Entradas y Salidas con periféricos

Sistemas con Microprocesadores

Ing. Esteban Volentini (evolentini@herrera.unt.edu.ar)

http://www.microprocesadores.unt.edu.ar/procesadores

Cronograma

Actividad	Inicio	Descripción	Fin
Presentación	19/08	Reglamento de la Materia	✔
Tema 1	19/08	Estructura de las computadoras	✔
Tema 2	26/08	Proyecto con un microcontrolador	✔
Tema 3	30/08	Descripción funcional de microprocesador	✔
Tema 4	13/09	Programación en lenguaje ensablador	✔
Tema 5	25/09	Descripción general de un microcontrolador	✔
Tema 6	27/09	Estructura general de microcontrolador	✔
Parcial	09/10	Primer examen parcial	✔
Tema 7	14/10	Sistema de Interrupciones	✔
Tema 8	21/10	Entradas y salidas digitales	✔
Tema 9	28/10	Entrada/salida con perifericos	←
Tema 10	06/11	Temporizadores	
Proyectos	25/11	Seminarios de Proyectos	
Parcial	04/12	Segundo examen parcial	

Interrupciones Grupales en GPIO

- Se puede crear una interrupción a partir de varios pins de un GPIO
  - Puede ser un OR o un AND entre estos pins para generar el pedido.
  - Si el pedido resultante es por flanco o nivel.
- Se puede seleccionar la polaridad
  - En cada pin: positiva o negativa.
- Hasta dos grupos en total.
- Cada grupo un único pedido.
- Veremos solo interrupciones individuales.

Interrupciones Individuales en GPIO

- Se pueden configurar hasta 8 pedidos distintos de Interrupción.
- Cada interrupción se puede asociar a una línea de GPIO
- Solo se pueden utilizar 8 lineas en total de todos los GPIOs.
- Las interrupciones pueden producirse por flanco o por nivel.
- Puede deshabilitarse el pedido de interrupción y usar el mecanismo para detectar flancos por polling.

Selección de la fuente de interrupción

- Para seleccionar la fuente hay 2 registros:
  - PINTSEL0 para las interrupciones 0 a 3
  - PINTSEL1 para las interrupciones 4 a 3
- Un byte para cada interrupción:
  - 5 bits permiten configurar la linea y 3 bits configuran el puerto
- Se puede seleccionar cualquier linea de cualquier GPIO

Bit	Símbolo	Descripción
4:0	INTPIN0	Nº del bit dentro del GPIO (0 a 15)
7:5	PORTSEL0	Número de GPIO (0 a 7)
12:8	INTPIN1	Nº del bit dentro del GPIO (0 a 15)
15:13	PORTSEL1	Número de GPIO (0 a 7)
	…	Se repite para PIN2 y PIN3

Registros de configuración

Nombre	Descripción
PINTSEL0/1	Identifica bit y GPIO para INT0-3 (SEL0) y para INT4-7(SEL1)

Selección de Flanco o nivel

- Hay un registro ISEL para configurar las interrupciones por flanco o nivel.
- Se usan solo 8 bits, uno por cada interrupción.
- Cada bit indica el modo de funcionamiento
  - El valor 0 configura la operación por flanco.
  - El valor 1 configura la operación por nivel
  - El bit 0 configura a INT 0 y asi sucesivamente.

Registros de configuración

Nombre	Descripción
PINTSEL0/1	Identifica bit y GPIO para INT0-3 (SEL0) y para INT4-7(SEL1)
ISEL	0 – por flancos, 1 – por nivel

Habilitar flanco positivo o nivel

- Un registro habilita el pedido de interrupciones al detectar un flanco positivo.
- Si esta configurado por nivel entonces simplemente interrupciones.
  - Por flanco (ISEL = 0)
    - 1 – habilita el pedido de interrupción por flanco positivo
    - 0 – inhabilita el pedido de interrupción por flanco positivo
  - Por nivel (ISEL = 1)
    - 1 – habilita el pedido de interrupción en el nivel activo
    - 0 – inhabilita el pedido de interrupción en el nivel activo

SIENR – Set interrupt Enable

- Si bien se puede escribir directamente en IENR este registro permite escribir los “1”
  - Si se escribe “1” en un bit entonces escribe “1” en bit correspondiente de IENR.
  - Si se escribe un “0” entonces no hay cambios en el bit correspondiente de INER
- Es para facilidad de programación.
  - Solo para poner en 1 algunos bits.
  - Se escribe directo en SIENR.
  - De este modo se evita leer, modificar y escribir.

CIENR – Clear Int Enable Reg

- Idem registro anterior, pero para poner en 0 bits del IENR.
- Si se escribe “1” en un bit entonces escribe “0” en bit correspondiente de IENR.
- Si se escribe un “0” entonces no hay cambios en el bit correspondiente de INER

Registros de configuración

Nombre	Descripción
PINTSEL0/1	Identifica bit y GPIO para INT0-3 (SEL0) y para INT4-7(SEL1)
ISEL	0 – por flancos, 1 – por nivel
IENR	1 – habilita flanco ascendente (ISEL = 0) o nivel (ISEL = 1) 0 - inhabilita flanco ascendente (ISEL = 0) o nivel (ISEL = 1)
SIENR	Para escribir solo los 1 en IENR
CIENR	Para escribir solo los 0 en IENR (se escribe 1 para un 0)

Habilitar flanco negativo o valor del nivel

- Un registro habilita el pedido de interrupciones al detectar un flanco negativo.
- Si esta configurado por nivel entonces selecciona el nivel que provoca interrupción.
  - Por flanco (ISEL = 0)
    - 1 – habilita el pedido de interrupción por flanco negativo
    - 0 – inhabilita el pedido de interrupción por flanco negativo
  - Por nivel (ISEL = 1)
    - 1 – el nivel alto (“1”) es el que provoca interrupción
    - 0 – el nivel bajo (“0”) es el que provoca interrupción

SIENF y CIENF – Clear y Set Register

- Idénticas funciones a SIENR y CIENR, pero para manipular los bits de IENF.
- SIENF
  - Si se escribe “1” en un bit de SIENF entonces escribe “1” en bit correspondiente de IENF.
  - Si se escribe un “0” en un bit de SEINF entonces no hay cambios en el bit correspondiente de INEF
- CEIF
  - Si se escribe “1” en un bit de CEINF entonces escribe “0” en bit correspondiente de IENF.
  - Si se escribe un “0” en un bit de CEINF entonces no hay cambios en el bit correspondiente de INEF

Registros de configuración

Nombre	Descripción
PINTSEL0/1	Identifica bit y GPIO para INT0-3 (SEL0) y para INT4-7(SEL1)
ISEL	0 – por flancos, 1 – por nivel
IENR	1 – habilita flanco ascendente (ISEL = 0) o nivel (ISEL = 1) 0 - inhabilita flanco ascendente (ISEL = 0) o nivel (ISEL = 1)
SIENR	Para escribir solo los 1 en IENR
CIENR	Para escribir solo los 0 en IENR (se escribe 1 para un 0)
IENF	1 - habilita flanco descendente (ISEL = 0) o nivel alto (ISEL = 1) 0 - inhabilita flanco descendente (ISEL = 0) o nivel bajo (ISEL = 1)
SIENF	Para escribir solo los 1 en IENF
CIENF	Para escribir solo los 0 en IENF (se escribe 1 para un 0)

RISE – Rising Edge detector

- Indica la detección de un flanco positivo.
- La detección esta activa independientemente de si se configuró el pedido de interrupción.
- Al leer el registro los “1” indican las líneas donde se detectaron flancos ascendentes.
- El evento se borra escribiendo un “1” donde ya había un “1” al leer el registro.
- Si se escribe “0” no hace nada.

FALL – Falling Edge detector

- Indica la detección de un flanco negativo.
- La detección esta activa independientemente de si se configuró el pedido de interrupción.
- Al leer el registro los “1” indican las líneas donde se detectaron flancos descendentes.
- El evento se borra escribiendo un “1” donde ya había un “1” al leer el registro.
- Si se escribe “0” no hace nada.

Registros de configuración

Nombre	Descripción
PINTSEL0/1	Identifica bit y GPIO para INT0-3 (SEL0) y para INT4-7(SEL1)
ISEL	0 – por flancos, 1 – por nivel
IENR	1 – habilita flanco ascendente (ISEL = 0) o nivel (ISEL = 1) 0 - inhabilita flanco ascendente (ISEL = 0) o nivel (ISEL = 1)
SIENR	Para escribir solo los 1 en IENR
CIENR	Para escribir solo los 0 en IENR (se escribe 1 para un 0)
IENF	1 - habilita flanco descendente (ISEL = 0) o nivel alto (ISEL = 1) 0 - inhabilita flanco descendente (ISEL = 0) o nivel bajo (ISEL = 1)
SIENF	Para escribir solo los 1 en IENF
CIENF	Para escribir solo los 0 en IENF (se escribe 1 para un 0)
RISE	1 - hubo un flanco ascendente, al escribir 1 se borra el evento
FALL	1 - hubo un flanco descendente, al escribir 1 se borra el evento

IST - Interrupt Status Register

- Permite simplificar la rutina de servicio de la interrupción
- Al leerlo indica cuales líneas pidieron interrupción
- Cuando los pedidos son por flancos, al escribirlo borra automáticamente el pedido adecuado (RISE o FALL)
- Cuando los pedidos son por nivel, entonces cambia el nivel activo alterando el bit correspondiente del IENF

Registros de configuración

Nombre	Descripción
PINTSEL0/1	Identifica bit y GPIO para INT0-3 (SEL0) y para INT4-7(SEL1)
ISEL	0 – por flancos, 1 – por nivel
IENR	1 – habilita flanco ascendente (ISEL = 0) o nivel (ISEL = 1) 0 - inhabilita flanco ascendente (ISEL = 0) o nivel (ISEL = 1)
SIENR	Para escribir solo los 1 en IENR
CIENR	Para escribir solo los 0 en IENR (se escribe 1 para un 0)
IENF	1 - habilita flanco descendente (ISEL = 0) o nivel alto (ISEL = 1) 0 - inhabilita flanco descendente (ISEL = 0) o nivel bajo (ISEL = 1)
SIENF	Para escribir solo los 1 en IENF
CIENF	Para escribir solo los 0 en IENF (se escribe 1 para un 0)
RISE	1 - hubo un flanco ascendente, al escribir 1 se borra el evento
FALL	1 - hubo un flanco descendente, al escribir 1 se borra el evento
IST	Lectura: 1 - indica un pedido de interrupción en la línea GPIO Escritura: 1 - borra el pedido de interrupción (ISEL = 0) Escritura: 1 - cambia el nivel para pedir interrupciones (ISEL = 1)

Resumen de funcionamiento

ISEL	IENR	IENF	Causas de Interrupción
0	0	0	Sin interrupciones
0	1	0	Flanco Ascendente
0	0	1	Flanco Descendente
0	1	1	Flanco Ascendente y descendente
1	0	0	Sin interrupciones
1	0	1	Sin interrupciones
1	1	0	Nivel Bajo
1	1	1	Nivel Alto

Direcciones de los registros

- Los registros PINSEL corresponden al SCU
- Todos los demás en direcciones seguidas
- Solo se usan los bits 0 al 7, uno por pin.
- Se puede usar un puntero a ISEL y de allí direccionar con desplazamiento.

Dirección	Indice	Registro
0x4008 7000	1	ISEL
0x4008 7004	2	IENR
0x4008 7008	3	SIENR
0x4008 700C	4	CIENR
0x4008 7010	5	IENF
0x4008 7014	6	SIENF
0x4008 7018	7	CIENF
0x4008 701C	8	RISE
0x4008 7020	9	FALL
0x4008 7024	10	IST

Sobre la rutina de servicio

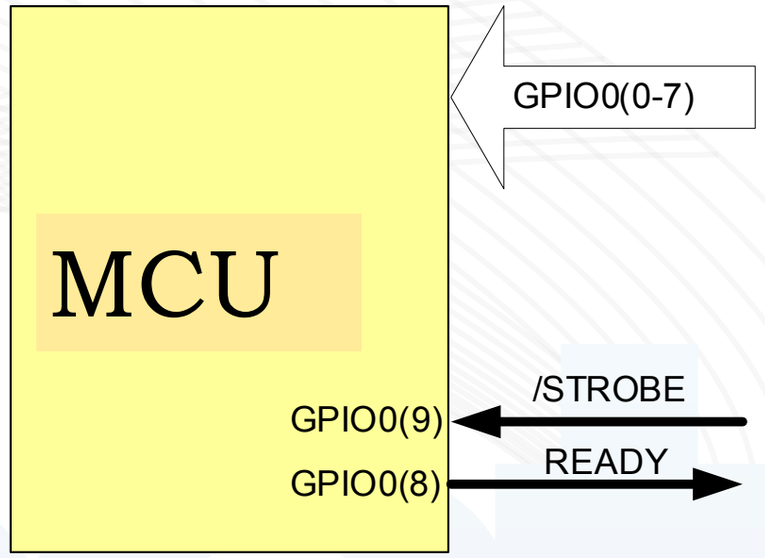
- Cuando trabaja por flancos se debe borrar el evento escribiendo en IST
- Debe ser lo primero para evitar perder eventos muy seguidos
- Cuando se trabaja por nivel no hace falta borrar el evento
  - Si al terminar la rutina sigue en nivel activo nueva interrupción.
  - Si es escribe en IST entonces se cambia el nivel activo.

Métodos de E/S

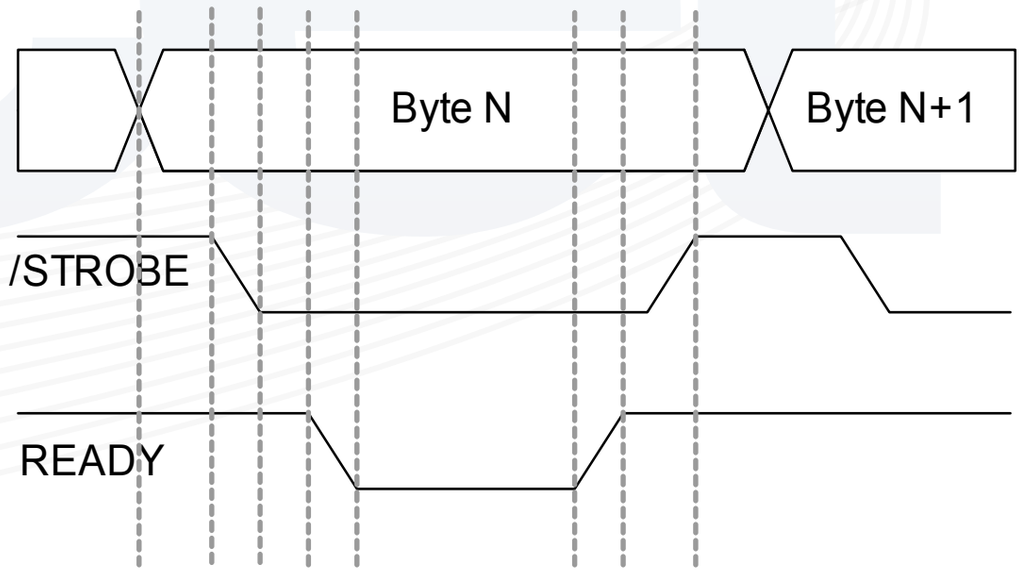
- Existen diversas maneras de manejar las entradas/salidas desde el microprocesador.
- **Polling (Encuestas)**: Consiste en testear permanentemente una señal de status, hasta que esta tome un valor. Entonces se ejecuta la E/S.
  - Requiere que el programa principal se ocupe de testear.
  - Puede testear continuamente y no se puede hacer más nada.
  - Puede testear periódicamente sin que pare el programa principal.
  - Si está bien diseñado, atiende más rápido una señal.
- **Interrupción**: Cuando la señal de status toma un valor, el programa se interrumpe para manejar la E/S.
  - Mucho más complicado de reproducir para probar.
  - El programa principal está libre para realizar otras tareas.
  - No hay desperdicio de tiempo testeando.
  - Mayor sobrecarga (overhead) pero solo cuando es necesario.

Ejemplo de E/S Asincrónica.

- Se ingresa un byte en paralelo al MCU vía GPIO0[7:0].
- Se ocupan dos líneas: GPIO0[8] para Ready y GPIO0[9] para /Strobe.



- Inicio: /STB=RDY=1 - Listo
- Perif: /STB=0 – Dato listo.
- µC: RDY=0 - Dato reconocido
- µC: RDY=1 - Dato recibido
- Perif: /STB=1 – Próximo dato

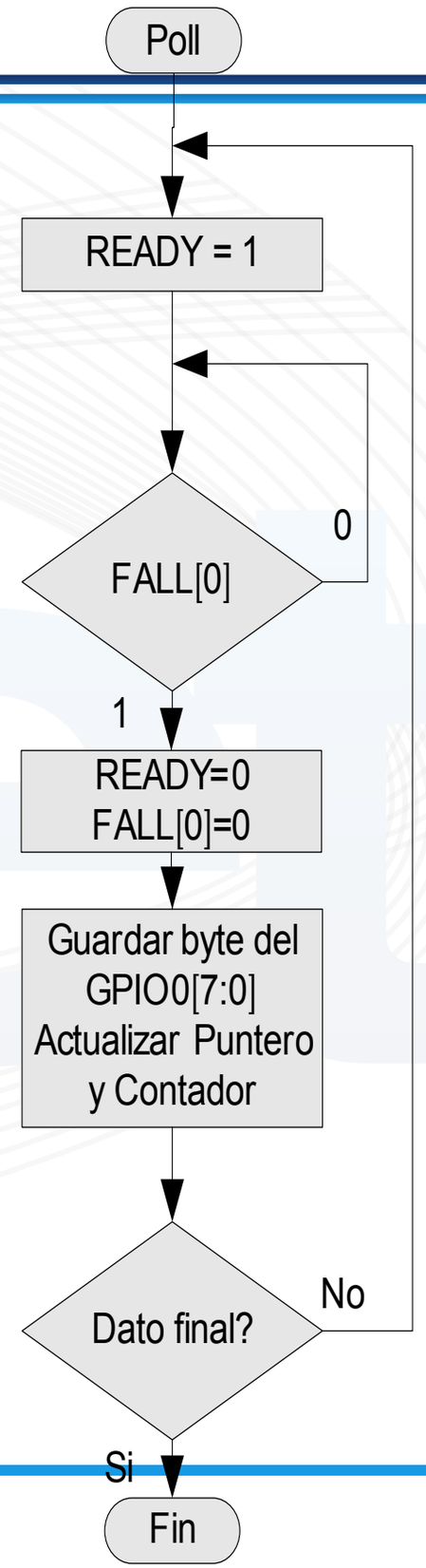


Inicialización: INT0 y GPIO_0

- GPIO_0[7,0] modo input.
- GPIO_0[8], ready, output.
  - Nombre: “Ready”, bit banding.
  - **¿Cómo doy un nombre?**
- GPIO_0[9], /STB, input, pin 0 IRQ
  - por flanco negativo, sin interrupciones
  - por polling, se chequea FALL[0]
- Enmascarar todo el puerto, menos 10 lsb. MASK_0.

Entrada por Polling

- Estructura de Datos
  - Puntero: a una tabla de **bytes.**
  - Contador: n datos a tomar
- Aunque no haya interrupciones, es posible utilizar el estado del flag Falling edge: FALL[0]
- Para referenciar Ready usaremos bit-banding y EQU.
- Para reset de FALL[0], se escribe 1 allí.



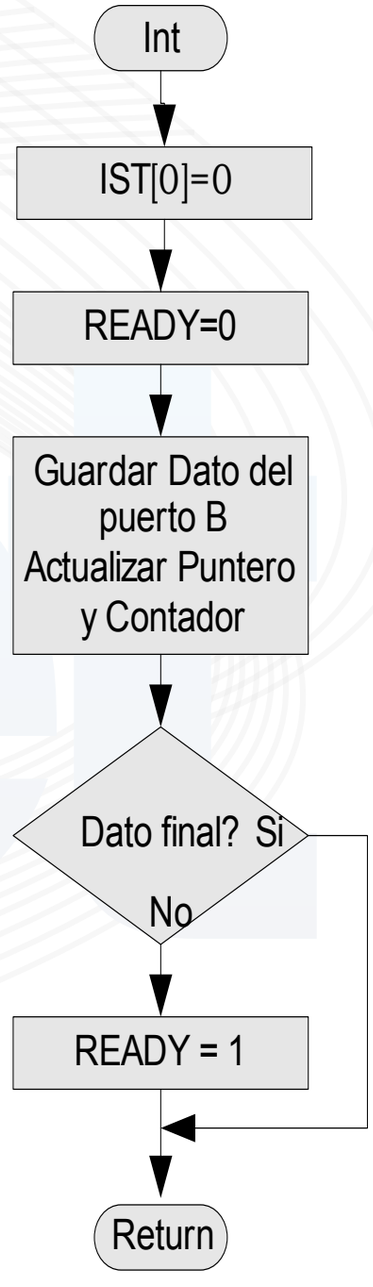
Programa Polling

```
In_P    LDR     R0,=MPIN_GPIO_0    //Suponemos GPIO0 inicializado.
        LDR     R1,=puntero        //inicialiación de variables
        LDR     R1,[R1]
        LDR     R2,=contador
        LDR     R2,[R2]
        LDR     R3,=FALL           //Falling edge status[0]
        LDR     R4,=Ready
        MOVE    R6,#1              //valores iniciales
        MOVE    R7,#0
Pol0    STR     R6,[R4]            //2T Rdy=1
Pol1    LDRB    R5,[R3]            //2T Revisar FALL[0]
        TST     R5,#1              //1T
        BEQ     Pol1               //1T (predice)
        STRB    R6,[R3]            //2T FALL[0]=0 – no cambia el resto
        STR     R7,[R4]            //2T RDY=0
        LDRB    R5,[R0]            //2T INPUT DATA
        STRB    R5,[R1],#1         //2T guardar dato e incr puntero
        SUBS    R2,#1              //1T decr. Contador y cambia flags
        BGE     Pol0               //1T (predice) Si cont<>0 otro poll
        …
```

Máx. Transfer Rate: 1 B/16T

E/S por Interrupciones

- Usando interrupciones, es muy simple manejar una rutina de E/S asincrónica.
- El programa principal debe ser responsable de inicializar:
  - Inicialización en general
  - Habilitar la interrupción
  - Variables globales
    - Puntero inicial a tabla **de bytes.**
    - Dimensión de la tabla (Contador)
  - RDY = 1
- R0-R3 y R12 libres para usar.
- Atención: Fin de Bloque - no se pone Ready=1. ¿Por qué?.



Programa Interrupción

```
In_I    LDR     R0,=IST             //2T Puntero IST
        MOVE    R1,#1               //1T
        STR     R1,[R0]             //2T IST[0]=0 – no cambia el resto
        LDR     R3,=Ready           //2T Puntero Ready
        MOVE    R2,#0               //1T
        STR     R2,[R3]             //2T RDY=0
        LDR     R0,=MPIN_GPIO_0 //2T
        LDRB    R2,[R0]             //2T Ingresa Datos
        LDR     R0,=puntero         //2T R0 dirección puntero
        LDR     R1,[R0]             //2T R1 apunta a Datos
        STRB    R2,[R1],#1          //2T guardar dato e incr puntero
        STR     R2,[R0]             //2T guardar puntero
        LDR     R0,=contador        //2T apunta a contador
        LDR     R2,[R0]             //2T
        SUBS    R2,#1               //1T decr. Contador y cambia flags
        STR     R2,[R0]             //2T guardar contador
        IT      NEQ                 //1T calcula condición
        STRBNEQ R1,[R3]             //2T Rdy=1 (no se cuenta su tiempo?)
Fin:    BX      LR                  //1T (predice) Retorna
```

Duración 31T en forma sostenida. No sostenida: sumar 12+10T →53T.

Conceptos

- Cuando llega una Interrupción, se termina una instrucción y se produce un efecto similar al salto (3T) que está incluido en los 12T.
- Latencia Interrupción:
  - 12T Cambio de Contexto.
  - 10T Retorno de Interrupción.
  - Si en ISR viene otro pedido, queda pendiente y no se quita estado del stack, si esto es permanente: latencia=0

Comparación Polling Vs. Int.

- Cuando el periférico es más lento que el CPU: “I/O bounded”
  - Caso contrario: “CPU bounded”.
- ¿Cuál es el caso más común?
- El ingreso de un byte en el caso más común dura:
  - 16T en polling
  - 53T en interrupción (overhead).
- La ventaja es que con interrupciones el CPU está libre para otras tareas.

Tasa Máxima de Transferencia

- Tasa máxima a la que el dispositivo puede transferir datos al MCU sin perder ninguno.
- El tiempo que tarda el MCU en procesar un dato es $t = T_o + T_d$.
  - T_d (tiempo de dato) es el tiempo para poner un dato en Memoria (idealmente 1 ciclo de M= 2T)
  - T_o (tiempo de overhead) es el tiempo que se tarda en detectar la señal, saltar al código adecuado, manejar la linea READY, mantener los punteros, buscar instrucciones, etc.

Tasa Máxima en Polling

- En caso de tasa máxima, el lazo de detección solo se ejecuta una vez por cada dato que llega, más rápido se pierden datos.
- Los tiempos son:
  - T_d= 4 T
  - T_o= 12 T
  - total= 16 T.
- Si f(bus)=32 MHz
  - T=31,25 ns.
- Tasa máxima=2 MB/s
- Eficiencia=4/16 =25 %
  - Trabajo útil / Trabajo Total.
- En transferencia, los MB son potencias de 10, 1K=10^3

```
//*********LAZO POLLING***********
Pol0    STR     R6,[R4]          //2T
Pol1    LDR     R5,[R3]          //2T
        TST     R5,#1            //1T
        BNE     Pol1             //1T
        STR     R6,[R3]          //2T
        STR     R7,[R4]          //2T
        LDRB    R5,[R0]          //2T
        STRB    R5,[R1],#1       //2T
SUBS    R2,#1            //1T
        BGE     Pol0             //1T
```

Tasa Máx. con Interrupciones

- Con IRQ=0, el CPU termina la instrucción actual, guarda el estado, y salta al lugar escrito en su vector de interrupción.
  - Con tasa máx. termina una rutina de INT y comienza otra.
  - Con Cortex-M4 latencia IRQ: 6 ciclos.
- Los tiempos son, a 32 MHz (PCLK):
  - T_d= 4 T
  - T_o= 33 T
  - Total = 37 T = 1,156 ms.

  Tasa máxima 0,86 MB/s
- Eficiencia = 4/37 = 10,81%
  - Menos de la mitad que en Polling
  - Solo en Tasa Máxima y gracias al CORTEX (latencia repetitiva=6T)

Modo Bloque

- La Tasa Máx. de Transf. con Polling es muy superior que con Interrupciones.
  - Sin embargo, con interrupciones se permite al CPU realizar otras tareas, mientras que el polling (como lo hemos diseñado) no.
  - Desde el punto de vista de la eficiencia, la verdadera diferencia está en el Tiempo de Overhead, mucho mayor.
- Para mejorar la eficiencia con ***dispositivos rápidos*** se ingresan ***bloques*** o ***ráfagas*** de n bytes con cada interrupción.
- El dispositivo pide INT y luego transmite el bloque completo.
  - De esta manera, gran parte del overhead solo se ejecuta una vez cada n datos y mejora la eficiencia.

Resultado

- En ISR se escribe un lazo para todo el bloque.
  - Similar al de polling.
  - Se ejecuta n veces, n=cantidad de datos.
- Cuando se calcula las medidas de performance y eficiencia por byte
  - Todo lo que se ejecuta una única vez es despreciable frente a n veces del lazo.
  - Resultado igual al de polling.
- Desventaja adicional
  - No se recomienda ISR largas.

Limitaciones de E/S.

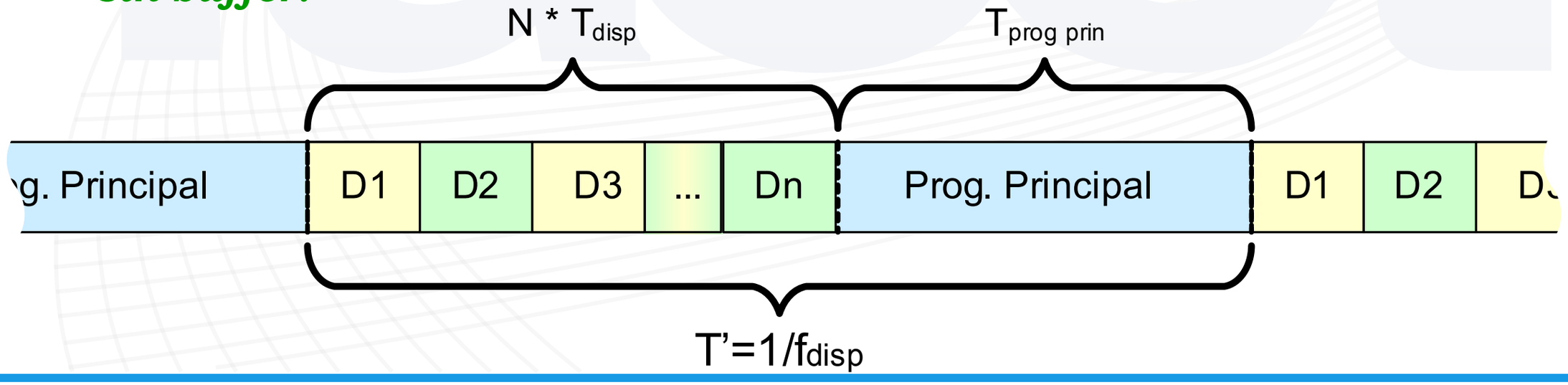
- Dispositivos Rápidos:
  - Existe un límite superior de la tasa de envío de datos, a partir de allí se pierden datos.
- ¿Dispositivos Lentos no tienen limitaciones?
- ¿Provocan algún otro problema?

Múltiples Dispositivos Lentos

- Sean n dispositivos que transfieren datos
  - Todos a igual tasa: f_{disp}
  - Todos consumen igual tiempo de CPU, T_{disp}=D (Identificación por Hw).
- ¿Efecto en la performance con que corre el programa en el CPU?
- ¿Existe un límite a la cantidad de dispositivos que puedo conectar?

Performance con múltiples dispositivos

- El tiempo necesario para hacer correr un programa resulta
1. Sin interrupciones, $T = T_{progprinc}$
2. Con interrupciones, $T' = T_{progprinc} + n.T_{disp}$ con $T'=1/f_{disp}$
- Performance de un programa inversamente proporcional a su tiempo de ejecución.
  - De (1) y (2) resulta $T/T' = 1 - n.(T_{disp}/T') = 1 - n.D.f_{disp}$
  - La máxima cantidad de dispositivos a conectar sale igualando la performance a 0 → $T/T'=0$:$f_{disp} = 1/n.D$ o $T'=n.D$
    - En este caso, el CPU no ejecuta el programa principal.
    - Más dispositivos o más tiempo para cada uno implica que se perderán datos, ***con o sin buffer.***



Ej. Efecto de dispositivos lentos.

- Para Interrupciones: MPU a 32 MHz en el bus y 20 dispositivos a igual tasa =10 KB/s.
- El mejor caso: todos las interrupciones llegan juntas.
- f_{disp}=10 KB/s, n = 20.
- T_{disp}= 37 T= 1,15 μs (31 ya visto).
- $T/T' = 1 - n.f_{disp}.T_{disp} = 0,77$
- ***¡La performance bajó un 23%! En el mejor caso.***
- $n_{max} = 1/f_{disp}.T_{disp} =$ ***86***.
- Gracias a sistema interrupciones ARM, sino peor

En caso de Polling

- Se realizan encuestas periódicas y así queda tiempo para correr el programa.
- Se debe realizar encuestas al ritmo máximo de ingreso de datos.
  - Difícil si no se cuenta con una interrupción periódica (SysTick por ejemplo).
  - La duración de cada encuesta depende de cuántos periféricos hay y cuantos datos se ingresan.
  - Puede haber periféricos que no estén listos (lentos).
- Así se calcula el tiempo en que el CPU no ejecuta el programa principal.
- El resto del cálculo es similar que en Int.

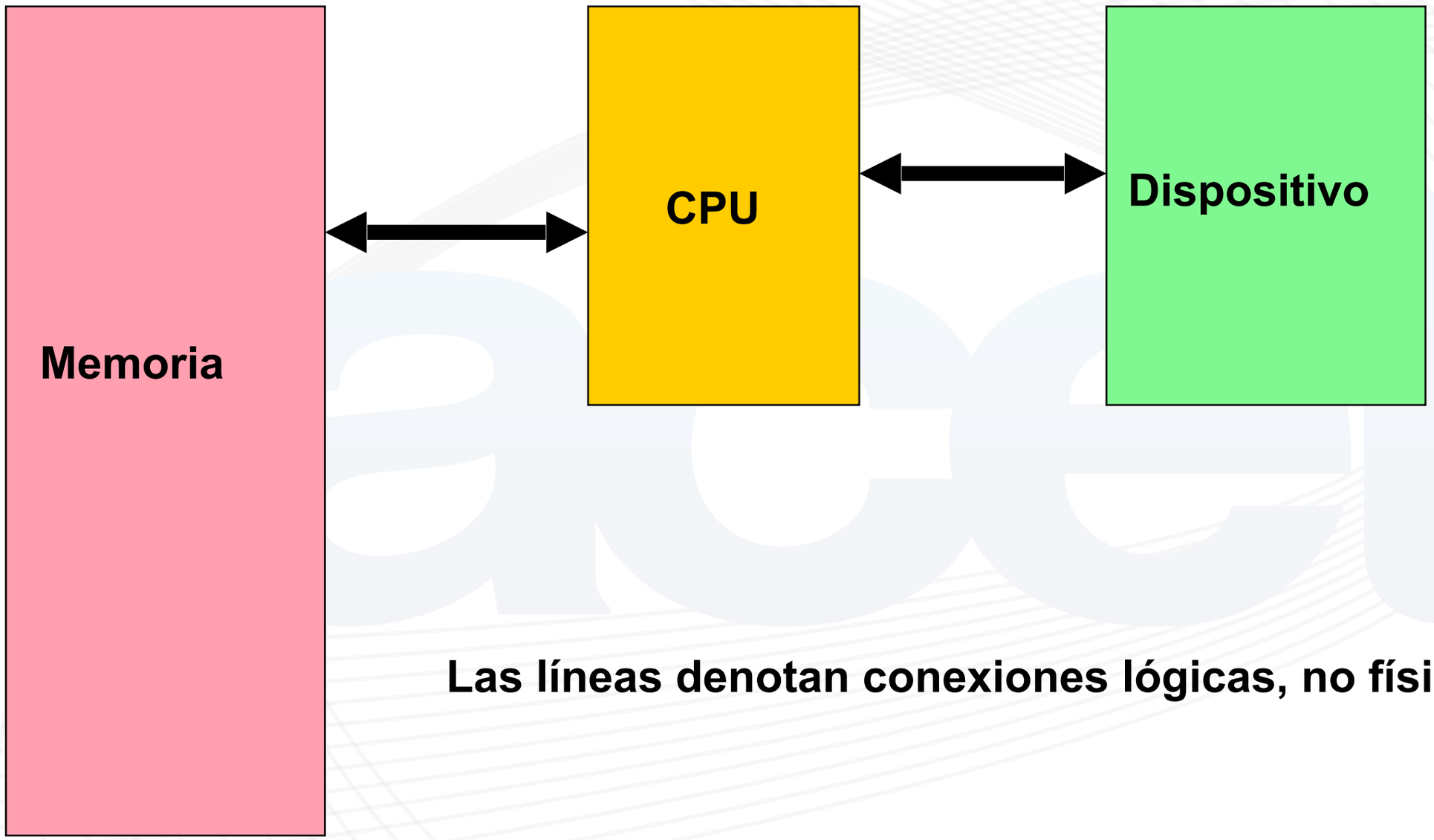
Limitaciones de E/S

- En conclusión:
  - Hay un límite en velocidad de transferencia para dispositivos rápidos.
  - Hay un límite en cantidad de dispositivos lentos que se pueden conectar.
    - En ambos casos el Programa Principal corre más lento.
    - ¡CUIDADO CON Watchdogs!
- Nos interesa incrementar estos límites.
- Dos alternativas:
  - Fuerza Bruta – transf más anchas o mayor frecuencia CPU. Ya lo hemos analizado.
  - Diseño (arquitectura).

Problemas Métodos Anteriores

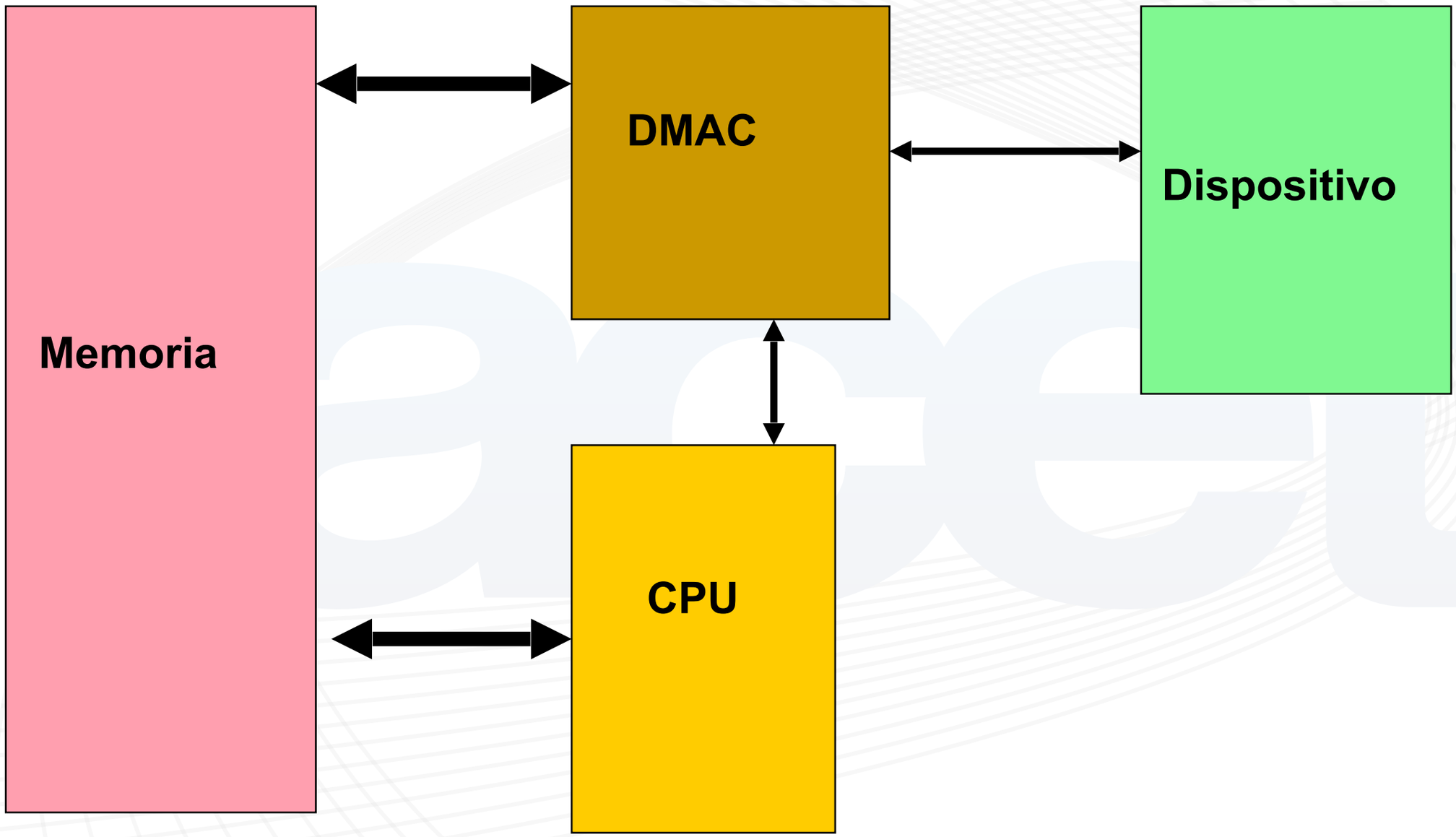
- El CPU es el cuello de botella: corre el programa para transferir datos, y otros.
- Cada vez que ingresamos un dato, tenemos que ejecutar múltiples instrucciones.
- Cada una tiene su propio Fetch. Por tanto son muchos accesos a memoria
- En realidad la transferencia de un dato requiere, al mínimo un único acceso a M mejor caso→1T.
- Gran diferencia entre lo que tenemos (16 a 53T) y lo mínimo (1T).

Problema: CPU es Intermediario

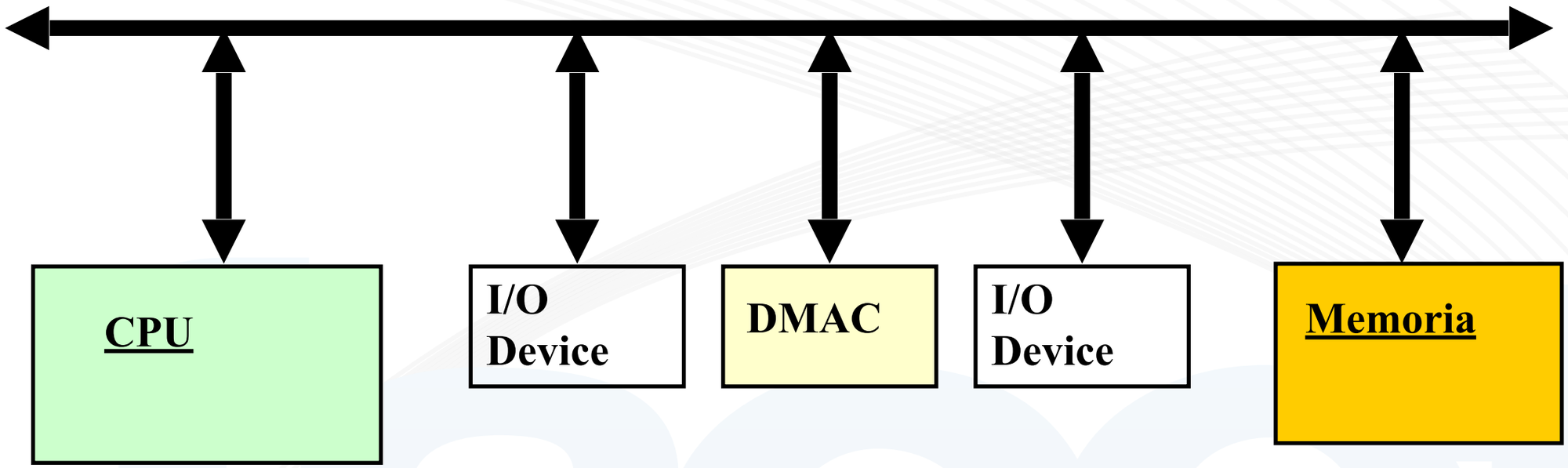


Las líneas denotan conexiones lógicas, no físicas.

Solución: Acceso Directo a M: DMA

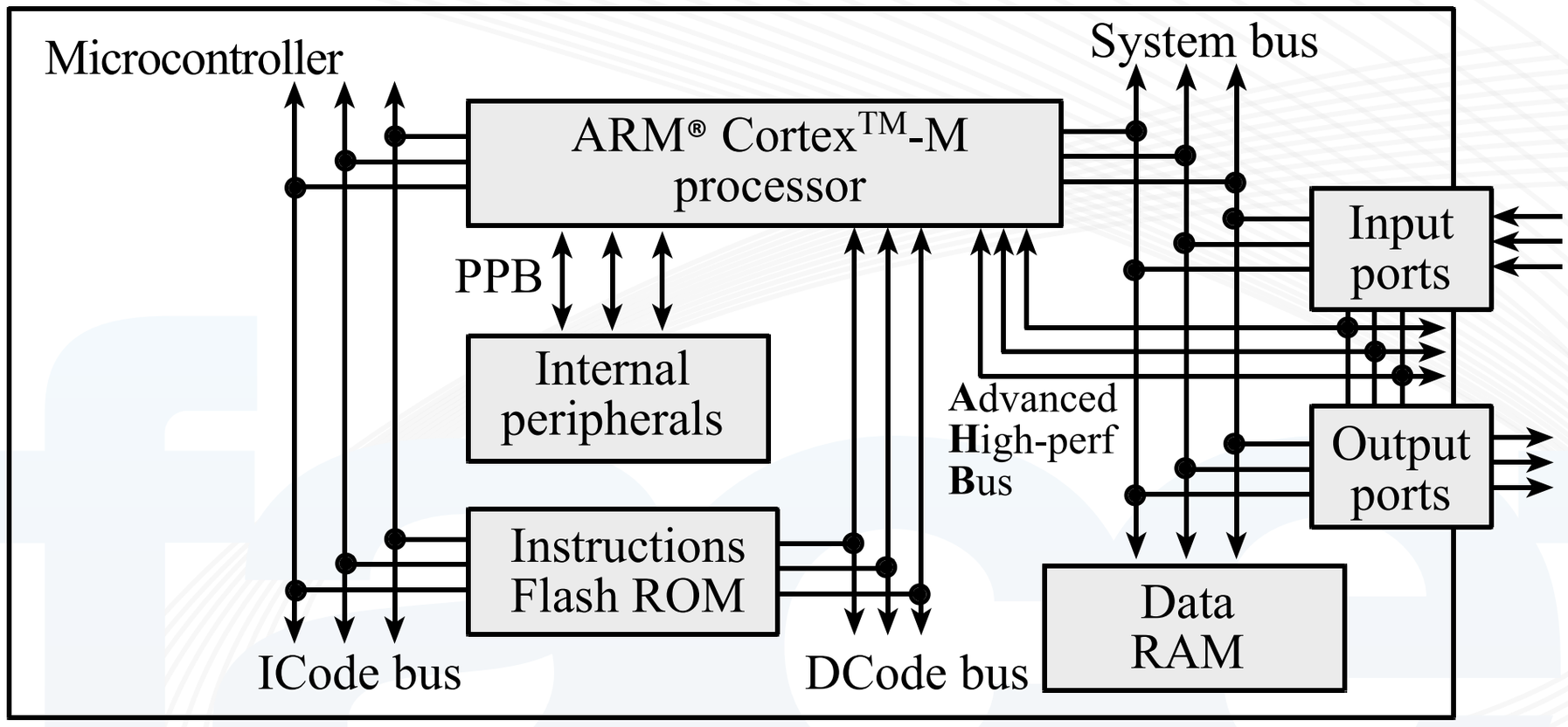


Bus en la práctica



- Un bus es un conjunto de cables.
- Necesita un maestro que lo maneje.
  - Pone todas las señales (AB,DB,CB)
  - El resto son esclavos.
- Otro dispositivo puede ser maestro.
  - No puede haber más de 1 maestro simultáneo.
  - ¿Por qué? ¿Y si tuviera dos buses?

Arquitectura Buses Cortex-M



- El CORTEX-M4 implementa varios buses.
- Permite leer instrucciones y datos al mismo tiempo (Harvard).
- También realizar varias transferencias de datos en paralelo (varios bancos de M, periféricos)

Controlador DMA (DMAC)

- Un dispositivo controlador (DMA Control) realiza las siguientes funciones:
  1) El DMAC pide ser maestro del Bus
  2) El DMAC se adueña (Master) del bus y provee todas las señales y la dirección en donde se debe realizar la transferencia del dato en M.
  3) Una vez finalizada la transferencia, DMA sale del BUS y otro (ej: CPU) se hace dueño del mismo.

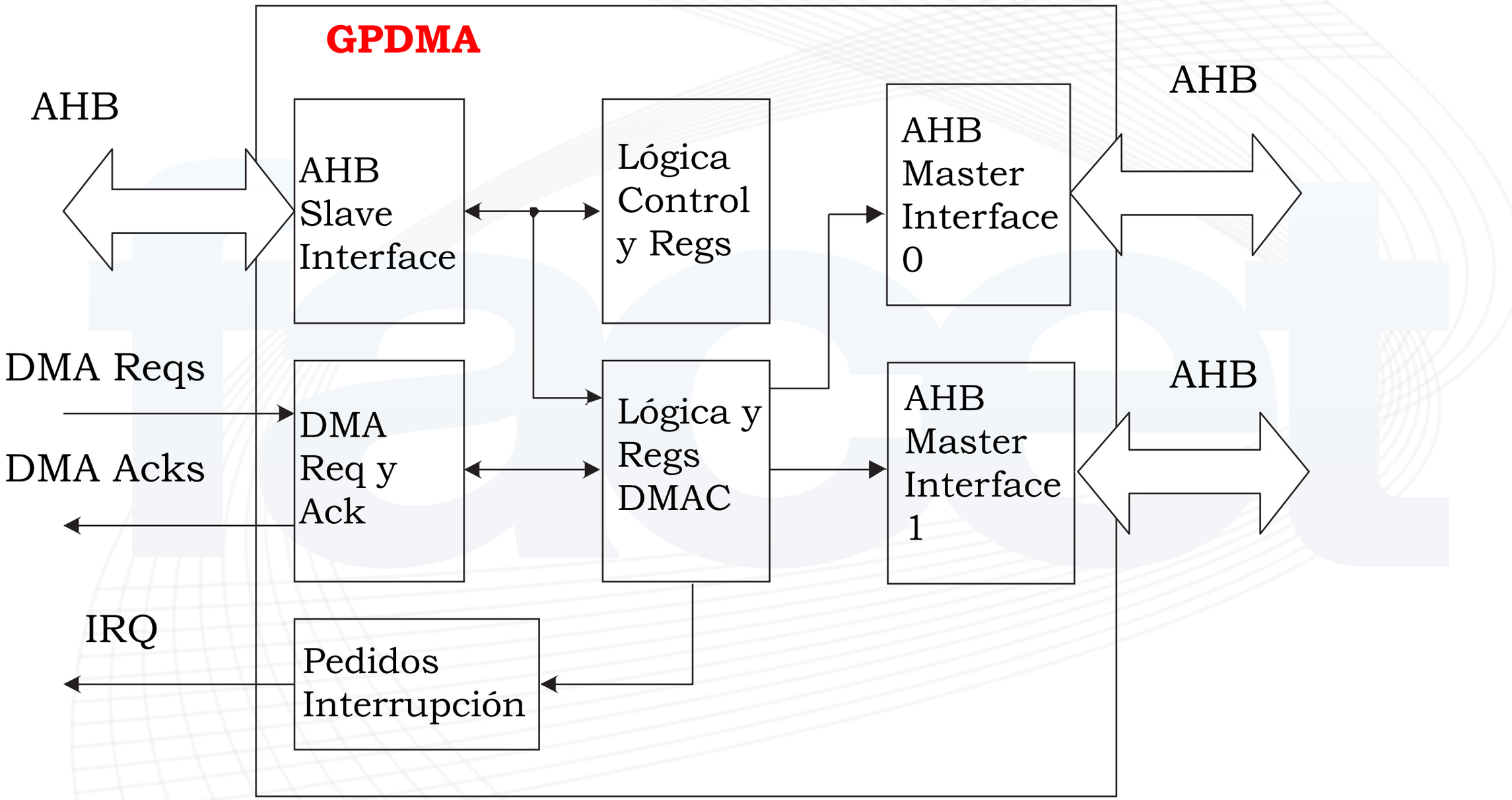
Comunicación DMA-CPU

- ¿Cómo sabe el DMA dónde poner los datos?
- El CPU lo configura inicialmente.
  - Para transferir un bloque de datos
- DMAC transfiere (lo que pueda en paralelo)
  - Cuando DMAC termina o si tiene un problema
    - Interrumpe y avisa al CPU.
- Hay casos de DMAC en que se puede configurar una “cadena” de bloques de datos
  - Con un puntero a ID de bloque siguiente
  - El DMAC lee el nuevo ID de bloque de Memoria, sin requerir interrupción y continúa.

Registros DMAC

- Puntero a fuente: periférico.
- Puntero incrementable a destino: memoria.
- Puede ser al revés: de Mem a periférico.
- También las otras dos combinaciones.
  - M a M requiere ambos punteros incrementables
  - P a P requiere ambos fijos.
- Contador de cantidad de bytes a transferir.
- Se configuran desde el CPU, por programa.
- Son los mismos requerimientos que usan las rutinas de E/S por Polling o por Interrupciones
  - La naturaleza del problema es el mismo.
  - Solo varía la forma de implementarlo.

GPDMA – NXP LPC 43xx



En LPC4337 GPDMA

- DMA de propósito general.
- Descripción simplificada: mayoría de los casos.
- Se configura para transferir datos de uno entre16 posibles periféricos.
- Puede configurarse en modo byte o modo bloque.
- Señales
  1. De entrada: DMA_Req – de un periférico.
  2. De salida: DMA_Ack – a un periférico.
  3. Solo para iniciar. Buses cuentan con todas las señales de sincronización.
  4. Pedido de Interrupción a CPU.

GPDMA (II)

- Registros DMA
  - ID de Fuente y Origen.
  - Punteros (direcciones) (2).
  - Cuenta.
  - Puntero a Próximo bloque (si existe).
- Registros de control o configuración
  - Habilitación
  - Modos de trabajo (Dato / Ráfaga).
  - Interrupciones
    - Fin de cuenta
    - Error.

DMA Modo Bloque - Secuencia

- El CPU configura al DMA: cuántos datos va a tomar, de dónde y dónde los va a poner (PUNTs y CONT). Lo habilita.
- El periferico solicita comenzar al DMA: requiere enviar datos: DMA-BREQ (burst request).
- El DMA se adueña de un bus del AHB. ¿Qué hace el CPU?
- El DMA permite transferencias directas entre Periférico y M e incorpora las direcciones (a partir de sus punteros) y señales de control (AB y CB): [P→M, M→P, P→P, M→M]
- Cuando CONT=0, DMA pone DMA-ACK para informar fin al dispositivo.
- El DMA deja de ser master del bus.
- El DMA pide interrupción al CPU.
- La rutina de interrupción de CPU
  - pone FLAG=1 para indicar al Programa Principal que hay Datos a Procesar.

DMA Modo Dato – Secuencia.

1. Programa Principal Inicializa DMA cargando Contador y Punteros y lo habilita a funcionar.
2. Periférico Pide DMA-SREQ para enviar un solo dato.
3. DMA se adueña de un Bus del AHB. Pone DMA-Ack.
4. DMA Control permite al Periférico enviar un único Dato y maneja AB y CB.
   - Interfase DMA a Bus contiene señales de sincronización.
   - Actualiza contador y Puntero/s.
5. DMA quita ACK al periférico.
6. DMA deja de ser master del bus.
7. Si CONT > 0 espera otro pedido DMA-SREQ y vuelve a 3.
8. CONT=0, DMA:
   - Pide INT al CPU.
   - CPU ejecuta interrupción
   - FLAG=1 avisa al programa principal que la transferencia terminó.

Límites DMA Modo Bloque

- $\mathbf{T_o}$ es el tiempo para adueñarse del bus.
  - Despreciable con n grande.
- ¡¡ Para cada Byte se requiere 1T !!
  - Es el límite dado por fbus.
  - Imposible más rápido con ese bus.
- El CPU puede continuar
  - En paralelo si dispone bus y banco M libre.
  - Intercalándose siempre que el Periférico no requiera el bus (su tasa no es tan alta).
- Tasa máxima a 32 MHz = 32 MB/seg. (Vs 1,1 MB/s modo bloques).
  - CPU parado cuando es interrupción por bloques
- Esta transmisión es sincrónica en bloques – ¿Trampa?

DMA Modo Dato.

- Difícil calcular el tiempo.
  - Latencia para tomar y soltar el bus.
  - Si CPU sale del bus, latencia de salida y de entrada.
  - Supongamos 6T
- Si cada transferencia durara 7T (Vs. 53T o 16M).
  - Y si hay paralelismo mucho menor, equivale a < 7T.
- Cuando no hay paralelismo, se dice que con cada transferencia DMA roba ciclos de bus al CPU.
  - A este modo se le llama también **“Robo de Ciclo”**.
- **Repetir los mismos cálculos que ya hicimos y comparar con interrupciones.**
  - **Tasa de transferencia máxima.**
  - **Eficiencia.**
  - **% de caída de performance del CPU en el peor caso.**

DMA Cortex-M4 LPC4370

- En realidad el MPU cuenta con 8 DMA's.
  - Se dice que tiene un solo DMA y 8 canales.
  - Canal = DMA con léxico de IBM.
- Se pueden asignar prioridades a los canales para tomar el bus.
- El de mayor tasa de transferencia menor prioridad.
  - Siempre que los datos no se pierdan.
  - Transferencias: M-M, M-P, P-M, P-P.
  - Múltiples transferencias a bloques distintos de M.
  - Punteros fijos a Perif o incrementables a M.
  - Tamaño de ráfaga configurable.
  - Buffer de 4 palabras por canal.
  - Ancho de palabra configurable (B, HW, W).
  - Soporta Big Endian o Little Endian.
  - Varios registros de configuración.