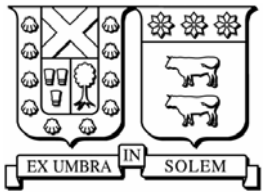

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE ELECTRÓNICA

LECTURA 4
MANEJO DE PERIFÉRICOS EN PROCESADORES DE PUNTO FIJO.

CURSO	LABORATORIO DE PROCESAMIENTO DIGITAL DE SEÑALES
SIGLA	ELO–385
PROFESOR	PABLO LEZANA ILLESCA
AYUDANTE	JORGE MURATT RODRÍGUEZ

Valparaíso, 31 de Agosto de 2004

INTRODUCCIÓN

Una de las principales ventajas de los DSP de punto fijo, es la de poseer una amplia gama de periféricos, lo que hace que estos DSP sean herramientas sumamente flexibles.
Un nicho importante de los DSP de punto fijo, se encuentra en el área del control automático, el poseer conversores A/D y resolvedores de encoder incorporados permite la adquisición de la mayoría de las variables a controlar, mientras que las unidades PWM internas y la gran cantidad de I/Os permite una fluida comunicación con casi cualquier actuador externo reduciendo el costo, espacio y posibilidades de error a la hora de implementar algoritmos de control.
Por otra parte, 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 presente documento describe de manera breve los principales registros y bits a utilizar durante la experiencia 4 del laboratorio de DSP, incluyendo al final una descripción de la implementación de un controlador PI en punto fijo.

1. Timers

TxCON (pag. 6-32 spru357b-peripheral.pdf):

Registro de control de operación del timer x.

Figure 6–9. Timer x Control Register (TxCON; x = 1, 2, 3, or 4) — Addresses 7404h (T1CON), 7408h (T2CON), 7504h (T3CON), and 7508h (T4CON)

15	14	13	12	11	10	9	8
Free	Soft	Reserved	TMODE1	TMODE0	TPS2	TPS1	TPS0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
7	6	5	4	3	2	1	0
T2SWT1/ T4SWT3†	TENABLE	TCLKS1	TCLKS0	TCLD1	TCLD0	TECMPR	SELT1PR/ SELT3PR†
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

Bits 15-14: Establece el comportamiento del conversor A/D en un *breakpoint*. El timer se detiene inmediatamente al llegar a un *breakpoint*, se detiene al llegar al *overflow*, o no se detiene.

Bits 12-11: Setea el tipo de cuenta del contador: incremental, decremental incremental-decremental o detenido.

Bits 10-8: Preescalador de reloj. Divide la frecuencia del reloj base (usualmente el reloj del DSP)

Bit 7: El contador utiliza su propio habilitador de operación o depende de otro contador.

Bit 6: Habilitador de operación del timer.

Bits 5-4: Selecciona la fuente del reloj. Interno, externo o un encoder.

Bits 3-2: Condición de recarga del registro de comparación del timer (TxCMPR).

Bit 1: Habilitador de comparador para el registro TxCMPR.

Bit 0: Uso del registro de período propio o del de otro timer.

GPTCONA (pag. 6-34 spru357b-peripheral.pdf):

Regula la relación de los timer 1 y 2 (3 y 4 en el caso de GPTCONB) con los demás periféricos del DSP.

Figure 6–10. GP Timer Control Register A (GPTCONA) — Address 7400h

15	14	13	12–11	10–9	8–7
Reserved	T2STAT	T1STAT	Reserved	T2TOADC	T1TOADC
RW-0	R-1	R-1	RW-0	RW-0	RW-0

6	5–4	3–2	1–0
TCOMPOE	Reserved	T2PIN	T1PIN
RW-0	RW-0	RW-0	RW-0

Bit 14: Indica el estado del timer 2: cuenta incremental o decremental.
Bit 13: Indica el estado del timer 1 cuenta incremental o decremental.
Bits 10-9: Indica que evento del timer 2 inicia una secuencia en el ADC.
Bits 8-7: Indica que evento del timer 1 inicia una secuencia en el ADC.
Bit 6: Habilitador los pines de salida de los comparadores (TxCMPR).
Bits 3-2: Polaridad de la salida del resultado de la comparación con T2CMPR.
Bits 1-0: Polaridad de la salida del resultado de la comparación con T1CMPR.

COMCONA (pag. 6-41):

Controla las unidades PWM del timer 1 (COMCONB del timer 3).

Figure 6–13. Compare Control Register A (COMCONA) — Address 7411h

15	14	13	12	11	10	9	8
CENABLE	CLD1	CLD0	SVENABLE	ACTRLD1	ACTRLD0	FCOMPOE	$\overline{\text{PDPINTA}}$ STATUS
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	R-$\overline{\text{PDPINTA}}$ PIN

7–0
Reserved
R-0

Bit 15: Habilita la comparación de los registros CMPR1, CMPR2 y CMPR3 (notar que estos registros **son diferentes** a TxCMPR) con GPT1.
Bits 14-13: Evento que provoca la recarga de los registros CMPRx.
Bit 12: Habilita la modulación vectorial PWM.
Bits 11-10: Evento que provoca la recarga en los registros de actuación.
Bit 9: Habilita las salidas PWM.

ACTRA (pag. 6-43):

Setea la polaridad de los pines de salida PWM del timer 1. (ACTRB maneja las salidas del timer 3).

2. Conversor A/D

ADCTRL1 (pag. 7-20 spru357b-peripheral.pdf):

Setea el funcionamiento básico del conversor.

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				

Bit 14: Resetea todos los registros del conversor A/D.

Bits 13-12: Establece el comportamiento del conversor A/D en un breakpoint. Detención inmediata, detención después de terminar la conversión actual o no detener.

Bits 11-8: Preescalador de reloj para la conversión (ancho de la ventana).

Bit 7: Preescalador de reloj general del DSP.

Bit 6: Conversión continua (termina un ciclo y se reinicia automáticamente) o iniciada por un evento externo al conversor.

Bit 5: Prioridad de una interrupción del ADC.

Bit 4: Un secuenciador de 16 estados o dos secuenciadores de 8 de conversiones.

Bit 3-0: Control de calibración.

ADCTRL2 (pag. 7-25 spru357b-peripheral.pdf):

Indica que eventos inician una conversión A/D, incluye la máscara y flag de interrupción de final de conversión.

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

Bit 15: Las conversiones seteadas en el secuenciador de 16 estados es iniciado por un evento del Event Manager B (EVB)

Bit 14: Reinicia el secuenciador 1 (8 estados) al estado 0.

Bit 13: Señal de inicio para el secuenciador 1 (seteado automáticamente al ocurrir el evento predefinido), escribir un 0 cancela el inicio automático.

Bit 12: Indica si la secuencia 1 está en proceso.

Bit 11-10: Habilita el generador de interrupciones al terminar la secuencia 1 o una conversión (máscara).

Bit 9: Indicador de que se produjo un evento de interrupción generado por la secuencia 1 (flag) .
Bit 8: Permite que un evento del Event Manager A (EVA) inicie la secuencia 1.
Bit 7: Permite que un evento externo inicie la secuencia 1.
Bit 6: Reinicia el secuenciador 2 (8 estados) al estado 8 (estado inicial de este secuenciador).
Bit 5: Señal de inicio para el secuenciador 2 (seteado automáticamente al ocurrir el evento predefinido), escribir un 0 cancela el inicio automático.
Bit 4: Indica si la secuencia 2 está en proceso.
Bit 3-2: Habilita el generador de interrupciones al terminar la secuencia 2 o una conversión (máscara).
Bit 1: Indicador de que se produjo un evento de interrupción generado por la secuencia 2 (flag).
Bit 0: Permite que un evento del Event Manager B (EVB) inicie la secuencia 2.

MAXCONV (pag. 7-31 spru357b-peripheral.pdf):

Cantidad de conversiones a utilizar en cada secuenciador. Máximo 8 por secuenciador.

CHSELSEQn (pag. 7-35 spru357b-peripheral.pdf):

Indica el orden de canales a convertir.

RESULTn (pag. 7-37 spru357b-peripheral.pdf):

n=0..15 registros donde se guarda el resultado de la conversión. El secuenciador 1 utiliza los registros 0..7, y el secuenciador 2 los registros 8..15. el formato es el siguiente:

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

Si se quiere que el resultado de una conversión se guarde siempre en el mismo registro, el secuenciador debe ser reiniciado al final de cada secuencia. De otro modo, los resultados de las conversiones se guardarán en orden correlativo hasta llenar todos los registros RESULT asignados al secuenciador.

3. Controlador PI:

a. Ecuaciones básicas.

Un controlador PI en tiempo continuo está expresado por:

$$y(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau \tag{1}$$

Para trabajar en un DSP con esta ecuación, es necesario discretizarla, una forma de hacerlo es la siguiente:
La ecuación (1), para el siguiente instante de tiempo es:

$$y(t+\Delta t) = K_p e(t+\Delta t) + K_i \int_0^{t+\Delta t} e(\tau) d\tau \tag{2}$$

restando (1) y (2):

$$y(t+\Delta t) - y(t) = K_p \big(e(t+\Delta t) - e(t)\big) + K_i \left(\int_0^{t+\Delta t} e(\tau) d\tau - \int_0^t e(\tau) d\tau \right) \tag{3}$$

lo que puede expresarse como:

$$y(t+\Delta t) = y(t) + K_p \big(e(t+\Delta t) - e(t)\big) + K_i \int_t^{t+\Delta t} e(\tau) d\tau \tag{4}$$

aproximando trapezoidalmente la integral de (4), se llega a:

$$y(t+\Delta t) = y(t) + K_p \big(e(t+\Delta t) - e(t)\big) + K_i \frac{e(t+\Delta t) + e(t)}{2} \Delta t \tag{5}$$

para que esta ecuación sea implementable en el DSP es necesario retardar todas las señales en un instante de tiempo, ya que en (5) se requiere conocer valores futuros del error, además en ella hay varios cálculos que pueden evitarse redistribuyendo los términos:

$$y(t) = y(t-\Delta t) + e(t) \left(\frac{K_i \Delta t}{2} + K_p \right) + e(t-\Delta t) \left(\frac{K_i \Delta t}{2} - K_p \right) \tag{6}$$

o si se prefiere:

$$y(t) = y(t-\Delta t) + e(t) K_a + e(t-\Delta t) K_b \tag{7}$$

b. Implementación en el DSP.

El conversor A/D del DSP sólo convierte valores positivos, por lo que al trabajar con señales positivo-negativas, es necesario sumar un offset a la señal de manera que sea convertida correctamente por el ADC, en este caso, una vez convertida la señal, es necesario restar este offset, de modo de no ingresar al lazo de control valores continuos no existentes en la realidad.
Una técnica bastante útil es utilizar valores “reales” para las variables, es decir, multiplicar la señal convertida por alguna ganancia K_1 de manera que el resultado de la conversión (sin el offset ya mencionado) sea igual al valor real de la variable medida en algún formato Q_n apropiado. Esto permite determinar los parámetros del controlador en términos de los valores reales a controlar y utilizar valores de referencia en “metros”, “amperes”, “rpm”, etc.
Una vez realizadas las conversiones necesarias se obtiene la señal de error *e(t)* de manera normal, esta señal alimenta al controlador (7), cuyos parámetros se encuentran en el un formato Q_n apropiado (generalmente el mismo utilizado para la variable de medición). Un aspecto a considerar es el de las saturaciones del actuador y el problema del

antienrrollamiento del integrador del PI. Una estrategia simple de abordar este problema es la siguiente:

1) Ya que se trabaja con valores reales, saturar la salida del controlador a los valores reales de actuación del actuador.
    a. Precalcular la salida del controlador en una variable temporal *temp(t)*.
    b. Si *temp(t)* supera la actuación máxima del actuador: *y(t)*=satup, por lo que en la siguiente evaluación del controlador *y(t-Δt)*= satup.
    c. Si *temp(t)* baja de la actuación mínima del actuador: *y(t)*=satdown,
    d. Si *temp(t)* se encuentra entre satdown y satup, *y(t)=temp(t)*.
2) Si se utiliza un modulador PWM para salir del DSP, multiplique la salida del controlador por un factor adecuado de modo que para *y(t)*=satup, la señal a comparar con la triangular tenga un valor máximo (igual al máximo de la triangular), de modo que efectivamente se tenga la máxima actuación posible, del mismo modo que para *y(t)*=satdown se obtenga la actuación mínima. En algunos es necesario utilizar estructuras un poco más complejas para cumplir ambas condiciones.

Nota: El resultado de la mayoría de los conversores de los DSP entregan los datos en el formato

Dato de x bits	0	0	...	0

por lo que resulta útil desplazar el resultado de la conversión de modo que quede alineado a la derecha.

4. Sistema de archivos.

Los archivos necesarios para trabajar con el DSP TMS320F2407 son un poco diferentes a los utilizados por el 6711, la descripción de ellos es la siguiente:

cvectors.asm: Trozo de código en assembler que se escribe a partir de la dirección 0 de la memoria de programas del DSP. Cada vez que un nivel de interrupción genera una interrupción válida (pasando por todas las máscaras necesarias) el DSP detiene su proceso actual y "salta" a una dirección específica correspondiente a dicho nivel de interrupción entre 0 y 0x003E, justamente las direcciones de memoria que son escritas por cvectors.asm. Ya que cada nivel de interrupción posee sólo dos bytes de memora para el código de la interrupción, es usual colocar un salto a una rutina más larga. La manera de hacerlo es:

B _{nombre_de_función_en C}.

main.c: Archivo de C que contiene los llamados a funciones de inicialización e inicio del loop infinito.

init.c: Archivo de C que contiene las funciones de inicialización llamadas por main.c, configuraciones de timers, ADC, comunicaciones seriales, registros de sistema.

interr.c: Archivo de C que contiene las funciones llamadas por las rutinas de interrupción llamadas por cvectors.asm. Estas funciones son del tipo *interrupt:*

```
interrupt void nombre_de_función_enC(void)
{codigo de la interrupción}
```

f2407_c.h: Archivo de cabecera que contiene las definiciones de todos los registros del 2407.

memoria.cmd: Archivo de configuración de memoria que indica al compilador el mapa de memoria del dispositivo.

lf2407.gel: Archivo de formato GEL que permite tener un rápido acceso a los registros internos del DSP.

rts2xx.lib: Archivo de librería que permite la depuración en tiempo real, además de contener la definición de variables utilizadas en la compilación. Se encentra en c:\ti2000\c2400\cgtools\lib.

Los archivos .c pueden reunirse en un sólo archivo, pero por orden se recomienda trabajarlos en forma separada.