Problema (Tiempo Estimado:1h 30min. 4,5 puntos).

En una iniciativa para reducir el consumo de agua, hemos decidido construir un sistema de bajo coste que pueda ser entregado por la administración en cada hogar para medir el flujo de agua que se consume.

El sistema está basado en el caudalímetro cuyas hojas de características se entregan a continuación. La señal de este dispositivo es entregada directamente al microcontrolador que gobierna el sistema, el cual deberá encargarse de mostrar en un display 7 segmentos de 3 dígitos la medida (en litros) que han pasado por el caudalímetro.

El sistema dispone de dos pulsadores, UZERO y MEAS.

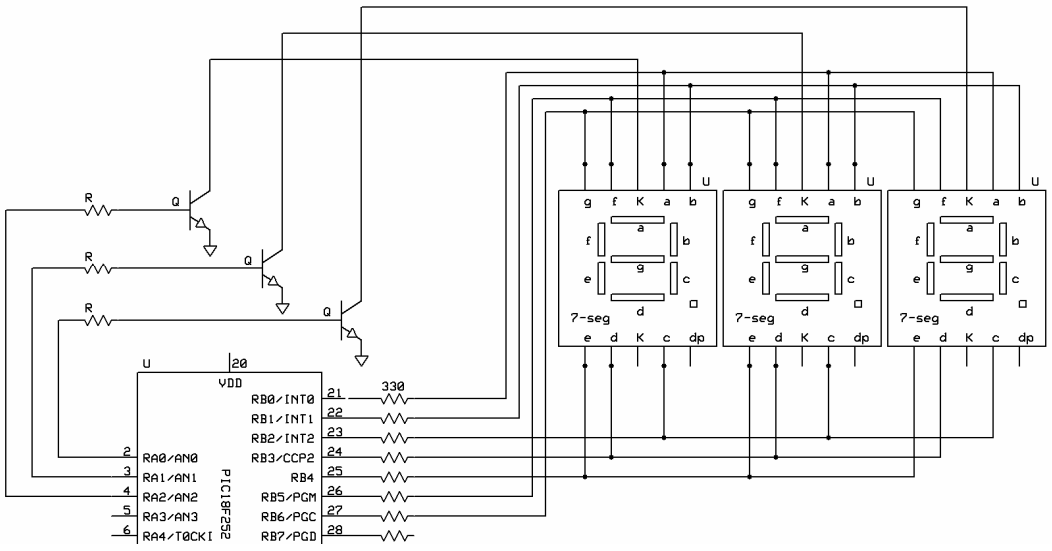
El pulsador UZERO debe servir para poner a cero el contador de litros del usuario, permitiéndole medir el número de litros consumidos desde la última vez que pulso este botón.

El pulsador MEAS debe servir para alternar la medida visualizada en el display entre el contador de litros del usuario y el contador de litros del sistema (este último almacenará el número total de litros que han pasado por el caudalímetro desde que se puso en funcionamiento).

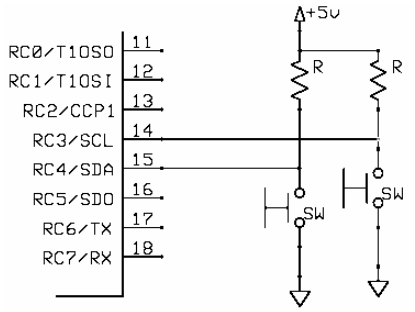
Responde a las siguientes preguntas:

1. Dibuja el esquemático de conexión entre el Microcontrolador y el display 7 segmentos y los pulsadores.

Display 7-Segmentos



Pulsadores



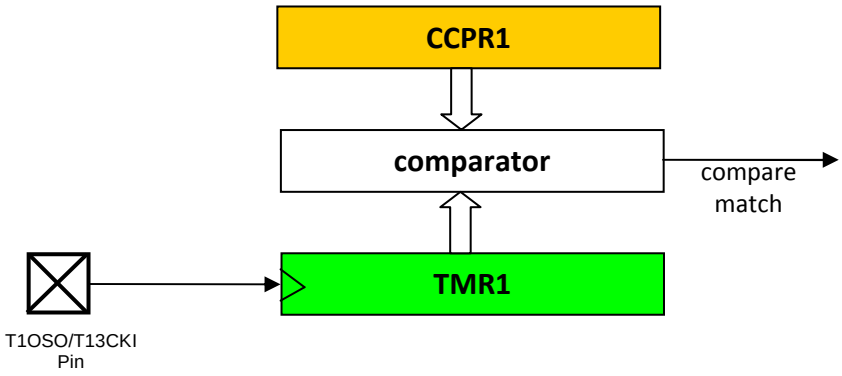
2. Dibuja el esquemático de conexión entre el Microcontrolador y el caudalímetro, justificando el pin del microcontrolador elegido. Indique los recursos del PIC que piensa utilizar en la aplicación.

Este es el punto clave del problema. ¿A qué pines conectar el caudalímetro para reducir el coste computacional en el microcontrolador? Para ello hay que entender cómo opera el caudalímetro, y qué es lo que queremos obtener de este.

En primer lugar, el caudalímetro genera pulsos. La frecuencia de los pulsos es proporcional al caudal, de forma que genera 730 pulsos/litro, entre los rangos de 1 litro/min y 20 litros/min.

En segundo, lo que queremos medir el número de litros que pasan por el caudalímetro, es decir, el hecho de que aparezcan 730 pulsos.

Para ello podemos utilizar el modulo CCP en modo COMPARE, y el temporizador en modo CONTADOR. De esta forma, el sistema queda



Conectando el cable verde (Output signal) del caudalímetro al pin T1OSO, el valor del contador se incrementa con cada pulso de este. Cuando TMR1 alcanza el valor del registro CCPR1 (convenientemente inicializado a 730), entonces, se genera un compare match que activa el flag CCP1IF (que entonces se nos indica que ha pasado 1 litro por el caudalímetro -730 pulsos-) y automáticamente se reinicia el TMR1 (El special event puede generar esta puesta a cero).

Para configurar adecuadamente este módulo, es recomendable ir a las hojas de catálogo de los dos bloques que es necesario configurar, **TMR1** y **CCP**. Empezamos por el bloque CCP.

1. Ir a las hojas del catálogo en las que se describe el modulo CCP, en las que encontramos el registro de configuración que hay que tocar:

REGISTER 15-1: CCPxCON REGISTER (CCP2 MODULE, CCP1 MODULE IN 28-PIN DEVICES)

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	DCxB1	DCxB0	CCPxM3	CCPxM2	CCPxM1	CCPxM0
bit 7							bit 0

bit 7-6 **Unimplemented:** Read as '0'

bit 5-4 **DCxB1:DCxB0**: PWM Duty Cycle bit 1 and bit 0 for CCP Module x

Capture mode:
Unused.

Compare mode:
Unused.

PWM mode:
These bits are the two LSbs (bit 1 and bit 0) of the 10-bit PWM duty cycle. The eight MSbs (DCx9:DCx2) of the duty cycle are found in CCPRxL.

bit 3-0 **CCPxM3:CCPxM0**: CCP Module x Mode Select bits

0000 = Capture/Compare/PWM disabled (resets CCP module)
0001 = Reserved
0010 = Compare mode, toggle output on match (CCPIF bit is set)
0011 = Reserved
0100 = Capture mode, every falling edge
0101 = Capture mode, every rising edge
0110 = Capture mode, every 4th rising edge
0111 = Capture mode, every 16th rising edge
1000 = Compare mode: initialize CCP pin low; on compare match, force CCP pin high (CCPxIF bit is set)
1001 = Compare mode: initialize CCP pin high; on compare match, force CCP pin low (CCPxIF bit is set)
1010 = Compare mode: generate software interrupt on compare match (CCPxIF bit is set, CCP pin reflects I/O state)
1011 = Compare mode: trigger special event, reset timer, start A/D conversion on CCP2 match (CCPxIF bit is set)
11xx = PWM mode

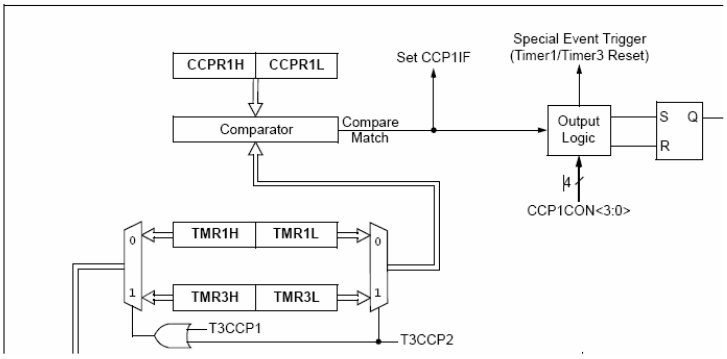
De las opciones ofertadas, seleccionar **compare**, sin que necesite más que activar el flag.

CCPxM3:0 = 0010

Asumimos que utilizamos el modulo CCP1, los bits corresponden al CCP1CON

2. Hay que indicar qué temporizador se asocia al módulo CCP. Ir al Block Diagram del modulo CCP en modo COMPARE de las hojas de catálogo, e identifica los bits que hay que tocar para el **modo compare**:

FIGURE 15-2: COMPARE MODE OPERATION BLOCK DIAGRAM



T3CCP1 y T3CCP2

3. Ir a la tabla de los registros del módulo, y localizar en qué registro se encuentras estos bits de configuración:

TABLE 15-3: REGISTERS ASSOCIATED WITH CAPTURE, COMPARE, TIMER1 AND TIMER3

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	49
RCON	IPEN	SBOREN(1)	—	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$	48
PIR1	PSPIF(2)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	52
PIE1	PSPIE(2)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	52
IPR1	PSPIP(2)	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	52
PIR2	OSCFIF	CMIF	—	EEIF	BCLIF	HLVDIF	TMR3IF	CCP2IF	52
PIE2	OSCFIE	CMIE	—	EEIE	BCLIE	HLVDIE	TMR3IE	CCP2IE	52
IPR2	OSCFIP	CMIP	—	EEIP	BCLIP	HLVDIP	TMR3IP	CCP2IP	52
TRISB	PORTB Data Direction Control Register								52
TRISC	PORTC Data Direction Control Register								52
TMR1L	Timer1 Register Low Byte								50
TMR1H	Timer1 Register High Byte								50
T1CON	RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{T1SYNC}$	TMR1CS	TMR1ON	50
TMR3H	Timer3 Register High Byte								51
TMR3L	Timer3 Register Low Byte								51
T3CON	RD16	T3CCP2	T3CKPS1	T3CKPS0	T3CCP1	$\overline{T3SYNC}$	TMR3CS	TMR3ON	51

T3CON

4. Ir a la descripción de T3CON (registro asociado a TMR3), para configurarlos de modo que el modulo CCP opere con TMR1:

REGISTER 14-1: T3CON: TIMER3 CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RD16	T3CCP2	T3CKPS1	T3CKPS0	T3CCP1	$\overline{T3SYNC}$	TMR3CS	TMR3ON
bit 7							bit 0

bit 7 **RD16:** 16-bit Read/Write Mode Enable bit
1 = Enables register read/write of Timer3 in one 16-bit operation
0 = Enables register read/write of Timer3 in two 8-bit operations

bit 6,3 **T3CCP2:T3CCP1:** Timer3 and Timer1 to CCPx Enable bits
1x = Timer3 is the capture/compare clock source for the CCP modules
01 = Timer3 is the capture/compare clock source for CCP2;
Timer1 is the capture/compare clock source for CCP1
00 = Timer1 is the capture/compare clock source for the CCP modules

T3CCP2:1 = 00

Ahora ya solo queda configurar el TMR1 para que funcione como CONTADOR. Queda como trabajo personal. Te acabamos de presentar la forma de trabajar. Ahora te toca a ti.

Y ahora que ya está planteado el problema, puedes además intentar resolver las siguientes cuestiones adicionales:

- Dibuja el diagrama de flujo del programa
- Indica cómo vamos a almacenar los datos de cantidad de litros, y cómo se van a representar en los LED 7-segmentos