Robótica Inteligente
Programación

Marco Antonio López Trinidad
Luis Enrique Sucar Succar
Departamento de Computación

Control en tiempo real

- Tres estrategias
  - ***Polling o poleo***
  - ***Manejo por interrupción***
  - ***Input capture***

Control en tiempo real

- ***Polling o poleo***, método donde el software esta ciclado y continuamente esta verificando una entrada. Se apropia del funcionamiento de la computadora

Control en tiempo real

- ***Manejo por interrupción***, permite implementaciones mas eficientes de software. En este método un evento externo genera una señal que indica al procesador posponer cualquier tarea que este realizando y responder al evento inmediatamente

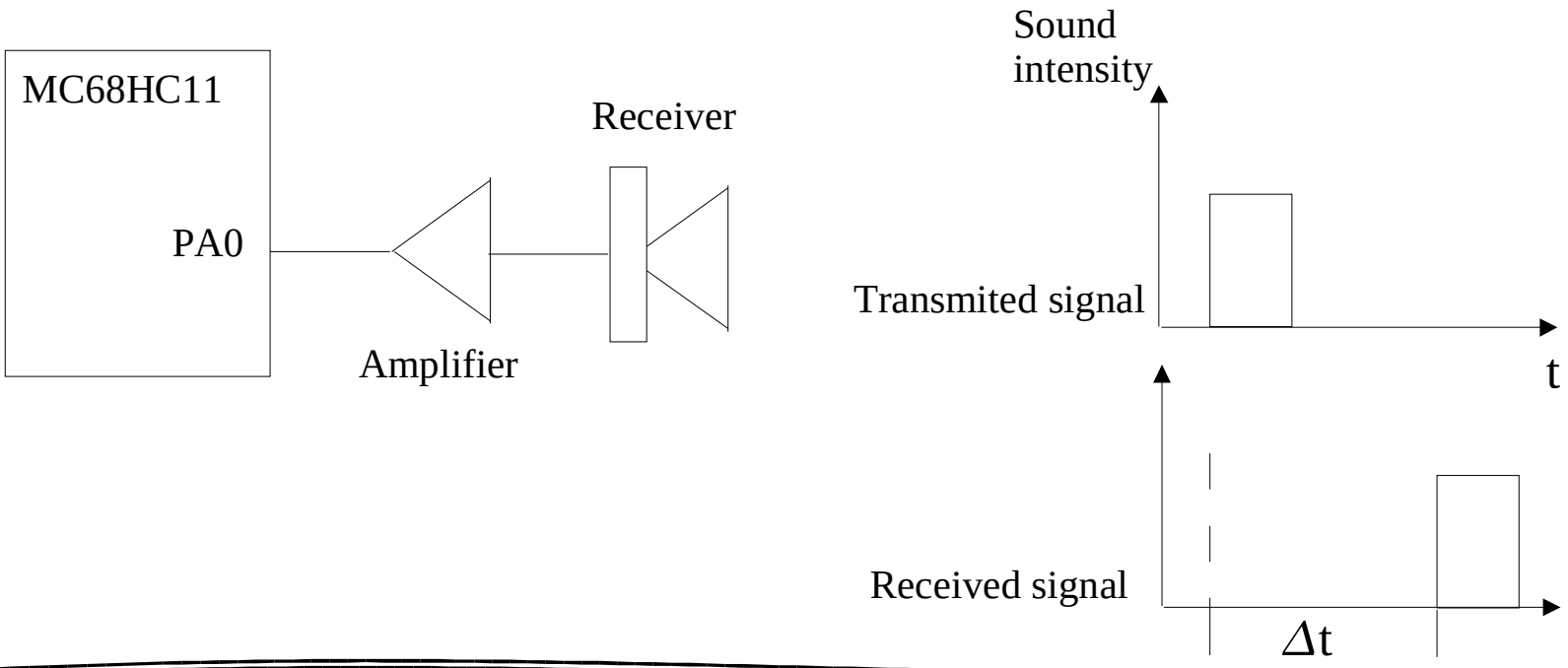
Control en tiempo real

- ***Input capture***, solamente si el microcontrolador cuenta con hardware especial para capturar eventos. Con esta estrategia no se interrumpe el trabajo del procesador

Control en tiempo real

- Ejemplo de polling

Se quiere medir el tiempo de vuelo de un pulso de sonar



Control en tiempo real

- Se requiere medir la diferencia de tiempo entre la emisión del pulso y la detección del rebote de la señal

Control en tiempo real

```
TIME-SONAR  ;Mide el tiempo de vuelo de un eco de sonar
    LDD TCNT            ;Obtiene el tiempo inicial del timer del sistema
    STD Sonar-tof       ;Guarda el tiempo de inicio
WAIT-FOR-ECHO
    BCLR PORTA, %00000001, WAIT-FOR-ECHO  ;Leer PA0
    LDD TCNT            ;Se detecto un eco y se registra el tiempo
    SUBD Sonar-tof      ;Se obtiene el tiempo de vuelo
    STD Sonar-tof       ;Se guarda el tiempo en un registro de 16 bits
    RTS                 ;Regresar a la ejecución normal del programa
```

Control en tiempo real

- El código principal que llama y responde a la subrrutina TIME−SONAR podría ser de la siguiente forma:

```
JSR     Turn−on−sonar     ;Inicializa el disparo del sonar
JSR     TIME−SONAR        ;Salto a nuestra subrrutina
JSR     Compute−distance  ;Utiliza el tiempo medido para
calcular la distancia
```

Control en tiempo real

- Interrupciones
  - Son eventos que disparan automáticamente una respuesta en un microprocesador
  - El código que responde a un evento se llama rutina de servicio a interrupción
  - Las subrrutinas de servicio a interrupción son similares a las subrrutinas, sin embargo estas solo se ejecutan cuando ocurre un evento, pero no con la instrucción JSR

Control en tiempo real

- Las interrupciones son asíncronas y un microprocesador no puede anticiparse a la ocurrencia de un evento
- Para atender a una interrupción, un microprocesador debe suspender la ejecución del código actual y guardar el estado de sus registros en el Stack
- Posteriormente debe localizar la subrrutina de servicio que corresponde al evento y transferir el control a esa subrrutina

Control en tiempo real

- Al terminar la ejecución de la subrrutina de servicio, el microprocesador debe recuperar el estado de sus registros que tenia antes de ser interrumpido

Control en tiempo real

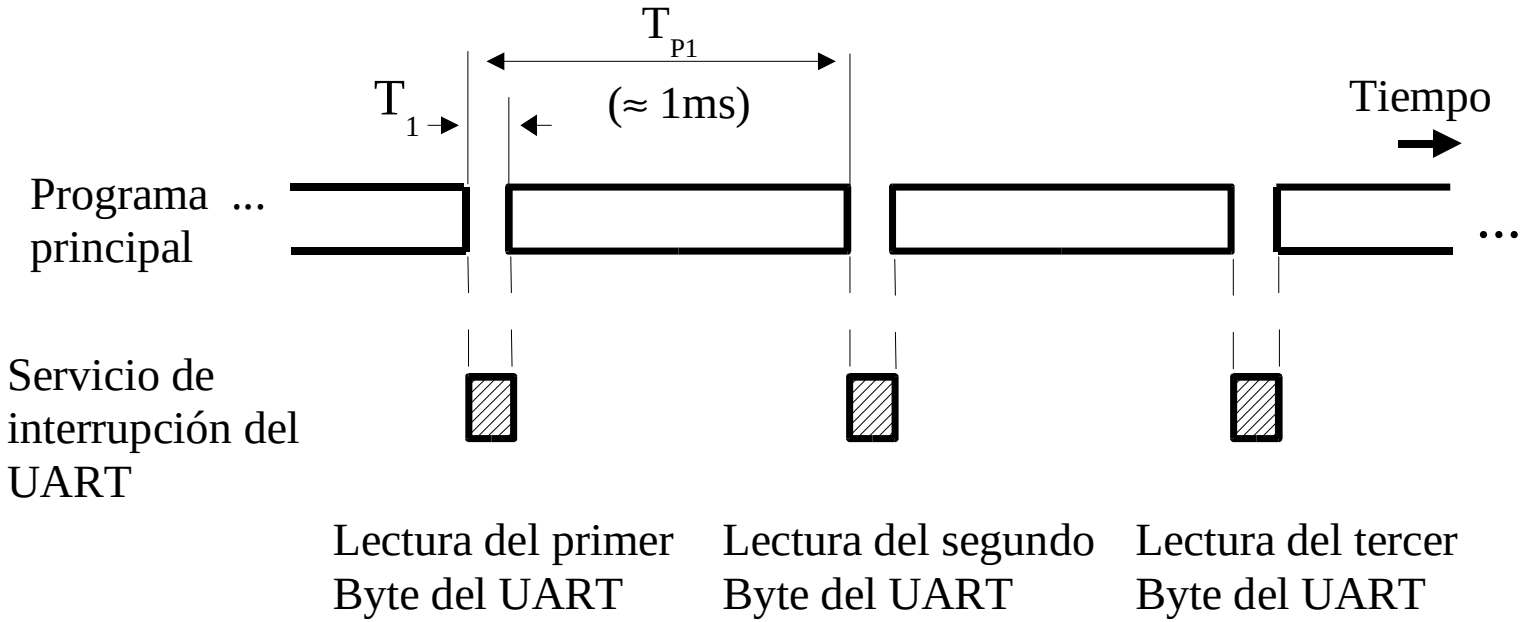
- Consideraciones para el manejo de interrupciones
  - Microcontrolador
    - El CPU debe tener el control en cualquier momento para deshabilitar interrupciones, ejecutar unas cuantas instrucciones y rehabilitar las interrupciones
    - El CPU debe permitir activar algunas interrupciones y deshabilitar el resto

Control en tiempo real

- Consideraciones para el manejo de interrupciones
  - Programador
    - Rápido servicio a el dispositivo que esta interrumpiendo
    - Minimizar cualquier retraso que ocurra, antes de proporcionar servicio a alguno de los dispositivos que al mismo tiempo soliciten servicio

Control en tiempo real

- Ejemplo: Leer un mensaje de 20 Bytes de un dispositivo conectado al UART configurado a un baud rate de 9600



Control en tiempo real

- Esta operación utiliza 20 pedazos de tiempo de CPU durante un periodo de aproximadamente 20 ms
- Cada milisegundo el UART interrumpirá la ejecución del CPU del programa principal, para tomar un Byte de dato y depositarlo en la memoria por lo tanto el diseñador debe considerar dos cosas:

Control en tiempo real

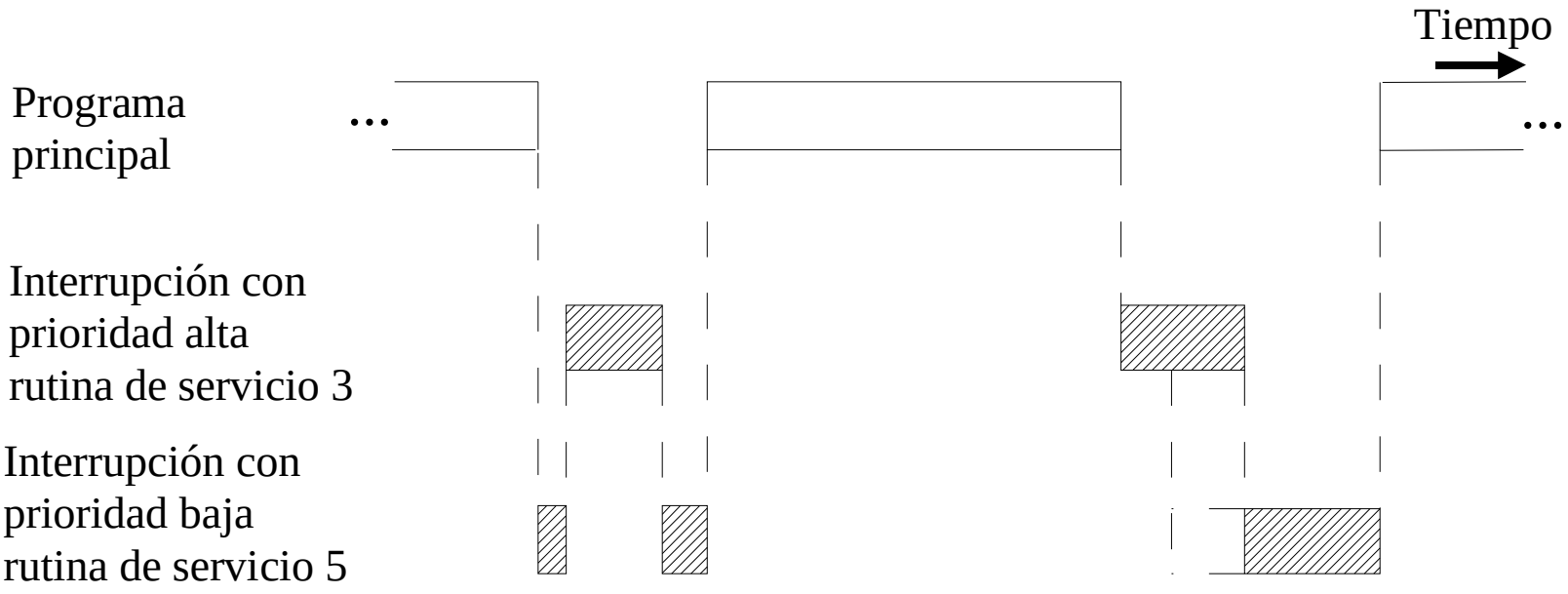
- Hacer el tiempo T1 tan corto que no sea necesario posponer otra interrupción por mucho tiempo, si ocurre un evento durante el tiempo T_1
- Hacer que el porcentaje de tiempo de CPU ocupado por T_1 sea pequeño, es decir hacer que la razón T_1/T_{p1} sea pequeña

Control en tiempo real

- Algunos dispositivos deben ejecutar una rutina de poleo cada vez que ocurre una interrupción, para determinar la fuente de la interrupción
- Otros microprocesadores utilizan un *vector de interrupciones*. Para este caso, cada interrupción lleva directamente al código que le da servicio a esa interrupción
- Un vector de interrupciones reduce el tiempo T1 y como consecuencia la razón T1/TP1

Control en tiempo real

- Algunos microprocesadores implementan el concepto de *vector de interrupciones con prioridad*

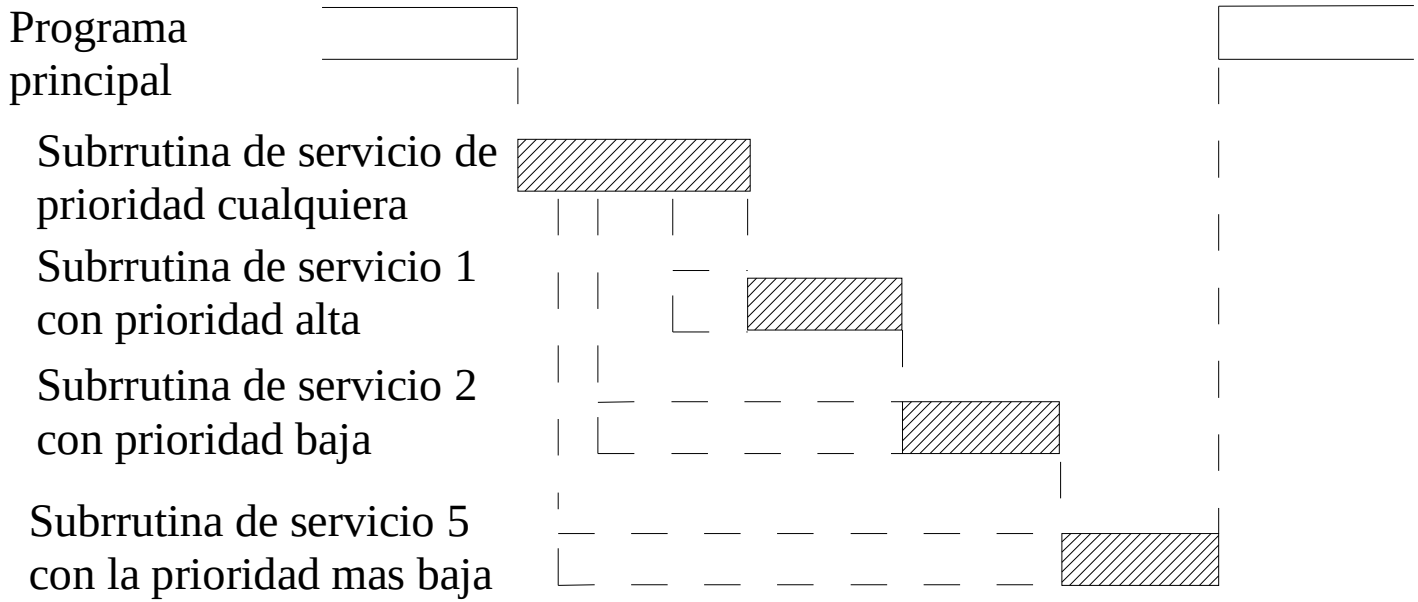


Control en tiempo real

- Una buena estrategia es deshabilitar las interrupciones cuando se esta ejecutando una subrrutina de servicio para un evento dado, dejando pendiente o en espera las demás subrrutinas de servicio
- Esta estrategia es buena siempre y cuando la subrrutina de servicio no tome mucho tiempo

Control en tiempo real

- Posteriormente se permite ejecutar las subrrutinas de mayor prioridad hasta llegar a las de menor prioridad



Control en tiempo real

- La diferencia entre esta estrategia y la anterior es la duración de tiempo de la subrrutina de servicio mas larga, pues es la cantidad del incremento de latencia (incremento de retardo) que un dispositivo debe esperar

Control en tiempo real

Interrupt sources (Highest priority first)	Vector Location	CPU Mask bit	Source* enable bit	Source flag
External reset pin	FFFE−F	Nonmaskable	−−	−−
Clock failure	FFFC−D	Nonmaskable	0039/3	−−
Watchdog timer	FFFA−B	Nonmaskable	003F/2	−−
Ilegal op code trap	FFF8−9	Nonmaskable	−−	−−
XIRQ pin	FFF4−5	Xbit	−−	−−
Any one of the following I−bit maskable interrupt sources can be assigned the nex priority	????−?	I	????/	????/?
IRQ pin/port C control pin	FFF2−3	I	0002/6	0002/7
Real time clock	FFF0−1	I	0024/6	0025/6
Timer input capture 1	FFEE−F	I	0022/2	0002/2
Timer input capture 2	FFEC−D	I	0022/1	0023/1
Timer input capture 3	FFEA−B	I	0022/0	0023/0

Control en tiempo real

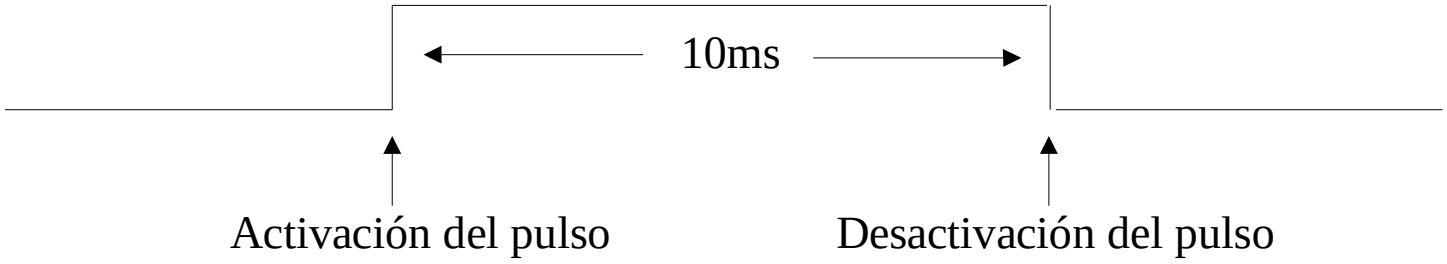
Interrupt sources (Highest priority first)	Vector Location	CPU Mask bit	Source* enable bit	Source flag
Timer output compare 1	FFE8−9	I	0022/7	0023/7
Timer output compare 2	FFE6−7	I	0022/6	0023/6
Timer output compare 3	FFE4−5	I	0022/5	0023/5
Timer output compare 4	FFE2−3	I	0022/4	0023/4
Timer output compare 5	FFE0−1	I	0022/5	0023/5
Timer overflow	FFDE−F	I	0024/7	0025/7
Pulse accumulator overflow	FFDC−D	I	0024/5	0025/5
Pulse accumulator input edge	FFDA−B	I	0024/4	0025/4
I/O serial port	FFD8−9	I	0028/7	0029/7
UART serial port−transmiter	FFD6−7	I	00D/7	002E/7
UART serial port−receiver	FFD6−7	I	00D/5	002E/5
Software interrupt instruction	FFF6−7	Nonmaskable	−−	−−

Control en tiempo real

- Timers controlables
  - Permiten interactuar con algún dispositivo externo por periodos de tiempo determinados, según sea necesario

    Ejemplo: activar un pulso y pedirle al timer interno del microcontrolador que nos avise o interrumpa cuando se hayan cumplido 10 ms.

Control en tiempo real



Durante 10 ms el microcontrolador esta libre para realizar alguna tarea útil.

Cuando se cumple ese periodo, el microcontrolador podría llamar a una subrrutina que apague el pulso

Control en tiempo real

- Ejemplo: Se quiere medir el tiempo de vuelo de un pulso de sonar

```
TIME−SONAR−ISR ;Rutina de servicio para una interrupcion
    LDD TCNT       ;Se registra el tiempo cuando sucede la interrupcion
    SUBD Sonar−tof ;Se obtiene el tiempo de vuelo del eco
    STD Sonar−tof  ;Se guarda el tiempo en un registro de 16 bits
    LDAA #1        ;Limpia la bandera de interrupcion
    STAA TFLG1     ;escribiendo un 1 en el registro TFLG1
    TRI            ;Regresa el control al codigo interrumpido
```

Control en tiempo real

- Registros involucrados

TMSK1 $1022

Bit 7							Bit 0
OC1I	OC2I	OC3I	OC4I	OC5I	IC1I	IC2I	IC3I
x	x	x	x	x	x	x	1

TFLG1 $1023

Bit 7							Bit 0
OC1F	OC2F	OC3F	OC4F	OC5F	IC1F	IC2F	IC3F
x	x	x	x	x	x	x	1

Control en tiempo real

- ICxI, ICxF Input Capture Interrupt habilita Input Capture Flags (x = 1, 2 o 3)
- La bandera de estatus ICxF se activa cuando un flanco (subida o bajada) se detecta
- El bit activado en TFLG1 se limpia escribiendo directamente sobre el bit asociado en el registro TMSK1

Control en tiempo real

- El bit de control ICxI permite configurar cada una de las entradas de captura para realizar funciones de poleo o de interrupción sin afectar los bits del registro TFLG1
- Cuando se escribe un 0 en alguno de los bits ICxI se inhibe la operación input−capture interrupt. Este tipo de configuración se utiliza cuando se esta realizando poleo sobre una entrada

Control en tiempo real

- Cuando se escribe un 1 en el bit ICxI, se genera una interrupción por hardware.
- Es responsabilidad del programador limpiar la bandera de interrupción ICxF en el registro TFLG1 antes de ejecutar una instrucción de retorno de la subrrutina de servicio

Control en tiempo real

- Registros involucrados

TCLT2 $1021

Bit 7							Bit 0
–	–	EDG1B	EDG1A	EDG2B	EDG2A	EDG3B	EDG3A
x	x	x	x	x	x	x	1

EDGxB	EDGxA	Configuration
0	0	Capture Disabled
0	1	Capture on Rising Edges Only
1	0	Capture Disabled on Falling Edges Only
1	1	Capture on Any Edge (Rising of Falling)

Control en tiempo real

- Ejemplo del proceso de inicialización

```
LDAA #%01          ;Se programa el registro TCTL2
STAA TCTL2         ;para detectar flancos de subida
LDAA #1            ;Se limpia la bandera IC3F
STAA TFLG1         ;escribiendo un 1 en el bit correspondiente del registro TFLG1
LDAA #1            ;Se configura el bit IC3I del registro TMSK1 para aceptar
STAA TMSK1         ;interrupciones por circuiteria externa
CLI                ;Se habilitan todas las interrupciones del sistema
JSR Turn−on−sonar      ;Se ejecuta la rutina que dispara el sonar
LDD TCNT           ;Se carga en el registro D el valor actual del timer
STD Sonar−tof      ;Se guarda el valor leido anteriormente
...                ;Se realiza algun trabajo utiil
JSR Compute−Distance ;Se calcula en algun momento la distancia
```

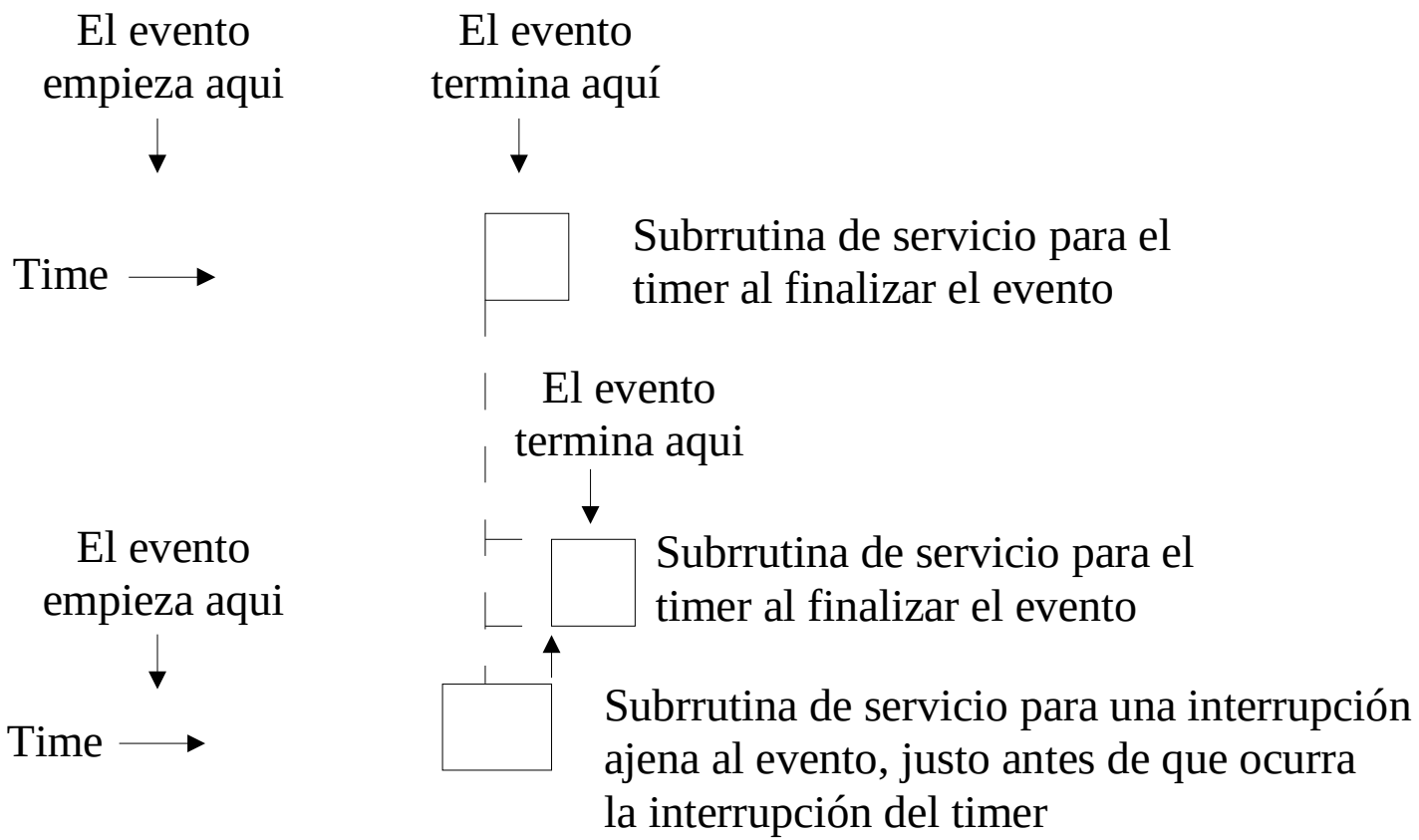
Control en tiempo real

- Para poder utilizar el código de nuestra rutina de servicio, el código se debe escribir en la dirección apuntada por el vector de interrupciones de la interrupción IC3I −> $FFEA

Control en tiempo real

- Problema
  - Se deben deshabilitar las interrupciones durante el tiempo de acción del timer, de otra manera podría aparecer una interrupción que dispare a la subrrutina de servicio asociada con esa interrupción, retardando la ejecución de la subrrutina de nuestro timer

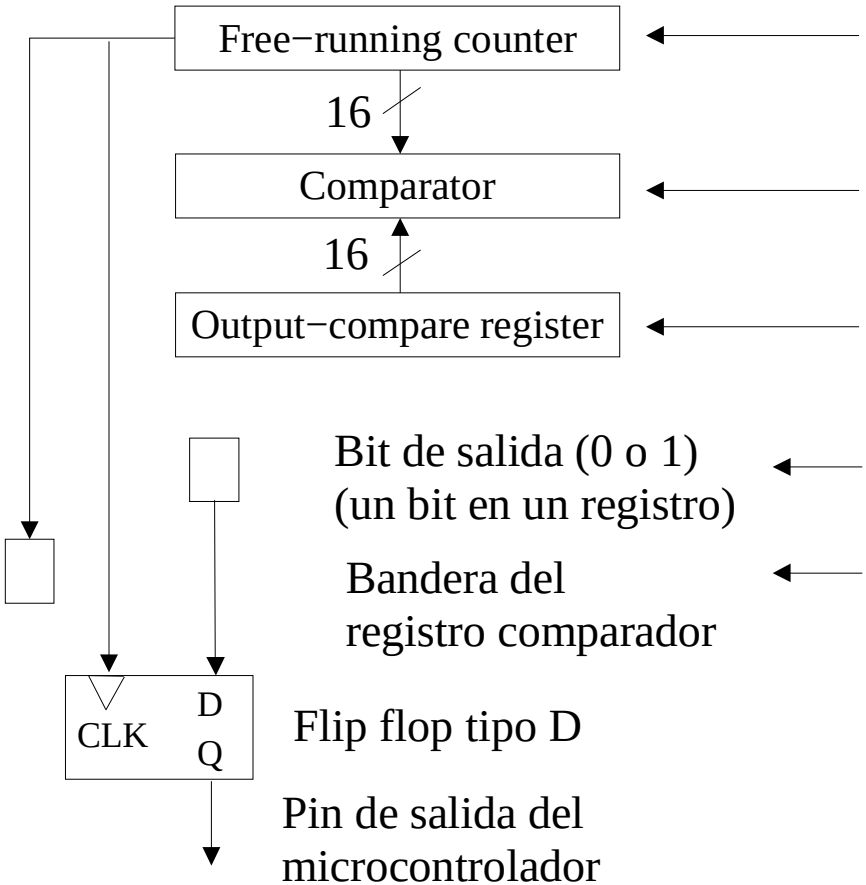
Control en tiempo real



Control en tiempo real

- Input Capture
  - Si se requiere precisión de tiempo para atender a un dispositivo, el microcontrolador debe generar un evento externo, independiente del trabajo que este realizando el CPU
  - Los elementos que se utilizan para ésta estrategia son los siguientes:

Control en tiempo real



Contador manejado por el reloj del sistema, el CPU puede leer de este

Detecta igualdad

El CPU puede escribir o leer de este registro

Bit de salida (0 o 1) (un bit en un registro)

El CPU puede escribir este bit

Bandera del registro comparador

Se activa cuando se detecta igualdad se puede leer o borrar por el CPU, puede disparar una interrupción

Flip flop tipo D

Pin de salida del microcontrolador

Control en tiempo real

- El contador de oscilación libre lleva la cuenta del tiempo transcurrido. El programador debe seleccionar la precisión del tiempo, mediante el prescalador del sistema. El prescalador permite modificar la frecuencia del contador de oscilación libre en múltiplos de 1, 2, 4 8 y 16

Control en tiempo real

- Los encargados de identificar cuando se ha cumplido el periodo de tiempo esperado, son el registro de comparación y el comparador
- Cuando se cumple el tiempo esperado, se libera el bit de nivel, anteriormente este bit se carga con un valor conocido (0 o 1)

Control en tiempo real

- Además la bandera del comparador se utiliza para enviar una señal de interrupción al CPU, para indicar que se ha cumplido el tiempo programado para un evento
- Al mismo tiempo se dispara el reloj del flip flop, pasando el valor del bit de nivel al pin de salida del microcontrolador

Control en tiempo real

- Ejemplo: Se desea generar un pulso positivo de 2 ms de duración
- Este problema requiere de dos procedimientos:
  - Una rutina que prepare lo necesario para disparar el pulso
  - Una rutina que prepare lo necesario para apagar el pulso

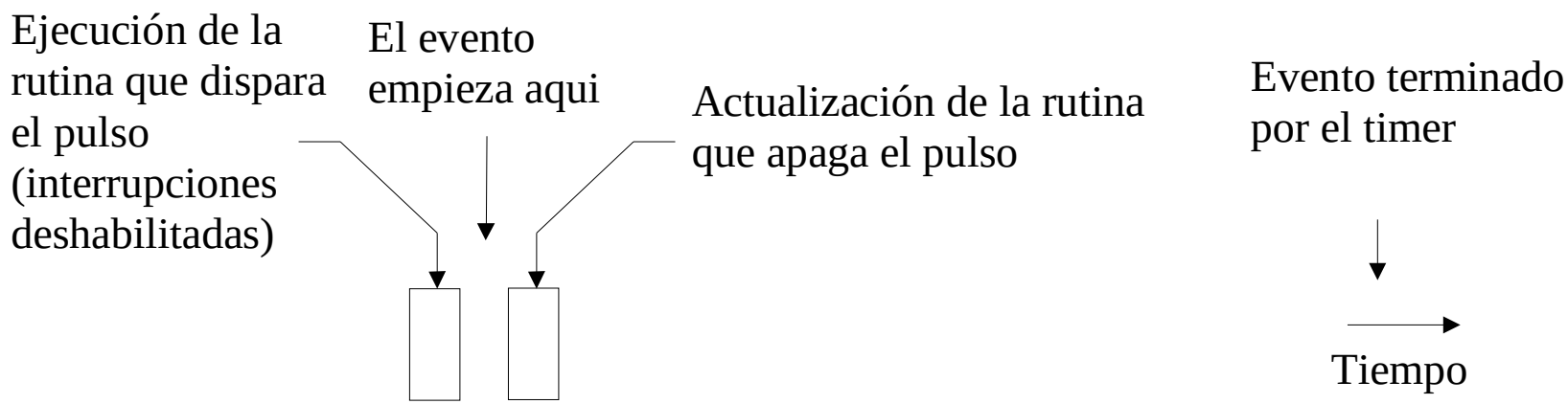
Control en tiempo real

- Rutina que prepara lo necesario para disparar el pulso
  - Deshabilitar interrupciones
  - Leer el contenido del contador de oscilación libre
  - Sumar un “numero mágico al valor leído anteriormente
  - Almacenar este numero en el *output−compare register*
  - Poner un 1 en el bit *output−level bit*
  - Habilitar interrupciones

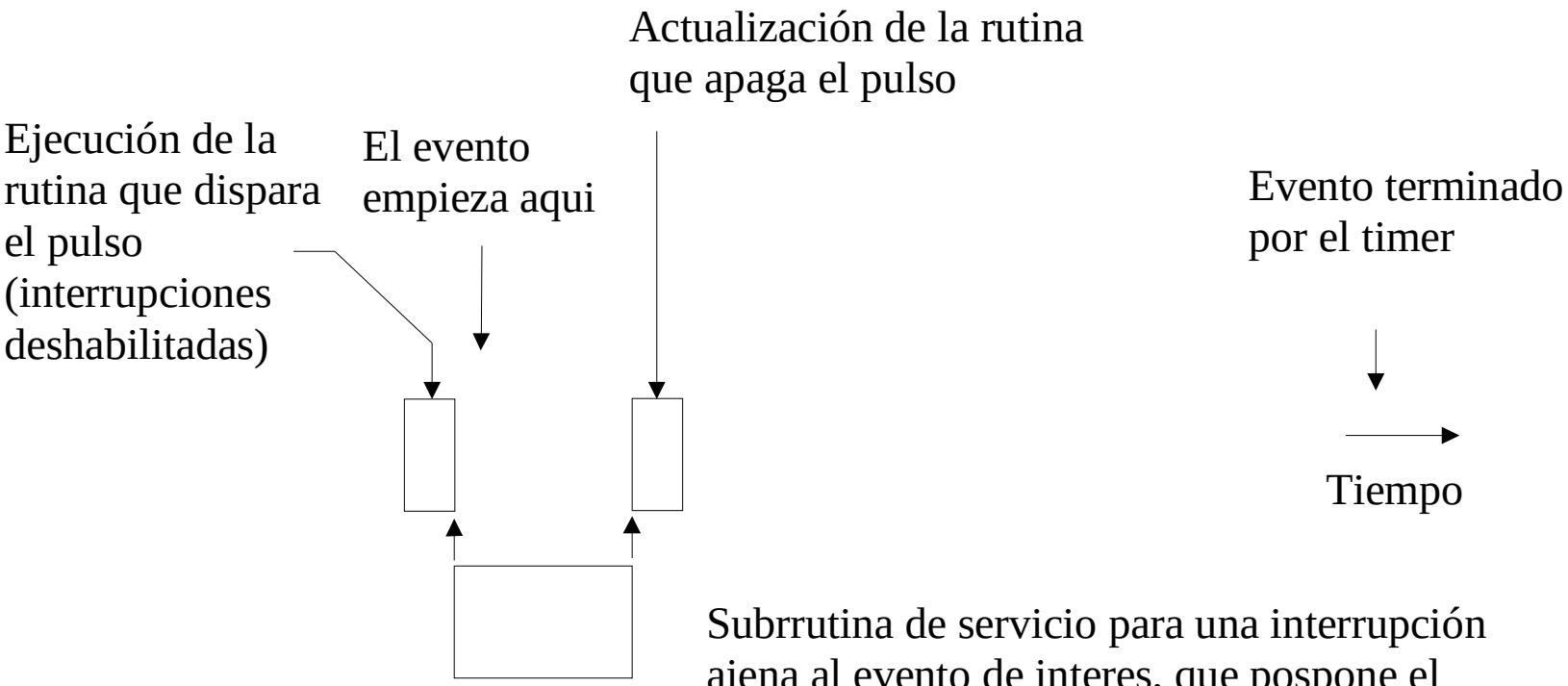
Control en tiempo real

- Rutina que prepara lo necesario para apagar el pulso
  - Leer el *output−compare register* para saber el momento exacto cuando empezó el pulso
  - Sumar 2000/INCREMENTO al valor leído anteriormente
  - Almacenar el resultado en el *output−compare register*
  - Escribir un 0 en el *output−level bit*

Control en tiempo real



Control en tiempo real

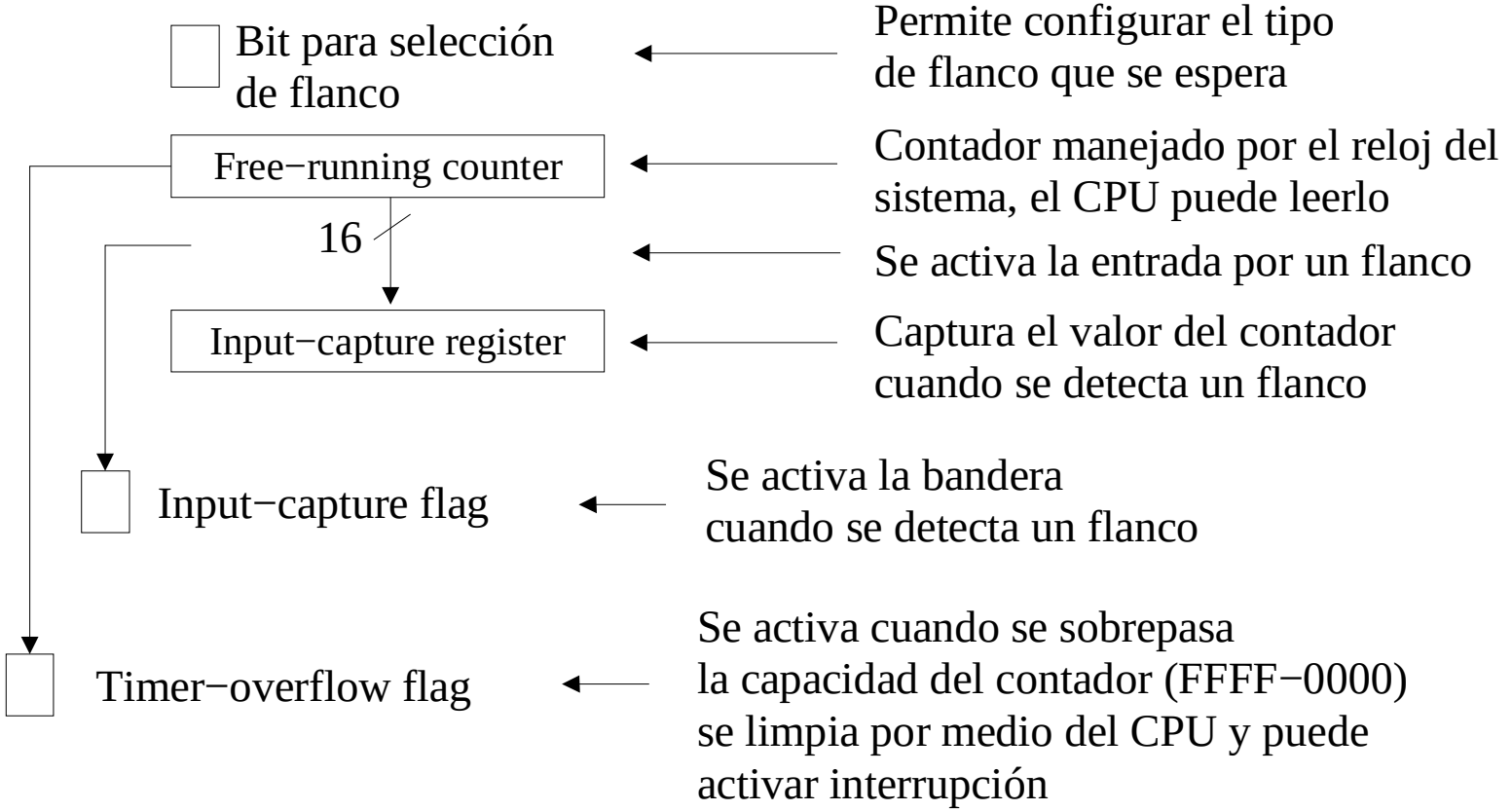


Control en tiempo real

- Timers controlables
  - También se utilizan para medir con alta precisión el ancho de un pulso
  - La estrategia es:
    - Rutina que prepara lo necesario para capturar el valor del instante cuando empieza el pulso
    - Rutina que prepara lo necesario para capturara el valor del instante cuando termina el pulso
    - La diferencia entre los dos valores capturados determinan la duración del ancho del pulso

Control en tiempo real

- Elementos involucrados



Control en tiempo real

- Rutina que prepara los elementos necesarios para capturar el valor del instante cuando aparezca un flanco de subida

```
Setup−IC3       ;Codigo para activar input capture
LDAA %#01       ;Disparo de IC3 con flanco de subida
STAA TCTL2      ;para capturar el tiempo en el registro TIC2
```

Control en tiempo real

- Capturando el valor del instante cuando se dispara el sonar

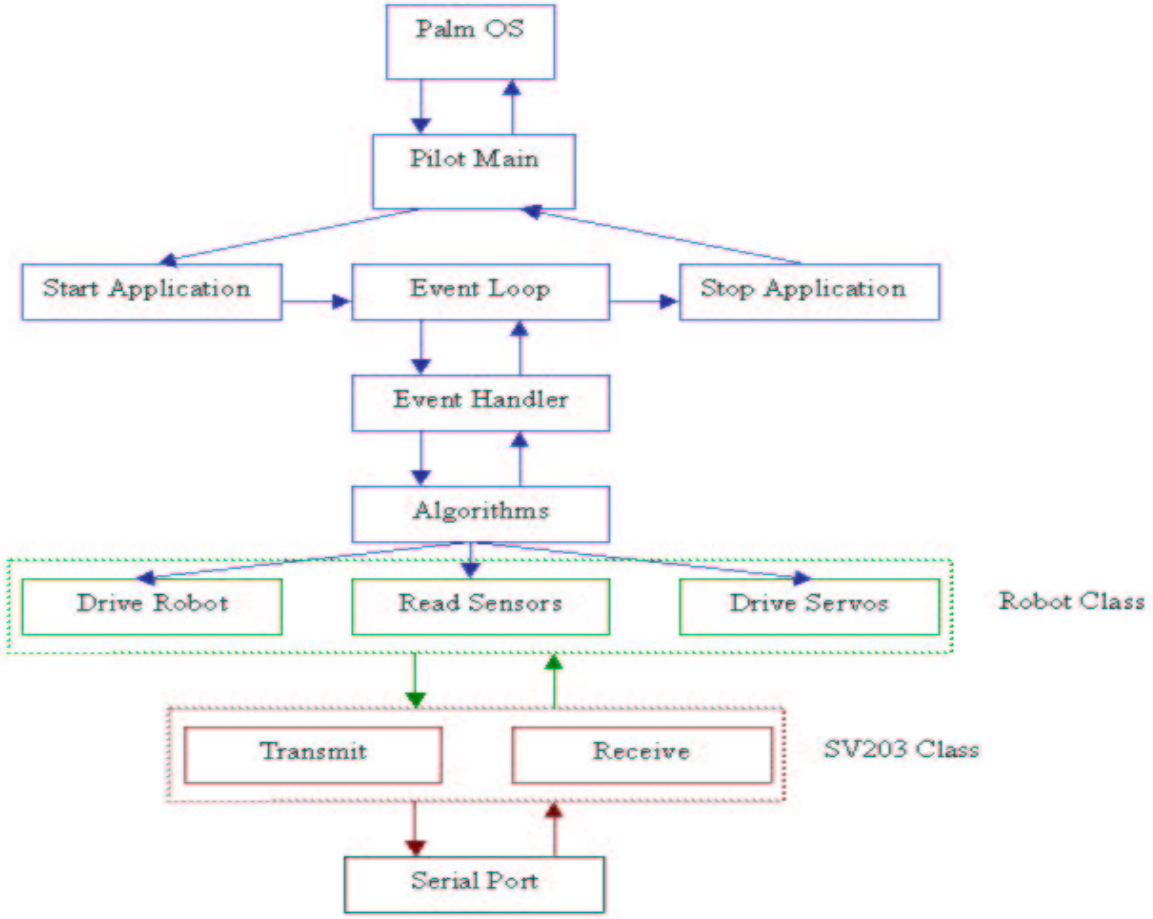
```
JSR Turn−on−sonar      ;Subrrutina que dispara el sonar
LDD TCNT      ;Se obtiene el valor del conatador de oscilacion libre
STD Sonar−start     ;Se guarda el valor leido anteriormente
```

Control en tiempo real

- Calcular la distancia una vez que se ha registrado el eco de regreso de la señal

```
LDD TIC3 ;Se carga el valor del oscilador una vez registrado el eco
SUBD Sonar−start ;Se resta este valor con el valor inicial
STD Sonar−tof ;Se almacena el resultado de la resta
JSR Computing−distance ;Llamar rutina que calcula la distancia
```

Estructura de una aplicación para el PPRK



Conclusiones

- Polling, se adueña del microcontrolador y no se puede predecir el tiempo de respuesta para un dispositivo que necesite interactuar con el microcontrolador

Conclusiones

- Manejo de interrupciones, permite hacer un uso mas optimo del hardware y del software, se suspende momentáneamente el trabajo del micro para atender a un posible evento

Conclusiones

- Input capture, las operaciones para realizar trabajo no distraen el trabajo del microcontrolador

Bibliografia

- Mobile Robots, Anita Flyn
- Design with Micronotrollers, John B. Peatman