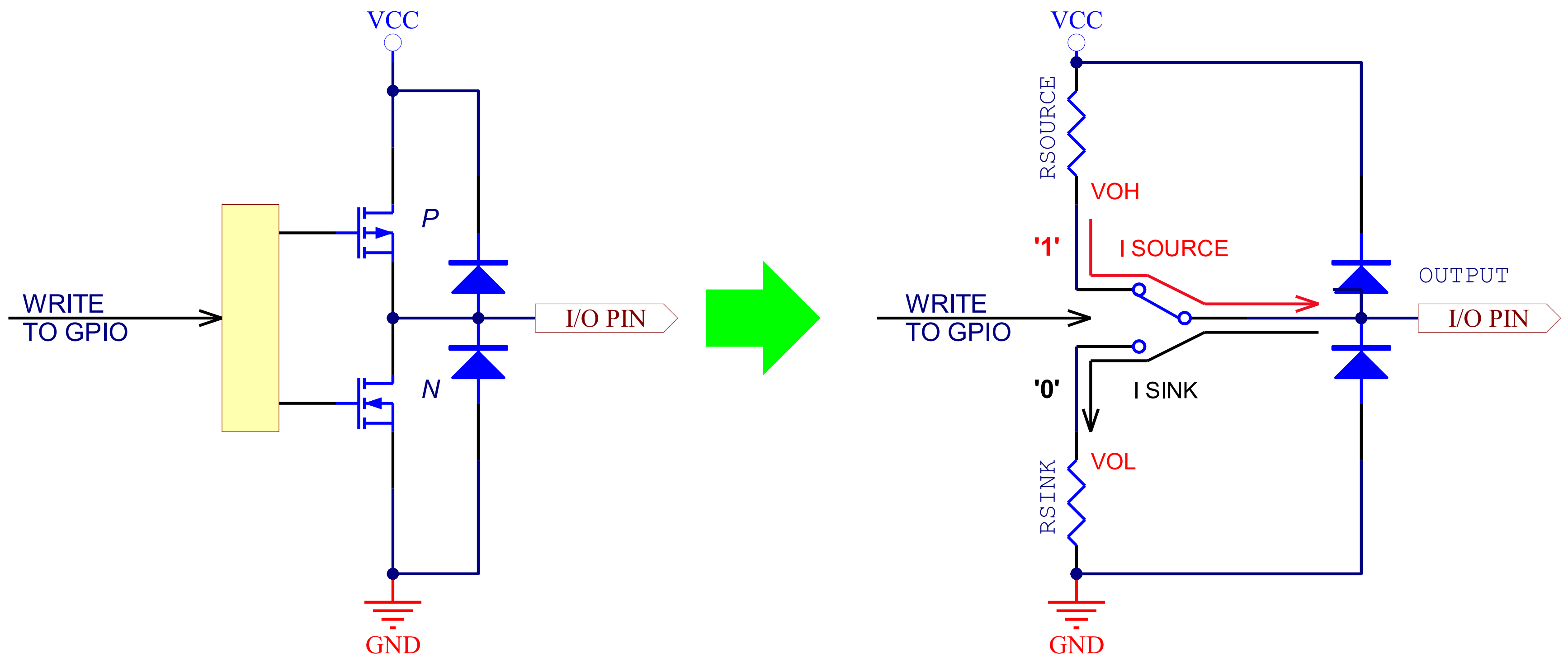
Page 137 (I/O Ports section)

http://ww1.microchip.com/downloads/en/devicedoc/30000684B.pdf

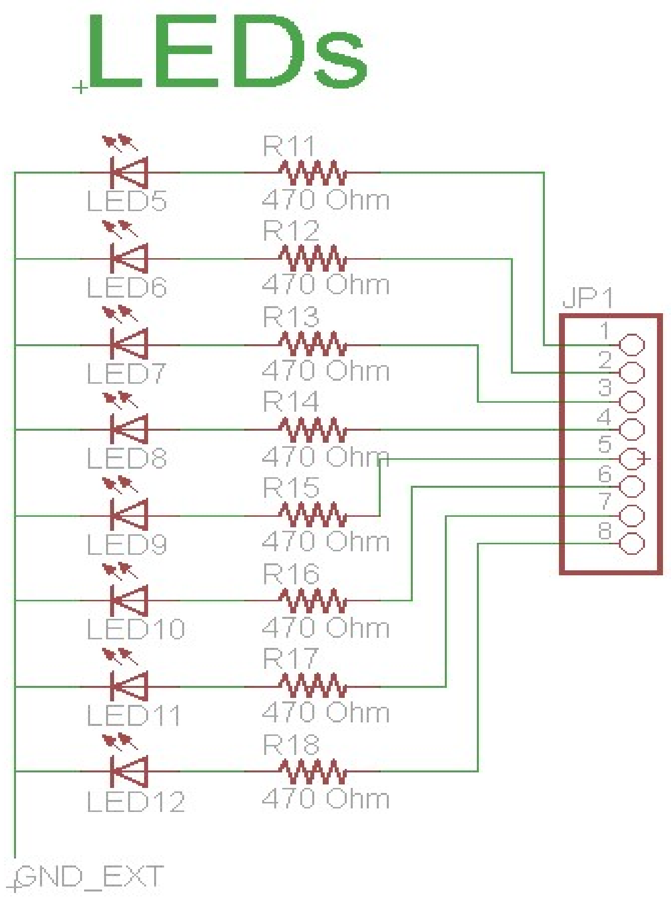
GPIO basic parameters

- GPIO from the electrical-logical point of view behave with similar characteristics of any discrete logic gate
  - INPUT:
    - VIH (Voltage threshold that defines a logic 1)
    - VIL (Voltage threshold that defines a logic 0)
    - Currents in CMOS inputs are negligible
  - OUTPUT
    - VOH @ IOH (Voltage output high sorcing certain current)
    - VOL @ IOL (Voltage output low sinking certain current)

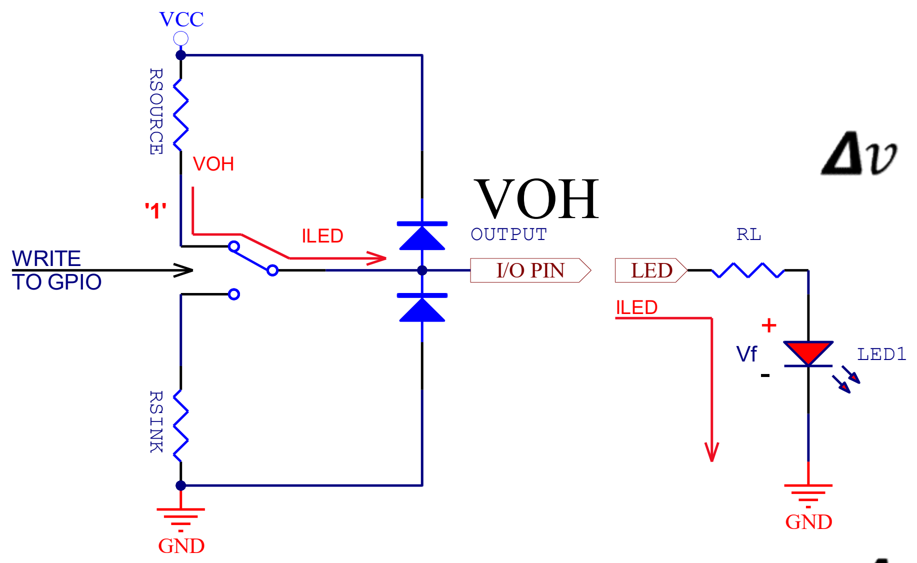
Anatomy of an output port

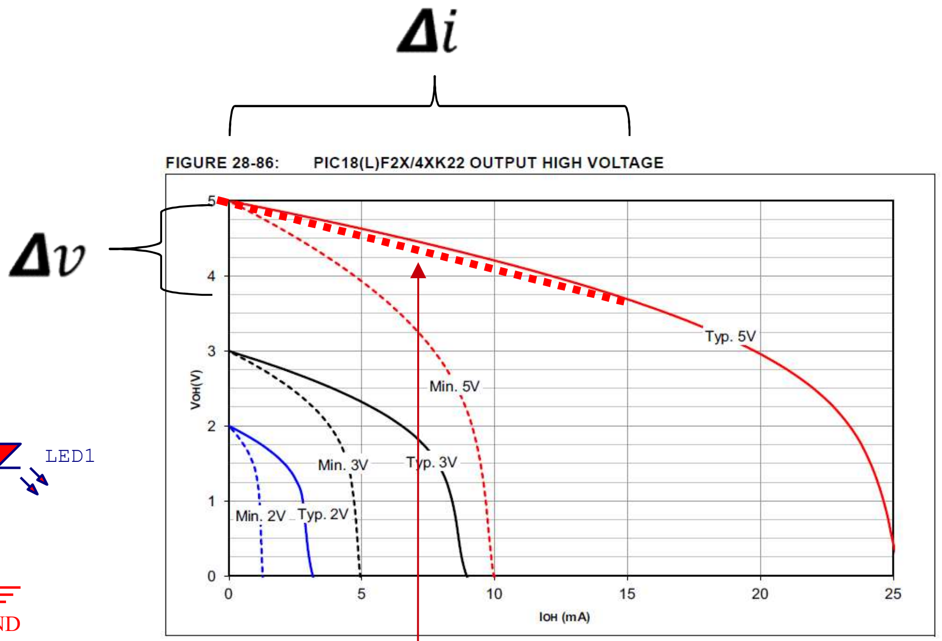


Lab board LED interface



LED interfacing





$$\frac{\Delta v}{\Delta i} = \frac{1.2}{0.015}$$

VCC = ILED*RSOURCE + ILED*RL + VF

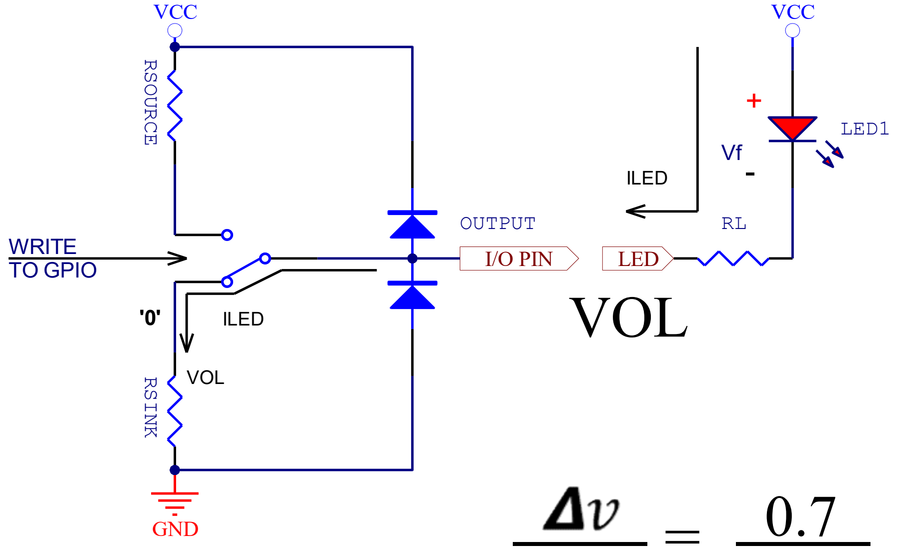
ILED = (VCC-VF)/(RSOURCE+RL)

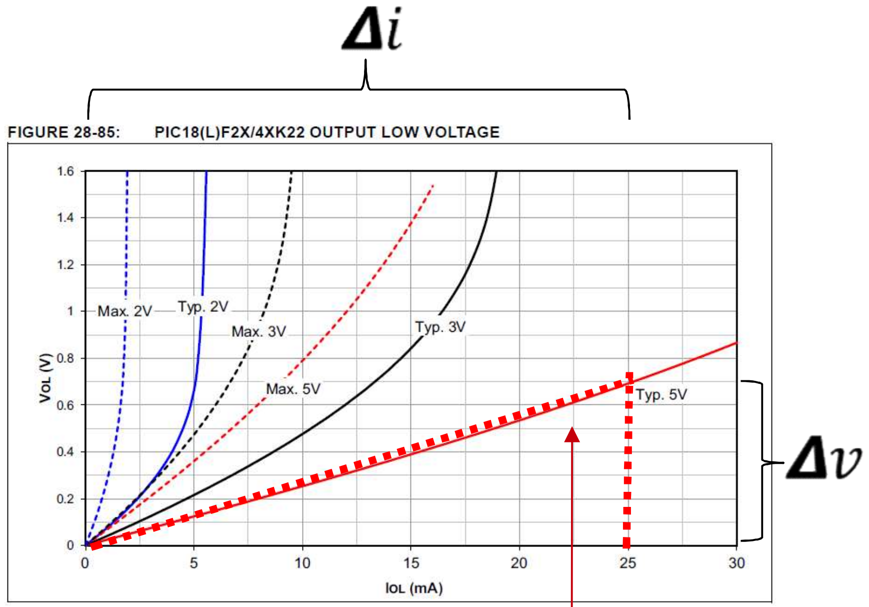
ILED = (5V-2V)/(80+470) = 5mA

VOH = 5- 5ma*80 = 4.6V

http://www.vishay.com/docs/83006/tlhg440.pdf

LED interfacing





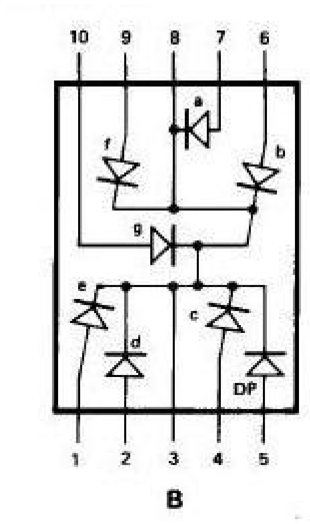
$$\frac{\Delta v}{\Delta i} = \frac{0.7}{0.025}$$

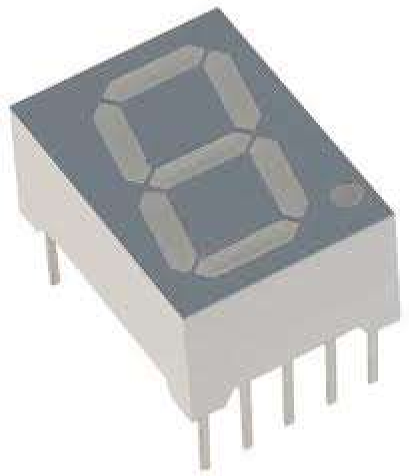
VCC = VF + ILED*RL + RSINK*ILED

ILED = (VCC-VF)/(RSINK+RL)

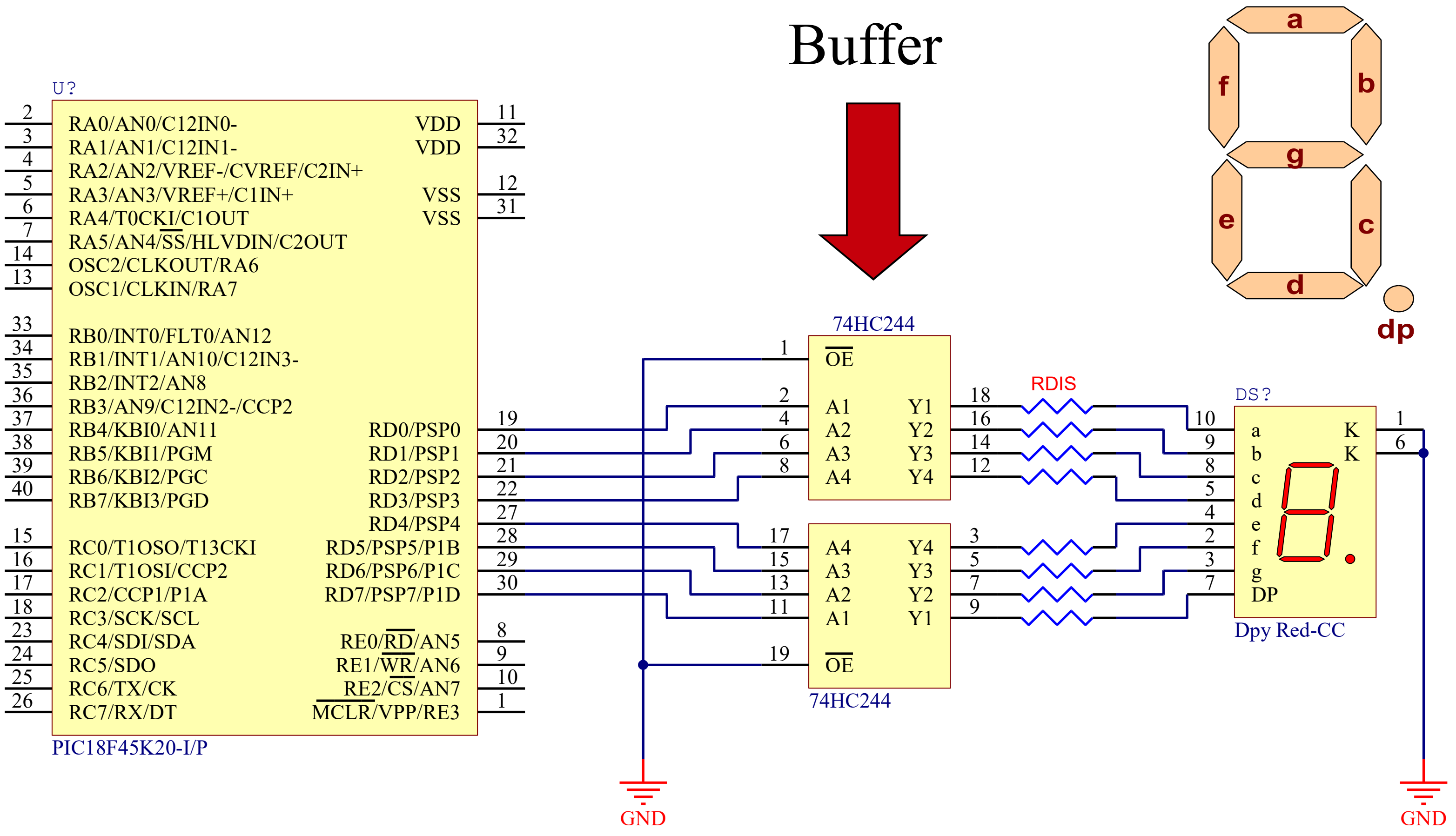
ILED = (5V-2V)/(28+470) = 6mA VOL = 6ma*28 = 0.17V

7 Segment displays

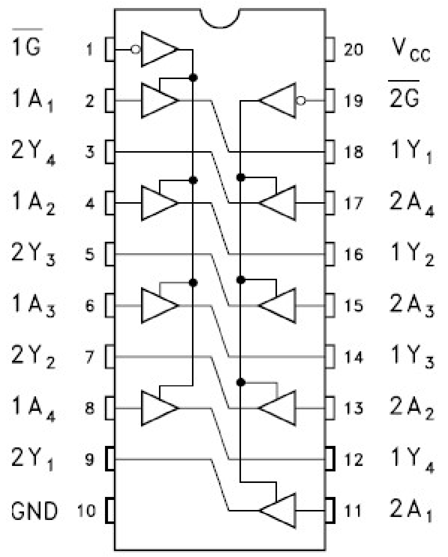




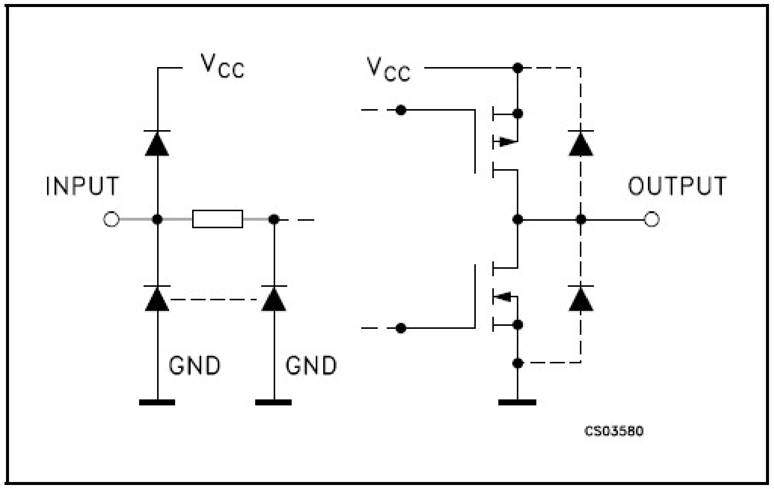
Buffer


U?
RA0/AN0/C12IN0-
RA1/AN1/C12IN1-
RA2/AN2/VREF-/CVREF/C2IN+
RA3/AN3/VREF+/C1IN+
RA4/T0CKI/C1OUT
RA5/AN4/SS/HLVDIN/C2OUT
OSC2/CLKOUT/RA6
OSC1/CLKIN/RA7
RB0/INT0/FLT0/AN12
RB1/INT1/AN10/C12IN3-
RB2/INT2/AN8
RB3/AN9/C12IN2-/CCP2
RB4/KBI0/AN11
RB5/KBI1/PGM
RB6/KBI2/PGC
RB7/KBI3/PGD
RC0/T1OSO/T13CKI
RC1/T1OSI/CCP2
RC2/CCP1/P1A
RC3/SCK/SCL
RC4/SDI/SDA
RC5/SDO
RC6/TX/CK
RC7/RX/DT
VDD
VDD
VSS
VSS
RD0/PSP0
RD1/PSP1
RD2/PSP2
RD3/PSP3
RD4/PSP4
RD5/PSP5/P1B
RD6/PSP6/P1C
RD7/PSP7/P1D
RE0/RD/AN5
RE1/WR/AN6
RE2/CS/AN7
MCLR/VPP/RE3
PIC18F45K20-I/P
74HC244
OE
A1 A2 A3 A4
Y1 Y2 Y3 Y4
74HC244
RDIS
DS?
a b c d e f g DP
K K
Dpy Red-CC
GND
GND
a b c d e f g dp

Buffer



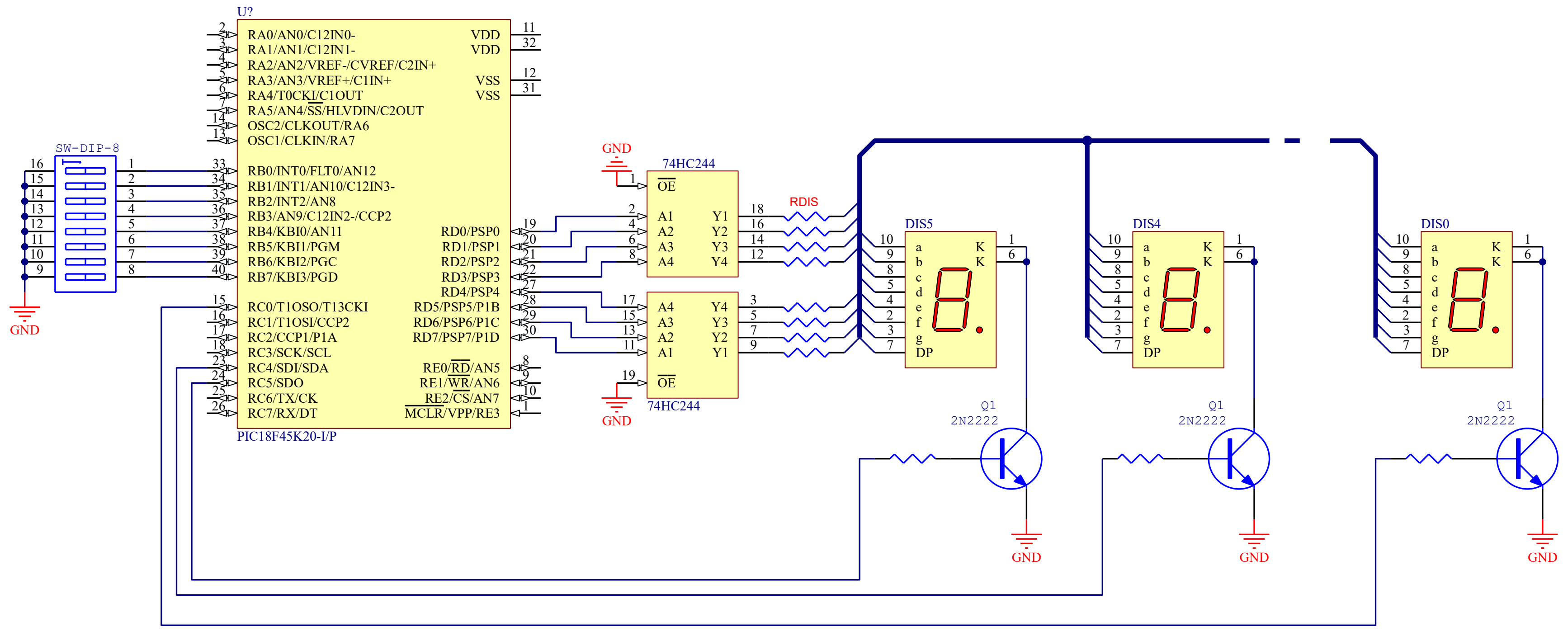
INPUT AND OUTPUT EQUIVALENT CIRCUIT



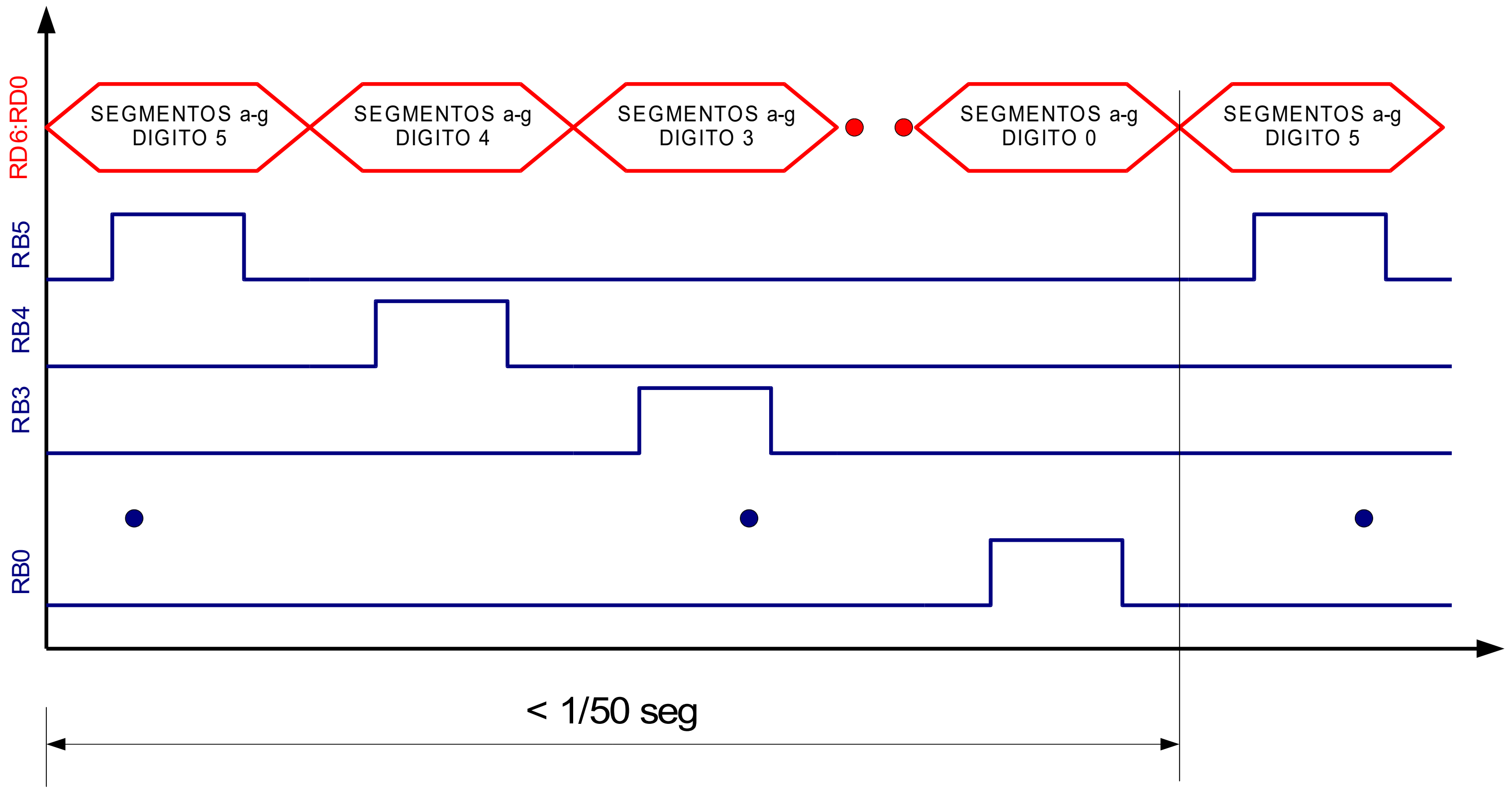
V_{OH}	High Level Output Voltage	2.0	I_O=-20 µA	1.9	2.0		1.9		1.9		V
		4.5	I_O=-20 µA	4.4	4.5		4.4		4.4		
		6.0	I_O=-20 µA	5.9	6.0		5.9		5.9		
		4.5	I_O=-6.0 mA	4.18	4.31		4.13		4.10		
		6.0	I_O=-7.8 mA	5.68	5.8		5.63		5.60		

Handling more than one digit

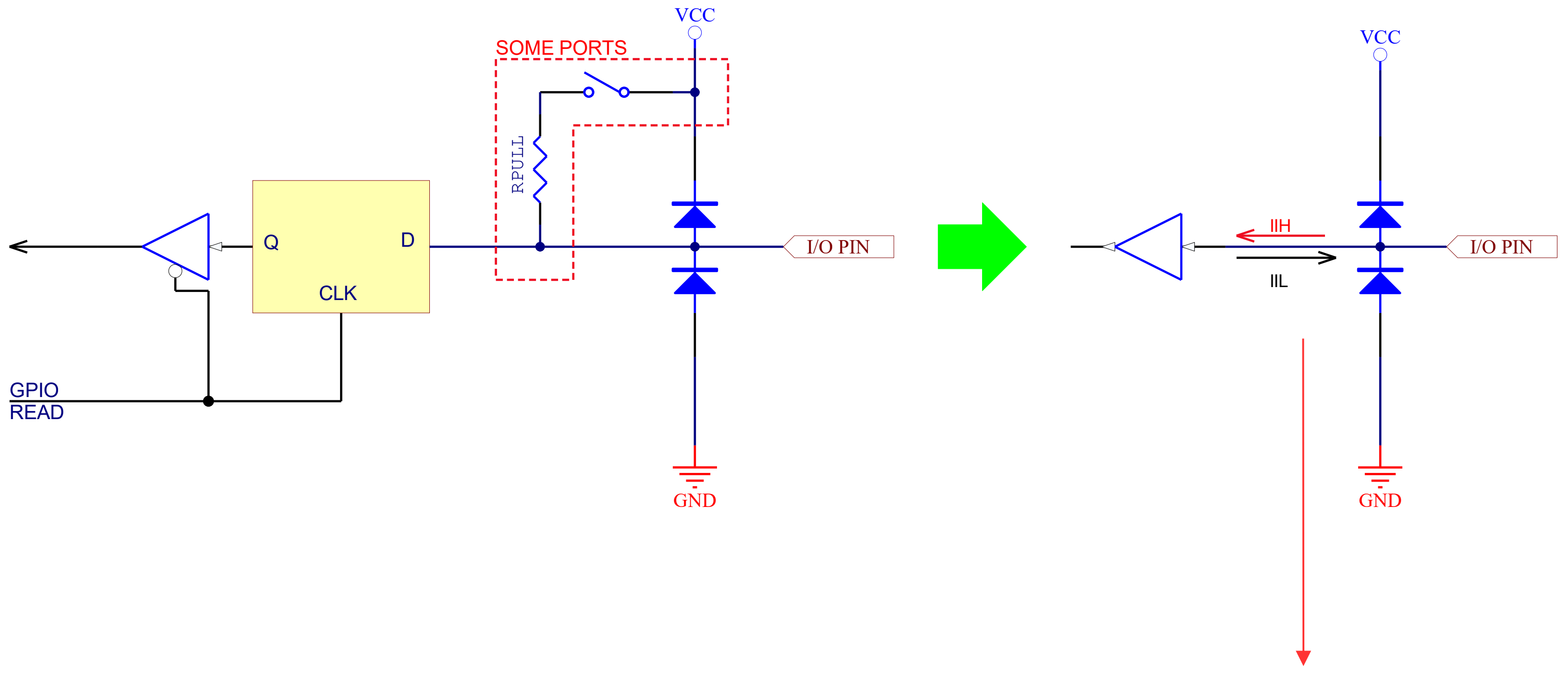
Multiplexing

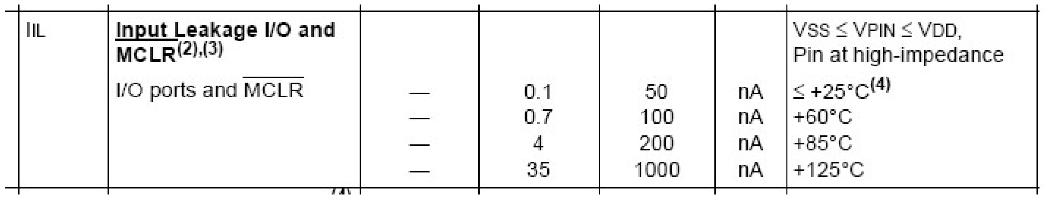


Multiplexing



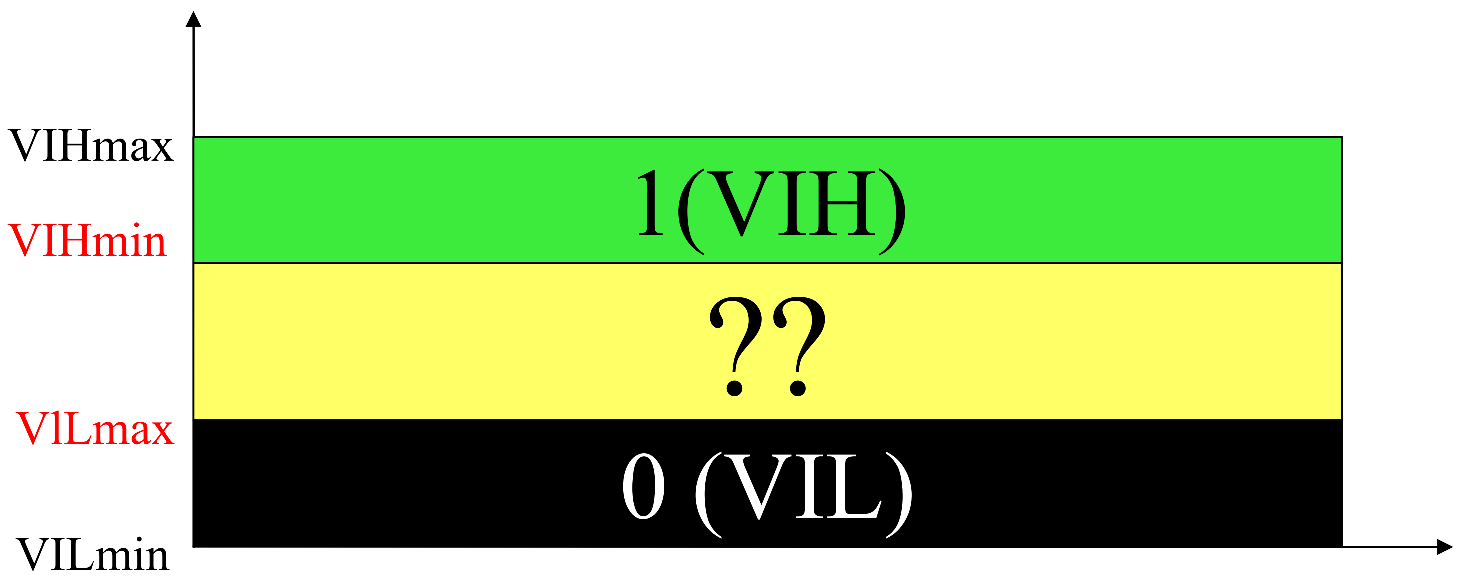
Anatomy of a input



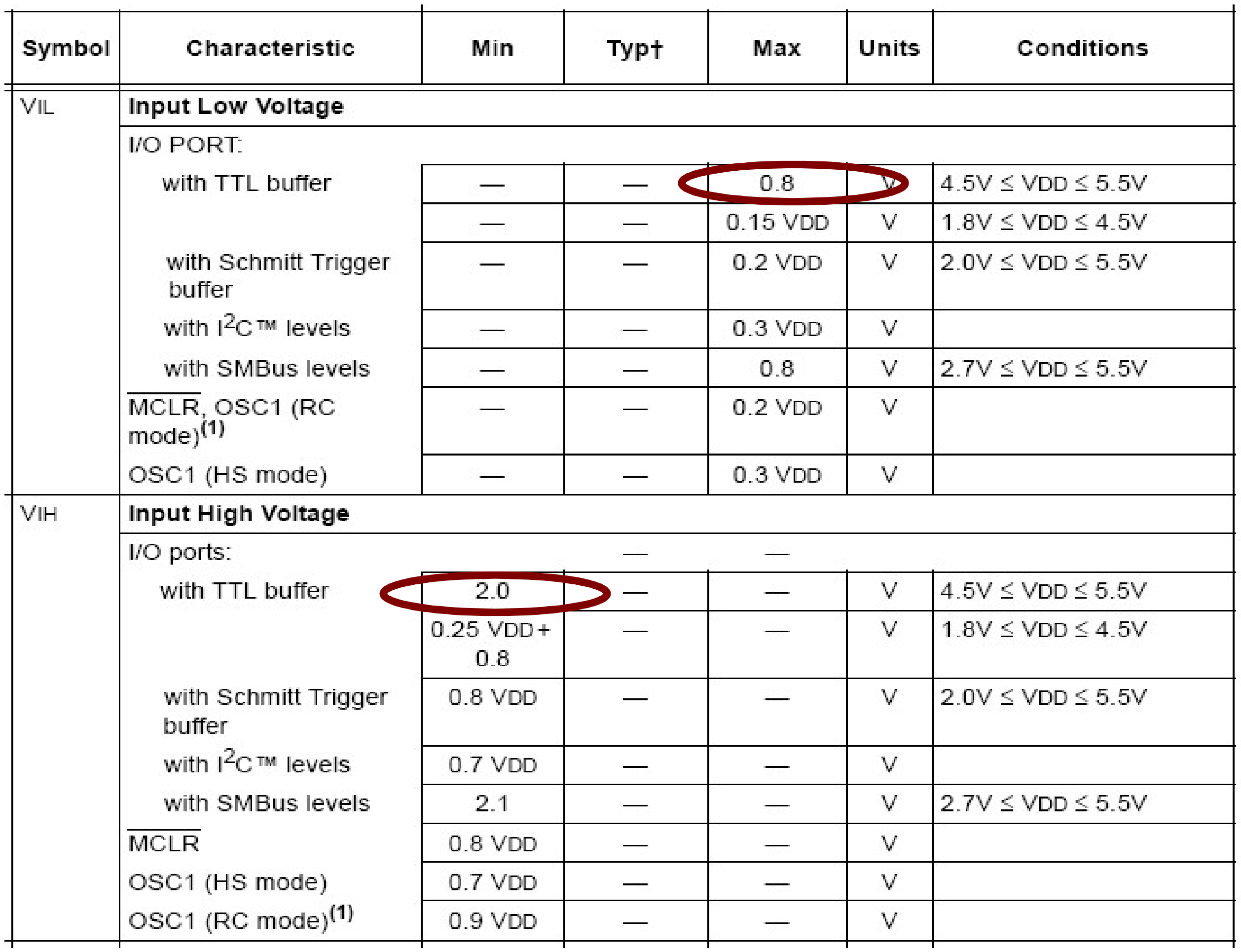


IIL	**Input Leakage I/O and MCLR**[(2),(3)]					VSS ≤ VPIN ≤ VDD, Pin at high-impedance
	I/O ports and MCLR	—	0.1	50	nA	≤ +25°C[(4)]
		—	0.7	100	nA	+60°C
		—	4	200	nA	+85°C
		—	35	1000	nA	+125°C

Input Voltages

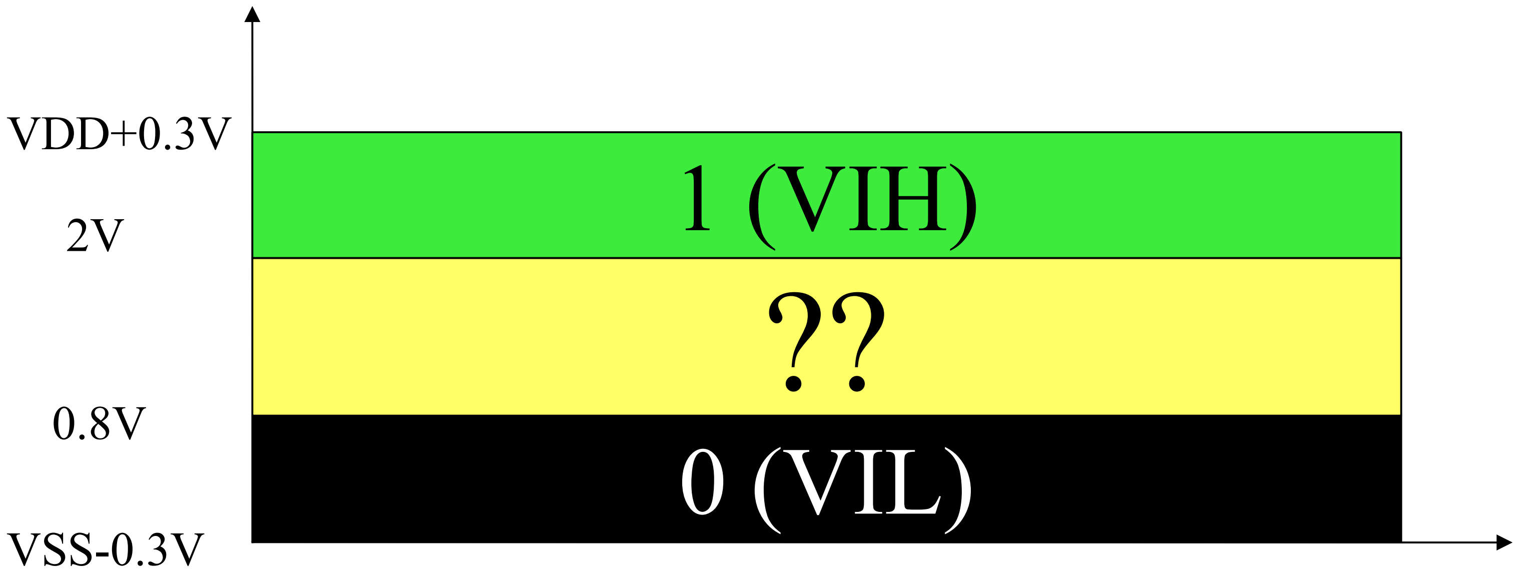


Input voltages

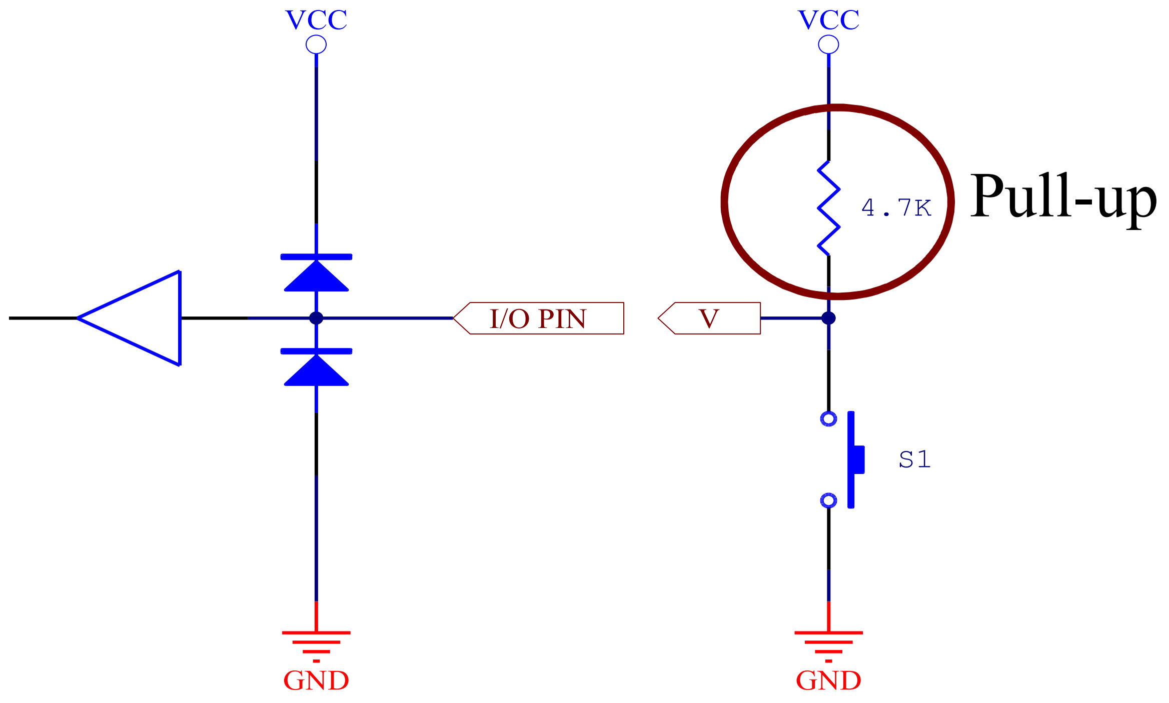


Symbol	Characteristic	Min	Typ†	Max	Units	Conditions
VIL	**Input Low Voltage**					
	I/O PORT:					
	with TTL buffer	—	—	0.8	V	4.5V ≤ VDD ≤ 5.5V
		—	—	0.15 VDD	V	1.8V ≤ VDD ≤ 4.5V
	with Schmitt Trigger buffer	—	—	0.2 VDD	V	2.0V ≤ VDD ≤ 5.5V
	with I^2C™ levels	—	—	0.3 VDD	V	
	with SMBus levels	—	—	0.8	V	2.7V ≤ VDD ≤ 5.5V
	$\overline{\text{MCLR}}$, OSC1 (RC mode)(1)	—	—	0.2 VDD	V	
	OSC1 (HS mode)	—	—	0.3 VDD	V	
VIH	**Input High Voltage**					
	I/O ports:		—	—		
	with TTL buffer	2.0	—	—	V	4.5V ≤ VDD ≤ 5.5V
		0.25 VDD + 0.8	—	—	V	1.8V ≤ VDD ≤ 4.5V
	with Schmitt Trigger buffer	0.8 VDD	—	—	V	2.0V ≤ VDD ≤ 5.5V
	with I^2C™ levels	0.7 VDD	—	—	V	
	with SMBus levels	2.1	—	—	V	2.7V ≤ VDD ≤ 5.5V
	$\overline{\text{MCLR}}$	0.8 VDD	—	—	V	
	OSC1 (HS mode)	0.7 VDD	—	—	V	
	OSC1 (RC mode)(1)	0.9 VDD	—	—	V	

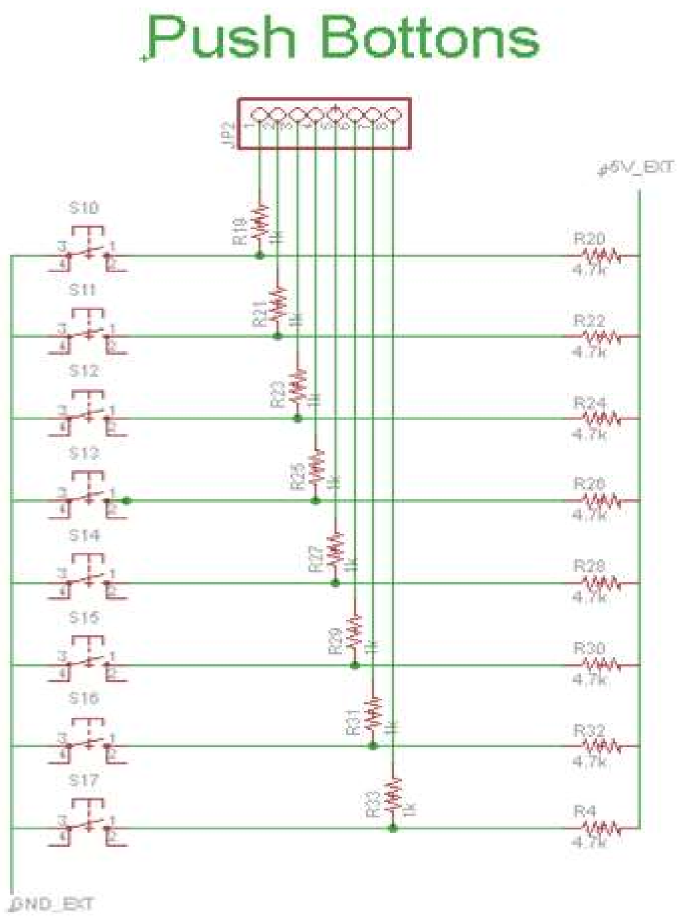
Input voltages



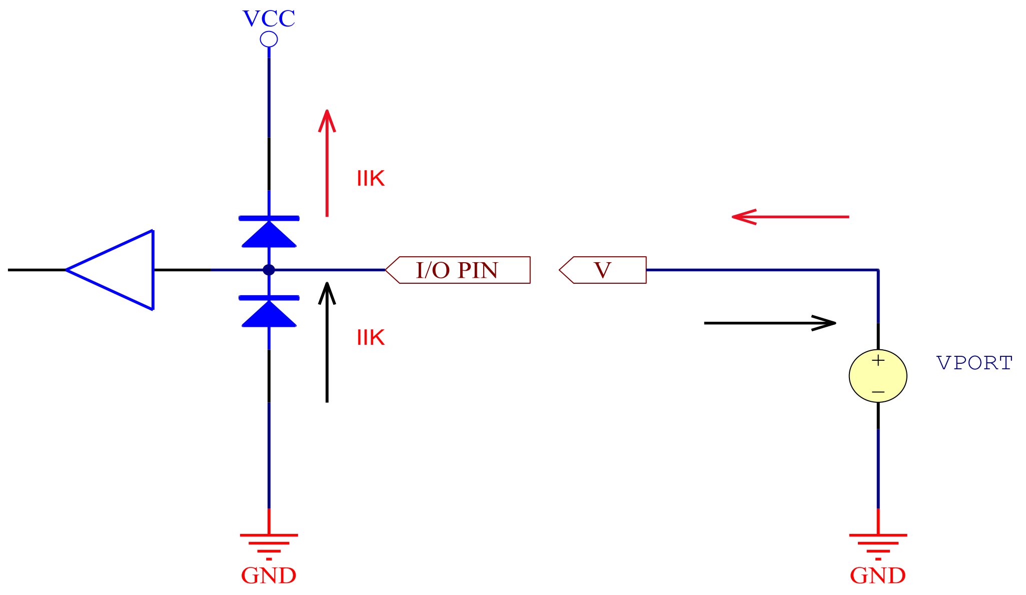
Interfacing switches



Switches in the lab board

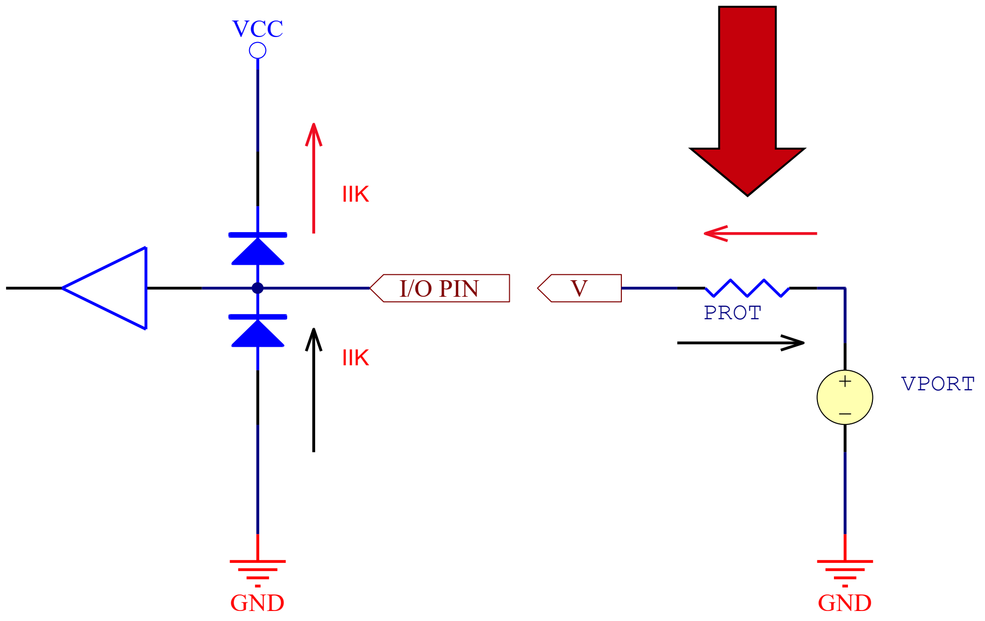


Current capacity of the input

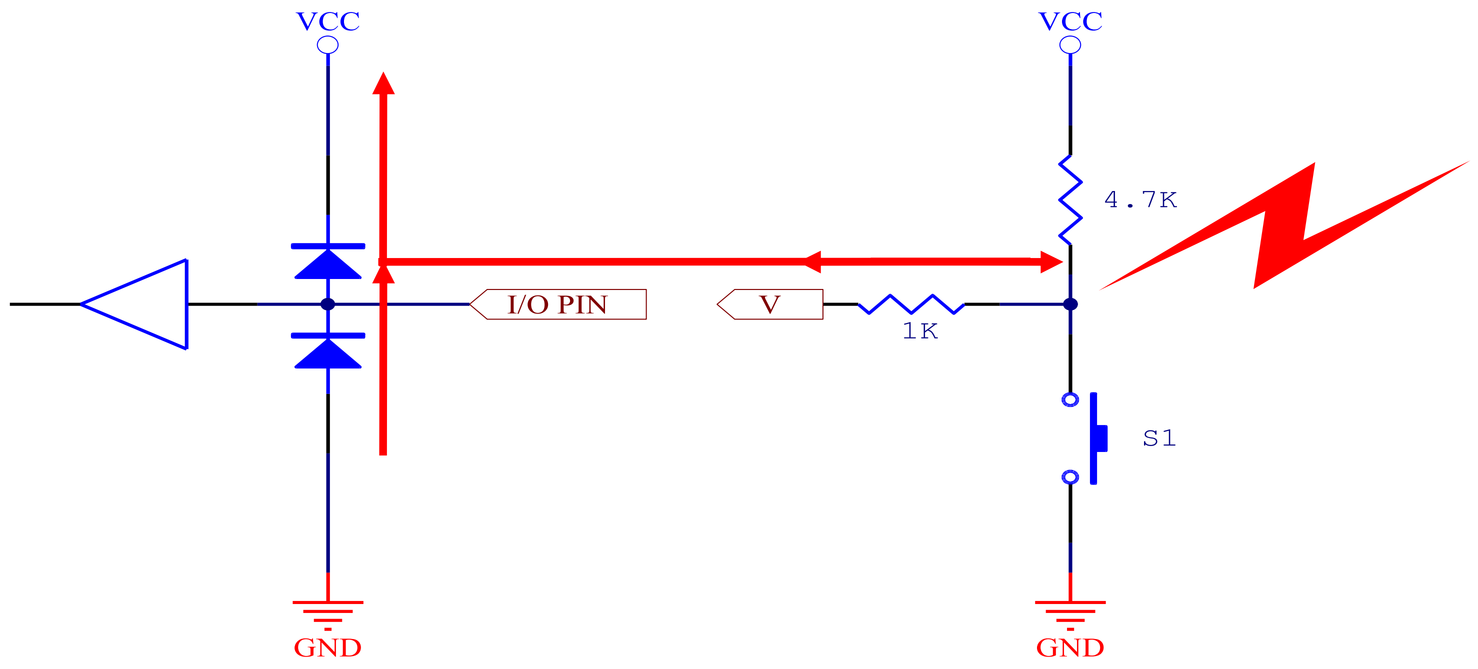


Input clamp current, IIK (VI < 0 or VI > VDD)............±20 mA

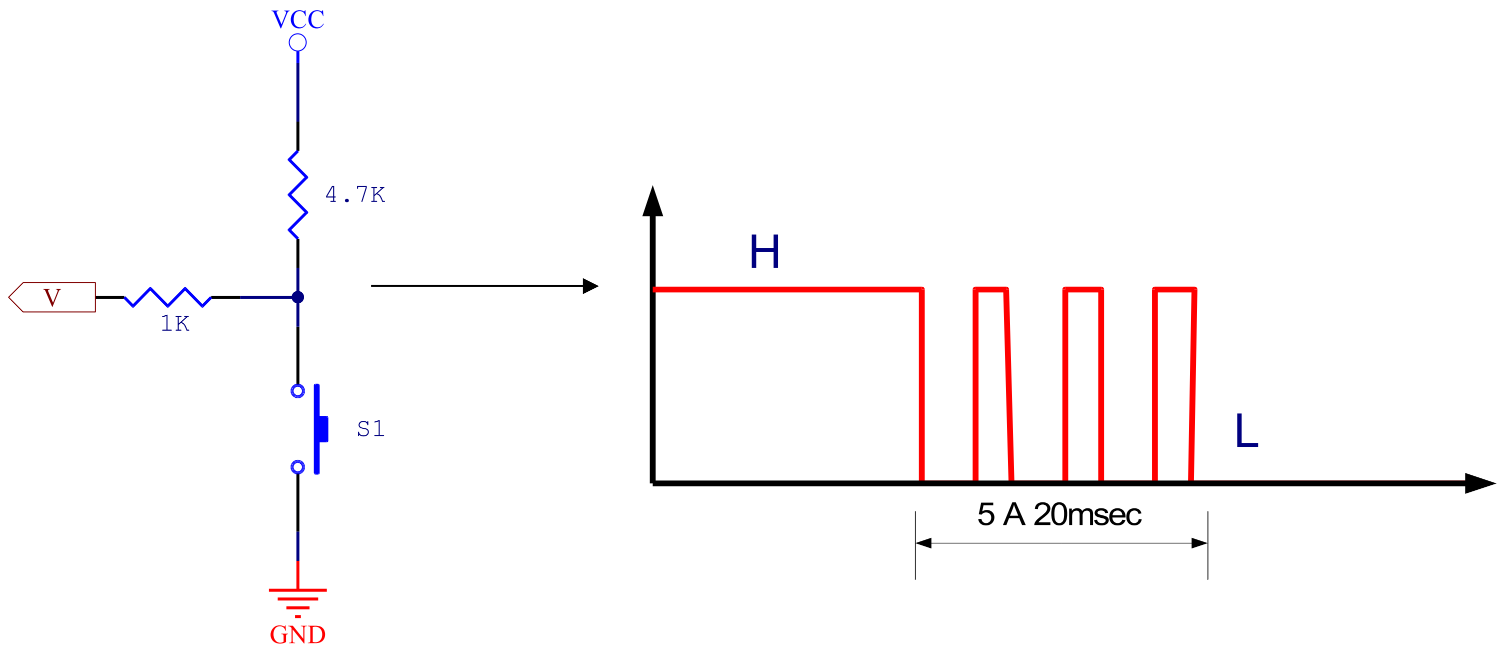
Resistor for input protection



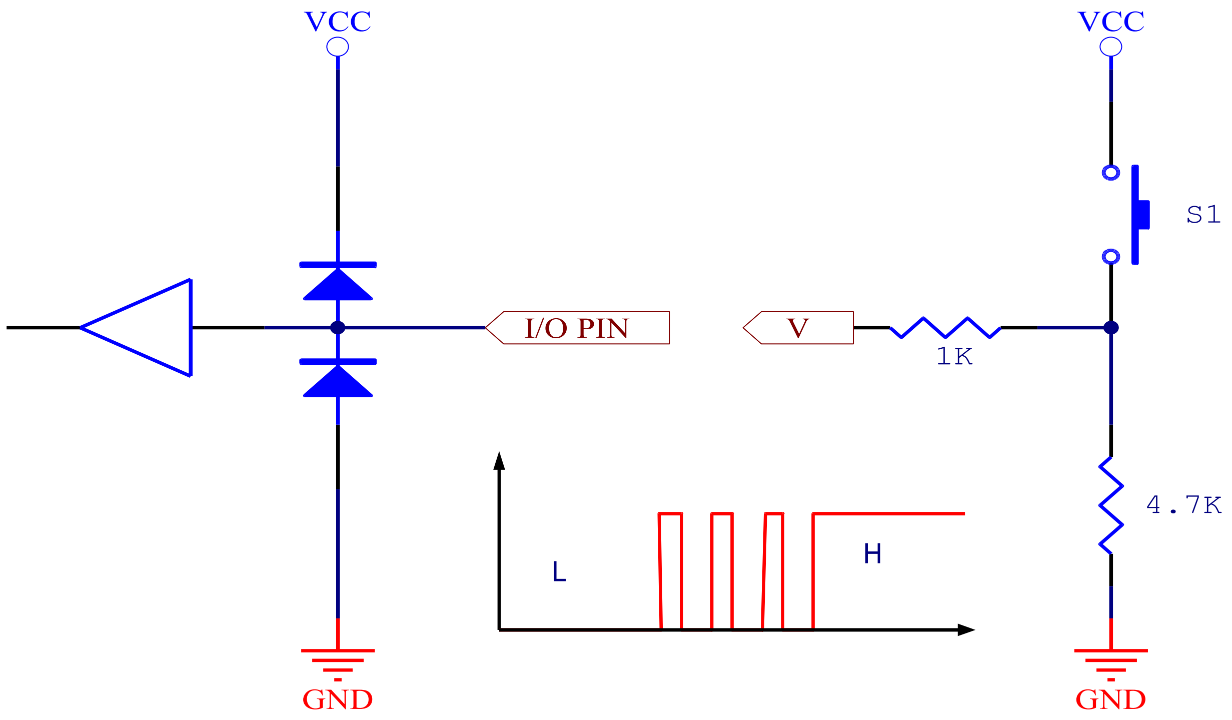
Input protections



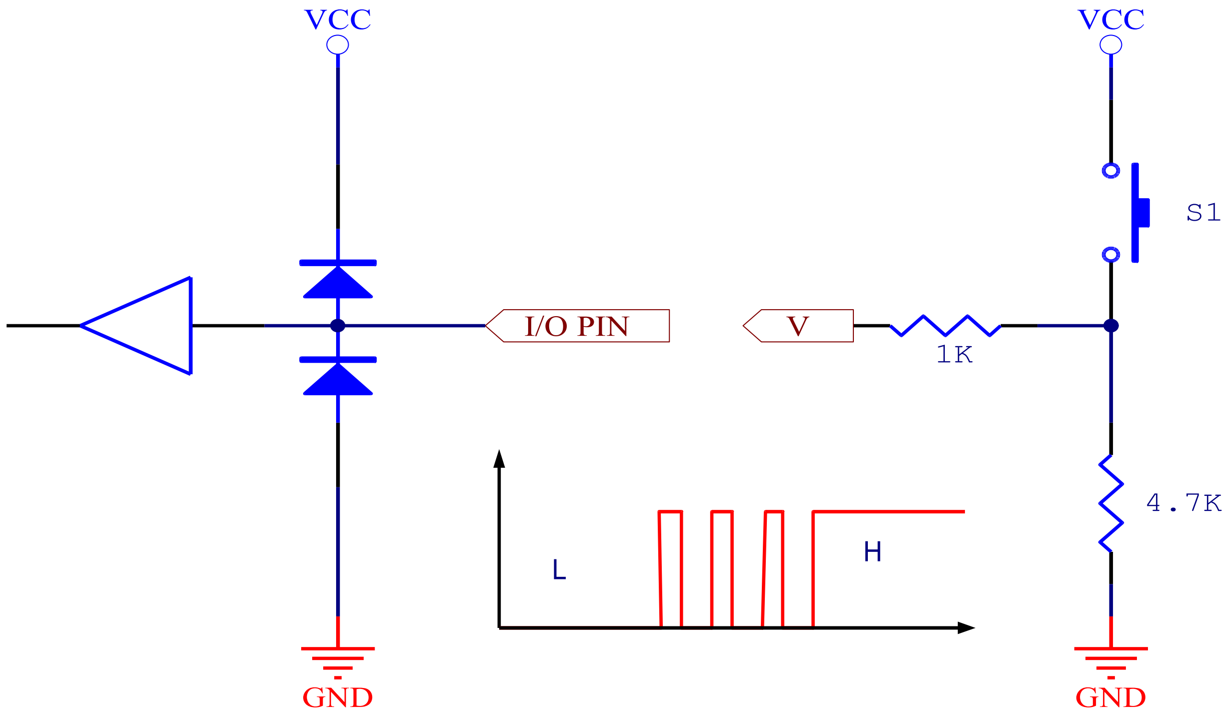
Bouncing in switches



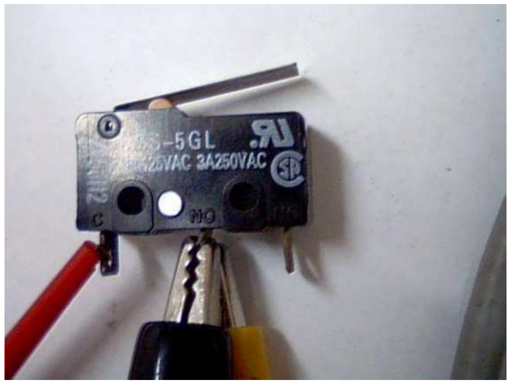
Active high switch

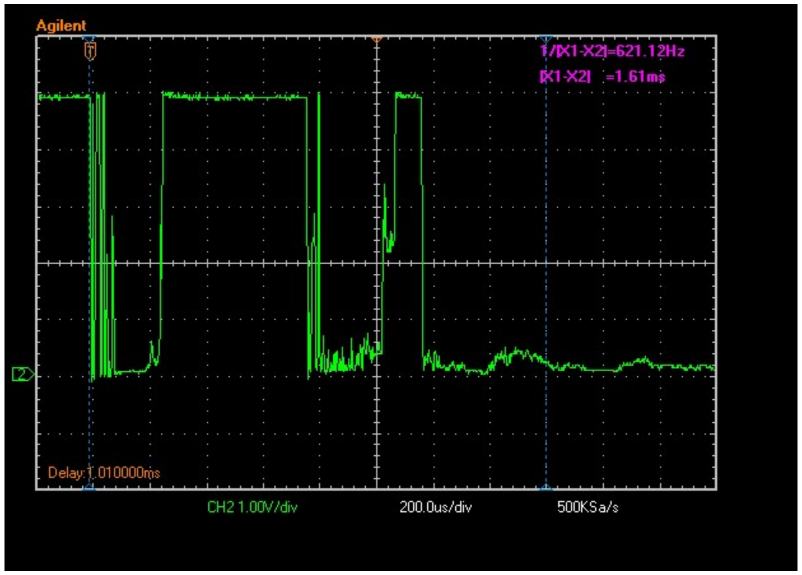


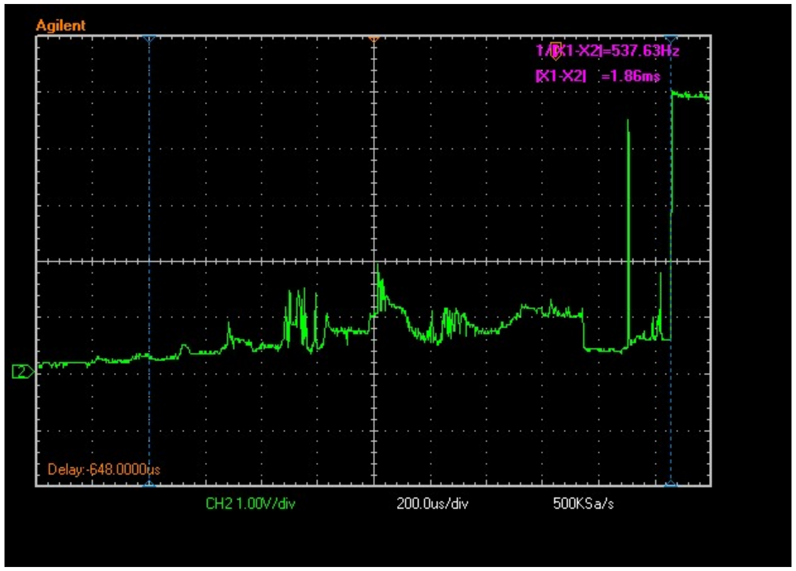
Active high switch



Bounces


S-5GL
125VAC 3A250VAC
NO


Agilent
1/|X1-X2|=621.12Hz
|X1-X2| =1.61ms
Delay:1.010000ms
CH2 1.00V/div
200.0us/div
500KSa/s

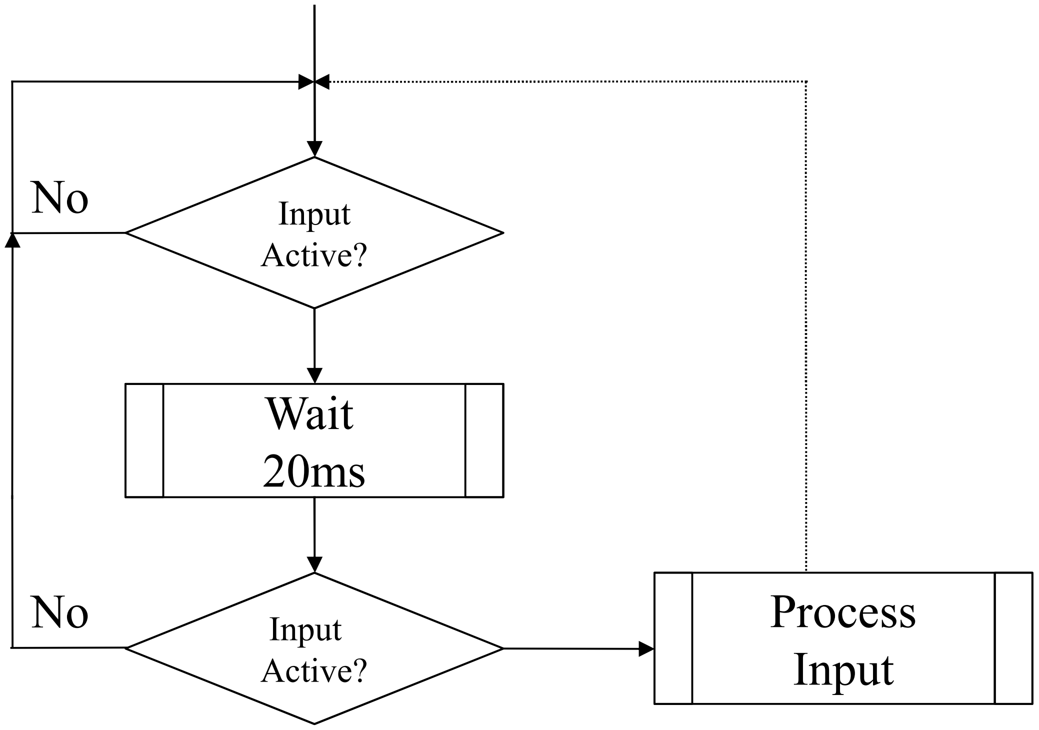

Agilent
1/|X1-X2|=537.63Hz
|X1-X2| =1.86ms
Delay:-648.0000us
CH2 1.00V/div
200.0us/div
500KSa/s

Debouncing techniques

- There are many debouncing techniques
- Use a set-reset latch before the switch
- Use a special integrated circuit to eliminate bounces
- RC filters
- Firmware

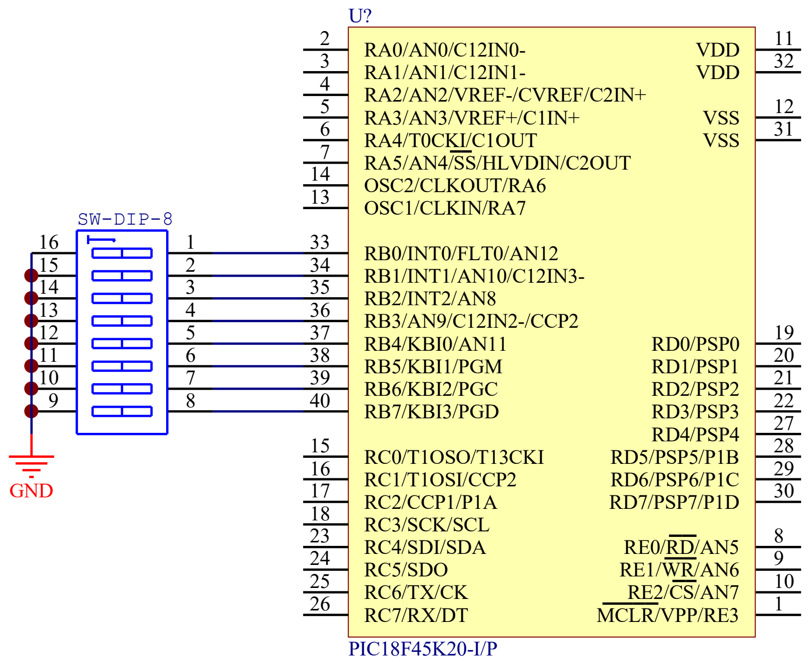
- See page 315 : Huang

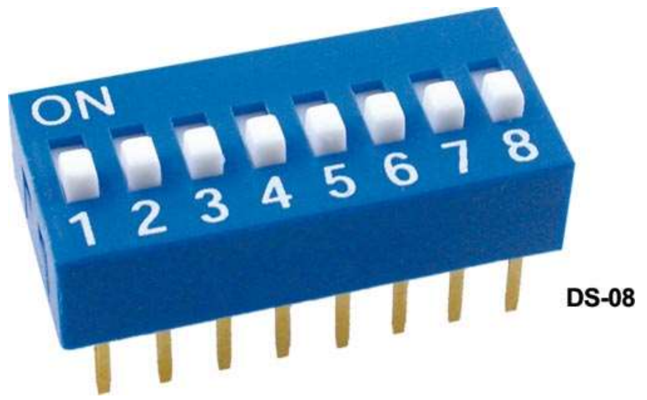
Anti debouncing using firmware



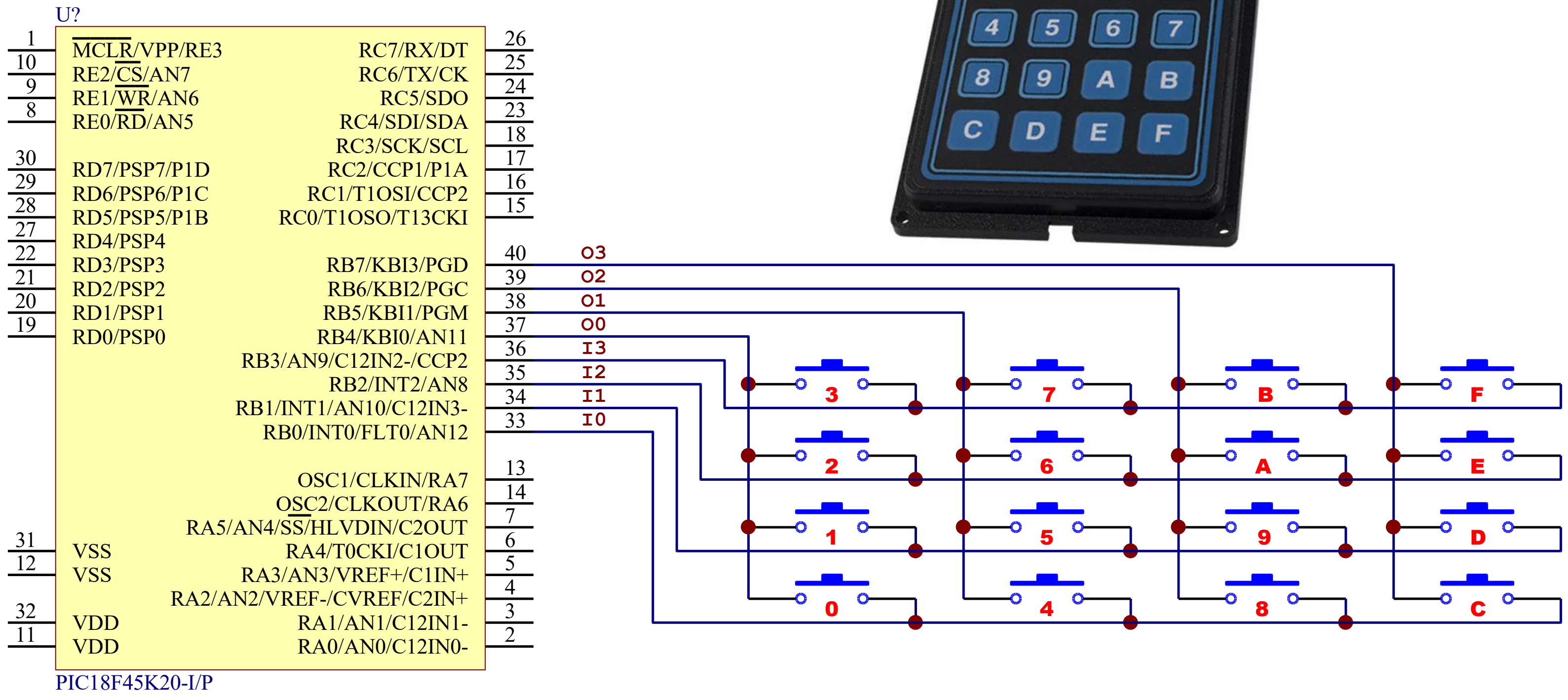
http://www.ganssle.com/debouncing.htm
https://youtu.be/4ZMiKMUec9o

Interface for switches using internal pull-ups

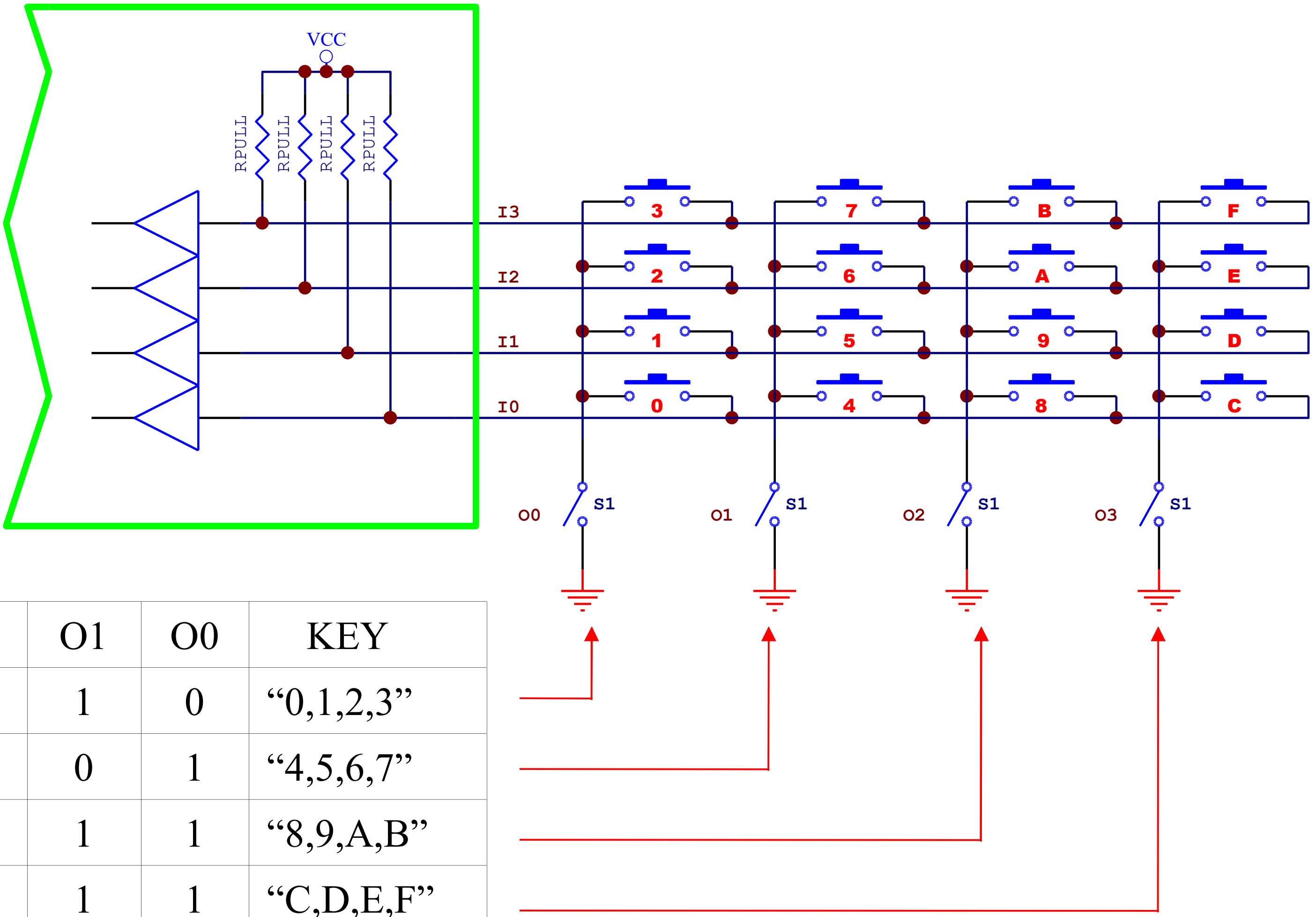




Manage of multiple inputs



Manage of multiple inputs



O3	O2	O1	O0	KEY
1	1	1	0	"0,1,2,3"
1	1	0	1	"4,5,6,7"
1	0	1	1	"8,9,A,B"
0	1	1	1	"C,D,E,F"

Port initialization general

- Check all possible configurations of each port (review the data sheet of the microcontroller)
- In many microcontrollers, the GPIOS are also shared with other input-output peripherals like serial ports, analog inputs, etc.
- Is a good practice to always configure the port even if the desired configuration is the default one (after reset)

Port initialization for PIC

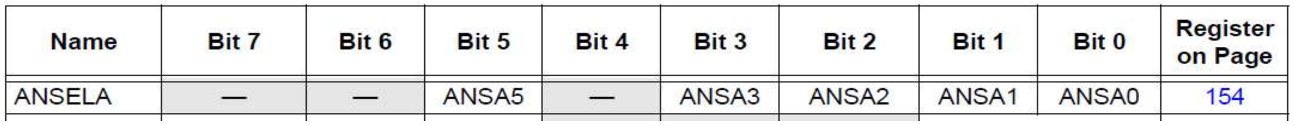
- Define if the port is digital or analong
- Define the direction (input or output)
- If digital input, set internal pull-up resistors if availale and required
- If output is better to define the default logic value before the configuration using the LATx register

Port initialization for PIC

- The port initialization is made using a SFR register called TRISx (TRISA, TRISB….etc)
  - Bit 0 of the register corresponds to the bit 0 of the PORTx
  - A logic 1 in any bit of the register defines a input, a logic 0 an output
- For the PIC18 some of the ports are also analog, so there is also a SFR register that must configured to tell the PIC if the port is analog or digital this is done with the ANSELx (ANSELA,ANSELB,..etc)

Example of port initialization

- Example, initialize bits 0 and 7 of PORTA as output and the rest of the signals as inputs:
- ANSELA = 0x00; //Define port as digital
- TRISA = 0b01111110; //Bit 0 and 7 as outputs
- Another way is to use the direct bit manipulation instructions



Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
ANSELA	—	—	ANSA5	—	ANSA3	ANSA2	ANSA1	ANSA0	154

Port B

- The PORTB provides internal pull-ups resistors that can be used activated individually by the user

```
ANSELB   = 0b11000000;   //Entras en operacion digital (no analogica)
TRISB    = 0b11111111;   //Definir direccion (0 salida 1 entrada)
WPUB     = 0b11111111;   //Define que todas los bits tendran pull-up
//Habilita la funcionalidad de pull-up en puerto B, apagando el bit 7
//del registro de control INTCON2
INTCON2 = INTCON2 & 0b01111111;
```

REGISTER 10-4: ANSELB – PORTB ANALOG SELECT REGISTER

U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
—	—	ANSB5	ANSB4	ANSB3	ANSB2	ANSB1	ANSB0
bit 7							bit 0

REGISTER 9-2: INTCON2: INTERRUPT CONTROL 2 REGISTER

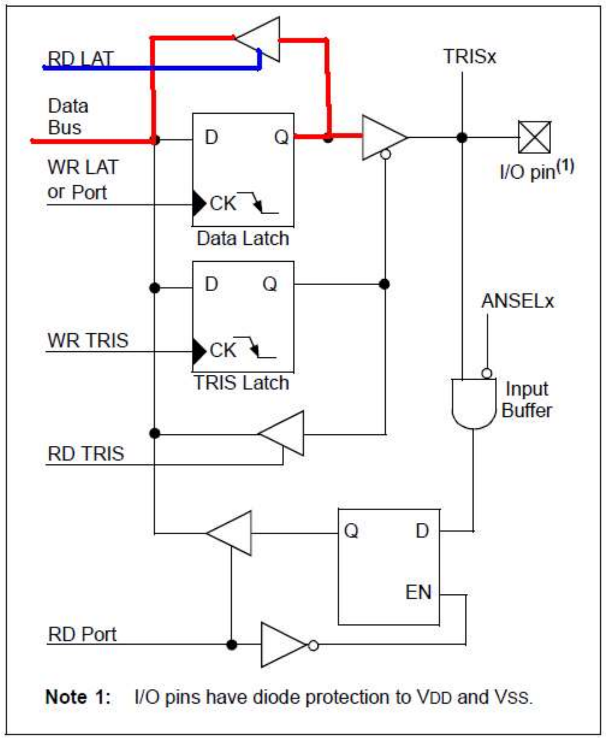
R/W-1	R/W-1	R/W-1	R/W-1	U-0	R/W-1	U-0	R/W-1
$\overline{\text{RBPU}}$	INTEDG0	INTEDG1	INTEDG2	—	TMR0IP	—	RBIP
bit 7							bit 0

REGISTER 10-12: WPUB: WEAK PULL-UP PORTB REGISTER

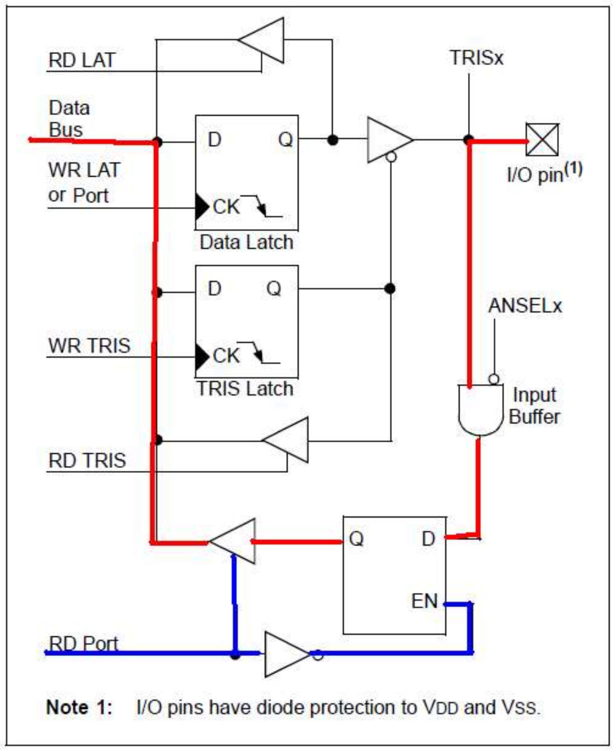
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
WPUB7	WPUB6	WPUB5	WPUB4	WPUB3	WPUB2	WPUB1	WPUB0
bit 7							bit 0

Port input reading

- To read the value on a port the programmer can use 2 registers



LATx



PORTx

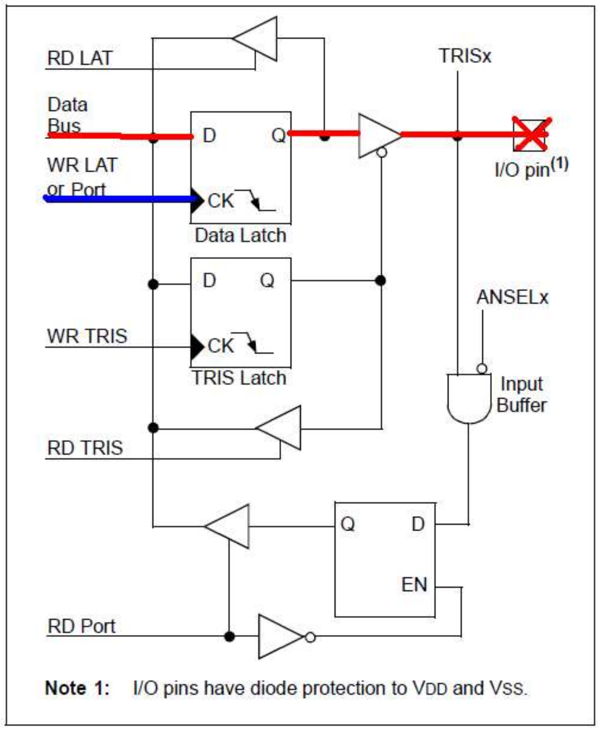
¿When do we use PORTx or LATx?

- When we what to read the logic value at the PIN you must use PORT instruction
- When you use operations that must read and write to a port, that is, to use the previous value written do to something then is better:

  LATA = LATA +1 tan PORTA = PORTA + 1;

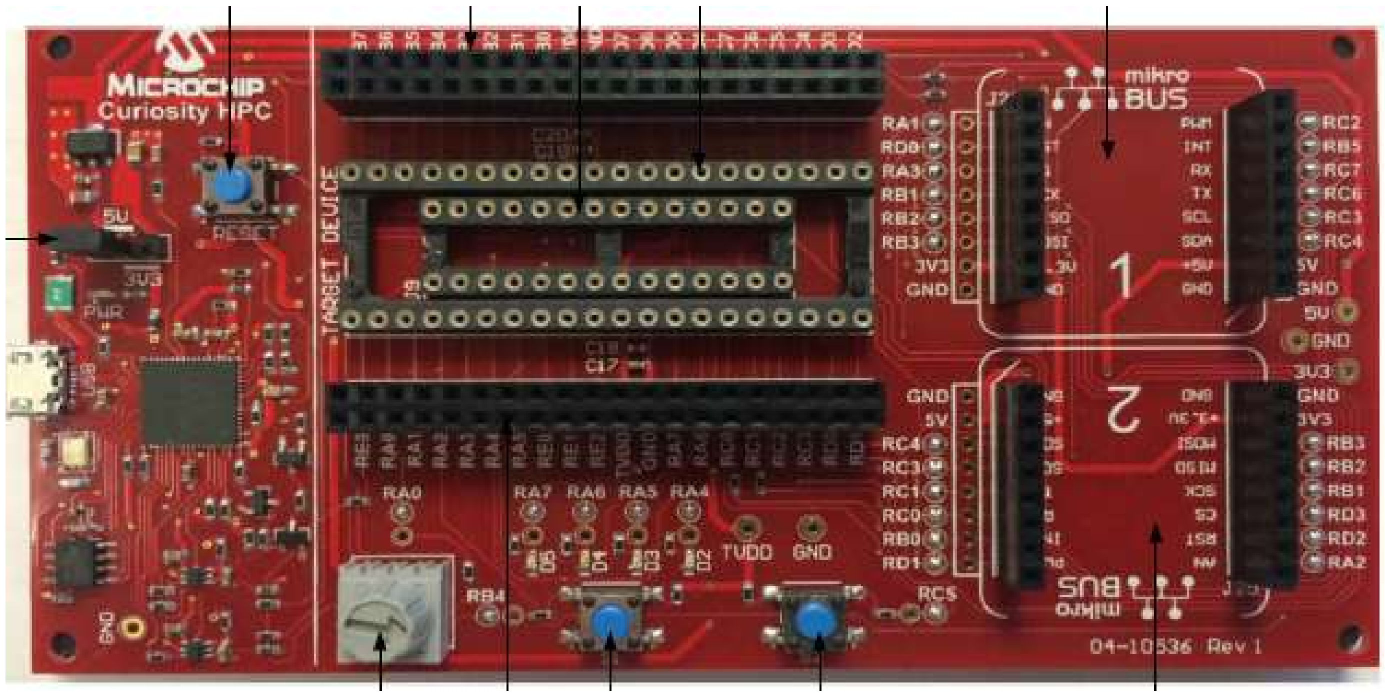
Writing to a port

- To write to a port, using the LATx or PORTx performs exactly the same action



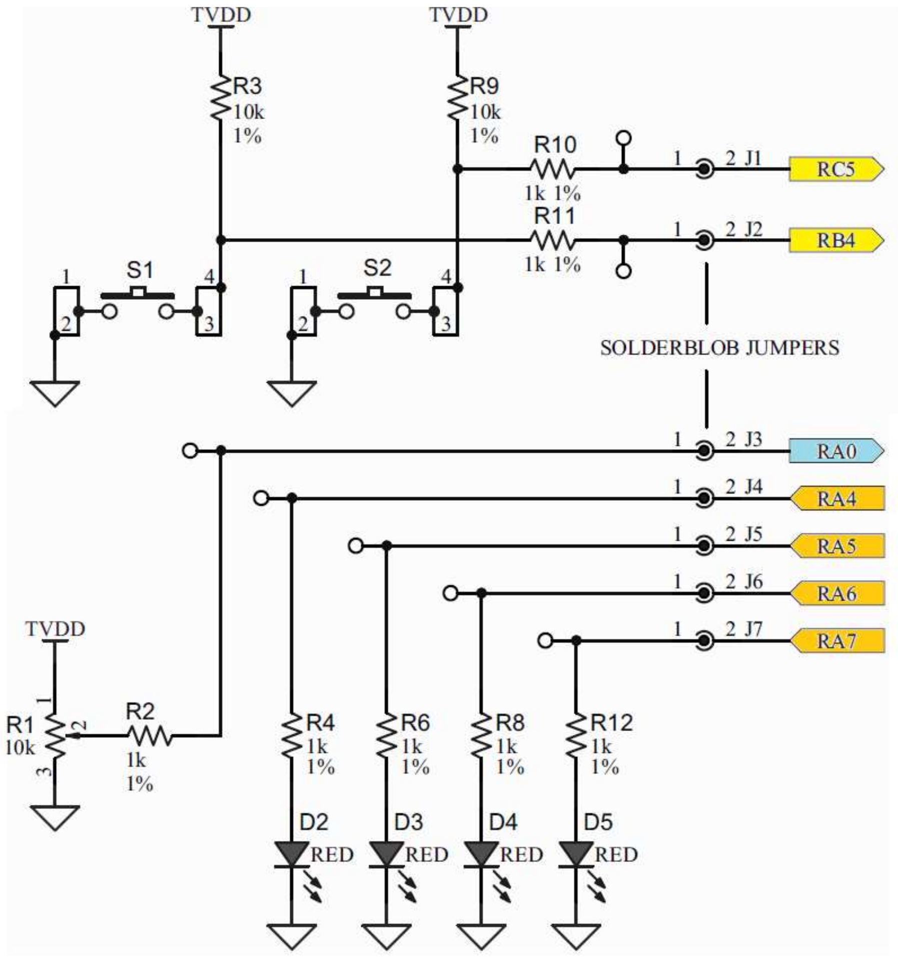
Note 1: I/O pins have diode protection to VDD and VSS.

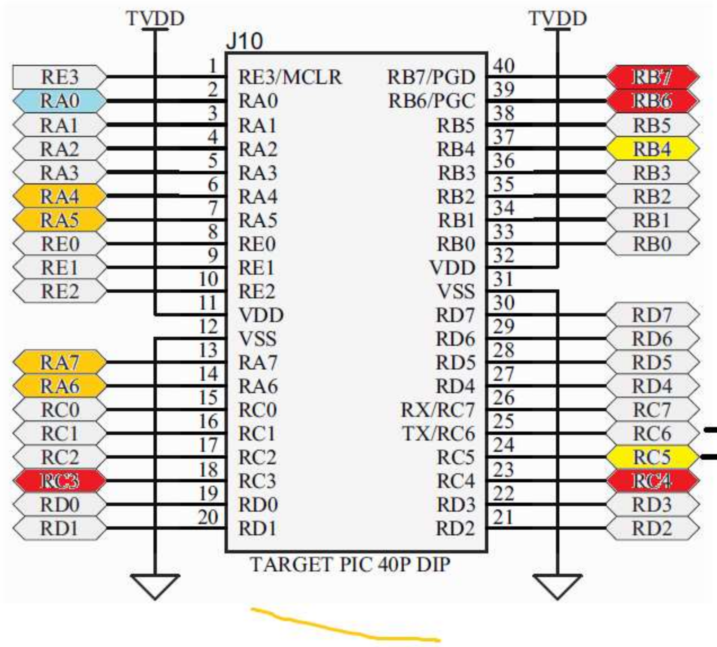
Available ports on the curiosity board



Board Manual

Available ports on the curiosity board





Reserved GPIO in the curiosity board

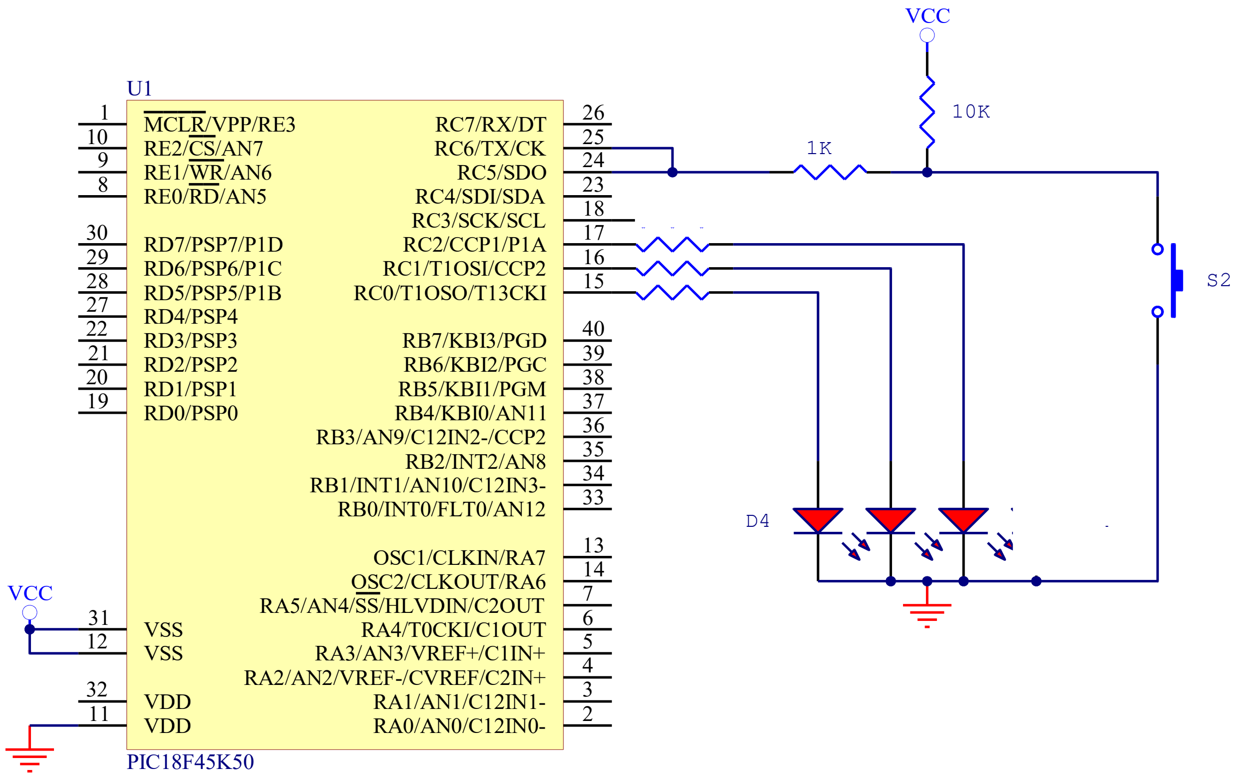
- RA0 is connected to potentiometer R1 for analog experiments
- RA4 is connected to LED D2 (active high)
- RA5 is connected to LED D3 (active high)
- RA6 is connected to LED D4 (active high)
- RA7 is connected to LED D6 (active high)
- RB4 is connected to push button S1 (active low)
- RC5 is connected to push button S2 (active low)**
- RB7 is the data signal for the debugger-programmer
- RB6 is the clock signals for the debugger-programmer
- RC3,RC4 and RC5 are USB signals

GPIO connected to switches, LED and pots can be reclaimed by removing the soldering blob jumpers, but avoid using the RB6-RB7 since you can interfere with the firmware download

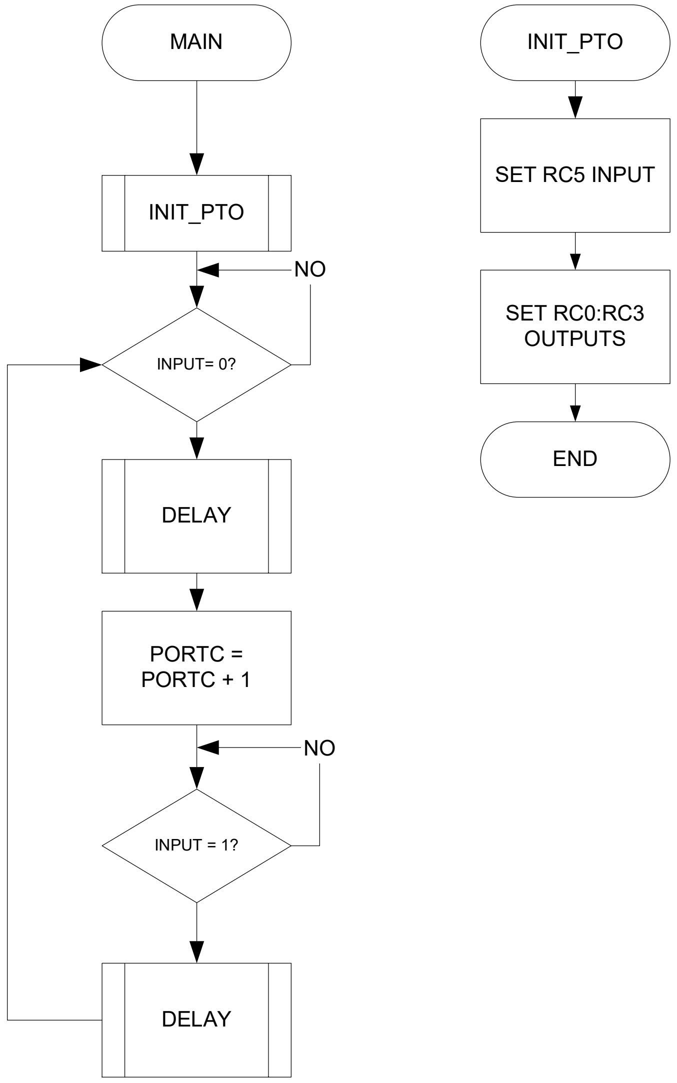
****To enable switch at RC5, you can place a jumper to RC6**

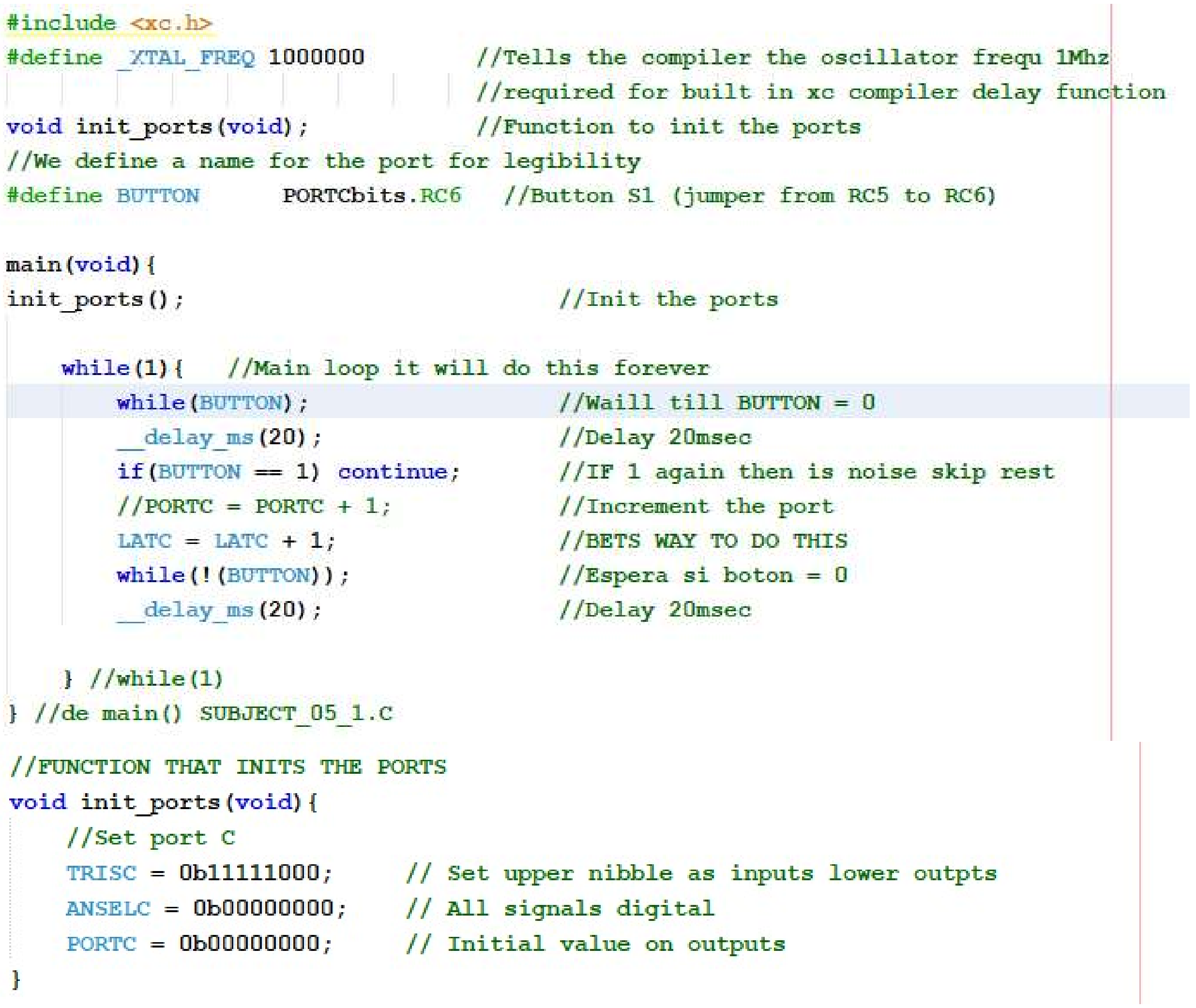
Example

- Implement a 3 bit binary counter using the following design. Each time we press the button, RC0:RC2 will increment by one. The button input must be "filtered" from debounces by firmware



Example


MAIN
INIT_PTO
NO
INPUT= 0?
DELAY
PORTC = PORTC + 1
NO
INPUT = 1?
DELAY
INIT_PTO
SET RC5 INPUT
SET RC0:RC3 OUTPUTS
END



```c
#include <xc.h>
#define _XTAL_FREQ 1000000        //Tells the compiler the oscillator frequ 1Mhz
                                  //required for built in xc compiler delay function
void init_ports(void);            //Function to init the ports
//We define a name for the port for legibility
#define BUTTON       PORTCbits.RC6   //Button S1 (jumper from RC5 to RC6)

main(void){
init_ports();                            //Init the ports

    while(1){   //Main loop it will do this forever
        while(BUTTON);                   //Waill till BUTTON = 0
        __delay_ms(20);                  //Delay 20msec
        if(BUTTON == 1) continue;        //IF 1 again then is noise skip rest
        //PORTC = PORTC + 1;             //Increment the port
        LATC = LATC + 1;                 //BETS WAY TO DO THIS
        while(!(BUTTON));                //Espera si boton = 0
        __delay_ms(20);                  //Delay 20msec

    } //while(1)
} //de main() SUBJECT_05_1.C
//FUNCTION THAT INITS THE PORTS
void init_ports(void){
    //Set port C
    TRISC = 0b11111000;     // Set upper nibble as inputs lower outpts
    ANSELC = 0b00000000;    // All signals digital
    PORTC = 0b00000000;     // Initial value on outputs
}
```

https://youtu.be/Xj5WgCN_IJ0