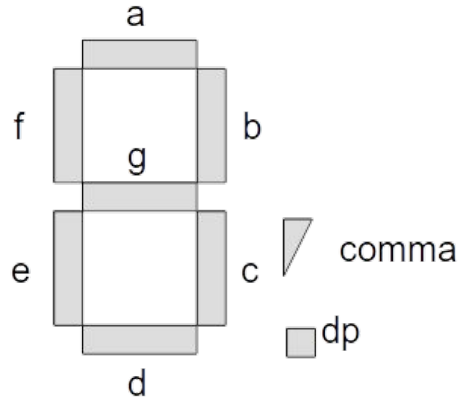
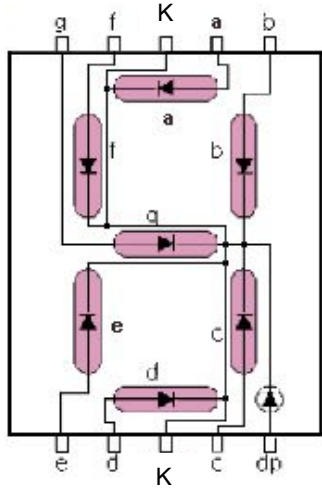


Periféricos elementales: Display 7 segmentos

Informática II - R2004
2021

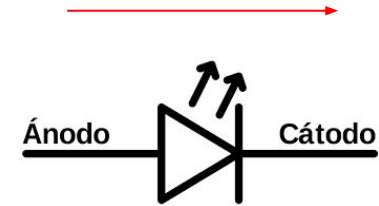
Displays de 7 segmentos

Son displays formados por 7 elementos luminosos (LEDs) más un punto (Digital Point, o dp). Para formar los diferentes números se prenden los segmentos correspondientes y para tener varios dígitos se ponen varios displays.



Funcionamiento de un led

Sentido de circulación de la corriente



En los leds hay un único sentido de circulación de la corriente que permite prenderlo. Para ello hay que colocar tensión en ánodo (un 1) y el cátodo a tierra (un 0).

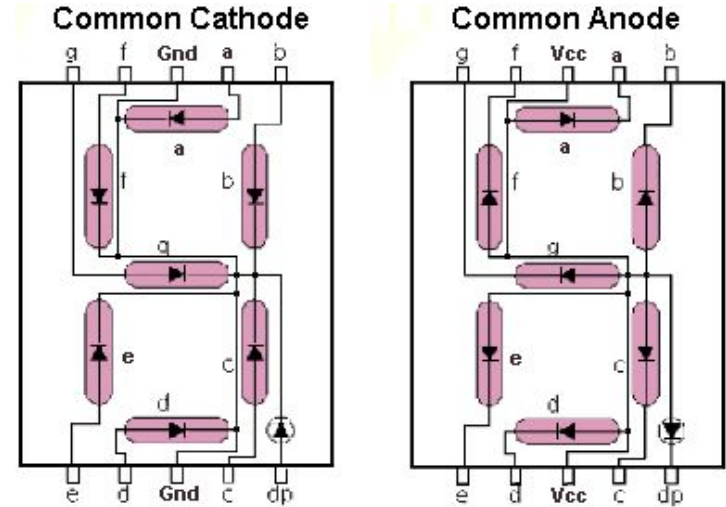
En cualquier otra combinación no se prenderá el led.

Display 7 segmentos

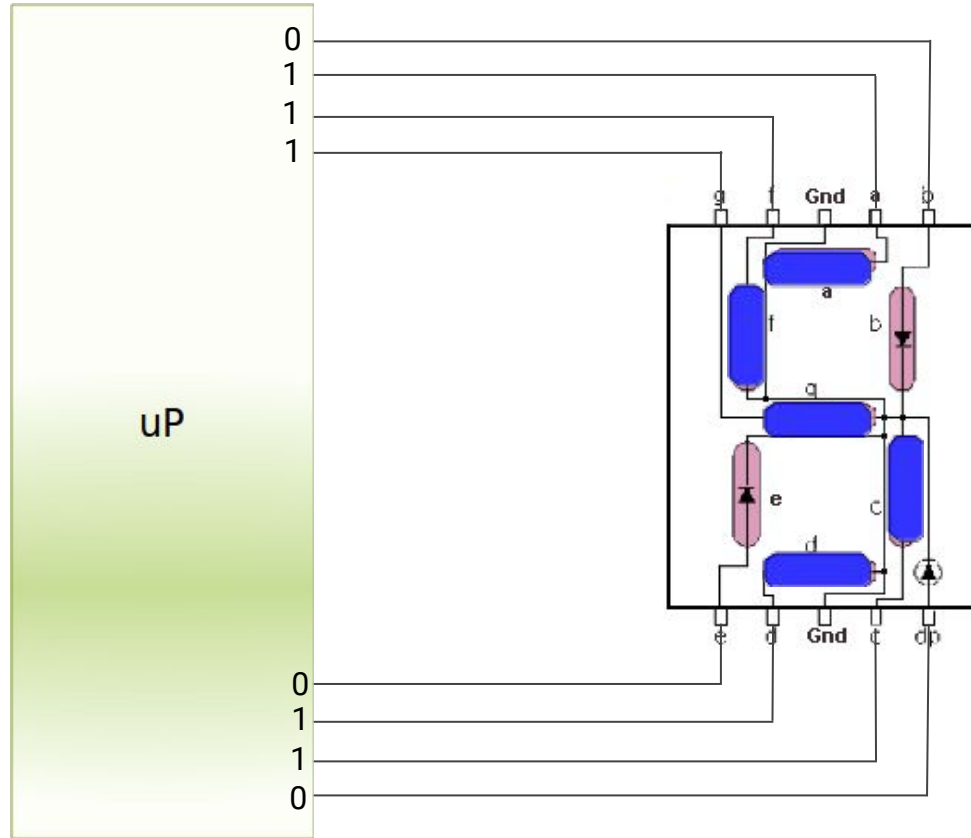
De acuerdo a cómo estén conectados estos LEDs, existen los displays de ánodo común o cátodo común.

Los display de cátodo común tienen todos los cátodos conectados entre sí, estos debe ponerse a tierra (0) y para prender cada segmento se debe poner a Vcc (1).

Los displays de ánodo común tienen todos los ánodos conectados entre sí, estos deben ponerse a Vcc (1) y para prender cada segmento se debe poner a tierra (0).



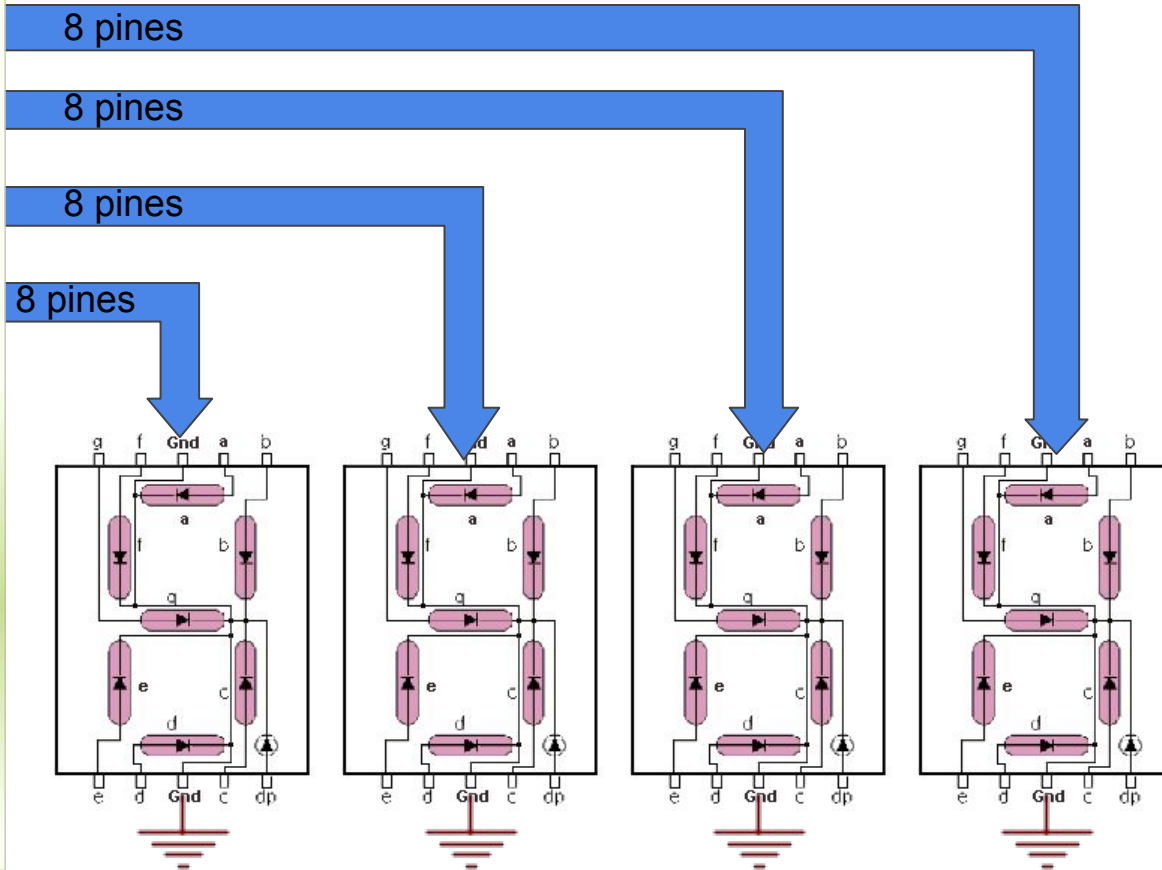
Display 7 segmentos



En este caso un display de cátodo común, prendo los segmentos que deseo colocando un 1 en los pines correspondientes.

Displays - Mostrando un valor

uP

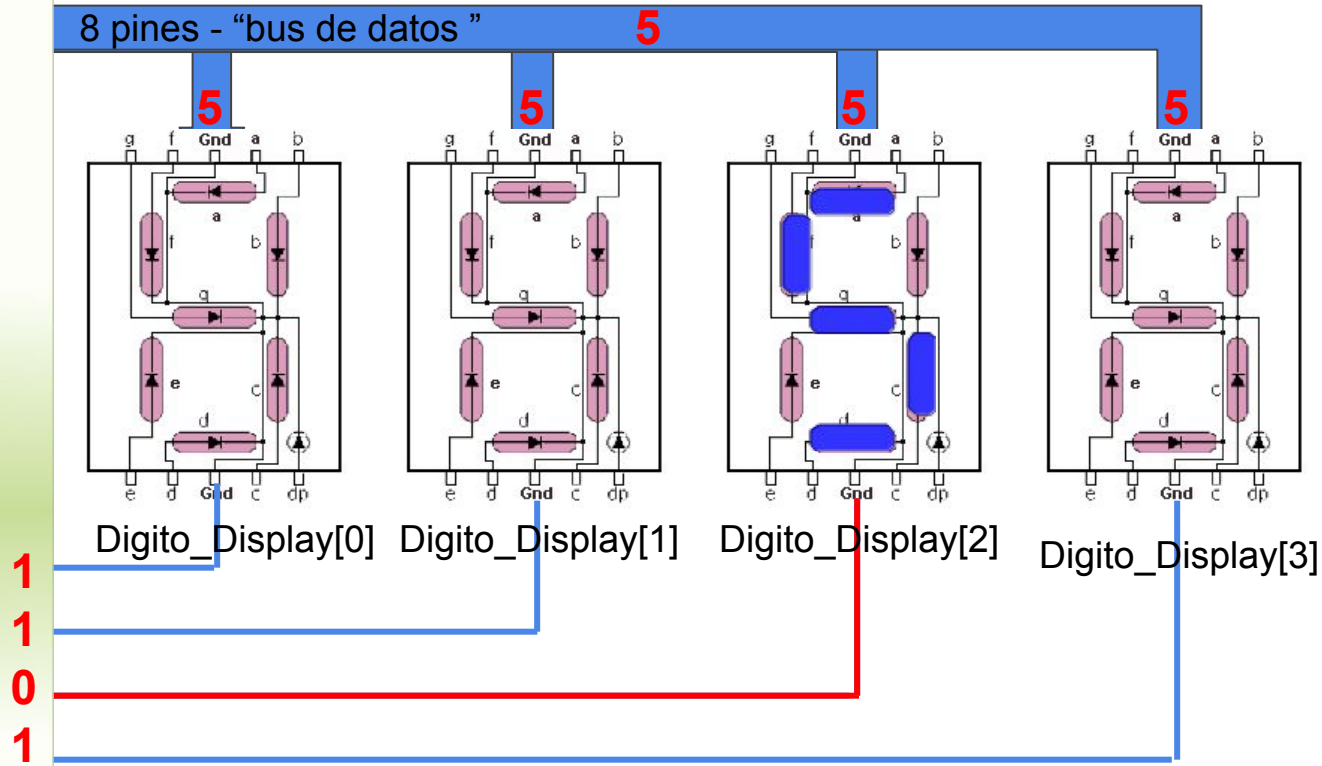


En general deseo tener más de un dígito para mostrar mi información.

Si quisiera tener 4 dígitos necesitaría $8 \times 4 = 32$ pines, esto es ocupar muchos pines para los displays, y cada vez que queremos agregar un dígito tenemos que sumar 8 pines más.

Displays - Lógica de “selección” del display

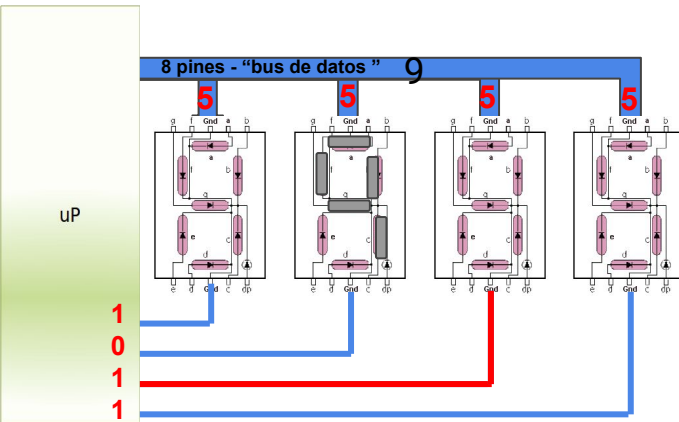
uP



Para no ocupar tantos pines por cada display, se conectan juntos todos los pines para prender los segmentos, y por medio del “común” se habilita o deshabilita cual de los displays muestra el dato.

Barrido de displays

Al tener todos pines para prender los segmentos conectados juntos tengo que prender un display a la vez para mostrar el dato que quiero en cada uno de ellos.



1956

Si prendo de a uno a la vez pero lo hago muy rápido y cíclicamente nuestro ojo va a ver a todos prendidos a la vez.

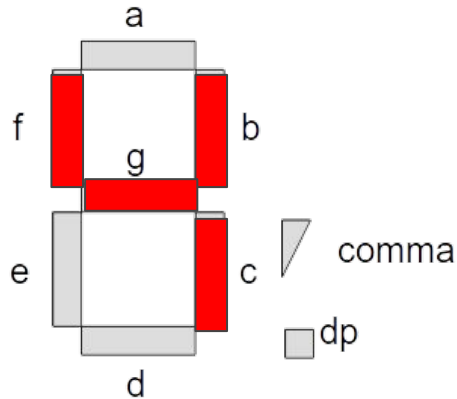
Fenómeno visual descubierto por el científico belga Joseph Plateau (1801-1883) que demuestra cómo una imagen permanece en la retina humana una décima de segundo después de desaparecer completamente.

Plateau descubrió que nuestro ojo ve con una cadencia de 10 imágenes por segundo, que nosotros no vemos como independientes gracias a la persistencia visual.

“Barrido” de displays

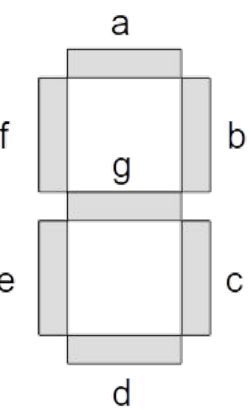
1. Desactivo todos los displays
2. Pongo el dato a mostrar en el display en el bus
3. Activo el display correspondiente

... y ¿cuál es el dato que tengo que poner en el bus? ¿Qué segmentos tengo que prender para mostrar cada número?

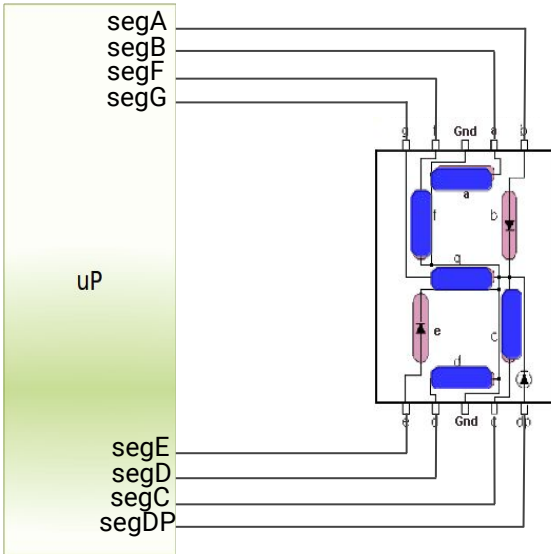


g	f	e	d	c	b	a	Num
0	0	0	0	1	1	0	1
1	0	1	1	0	1	1	2
1	0	0	1	1	1	1	3
1	1	0	0	1	1	0	4

¿Cómo hago todo esto?

	g	f	e	d	c	b	a	Num		
	0	1	1	1	1	1	1	0	→	0x3f,
	0	0	0	0	1	1	0	1	→	0x06,
	1	0	1	1	0	1	1	2	→	0x5b,
	1	0	0	1	1	1	1	3	→	0x4f,
	1	1	0	0	1	1	0	4	→	0x66,
	1	1	0	1	1	0	1	5	→	0x6d,
	1	1	1	1	1	0	1	6	→	0x7d,
	0	0	0	0	1	1	1	7	→	0x07,
	1	1	1	1	1	1	1	8	→	0x7f,
	1	1	0	0	1	1	1	9	→	0x67};

¿Como hago todo esto?



Si quiero mostrar el número 5, ingreso a la posición 5 de la tabla

Tabla_Digitos_BCD_7segmentos[5]

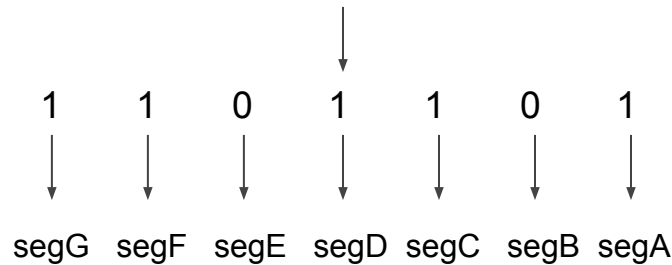
1	1	0	1	1	0	1
↓	↓	↓	↓	↓	↓	↓
segG	segF	segE	segD	segC	segB	segA

Cada uno de los segmentos se encuentran conectados a un
PUERTO y un PIN.

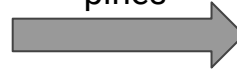
```
#define segA PUERTO0,22
```

¿Como hago todo esto?

```
aux = Tabla_Digitos_BCD_7segmentos[ 5 ]
```

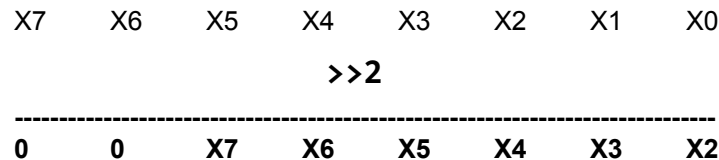


Para setear los
datos en los
pines

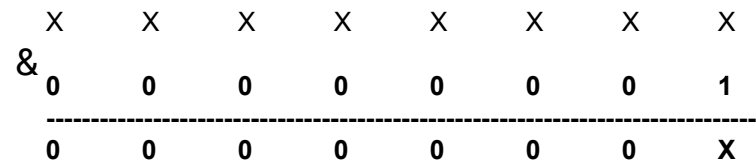


```
setPIN(segA, aux & 0x01 )  
setPIN(segB, (aux>>1) & 0x01 )  
setPIN(segC, (aux>>2) & 0x01 )  
setPIN(segD, (aux>>3) & 0x01 )  
.  
.  
.  
segPIN(segG, (aux>>7) & 0x01 )
```

Desplazando los bits llevo al bit menos
significativo el que me interesa para después
aplicarle la & con 0x01



Realizando un & con 0x01 me quedo con el
valor del bit menos significativo y el resto
queda en 0.



Barrido de displays

1. Desactivo todos los displays
2. Pongo el dato a mostrar en el display en el bus
3. Activo el display correspondiente

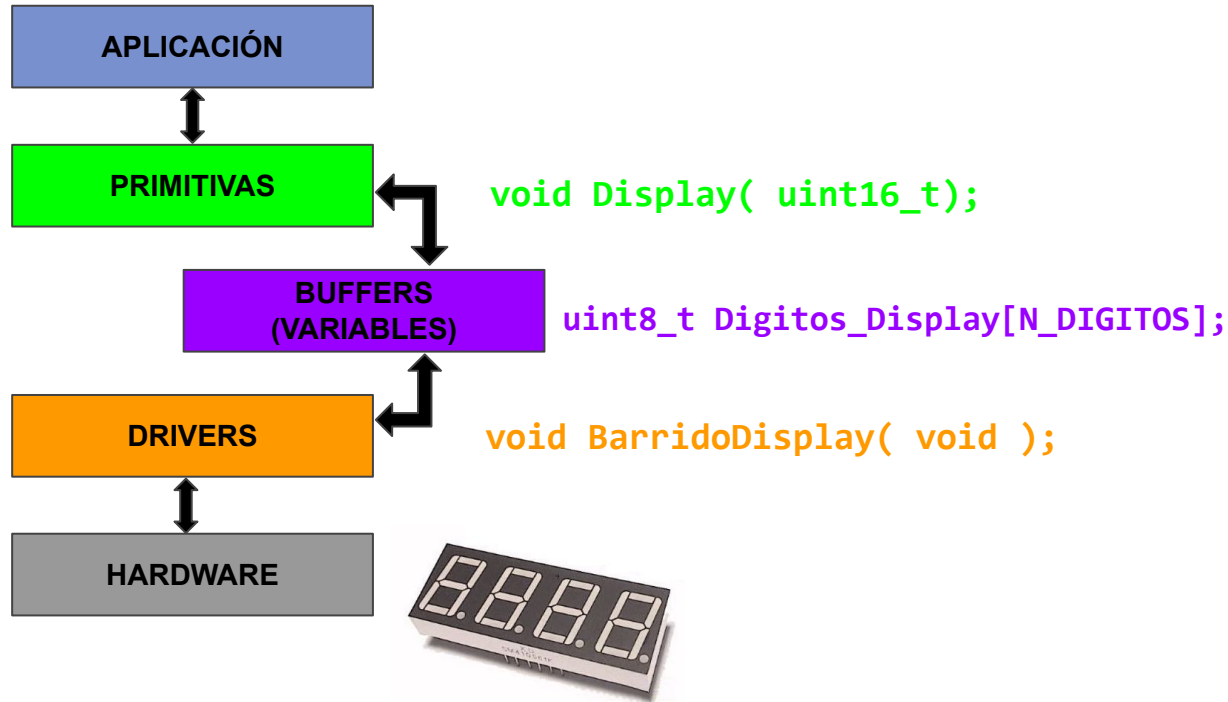
... ¿qué pasa si no sigo este orden?

Aparecen “Fantasmas”: Si la secuencia de apagado y encendido de los dígitos no se realiza como fue indicado, es posible que durante un breve tiempo aparezca un valor no deseado en los dígitos, lo que en la práctica se verá como el valor con una luminosidad más tenue.

¿Cada cuanto tiempo tengo que prender cada uno de los displays?

La diferencia entre que el ojo humano vea encender y apagarse los dígitos o vea un valor continuo depende de la frecuencia con que refresco los displays. Aproximadamente debería refrescar todos los displays en 30mseg. Si me demoro más tiempo veré un “Parpadeo” en lugar de todos encendidos a la vez.

Diagrama en Capas driver



La APLICACIÓN hace uso de la primitiva para “acceder” al recurso de hardware.

La PRIMITIVA carga el buffer independizando a los datos del hardware específico.

El BUFFER vincula la primitiva con el driver independizando las funciones.

El DRIVER lee el buffer y maneja el hardware conociendo las especificaciones del mismo. Esta función es ejecutada por el SysTick.

Invocando el Driver

- Se debe llamar constantemente.
- Para que el ojo humano visualice de forma correcta los displays se deben barrer TODOS los displays en 30ms.

PARA QUE LOS TIEMPOS SEAN ESTABLES EL DRIVER DE TECLAS SERÁ INVOCADO EN EL SYSTICK

```
void SysTick_Handler(void)
{
    DebounceEntradas( );

    BarridoDisplay( );

    RefrescoSalidas( );
    AnalizarTimers( );
    EscrituraLCD( );
    DriverTecladoSW ( );
    RefrescoBuzzer( );
}
```

Función BarridoDisplay - Driver

```
uint8_t auxiliar;  
static uint8_t Indice_Display = 0;
```

```
SetPIN (dg0,OFF);  
SetPIN (dg1,OFF);  
SetPIN (dg2,OFF);  
SetPIN (dg3,OFF);  
SetPIN (dg4,OFF);  
SetPIN (dg5,OFF);
```

Desactivo todos los displays

Convierto el valor a mostrar a 7 segmentos

```
auxiliar = Tabla_Digitos_BCD_7seg[ Digito_Display[Indice_Display] ];
```

```
SetPIN (seg_a,(auxiliar & (uint8_t)0x01));  
SetPIN (seg_b,(( auxiliar >> 1 ) & (uint8_t)0x01));  
SetPIN (seg_c,(( auxiliar >> 2 ) & (uint8_t)0x01));  
SetPIN (seg_d,(( auxiliar >> 3 ) & (uint8_t)0x01));  
SetPIN (seg_e,(( auxiliar >> 4 ) & (uint8_t)0x01));  
SetPIN (seg_f,(( auxiliar >> 5 ) & (uint8_t)0x01));  
SetPIN (seg_g,(( auxiliar >> 6 ) & (uint8_t)0x01));  
SetPIN (seg_dp,(( auxiliar >> 7 ) & (uint8_t)0x01));
```

Pongo el valor en el bus de datos



Función BarridoDisplay - Driver

```
switch( Indice_Display)
{
    case DIGITO_0:
        SetPIN (dg0,ON); break;
    case DIGITO_1:
        SetPIN (dg1,ON); break;
    case DIGITO_2:
        SetPIN (dg2,ON); break;
    case DIGITO_3:
        SetPIN (dg3,ON); break;
    case DIGITO_4:
        SetPIN (dg4,ON); break;
    case DIGITO_5:
        SetPIN (dg5,ON); break;
}
```

Enciende el display correspondiente

```
// Incremento el indice del display
Indice_Display++;
Indice_Display %= N_DIGITOS;
```

Indico cual es el próximo
dígito a mostrar

Carga de dato para barrido - Primitiva

```
void Display(unsigned int Val)
{
    unsigned char a ;

    // Convierto a 7 Seg.
    for(a = (CANT_DIGITOS - 1) ; a ; a-- )
    {
        Digito_Display[a] =
            Val % 10;
        Val /= 10;
    }
}
```

buffer para comunicación
driver - primitiva

```
volatile uint8_t Digito_Display[ N_DIGITOS ];
```

Val

1	5	6	7
↓	↓	↓	↓
DSP0	DSP1	DSP2	DSP3

Cada display se corresponde con la
posición del buffer Digito_Display

$1567 \% 10 = 7$	→	Posición 3 del buffer
$1563 / 10 = 156$		
$156 \% 10 = 6$	→	Posición 2 del buffer
$156 / 10 = 15$		
$15 \% 10 = 5$	→	Posición 1 del buffer
$15 / 10 = 1$		
$1 \% 10 = 1$	→	Posición 0 del buffer

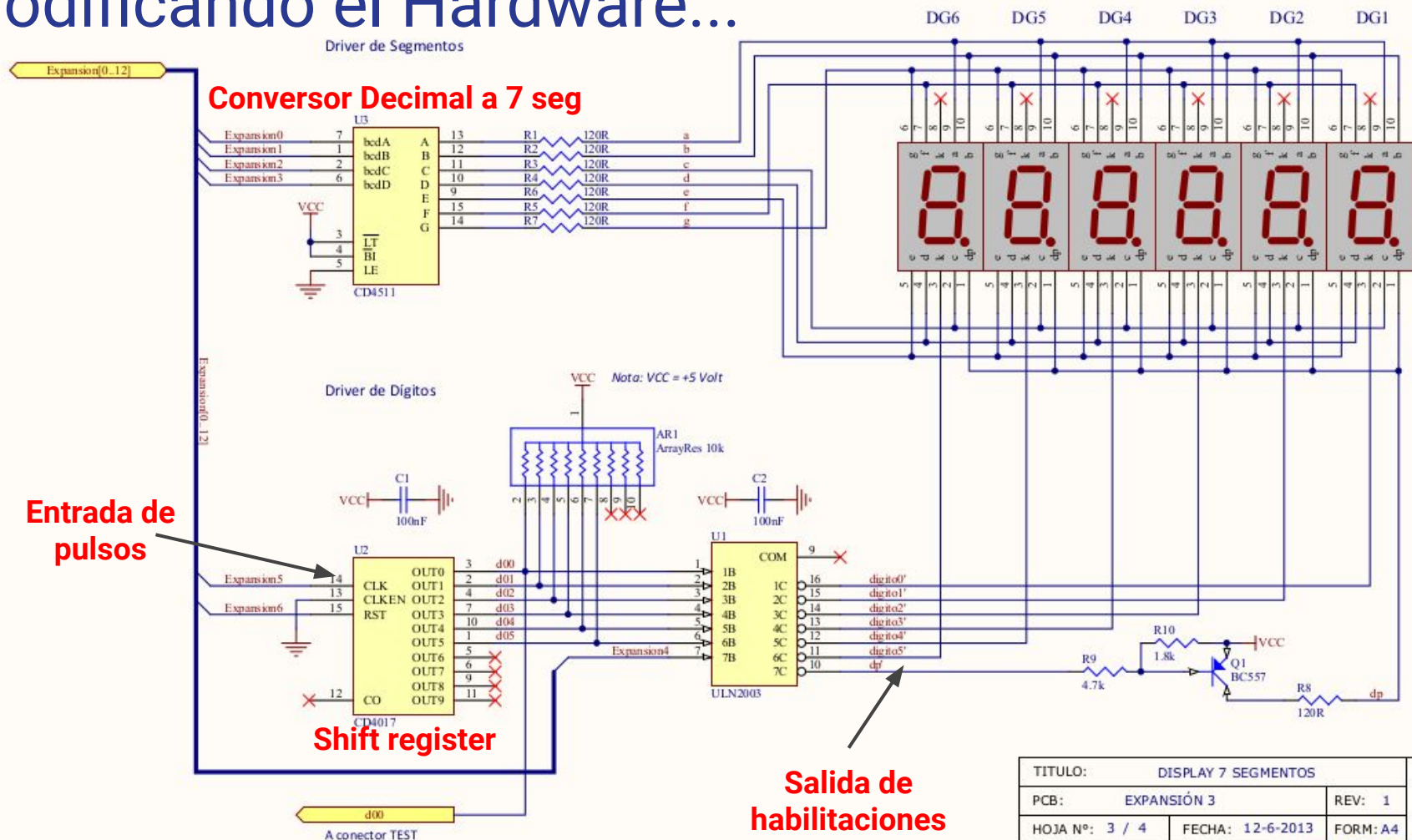
Ejemplo...

Tomando como base el ejemplo del teclado anterior, realizar un programa que escriba en un display 7 segmentos un número que se vaya incrementado o decrementado utilizando las teclas del teclado.



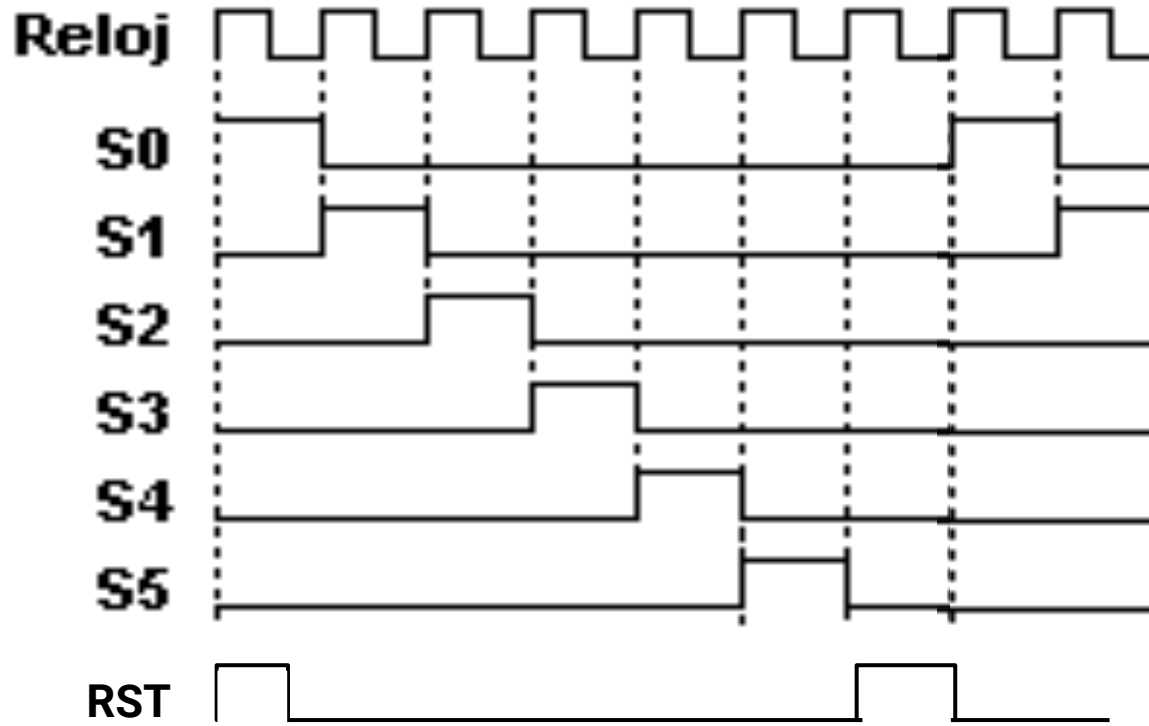
Modificando el Hardware...

Display de 7 Segmentos



TITULO: DISPLAY 7 SEGMENTOS		
PCB: EXPANSIÓN 3	REV: 1	
HOJA N°: 3 / 4	FECHA: 12-6-2013	FORM: A4

Diagrama de Tiempos (Habilitación de los dígitos)



El primer dígito se activa con el flanco descendente del pulso de RST, y luego se va cambiando de dígito con el flanco ascendente del pulso de CLK.

Modificando el driver...

```
void BarridoDisplay(void)
{
    uint8_t aux;
    static uint8_t digito = 0;
```

```
SetPIN(EX3_BCDA, ON);
SetPIN(EX3_BCDB, ON);
SetPIN(EX3_BCDC, ON);
SetPIN(EX3_BCDD, ON);
SetPIN(EX3_DP, OFF);
```

**BORRO
EL BUS**

```
if(digito == 0)
{
    SetPIN(EX3_RST, ON);
    SetPIN(EX3_RST, OFF);
}
```

**ACTIVO EL DÍGITO
CORRESPONDIENTE**

```
else
{
    SetPIN(EX3_CLK, OFF);
    SetPIN(EX3_CLK, ON);
}
aux = Digito_Display[digito] ;
```

```
SetPIN(EX3_BCDA, (aux >> 0) & 0x01);
SetPIN(EX3_BCDB, (aux >> 1) & 0x01);
SetPIN(EX3_BCDC, (aux >> 2) & 0x01);
SetPIN(EX3_BCDD, (aux >> 3) & 0x01);
```

**PONGO EL NUEVO
DATO EN EL BUS**

