Electrónica Digital II

Santiago Rúa Pérez, PhD.

18 de septiembre de 2022

GENERAL PURPOSE INPUT-OUTPUT GPIO

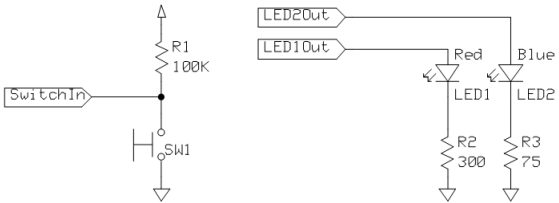
Objetivos

- Entender como utilizar los puertos de propósitos generales de entrada y salida.
- Entender los registros de acceso a los puertos de propósito generales.
- Comprender el funcionamiento del LCD y teclado matricial.

Panorama general

- Cómo podemos hacer para encender un led que dependa de la respuesta de un suiche?
- Entender como es la interfaz de los circuitos.
- Periféricos GPIO.
- LCD y teclado matricial.

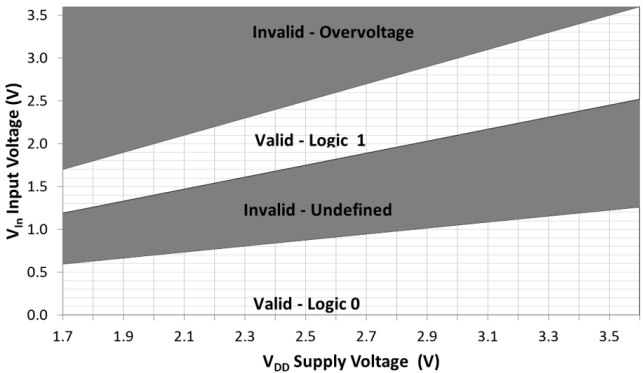
Conceptos básicos



- Fin: encender LED1 o LED2 de acuerdo al estado del suiche SW1.
- GPIO: General-purpose input and output
  - Entrada: el programa puede determinar si la señal de entrada es 1 o 0.
  - Salida: el programa pone un 1 o 0 en el pin.
- Pueden ser utilizado como interfaz con dispositivos externos.
  - Entrada: suiche
  - Salida: LEDs.

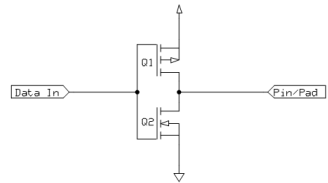
Entradas: Qué es un cero y qué es un uno?

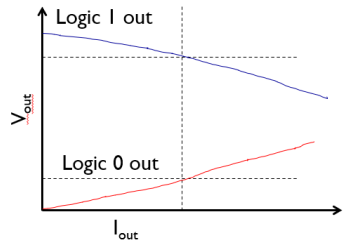
- El valor de la entrada de señal es determinada por el voltaje.
- Los límites de voltaje dependerán de la alimentación del sistema. Es una medida relativa.
- Alimentar con un valor mayor al de alimentación V_{DD} o inferior a GND puede generar un daño en el dispositivo.



Salida: Qué es un cero y qué es un uno?

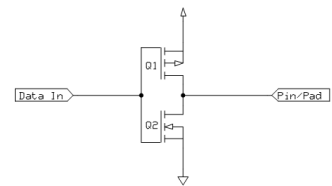
- Salida de voltajes nominales
  - 1: V_{DD}-0.5 V to V_{DD}
  - 0: 0 to 0.5 V.
- Nota: La salida de voltaje dependerá de la corriente demandada por el circuito.
  - Se necesita considerar la resistencia entre la fuente y el drenaje en el transistor.
  - Los valores anteriores solo se especifican para corrientes de menos de 5 mA (18 mA para pines de alta corriente) y V_{DD} mayor a 2.7 V

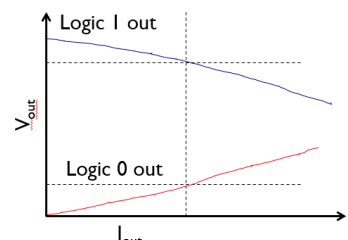




Ejemplo de salida: Conectar un LED

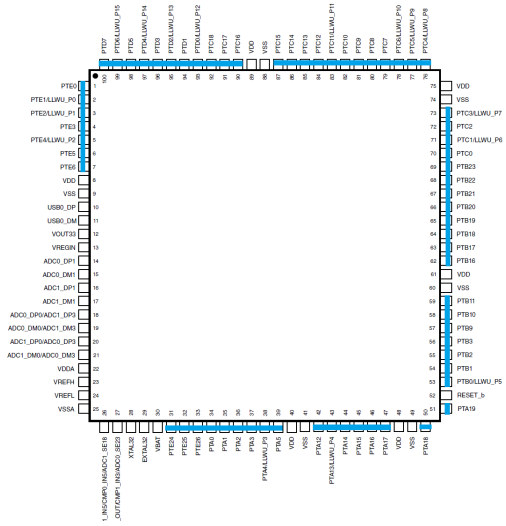
- Limitar la corriente a un nivel seguro para el LED y el pin del microcontrolador.
- Usar resistencia limitadora de corriente.
  - $R = (V_{DD} - V_{LED})/I_{LED}$
- Seleccionar la corriente del LED alrededor de $4\,\text{mA}$
- El voltaje en el led dependerá de su color.
  - Rojo: $\approx 1.8\,\text{V}$
  - Azul: $\approx 2.7\,\text{V}$
- Resolver R dado que la alimentacion es alrededor de $3\,\text{V}$.
  - Rojo: $300\,\Omega$
  - Azul: $75\,\Omega$





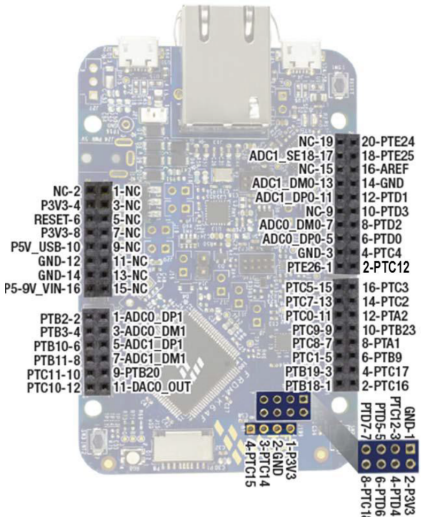
Hardware del GPIO

- Se tiene desde el puerto A al E.
- No todos los bits de los puertos estan disponibles.
- La cantidad dependerá del footprint del chip.

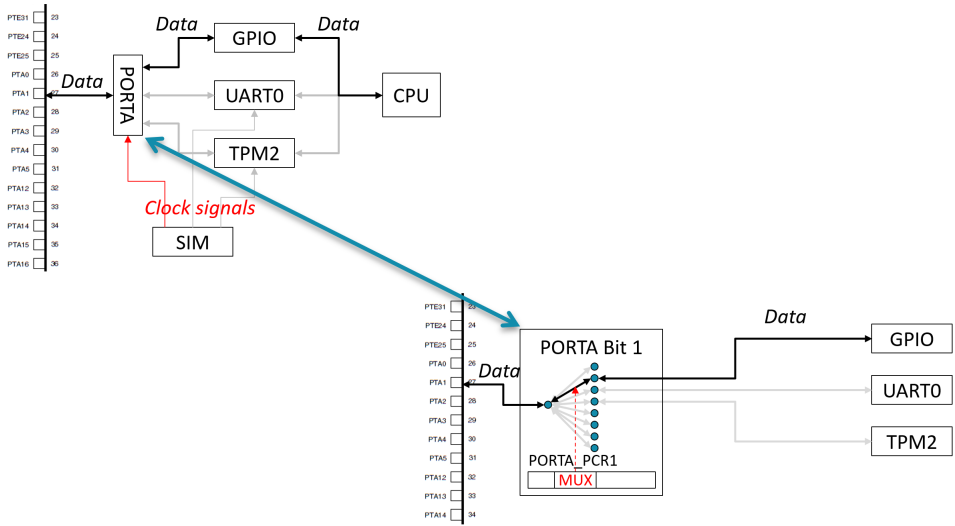


Hardware del GPIO - FRDM

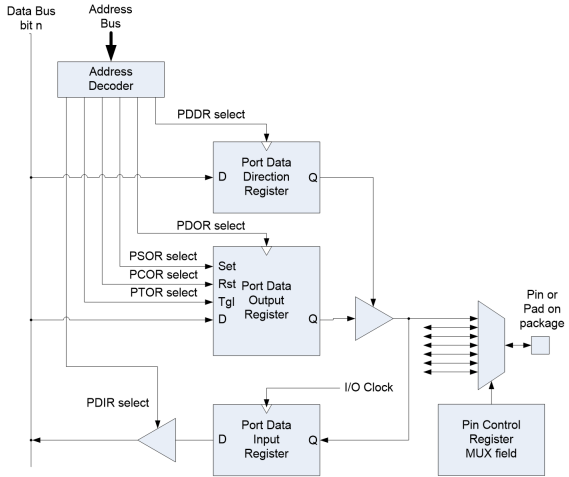
- No todos los pins estan accesibles.
- El led RGB esta conectado a 3 puertos.
  - Rojo: PTB22
  - Azul: PTB21
  - Verde: PTE26
- Los pulsadores también.
  - SW2: PTC6
  - SW3: PTA4



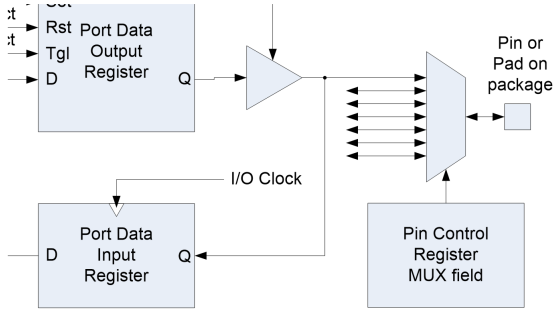
Pines Compartidos



Circuito del GPIO



Conectar una senal del GPIO al pin



- Se utiliza un multiplexador para incrementar la configurabilidad del sistema y sus capacidades.
- Cada pin configurable tiene un registro de control de pin, PCR (Pin Control Register)

Registro de control de pin PCRn

Address: Base address + 0h offset + (4d × i), where i=0d to 31d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0							ISF	0				IRQC			
W								w1c								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LK	0				MUX			0	DSE	ODE	PFE	0	SRE	PE	PS
W																
Reset	0	0	0	0	0	*	*	*	0	*	0	*	0	*	*	*

144 LQFP	144 MAP BGA	121 XFB GA	100 LQFP	Pin Name	Default	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5	ALT6	ALT7	ExPort
—	L5	L7	—	RTC_ WAKEUP_B	RTC_ WAKEUP_B	RTC_ WAKEUP_B								
—	—	B11	—	PTB12	DISABLED		PTB12	UART3_ RTS_b	FTM1_CH0	FTM0_CH4		FTM1_QD_ PHA		

El campo del MUX define las conexiones internas de la señal.

MUX (bit 10-8)	Configuración
000	Pin deshabilitado
001	ALT 1 - GPIO
010	ALT 2
011	ALT 3
100	ALT 4
101	ALT 5
110	ALT 6
111	ALT 7

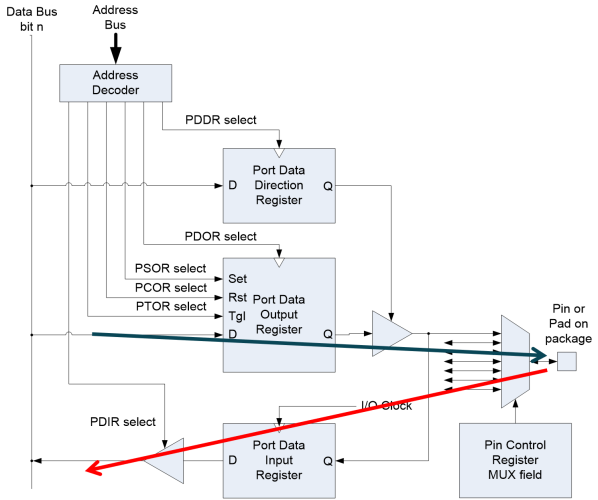
Registros control GPIO

Absolute address (hex)	Register name	Width (in bits)
400F_F000	Port Data Output Register (GPIOA_PDOR)	32
400F_F004	Port Set Output Register (GPIOA_PSOR)	32
400F_F008	Port Clear Output Register (GPIOA_PCOR)	32
400F_F00C	Port Toggle Output Register (GPIOA_PTOR)	32
400F_F010	Port Data Input Register (GPIOA_PDIR)	32
400F_F014	Port Data Direction Register (GPIOA_PDDR)	32

- Un conjunto de registro de control por puerto.
- Cada bit en un registro de control corresponde a un bit del puerto.

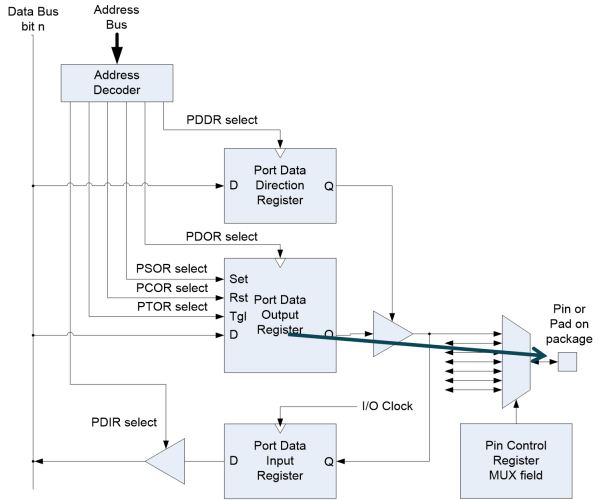
PDDR - Port Data Direction Register

- Cada bit puede ser configurado de manera independiente.
- Entrada: 0
- Salida: 1
- Un reset limpia el puerto y lo pone en cero.



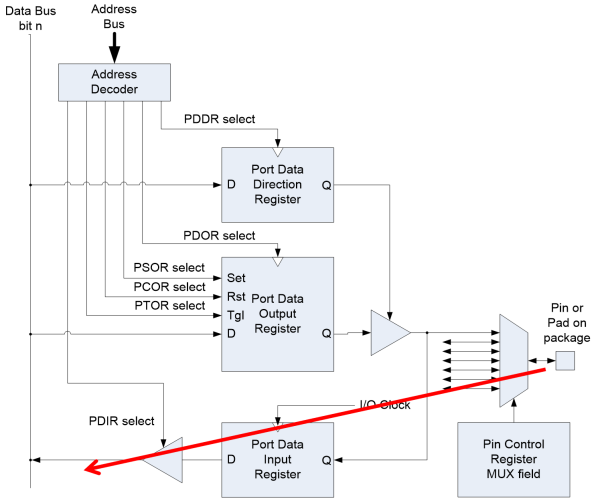
Escribiendo un puerto

- Directo: escriba el valor en el PDOR
- Toggle: escriba un uno 1 en PTOR
- Clear (a 0): escriba un 1 en PCOR
- Set (a 1): escriba un 1 en PSOR



Leyendo un puerto

- Leerlo desde el PDIR
- El bit mantiene el valor de lectura realizado.



Como configurar en software?

Pseudocódigo

```
1   // Make PTB21 and PTB22 outputs
2   set bits 21 and 22 of GPIOB_PDDR
3   // Make PTA4 input
4   clear bit 4 of GPIOA_PDDR
5   // Initialize the output data values: LED 1 off, LED 2 on
6   clear bit 21, set bit 22 of GPIOB_PDOR
7   // read switch, light LED accordingly
8   do forever {
9     if bit 4 of GPIOA_PDIR is 1 { // switch is not pressed, so light LED 2
10      set bit 22 of GPIOB_PDOR
11      clear bit 21 of GPIOB_PDOR
12    } else {          // switch is pressed, so light LED 1
13      set bit 21 of GPIOB_PDOR
14      clear bit 22 of GPIOB_PDOR
15    }
16  }
17
```

Estilo de codificacion y acceso a bits

- Es facil cometer errores poniendo binarios o hexadecimales
  - Para prender el bit 13 y 19 se usaria 0000 0000 0000 1000 0010 0000 0000 0000 or 0x00082000
- Use el valor literal para desplazar los bits

  n = (1UL << 19) | (1UL << 13);
- Realice macros para los pines

  #define GREEN_LED_POS (19)
  #define YELLOW_LED_POS (13)
  n = (1UL << GREEN_LED_POS) | (1UL << YELLOW_LED_POS);
- Cree macros para el desplazamiento

  #define MASK(x) (1UL << (x))
  n = MASK(GREEN_LED_POS) | MASK(YELLOW_LED_POS);

Usando las máscaras

- Sobrescribir todo el registro con el valor de la máscara.

  n = MASK(foo);

- Poner 1 en n todos los valores que correspondan con 1 en la máscara, sin modificar los otros.

  n |= MASK(foo);

- Poner 0 en n todos los valores que correspondan con 1 en la máscara, sin modificar los otros.

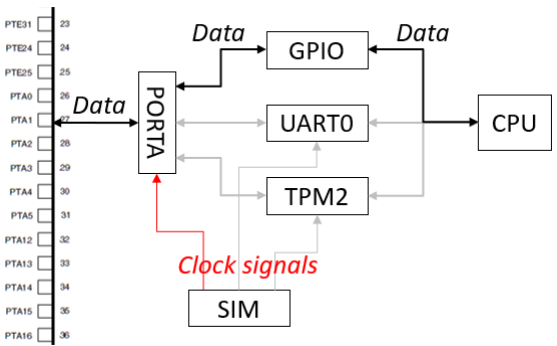
  n &= ∼MASK(foo);

- Nos ayudamos de CMSIS para acceder a los puertos. Analizar la hoja de datos la estructura GPIO. En donde PTx se define como un puntero a dicha estructura.

Ejemplo codificado

```c
#include "MK64F12.h"

#define LED1_POS (21)
#define LED2_POS (22)
#define SW1_POS (4)
#define MASK(x) (1UL << (x))
#define MASK(x) (1UL << (x))

int main(void){
  // Enable Clock Port A and B
  SIM->SCGC5 |= SIM_SCGC5_PORTA_MASK |
      SIM_SCGC5_PORTB_MASK;

  // Make 3 pins GPIO
  PORTB->PCR[LED1_POS] &= ~PORT_PCR_MUX_MASK;
  PORTB->PCR[LED1_POS] |= PORT_PCR_MUX(1);
  PORTB->PCR[LED2_POS] &= ~PORT_PCR_MUX_MASK;
  PORTB->PCR[LED2_POS] |= PORT_PCR_MUX(1);
  PORTA->PCR[SW1_POS] &= ~PORT_PCR_MUX_MASK;
  PORTA->PCR[SW1_POS] |= PORT_PCR_MUX(1);

  PTB->PDDR |= MASK(LED1_POS) | MASK(LED2_POS); //
      set bits to outputs
  PTA->PDDR &= ~MASK(SW1_POS); // clear switch bit
      to input

  PTB->PDOR = MASK(LED2_POS);  // turn on LED1,
     turn off LED2

  while (1) {
    if (PTA->PDIR & MASK(SW1_POS)) { // switch is
      not pressed, so light LED 2
      PTB->PDOR = MASK(LED2_POS);
    } else {                        // switch is
      pressed, so light LED 1
      PTB->PDOR = MASK(LED1_POS);
    }
  }
}
```

Chequeo para el uso de pines



- Módulo SIM le habilita el clock a
  - Módulo de puertos.
  - Algunos periféricos (puede o no ser puerto)
- Configurar el pin de control MUX
- Configurar el periférico
- Configurar la interrupción del sistema si es del caso.

Lógica para el clock

Bit	Puerto
13	PORTE
12	PORTD
11	PORTC
10	PORTB
9	PORTA

- Se necesita habilitar el clock al módulo GPIO.
- Por defecto vienen desactualizados para conservar energía.
- Si se escribe a un GPIO desactivado, sale error por hardware.
- El registro de control SIM_SCGC5 contiene el clock de los GPIO
- Habilitar el clock al puerto A.

  SIM->SCGC5 |= (1UL << 9);
- Utilizando las definiciones del Header.

  SIM->SCGC5 |= SIM_SCGC5_PORTA_MASK;

Soporte del CMSIS para el PCR

- El devive.h define la estructura PORT_type con los campos incluyendo PCR como un vector de 32 enteros.

```
/** PORT - Register Layout Typedef */
typedef struct {
  __IO uint32_t PCR[32];      /** Pin Control Register n, array offset: 0x0, array step: 0x4 */
  __O  uint32_t GPCLR; /** Global Pin Control Low Register, offset: 0x80 */
  __O  uint32_t GPCHR; /** Global Pin Control High Register, offset: 0x84 */
       uint8_t RESERVED_0[24];
  __IO uint32_t ISFR;  /** Interrupt Status Flag Register, offset: 0xA0 */
} PORT_Type;
```

Soporte del CMSIS para el PCR

- Header file defines pointers to PORT_Type registers

```
/* PORT - Peripheral instance base addresses */
/** Peripheral PORTA base address */
#define PORTA_BASE      (0x40049000u)
/** Peripheral PORTA base pointer */
#define PORTA           ((PORT_Type *)PORTA_BASE)
```

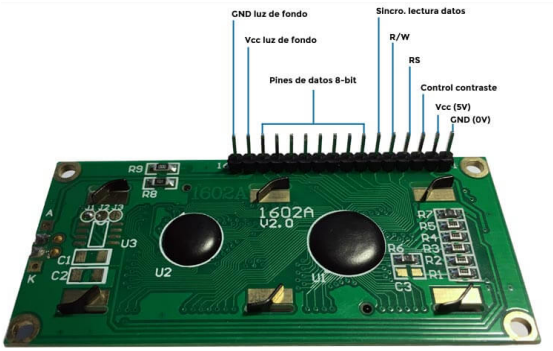
- Also defines macros and constants

```
#define PORT_PCR_MUX_MASK     0x700u
#define PORT_PCR_MUX_SHIFT    8
#define PORT_PCR_MUX(x)
(((uint32_t)(((uint32_t)(x))<<PORT_PCR_MUX_SHIFT)) &PORT_PCR_MUX_MASK)
```

Ejemplo LCD

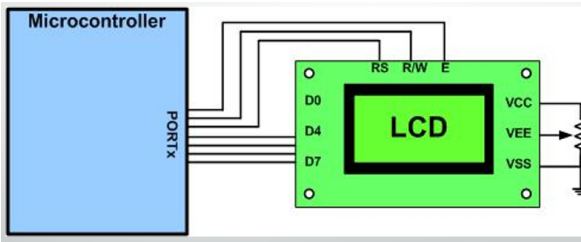
- Pantalla de cristal liquido
- Hay de diferentes tamaños: 16x2, 16x4, etc.
- Puede comunicarse con 4 o 8-bits

Pin	Símbolo	Descripción
1	Vss	Tierra
2	Vcc	Fuente
3	Vee	Contraste
4	RS	0 com, 1 dat
5	RW	0 escri, 1 lect
6	E	enable
7-14	D0-D7	Bus de datos
15	A	Ánodo
16	K	Cátodo



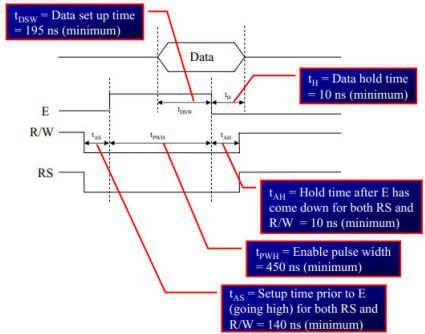
Ejemplo LCD - Comando y conexiones

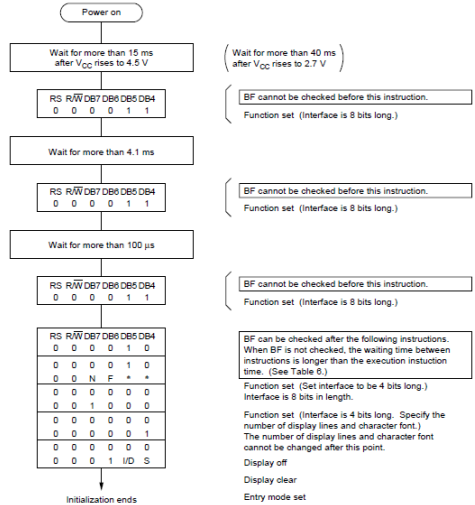
Codigo (Hex)	Comando
1	Limpia el display
2	Retorna cursor al home
6	Incrementa el cursor
F	Display on, y cursor parpadea
80	Inicia cursor en primera linea
C0	Cursor en segunda linea
28	2 lineas, 5x7 car, 4-bit



- Inicializar el LCD antes de usarlo.
- Crear funciones para envió de datos y envió de comandos.
- Respetar diagramas de tiempo.

Ejemplo LCD - Diagrama de tiempo e inicialización





Ejemplo LCD - Creando la Libreria

- Inicializar GPIO a utilizar
- Crear una función de retraso en ms. Aprox.
- Crear una función que dado un char me active los cuatros pines seleccionados.
- Crear la funcion de envio de datos.
- Crear la funcion de envio de comandos.
- Crear funciones auxiliares tales como: escribir una cadena, limpiar la pantalla, poner el cursor en una posición dada.

```
1  #define D7  16  /* PTC16 */
2  #define D6  17  /* PTC17 */
3  #define D5  9 /* PTB9 */
4  #define D4  1 /* PTA1 */
5  #define RS  23  /* PTB23 */
6  #define EN  2 /* PTA2 */
7  #define MASK(x) (1UL << (x))
8
```

El pin RW se pone a tierra ya que siempre se esta escribiendo la pantalla.

Ejemplo LCD - Software

- Creando la función que pone el dato en el bus.

```
1   void LcdDataBus(unsigned char Data){
2     if(Data & 1)
3     PTA->PSOR|= MASK(D4);
4     else
5     PTA->PCOR|= MASK(D4);
6     if(Data & 2)
7     PTB->PSOR|= MASK(D5);
8     else
9     PTB->PCOR|= MASK(D5);
10    if(Data & 4)
11    PTC->PSOR|= MASK(D6);
12    else
13    PTC->PCOR|= MASK(D6);
14    if(Data & 8)
15    PTC->PSOR|= MASK(D7);
16    else
17    PTC->PCOR|= MASK(D7);
18  }
19
```

Ejemplo LCD - Software

- Creando la función para escribir comandos.

```
1  void Lcd_CmdWrite(unsigned char cmd)
2  {
3    LcdDataBus((cmd>>4) & 0x0F);      //Send higher nibble
4    PTB->PCOR = MASK(RS);          // Send LOW pulse on RS pin for selecting Command register
5    PTA->PSOR = MASK(EN);         // Generate a Low-to-High pulse on EN pin
6    delayMs(4);
7    PTA->PCOR = MASK(EN);         // Generate a High-to-low pulse on EN pin
8
9
10   delayMs(4);
11
12   LcdDataBus(cmd & 0x0F); //Send Lower nibble
13   PTB->PCOR = MASK(RS);          // Send LOW pulse on RS pin for selecting Command register
14   PTA->PSOR = MASK(EN);         // Generate a Low-to-High pulse on EN pin
15   delayMs(4);
16   PTA->PCOR = MASK(EN);         // Generate a High-to-low pulse on EN pin
17
18   delayMs(5);
19 }
20
```

Ejemplo LCD - Software

- Creando la función para inicializar el LCD.

```
1     delayMs(30);                /* initialization sequence */
2   Lcd_CmdWrite(0x33);          /* init */
3   Lcd_CmdWrite(0x32);          /* init */
4
5   Lcd_CmdWrite(0x28);          /* set 4-bit data, 2-line, 5x7 font */
6   Lcd_CmdWrite(0x0C);          /* move cursor right */
7   Lcd_CmdWrite(0x01);          /* clear screen, move cursor to home */
8   Lcd_CmdWrite(0x06);          /* turn on display, cursor blinking */
9
```

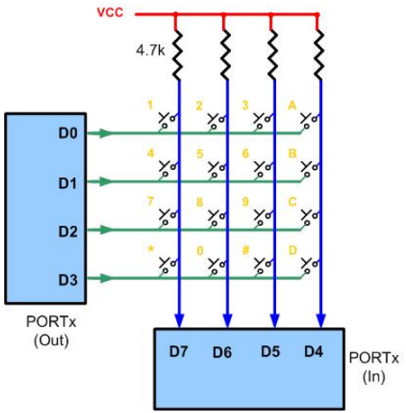
Laboratorio 1 (15 %)

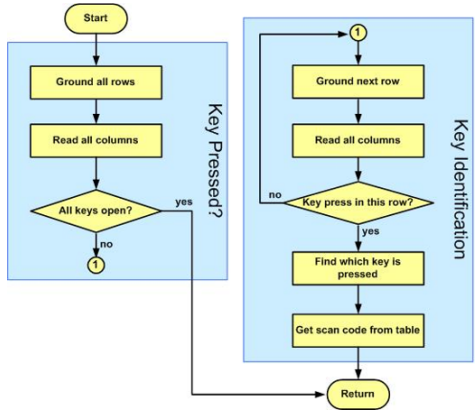
Hacer un programa en C que se pueda escribir cadenas en el LCD.

- Recuerde inicializar el clock de cada puerto utilizado.
- Recuerde inicializar el registro de control PCR para los puertos utilizados como GPIO.
- Cree una función que envié datos al LCD.
- Cree una función que reciba una cadena y la muestre en el LCD.
- Cree una función que ponga el cursor en cualquier parte de la pantalla.

Laboratorio - Opcional

Hacer un programa en C lea un teclado matricial de 4x4 o 5x5.





GENERAL PURPOSE INPUT-OUTPUT GPIO

GRACIAS