RESUMEN ED3 PRIMER PARCIAL

**PLACA**

**GPIO**

* **REGISTROS (32 Bits)**
* PINSEL: Permite seleccionar la función del pin en cuestión. Cada pin requiere 2 bits del registro, por lo que cada pin tendrá como máximo 4 funcionalidades.
* PINMODE: Permite seleccionar las características del pin: pulll up/down, repeater, etc.
* FIODIR: Selecciona si el pin es ENTRADA (0) o SALIDA (1)
* FIOSET: Setea el valor del pin (pone en 1), si este es salida.
* FIOCLEAR: Clearea el valor del pin (pone en 0), si este es salida.
* FIOPIN: Lee el estado de los pines (Si son entradas)
* FIOMAS: Mascara para las acciones de pines.

**SYSTICK**

Es un timer integrado en el CORTEX, por lo que se dice que esta integrado en el sistema. “SystemTimer”. Tiene la capacidad de interrumpir (IRQ) cada un cierto tiempo definido por software. El NVIC obliga al núcleo a ejecutarse la rutina de interrupciones: void SysTick\_Handler( ){}

Internamente es un contador que decrece con cada clock:

Gráfico

Descripción generada automáticamente

Reload = SysTick\_LOAD \* ClockFrequency – 1.

Donde:

-Reload: Ciclos de reloj en los que el Systick llegara a 0.

-SysTick\_LOAD: Variable que contiene el valor de carga que el usuario define por software.

-ClockFrequency: Frecuencia del clock.

* **REGISTROS (32 Bits)**
  + SysTick\_CTRL: Contiene el CountFlag (es 1 cuando el contador llega a 0), el ClockSource (clck interno o externo), el TICKINT (habilita la interrupción por Systick), el Enable (habilita el contador)
  + SysTick\_LOAD (24 bits, el resto reserved): Este registro contiene el valor de recarga del Systick. No contiene el valor actual del contador, es decir, su valor no cambia a menos que el usuario lo haga.
  + Systick\_VAL (24 bits, el resto reserved): Contiene el valor actual del contador del systick.
  + Systick\_CALIB: Es una recarga por defecto de 10ms

**CMSIS**

* **GPIO**
  + Void GPIO\_SetDir( uint8\_t portNum, uint32\_t bitValue, uint8\_t dir): Setea la direccion del bitValue (que pin). Si necesitamos poner mas de un pin, podemos hacerlo con la misma función y con una OR.
  + Void GPIO\_SetValue( uint8\_t portNum, uint32\_t bitValue ) : Setea el pin si es salida.
  + Void GPIO\_ClearValue ( uint8\_t portNum, uint32\_t bitValue ) : Clearea el pin si es salida.
  + Uint32\_t GPIO\_ReadValue( uint8\_t portNum ) : Lee el valor del puerto, de los pines que tienen input direction.
  + Void GPIO\_IntCmd( uint8\_t portNum, uint32\_t bitValue, uint8\_t edgeState ): Habilita la interrupcion por GPIO del portNum, y de los pines bitValue. Necesitamos pasarle el flanco de interrupción (Solo sirve: P0.0, P0.30, P2.0, P2.13)
* **SYSTICK**
  + Void SYSTICK\_InternalInit ( uint32\_t time ) : Inicializa el SYSTICK con el CLK interno. Time es la variable de carga. Tener en cuenta que es limitado en intervalos. Con 100MHz, puede contar hasta máximo 167 ms.
  + Void SYSTICK\_ExternalInit ( uint32\_t freq, uint32\_t time ): Inicializa con CLK externo.
  + Void SYSTICK\_Cmd ( FunctionalState NewState ): Comienza el conteo. El FunctionalState debe ser ENABLE.
  + Void SYSTICK\_IntCmd ( FunctionalState NewState ): Habilita las interrupciones por SYSTICK. El FunctionalState debe ser ENABLE.
  + Uint32\_t SYSTICK\_GetCurrentValue (void): Valor actual del SYSTICK
  + Void SYSTICK\_ClearCounterFlag (void): Limpia la bandera de interrupcion

**NVIC del LPC17xx**

CARACTERISTICAS

* Admite 35 interrupciones vectoriales.
* 32 niveles de prioridad.
* Tabla de vectores reubicable.
* Interrupciones no enmascarables.
* Interrupciones por software.
* Es NESTED (anida interrupciones)
* Es VECTORIZADA: Cada interrupción tiene su vector, y el sistema busca en este vector especifico, una direccion de memoria donde DEBE estar la rutina de servicio.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Los vectores de interrupción empiezan en la dirección 0x0..04, con el Reset\_Handler.

TABLA DE VECTORES

Es una tabla de dos columnas (si se quiere ver gráficamente) y n filas, siendo n el numero de tipos de interrupción, o vectores tenga el sistema. La columna 1 esta asociada con el “tipo” o “interruptID” de interrupción, por ejemplo, interrupción por SYSTICK. La columna 2 contiene la dirección de memoria de la subrrutina de interrupción de tal evento. Entonces, el sistema encuentra rápidamente a donde debe ir cuando una interrupción de tipo “i” llega.

Tabla

Descripción generada automáticamente

CÁLCULO DE LA DIRECCIÓN DE MEMORIA DE LOS PUNTEROS ISR

Observar que en la figura de la memoria de arriba (no la tabla, la otra), en el recuadro rojo están las excepciones. Nos interesa conocer las direcciones de punteros de las ISR, es decir: ¿En que posición exacta está la dirección de memoria de la rutina de interrupción de una determinada interrupción?

El primer vector dentro de la memoria es el 0x00…40 y pertenece al WWDG\_IRQHandler. En decimal, esto es 64. Así que a partir de la dirección 64 hacia arriba, estarán las siguientes ISR.

Además, cada posición de memoria son 4 bytes (se dice que cada dirección son 4 posiciones de memoria, porque trabajamos con un ARM de 32 bits de siza).

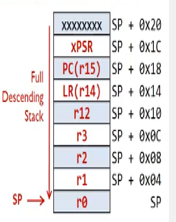
Además, el InterruptID crece a medida crecen las direcciones.

Entonces:

DATO: Las InterruptID decrecen hacia abajo del 0x00..40, es decir, a partir de ahí hacia abajo las InterruptID son de EXCEPCIONES y tienen valores NEGATIVOS.

STACKING

En el Stack se cargará el contexto de las variables y el contexto del PC correspondiente para volver al lugar adecuado. El encargado de hacer este proceso se denomina Stack Pointer.



Cuando una rutina de interrupción es finalizada, el Stack es “popeado” devolviendo contexto y PC.

ESTADO DE INTERRUPCIONES

* Active: Este valor estará activo cuando el stacking haya terminado y la rutina de interrupción este ejecutándose.
* Pending: Llegó la interrupción, pero se debe stackear.

CMSIS – IRQ

* NVIC\_DisableIRQ( IRQn ); //Deshabilita interrupcion por IRQn
* NVIC\_EnableIRQ( IRQn ); //Habilita interrupcion por IRQn
* NVIC\_ClearingPending( IRQn ) //limpia el estado de “pending” de IRQn
* NVIC\_SetPriority ( IRQn, priority ) //Setea la prioridad

PRIORIDADES

Menor numero significa mayor urgencia.

Tabla

Descripción generada automáticamente

INTERRUPCION POR SOFTWARE

--🡪>:>:>::>:>:>::>:>:ESCRIBIR

**CLOCK**

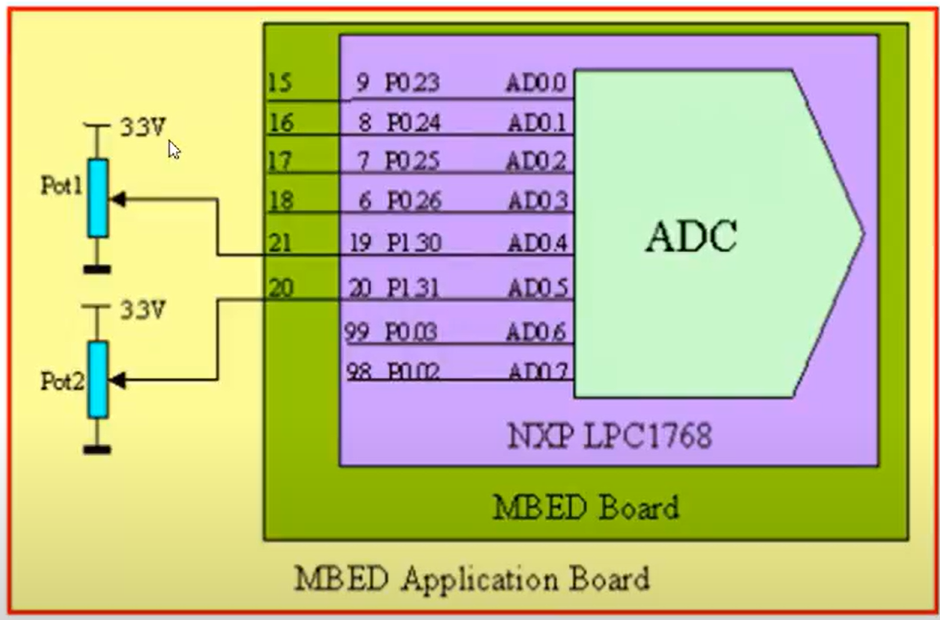
Tabla 40 PCLKSEL0, para el adc la tabla 531

**ADC**

CARACTERISTICAS:

* 12 Bits de aproximaciones sucesivas.
* Multiplexado de entrada de 8 pines (selector de señales analógicas)
* 200 KHz (velocidad de conversión)
* El voltaje de entrada de conversión no puede superar los 3,3V (típicamente, 3V)
* Modo ráfaga
* COnversion opcional a través de temporizador o con pin de entrada.
* Deshabilitar el ADC produce una interrupción en el NVIC.

EJEMPLO DE USO CON POTENCIOMETRO



COMO PROGRAMAR CUALQUIER PERIFERICO?

1. Encender el periférico (En el ADC, se debe poner un 1 en el bit 12 del registro PCADC)
2. Configurar el reloj del periférico (En el ADC, bits 24 y 25 de PCLK\_ADC) Tabla

   Descripción generada automáticamente
3. Configurar las funciones del pin (Seleccionar en el PINSEL, el AD0[7] (puerto 0))

PINES DEL ADC

* AD0 to AD7 (Pines de multiplexado de entrada)
* VREFP, VREFN: Pines de referencia. Comúnmente 3,3 V y masa
* VDDA, VSSA: Energía del ADC. Tienen que ser 3,3 V y masa.

REGISTROS DEL ADC (tabla 51)

* ADCR (Registro de control) *(tabla 532)*
  + Bit 21 = Enable (en 1)
  + Bit 0:7 = Habilitar las entradas del ADC (mux channels).
  + Bit 8:15 = divisor del clock (representación binaria de un numero n tal que PCLK/(n+1))
  + Bit 17:20 = RESERVED
  + Bit 24:26 = Modos de lanzamientos del ADC.
  + Bit 27: FLANCO (EDGE) del start of conversion (START: 001)
  + BIT 16: BURST: Rafaga (si esta en 1)
* ADGDR (Registro de datos globales): Esto tiene la data de la ultima conversión del ADC

Esto tiene un DONE global y un OVERRUN global *(Tabla 532)*

* ADINTEN (Habilitacion de interrupciones) *(Tabla 534)*
  + Bit 0:7 = Habilitacion de interrupción particular de cada canal.
  + Bit 8 = Enable de la interrupción particular (0) Enable de la interrupción global (1)
* ADDR0 to ADDR7 (Contienen la data de la ultima conversión de la entrada especifica) (Bits 4 to 15 la data, bit 31 DONE, bit 30 Overrun)
* ADSTAT (Status): bit de DONE (finalizo la conversion. Es un OR logico de todos los DONE particulares de cada canal), OVERRUN (sobreescrito flag), etc.
* ADTRM: condiciones iniciales del conversor (No lo modificamos)

FORMAS DE TRABAJAR

* Disparo Controlado o Continuo:
  + BURST = 1) Rafaga. Realiza la conversión de forma continua. Cuando necesitamos leer el valor, simplemente lo buscamos, pero debemos tener en cuenta que esta realizando conversiones todo el tiempo, en los canales habilitados.
  + BURST = 0) Controlado. Convertimos un dato de un canal determinado en el momento deseado.
    - Detenemos el programa hasta que el dato este disponible (DONE)
    - Interrumpe el programa cuando el dato este