UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA

Departamento de Electrónica

GUÍA DE LABORATORIO 4
MANEJO DE PERIFÉRICOS EN PROCESADORES DE PUNTO FIJO

CURSO	LABORATORIO DE PROCESAMIENTO DIGITAL DE SEÑALES
SIGLA	ELO–385
PROFESOR	MARCO E. RIVERA A.
AYUDANTE	IGNACIO LIZAMA A.

Valparaíso, Abril de 2009

Guía de Laboratorio 4
Manejo de Periféricos en Procesadores de Punto Fijo

El uso de DSPs a veces implica operaciones de punto fijo, ya que el uso de punto flotante no siempre está disponible y su emulación es costosa desde el punto de vista del cómputo. Los DSPs de punto fijo, a diferencia de sus contrapartes de punto flotante, están más orientados a labores de control de procesos que a velocidad de cálculo. A pesar de esto, mantienen las características básicas de la arquitectura DSP: memorias de datos y de programa independientes, multiplicación (de punto fijo) en un ciclo de instrucción, direccionamiento directo e indirecto de memoria, pipeline, etc. Además de estas características propias de un DSP, incluyen conversores A/D internos, varios timers, gran cantidad de I/Os, puertos de comunicación serial y salidas PWM, todos ellos asociados a una completa red de niveles de interrupciones. Todos estos periféricos integrados, una vez configurados, funcionan de forma autónoma sin interferir con la CPU. Si bien los DSP de punto fijo son capaces de trabajar con aritmética de punto flotante, la cantidad de tiempo que tardan estas rutinas obliga, en la mayoría de los casos, a limitar al máximo este tipo de operaciones, privilegiando las rutinas de punto fijo en formato Q_n.

El **objetivo de este laboratorio** es el aprender a programar operaciones matemáticas de punto fijo, conocer las ventajas respecto al punto flotante no nativo (emulado desde el punto fijo) que posee el DSP de la familia TMS320F2XX y ocupar las herramientas básicas que posee éste, en cuanto al manejo de interrupciones, timers, ADCs y unidades PWM nativas que son indispensables para el control de accionamientos eléctricos.
En esta guía se presentan sólo algunos conceptos básicos, a partir de la recopilación de los documentos disponibles de guías y trabajos anteriores, por lo que es de responsabilidad del alumno profundizar más sobre el tema.

1. Arquitecturas de Punto Fijo y Flotante

Arquitectura de un sistema se refiere a la forma que los DSPs utilizan para representar digitalmente la señal que se está procesando. Estas arquitecturas se dividen en dos grupos:

- Arquitecturas de punto fijo.
- Arquitecturas de punto flotante.

La arquitectura de punto fijo fue introducida a comienzos de los 80’ y está basada en una representación que contiene una cantidad fija de dígitos después del punto decimal. Al no requerir de unidad de punto flotante, la mayoría de los chips DSP de bajo costo utilizan esta arquitectura, aunque en determinados casos esta arquitectura ofrece también un mayor performance o mayor exactitud.
La representación en punto fijo de un número puede calcularse como:

$$2^{m-1} - \frac{1}{2^f} \qquad \text{Números Positivos}$$

$$-2^{m-1} \qquad \text{Números Negativos}$$

Donde m corresponde a los bits de magnitud y f los bits fraccionales.

Por ejemplo, si se disponen de 16 bits en total y se utilizan 11 bits para representar los enteros (m) y los restantes 5 bits para los fraccionales (f), se tiene que el máximo número positivo representable es 1023,96875. En cambio, si se asigna $m = 12$ y $f = 4$, el mayor número positivo que se puede representar es 2047,9375 y si $m = 13$ y $f = 3$, el número resulta ser 4095,875. Entonces, la arquitectura de punto fijo permite representar magnitudes mayores sólo a costo de reducir la precisión después del punto decimal.

La arquitectura de punto flotante es más moderna y resulta suficientemente exacta y rápida para la mayoría de las aplicaciones. Es frecuentemente usada para tener buenas aproximaciones, pero a menudo requiere de redondeos debido a su precisión limitada. Un número de representa de la forma:

$$a = m \cdot b^e$$

Donde m es la mantisa (número entero), b la base, e el exponente y a un número en punto flotante. La gran ventaja de esta arquitectura es que permite la representación de un rango de magnitudes mucho más amplio que el de la arquitectura de punto fijo. De acuerdo a la cantidad de bits utilizados, para almacenar un determinado número de punto flotante se tendrán números de punto flotante de precisión simple (32bits) y de precisión doble (64) bits.

2. Programación en Punto Fijo y Punto Flotante.

Los DSP de punto fijo, tal como su nombre lo indica, tienen unidades especializadas (acumuladores y multiplicadores) para trabajar en aritmética de punto fijo. A pesar de esto, es posible operar con aritmética de punto flotante, a través de rutinas por software proporcionadas por el compilador. El principal problema de este tipo de implementación es que la aritmética de punto flotante es emulada a través de un código bastante largo y complejo, lo que hace notablemente más lento el desempeño del algoritmo, por lo que no es aconsejable su uso, salvo en casos imprescindibles.

Note que el compilador interpretará que debe utilizar aritmética de punto fijo, cada vez que se opere con números decimales, por ejemplo:

a=c+1;	aritmética en punto fijo
a=c+1.0;	aritmética en punto flotante

Es por este motivo, que todas las constantes a utilizar deben estar escritas en el formato Q_n elegido. Una manera rápida para encontrar la representación en Q_n de cualquier número es la siguiente:

$\#Q_n = \text{round}(\#\text{decimal} * 2^n)$

Los cuidados que se deben tener son:

1. Elegir un n adecuado para no perder demasiada precisión.
2. Elegir un n adecuado de modo que $\#Q_n$ pueda ser representado con 16 bits (si se utilizan variables tipo int, o 32 bits si se utilizan variables tipo long).

Recuerde que el uso de formatos Q_n es una abstracción por parte del programador. El DSP SIEMPRE opera sobre números enteros.

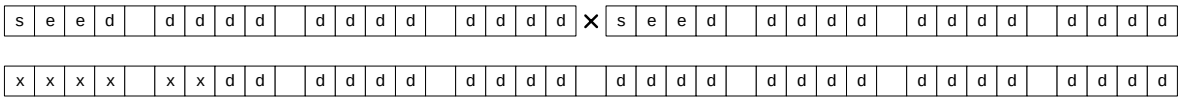
2.1 Suma en punto fijo.

La suma no se modifica al trabajar en Q_n, si es que ambos números se encuentran en el mismo *n*, de otro modo, es necesario multiplicar (o dividir), uno de los operandos por la potencia de 2 correspondiente de manera de dejar el "punto decimal" en la misma ubicación para ambos.

El único cuidado que debe mantenerse es que **nunca** se sobrepase el máximo número representable para el Q_n elegido.

2.2 Multiplicación en punto fijo.

Al multiplicar dos números de *k* bits, se obtiene un resultado de *2k* bits de largo. De la misma forma, si se opera con dos números en Q_n el resultado será de la forma:



Multiplicación de 2 Byte de 16 bits en Q_{13}.

El resultado de la multiplicación resulta ser un número en formato Q_{2n}. Para llevarlo al formato original, es necesario descartar los *n* bits menos significativos, lo que es equivalente a desplazar el resultado en *n* bits a la derecha. Nuevamente es necesario tener el cuidado de que el resultado de la multiplicación sea representable en el formato Q_n utilizado, otra solución es desplazar el resultado hacia la derecha en más posiciones que las necesarias, de modo de descartar más "dígitos decimales" para dar lugar a los "dígitos enteros", cambiando la base *n* a una nueva base *m*.

3. DSP TMS320F2407

Es un DSP de 16 bits de punto fijo y dentro de sus características principales están el que posee un CAD multiplexado en 16 canales, 4 timers de propósito general de 16 bits cada uno, unidades PWM, puertos de comunicación serial SPI, SCI, además de contar con una gran cantidad de I/Os.

Revisar el datasheet del DSP, disponible en la página del curso.

4. Manejo del Conversor Análogo Digital (CAD), Timers e Interrupciones.

3.1 Conversor Análogo Digital CAD.

El 2407 cuenta con 16 canales análogos (ADCIN0-ADCIN15), los cuales son multiplexados temporalmente para alimentar a un conversor A/D de 10 bits y que tarda 375 [ns] en procesar un dato. De acuerdo a los bits de control, es posible elegir qué canales se utilizarán y en qué orden se realizará la conversión. Además, es posible setear el tipo de evento que inicia un lote de conversiones, el que puede ser gatillado por eventos externos al DSP, eventos de timers como overflow o underflow o vía software. Los resultados de las conversiones se guardan respetando el orden de conversión en los registros RESULT0-RESULT15.

Algunos registros importantes de los CAD son presentados a continuación y en el material disponible en la página web puede revisárlos para una mayor comprensión.

- ADCTRL1 Reloj de conversión.
  Configuración de modo de secuenciadores.

15	14	13	12	11	10	9	8
Reserved	RESET	SOFT	FREE	ACQ PS3	ACQ PS2	ACQ PS1	ACQ PS0
	RS-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
7	6	5	4	3	2	1	0
CPS	CONT RUN	INT PRI	SEQ CASC	CAL ENA	BRG ENA	HI/LO	STEST ENA
RW-0	RW-0	RW-0	RW-0				

Fig. 1 Bits registro ADCTRL1.

- ADCTRL2 Maneja los eventos de inicio de las secuencias y manejo de interrupciones.

15	14	13	12	11	10	9	8
EVB SOC SEQ	RST SEQ1/ STRT CAL	SOC SEQ1	SEQ1 BSY	INT ENA SEQ1 (Mode 1)	INT ENA SEQ1 (Mode 0)	INT FLAG SEQ1	EVA SOC SEQ1
RW-0	RS-0	RW-0	R-0	RW-0	RW-0	RC-0	RW-0
7	6	5	4	3	2	1	0
EXT SOC SEQ1	RST SEQ2	SOC SEQ2	SEQ2 BSY	INT ENA SEQ2 (Mode 1)	INT ENA SEQ2 (Mode 0)	INT FLAG SEQ2	EVB SOC SEQ2
RW-0	RS-0	RW-0	R-0	RW-0	RW-0	RC-0	RW-0

Fig. 2 Bits registro ADCTRL2.

- RESULTn Registros donde se guarda el resultado de la conversión.

15	14	13	12	11	10	9	8
D9	D8	D7	D6	D5	D4	D3	D2
7	6	5	4	3	2	1	0
D1	D0	0	0	0	0	0	0

Fig. 3 Bits registro RESULTn.

- MAXCONV Cantidad de conversiones por secuencia.

15–8							
Reserved							
R-x							
7	6	5	4	3	2	1	0
Reserved	MAX CONV2_2	MAX CONV2_1	MAX CONV2_0	MAX CONV1_3	MAX CONV1_2	MAX CONV1_1	MAX CONV1_0
R-x	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

Fig. 4 Bits registro MAXCONV.

- CHSELSEQn orden de los canales a convertir.

	Bits 15–12	Bits 11–8	Bits 7–4	Bits 3–0	
70A3h	CONV03	CONV02	CONV01	CONV00	CHSELSEQ1
	RW-0	RW-0	RW-0	RW-0	

Note: R = Read access, W = Write access, -0 = value after reset

	Bits 15–12	Bits 11–8	Bits 7–4	Bits 3–0	
70A4h	CONV07	CONV06	CONV05	CONV04	CHSELSEQ2
	RW-0	RW-0	RW-0	RW-0	

Note: R = Read access, W = Write access, -0 = value after reset

	Bits 15–12	Bits 11–8	Bits 7–4	Bits 3–0	
70A5h	CONV11	CONV10	CONV09	CONV08	CHSELSEQ3
	RW-0	RW-0	RW-0	RW-0	

Note: R = Read access, W = Write access, -0 = value after reset

	Bits 15–12	Bits 11–8	Bits 7–4	Bits 3–0	
70A6h	CONV15	CONV14	CONV13	CONV12	CHSELSEQ4
	RW-0	RW-0	RW-0	RW-0	

Note: R = Read access, W = Write access, -0 = value after reset

Fig. 5 Descripción de bits registro CHSELSEQn.

3.2 Timers

El DSP TMS320F2407 cuenta con 4 timers de propósito general GPTx de 16bits, los que se encuentran divididos en dos bloques idénticos, Event Manager A (EVA) que maneja a GPT1 y GPT2 y Event Manager B (EVB) que maneja a GPT3 y GPT4. Debido a que ambos bloques son idénticos, sólo se explicarán las principales características de EVA. Los timers GPT1 y GPT2 son contadores de 16 bits c/u, pero pueden ser utilizados como un solo contador de 32 bits a través de un seteo por hardware. La cuenta de estos timers puede ser incremental, decremental o ambas; cada uno cuenta con un registro de período y es capaz de generar sus propias interrupciones de underflow, overflow, período y comparación. El reloj base para cada

contador puede ser seteado como un múltiplo del reloj principal del DSP, o en base a pulsos externos. Todas estas características hacen de estos contadores herramientas sumamente potentes y flexibles, pudiendo utilizarse como bases de tiempo real, para la ejecución de interrupciones programadas, contadores para recorrer tablas, decodificadores de encoder y generadores de patrones PWM.

Algunos de sus registros importantes son:

- TnCON Reloj de timer n, modo de cuenta detenido, ascendente, ascendente – descendente.
- GPTCONx Estado de los timers, manejo del CAD, polaridad de los pines de salida manejados por el TnCMPR.
- TnCNT Registros de cuenta.
- TnPR Valor hasta el que necesita el timer n.
- TnCMPR Valor con que se compara la cuenta del timer n.
- CMPRn Valor con que se compara la cuenta del timer n. Especializados para moduladores PWM.
- ACTRx Setea la polaridad de los pines de salida manejados por CMPRn.
- COMCONx Habilitación de los comparadores de salida dual. Maneja condiciones de recarga.
- CMPRn Valor con que se compara la cuenta del timer. El resultado de a comparación tiene una salida dual. Especializados para moduladores PWM.
- ACTRx Polaridad de los pines de salida manejados por CMPRn.

Para mayor información revise el manual de periféricos disponible en la página del curso.

3.3 Interrupciones

El manejo de interrupciones en el 2407 se basa en una serie de “flags” o avisos de interrupción y de “máscaras” o habilitadores de interrupción. Existe una gran cantidad de ***eventos*** capaces de generar interrupciones, los que se encuentran agrupados en 6 ***niveles de interrupción***, de acuerdo a su prioridad. Debido a que varios ***eventos de interrupción*** activan un mismo ***nivel de interrupción***, el DSP cuenta con una estructura muy flexible, que permite bloquear los eventos no necesarios e identificar de manera fácil qué evento es el que ha producido el ***llamado de interrupción***. La **Fig. 6** muestra el diagrama de flujo de un ***evento de interrupción*** relacionado con el Event Manager (EV). En el caso de eventos ajenos al EV los bloques 2 y 3 son reemplazados por los bits de máscara y flag propios del evento.

Los pasos 8 y 9 están relacionados con el archivo c vectors.asm, disponible en la página web. El paso 10 no es necesario si se encuentra habilitado sólo ***un evento de interrupción*** por nivel. En caso de no “limpiarse” el bit de flag respectivo (paso 12), escribiendo un “1” en la posición adecuada, la rutina de interrupción NO será atendida nuevamente.

- Flag Bit que avisa que se ha producido una interrupción.
- Máscara Bit que permite que el aviso de interrupción pase al siguiente nivel.

Algunas fuentes de interrupción requieren habilitaciones, flag y máscaras especiales. Éste es el caso de los timers y los registros EVx, donde algunos bits deben ser escritos por el usuario para producir la interrupción. Las fuentes de interrupción se encuentran agrupadas en niveles de 1 al 6 según su prioridad.

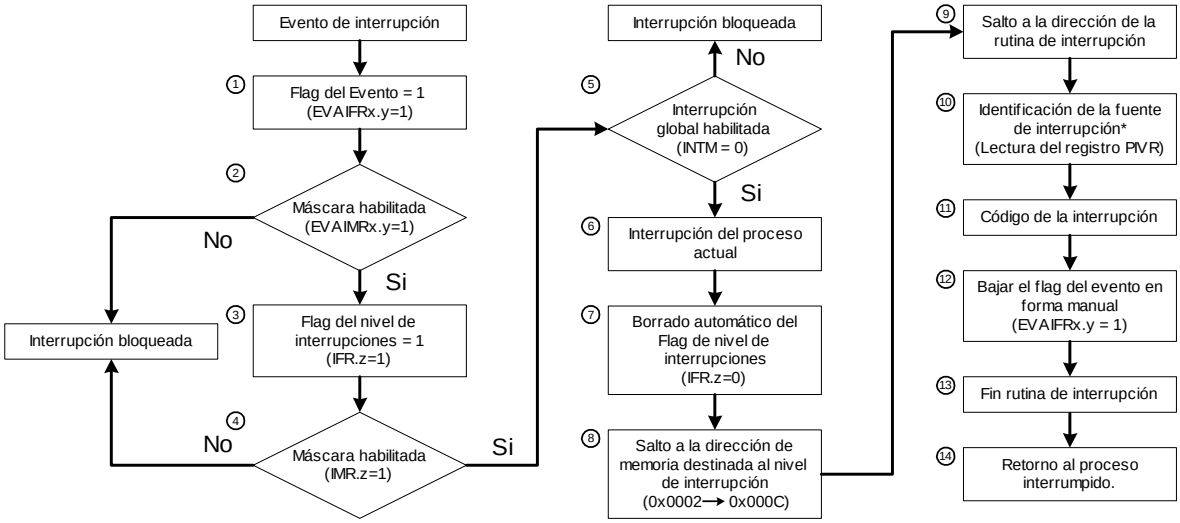


Fig. 6 Diagrama de flujo de un evento de interrupción.

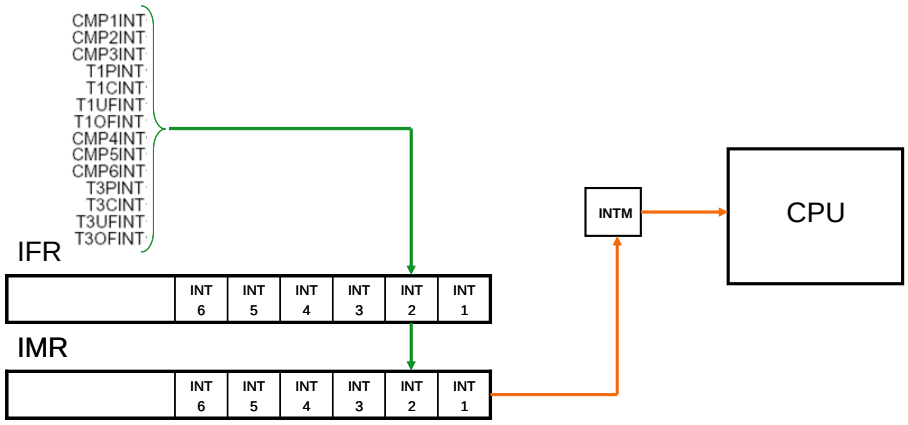


Fig. 7 Agrupaciones fuentes de interrupción según prioridad.

3.4 Manejo de I/Os.

Otra diferencia importante entre el 6711 y el 2407, es la cantidad de pines de I/Os que este último posee (40 en total). La mayoría de estos I/O poseen una doble función: función primaria (salida PWM, entrada de encoder, comunicación serial dependiendo del pin) o como I/O normal, tal como muestra la **Fig. 8.**

Para elegir entre un modo de funcionamiento u otro es necesario especificar en el registro MCRx correspondiente, la función deseada. Si la función seleccionada corresponde a la función primaria, el pin es conectado internamente al registro o bit que maneja dicha función, por otra parte, si se selecciona como I/O, es necesario configurar el bit correspondiente en el registro PxDATDIR (bits 15 a 8), para especificar si el pin será utilizado como entrada (0) o salida (1), el dato de entrada o salida es guardado en el registro PxDATDIR en las posiciones 7 a 0, dependiendo del pin elegido.

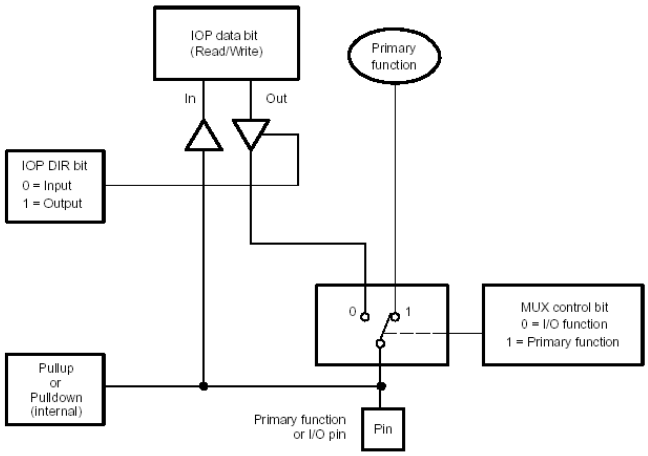


Fig. 8 Esquema de funcionamiento de un pin I/O.

5. Señales PWM

Uno de los principales problemas de los amplificadores transistorizados (Tipo A y B) es la alta cantidad de potencia que debe ser disipado por los semiconductores, debido a que éstos trabajan la mayor parte del tiempo en zona lineal. Una técnica utilizada para disminuir estas pérdidas es la utilización de amplificadores en estado de conducción discontinua (amplificadores tipo D). Tal como su nombre lo indica, la idea es que los semiconductores operen en estado de corte y saturación, de manera que el valor medio de la señal, durante un período (período de muestreo por ejemplo) corresponda al valor análogo de la señal sintetizada.

Una forma de obtener esta señal discontinua es a través de la modulación por ancho de pulso a través de una señal triangular (PWM triangular), esto se obtiene comparando una referencia de “baja frecuencia”, con una triangular de “alta frecuencia”, tal como muestra las figuras siguientes.

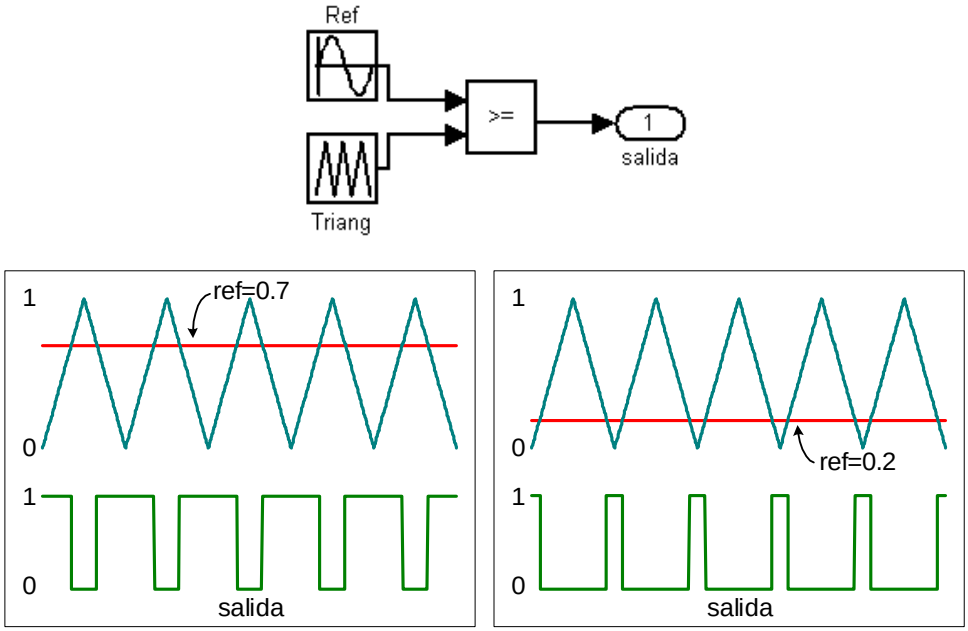


Fig. 9 Señales PWM para diferentes índices de modulación.

El DSP 2407 cuenta con varias unidades PWM, las que actúan por hardware, es decir, sin quitar tiempo a la ejecución normal del código.

Para que estos moduladores PWM operen se requiere que:

a) Se setee un timer para cuenta continua ascendente (diente de sierra) o ascendente-descendente (señal triangular).
b) Habilitar los comparadores asociados al timer antes descrito.
c) Habilitar las salidas de los comparadores con la polaridad deseada.
d) Recargar los registros de comparación a medida que se recalculan las salidas.

NOTA: No es necesario que el timer produzca interrupciones para que las salidas PWM funcionen adecuadamente

En el caso del 2407, cada timer cuenta con un registro de comparación denominado TxCMPR, **además**, los timers GPT1 y GPT3 tienen asociados 3 registros extras c/u denominados CMPRx que están pensados especialmente para ser utilizados en el control de inversores trifásicos, debido a que:

a) Es posible obtener la señal PWM y la señal PWM negada.
b) Se puede generar el tiempo muerto entre las señales anteriormente descritas a través de hardware.
c) Pueden configurarse para trabajar como un modulador vectorial.

6. Representaciones Numéricas

6.1 Complemento 2

El uso de complemento 2 permite trabajar con números positivos y negativos, pero disminuye el máximo entero representable a la mitad. Posee una representación única del cero. Se caracteriza por poder representar 2^n números negativos y $2^n - N$ números positivos.

El complemento a 2 de un número se define como $2^n - N$, donde n corresponde al número de bits y N el número que se quiere codificar.

Por ejemplo,

El complemento a 2 del decimal 7 se obtiene como:

a. El 7 en binario es 0111
b. Considerando 4 bits, es decir $n = 7$ y $N = 7$, entonces $2^n - N = 2^4 - 7 = 9$.
c. 9 en binario es 1001.

Otra forma más sencilla sería:

a. Intercambiar ceros por uno y viceversa, es decir $0111 \rightarrow 1000$
b. Al resultado sumar 1, es decir 1000 + 1 = 1001.

Así se tiene que el complemento a 2 de 7 es -7.

6.2 Formato Q_n.

Es un formato para trabajar en punto fijo y se caracteriza porque el punto "." está siempre en la misma posición. Por ejemplo, si se tienen un número de 8 bits y se trabaja en Q_4 dado por: 1001.0011, entonces ese número será:

$$1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 0 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3} + 1 \cdot 2^{-4}$$

Así se conoce dónde están los decimales. La idea es que los números se coloquen de esta forma para que el código sea mucho más rápido ya que de lo contrario, el compilador debe pasarlo primero a punto fijo y luego operar. Cabe destacar que la suma de dos números Q_n es un número en Q_n. La multiplicación de dos números en Q_n es un número en Q_{2n}. Es posible multiplicar números del tipo Q_{n1} y Q_{n2}, en este caso, el resultado tendrá $n_1 + n_2$ decimales, por lo que el programador deberá elegir la cantidad de decimales a utilizar, de manera que el resultado sea correcto. Los formatos Q_n permiten operar con decimales, pero el rango de enteros se ve disminuido mientras aumenta la precisión. La suma de Q_n es una suma normal, pero sólo se pueden sumar números de igual base Q_n. Es posible multiplicar números de diferente base Q_n.

7. Actividades

7.1. Parte Teórica

7.1.1. Realice *manualmente* y *paso a paso* la siguiente multiplicación:

1.5 x 1.5

utilizando variables de tipo entero de 8 bits en el formato Q_4.

7.1.2. Explique el funcionamiento del código interno de la función interr.c.

7.1.3. Setee los registros necesarios para que el timer GPT1 cuente ininterrumpidamente de manera ascendente y descendente (señal triangular) y que origine una interrupción de período cada 250[μs].

7.1.4. Deshabilite la interrupción del punto 7.1.3 y en lugar de generarla setee los registros necesarios para iniciar una conversión A/D en el canal 1 **de manera automática** cada vez que el timer GPT1 llegue a cero. Setee los registros necesarios para generar una interrupción al finalizar cada conversión A/D.

7.1.5. Setee los registros de control del timer GPT2 para que cuente de manera ascendente (señal diente de sierra) hasta un valor máximo de 0x07FF y una frecuencia de aprox. 15[kHz], y habilite el pin de comparación correspondiente a este timer (T2PIN).

7.1.6. Determine las ganancias K_1, K_2, K_a y K_b del controlador PI (ver apunte de Manejo de periféricos en DSP de P. Fijo) de la **Fig. 10**, el cual regula el voltaje del condensador de la **Fig. 11**.

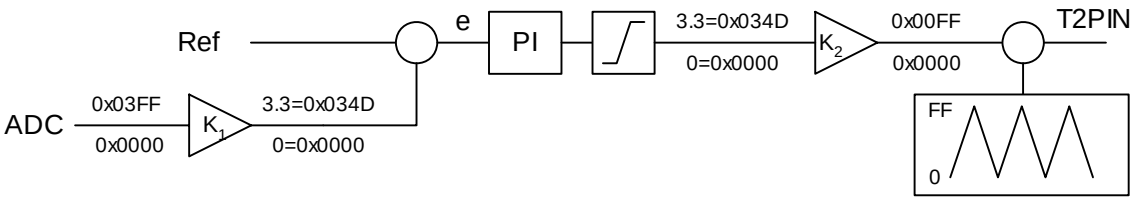


Fig.10 Esquema del controlador PI (Todos los factores en Q_8).

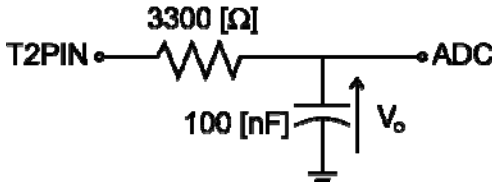


Fig. 11 Planta a controlar.

Recuerde que la salida de un controlador PI discreto está dada por:

$$y(t) = y(t-1) + u(t)\underbrace{\left(\frac{K_i}{2}\Delta t + K_p\right)}_{K_a} + u(t-1)\underbrace{\left(\frac{K_i}{2}\Delta t - K_p\right)}_{K_b}$$

La frecuencia de la señal triangular de **8 bits** debe ser de 10 [kHz] aproximadamente, cada vez que se produce un underflow de la señal, se inicia un ciclo de conversión AD, el que una vez terminado llama a una interrupción donde se ubica el código del controlador PI ($\Rightarrow \Delta t \approx 10.000^{-1}$).

7.2. Parte Experimental

7.2.1. Pruebe la correcta configuración de las rutinas de interrupción de los puntos 7.1.2, y 7.1.3 con el código disponible en la página web (archivo interr.c).

7.2.2. Usando la configuración del punto 7.1.3, almacene en un arreglo 150 conversiones A/D consecutivas de una señal generada por un generador de señales y despliéguelas en pantalla a través de un gráfico.

7.2.3. Mida la señal PWM obtenida en el punto 7.1.4 para señales de referencia tales que se observen ciclos de trabajo de: 50%, 20% y 75%.

7.2.4. Escriba un pequeño programa de prueba que realice las siguientes operaciones:

```
int a,b;
long c,d,e,f;
float g,h;
int i,j,k;
long l,m,n,o;
float p;

a=0x0180;          // 1.5 en Q8
b=0x0180;          // 1.5 en Q8
c=0x00000180;      // 1.5 en Q8
d=0x00000180;      // 1.5 en Q8
e=0x01800000;      // 1.5 en Q24
f=0x01800000;      // 1.5 en Q24
```

```
g=1.5;
h=1.5;
i=a*b>>8;
j=(a*b)>>8;
k=((long)a*b)>>8;
l=(a*b)>>8;
m=(a*c)>>8;
n=(c*d)>>8;
o=(e*f)>>24;
p=g*h;
```

En el modo ASM view, y con la ayuda de las ventanas Watch Window y Register Window revise **paso a paso** que sucede con las diferentes líneas. ¿Cuál(es) llegan a los resultados correctos?¿Cuánto aumenta el tiempo de ejecución una multiplicación de punto flotante con respecto a la de punto fijo?

Muestre los resultados de la implementación del punto 7.1.5 para una referencia fija y para referencia tipo señal cuadrada de valores 1-2[V].