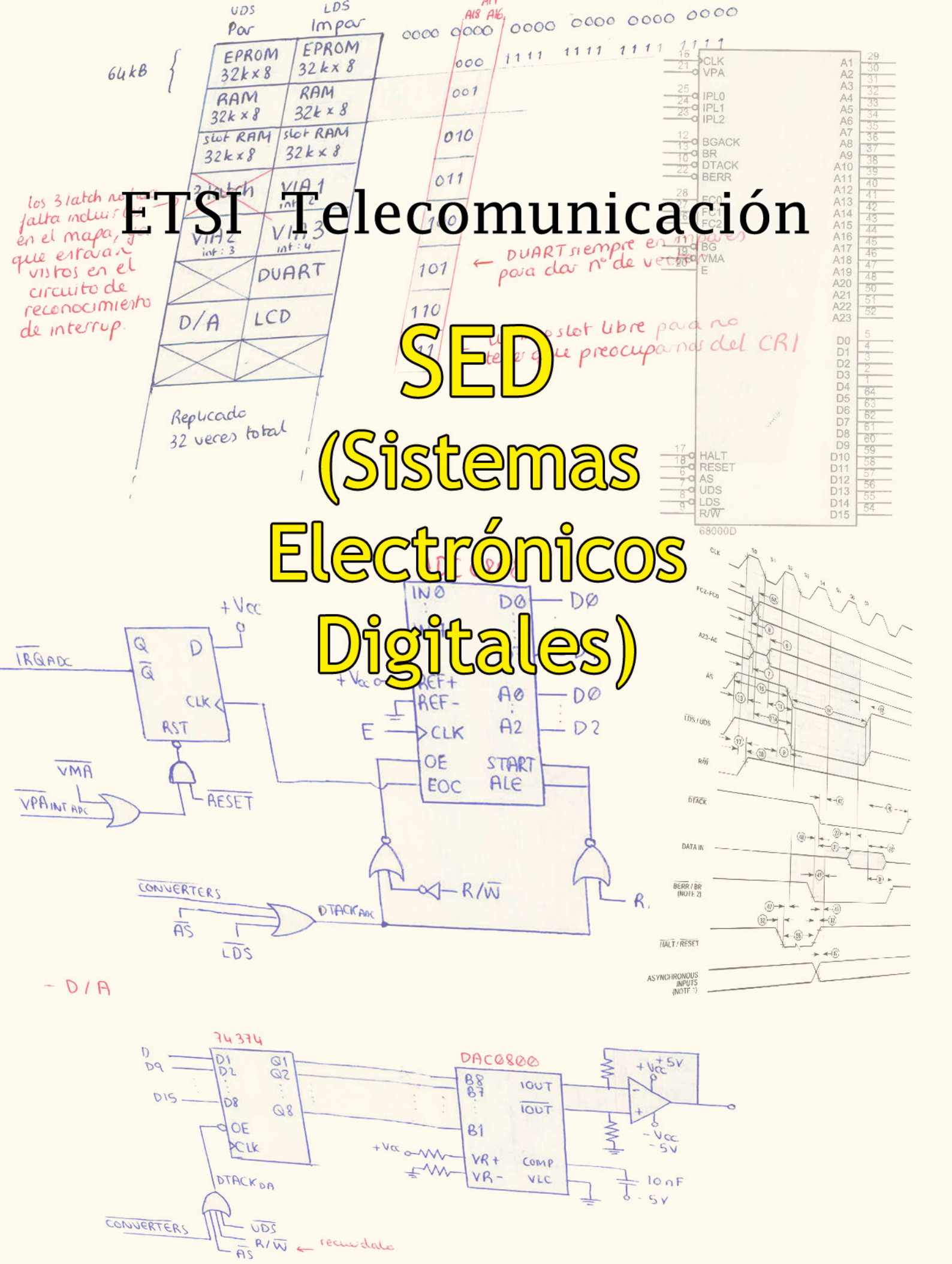
ETSI Telecomunicación

SED (Sistemas Electrónicos Digitales)



Apuntes de Pak

Sistemas Electrónicos Digitales (SED)

Apuntes de Pak (Francisco José Rodríguez Fortuño)
ETSI Telecomunicación. Universidad Politécnica de Valencia.
Primer cuatrimestre de 3er curso
Curso 2005/2006

Fecha de última actualización: 23 Agosto 2007

SED: Sistemas Electrónicos Digitales

TEMA 1. Introducción a los microprocesadores

Sistemas

cableados:
- rápidos, poco flexibles

mismos estímulos → mismas respuestas

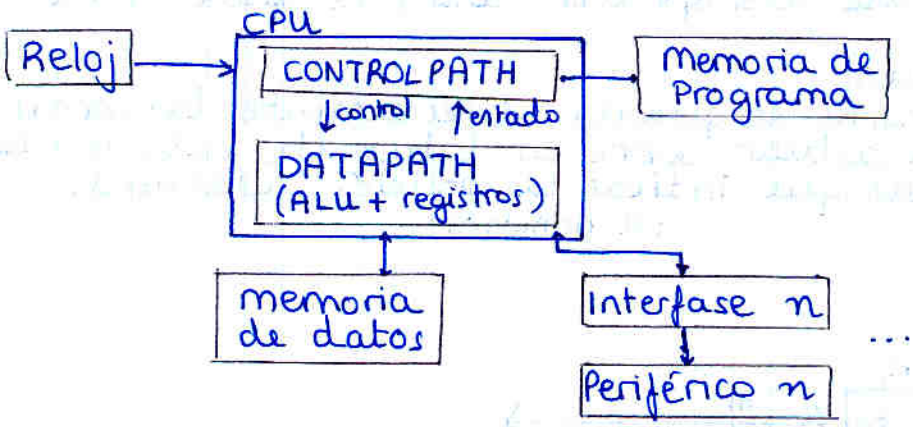
programables:
- sigue instrucciones de una memoria

↳ CPU: - en único C.I.
- mem. programa exterior

↳ Microcontrolador: mem. prog. interior
- mas específicos - robustos

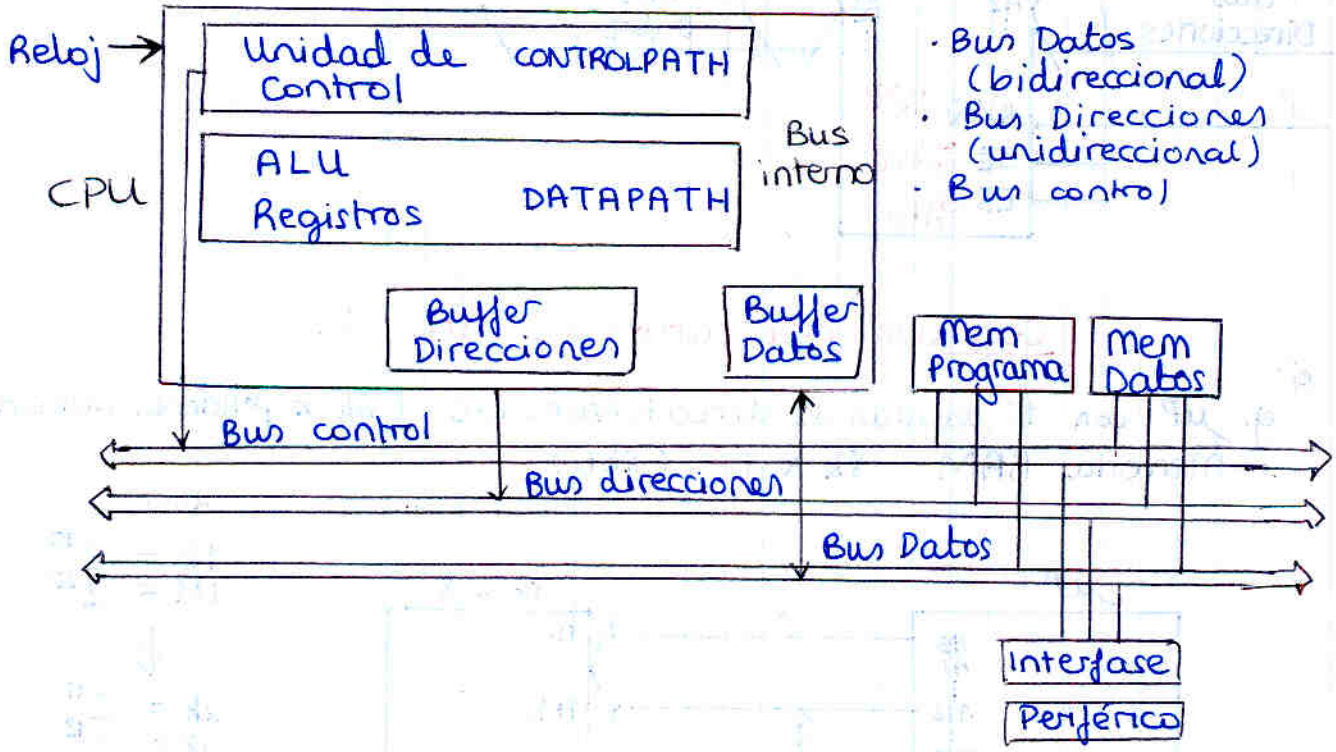
↳ DSP: DataPath MUY grande

Diagrama de bloques



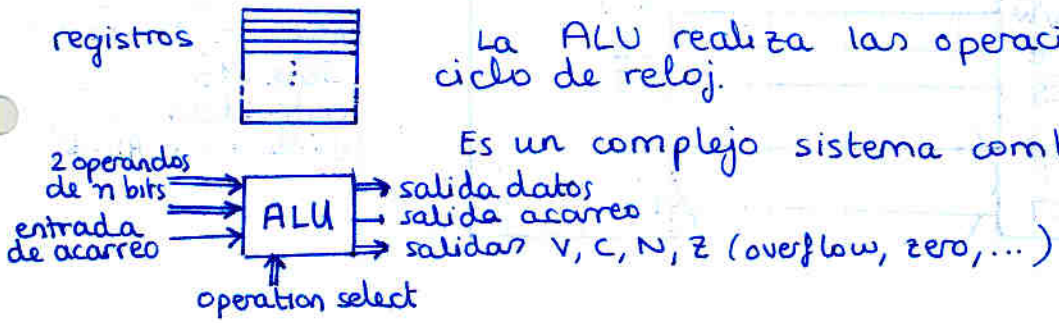
Se necesitan muchas conex.

Normalmente se utilizan buses.



- Bus Datos (bidireccional)
- Bus Direcciones (unidireccional)
- Bus control

Datapath



La ALU realiza las operaciones en un único ciclo de reloj.

Es un complejo sistema combinacional

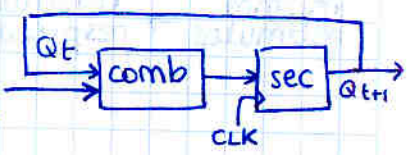
Controlpath

señales para temporizar las operaciones del datapath.

En cada instante se activan una serie de microoperaciones.

· Distintas arquitecturas:

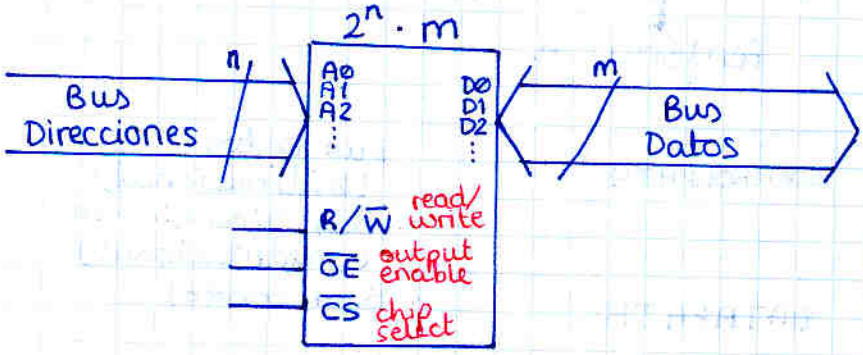
- **Cableadas**: gran maquina de estados
  el circ. combinacional puede retardar mucho



- **Segmentadas**: se divide en segmentos en los cuales puede ir haciéndose una operación antes de acabar otra.

- Microprogramadas:
  En una memoria se guardan directamente las señales que deben actuar sobre el datapath, cada instrucción no tiene mas que indicar la posición adecuada.
  (el firmware)

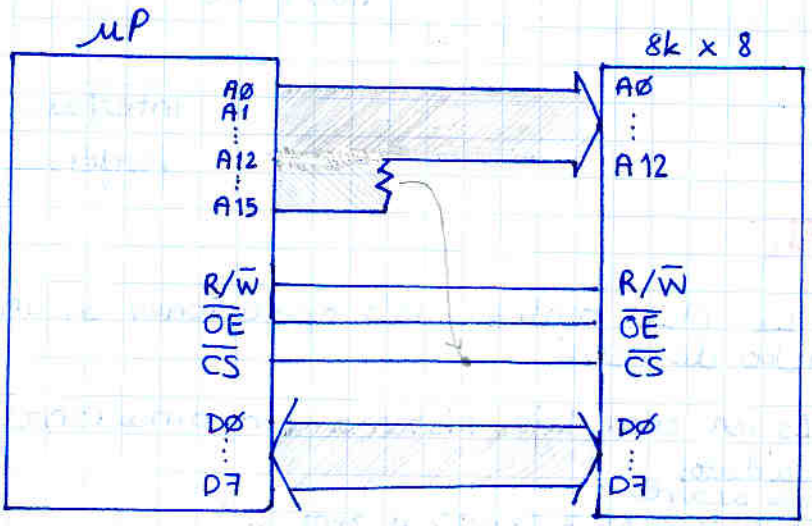
Memoria



Conexión procesador a memoria

ej:

- μP con 16 salidas de direccionamiento (64 k posiciones memoria)
- Memoria RAM 8k x 8 (8kB)



TRUCO

$1k = 2^{10}$
$1M = 2^{20}$

$\Downarrow$

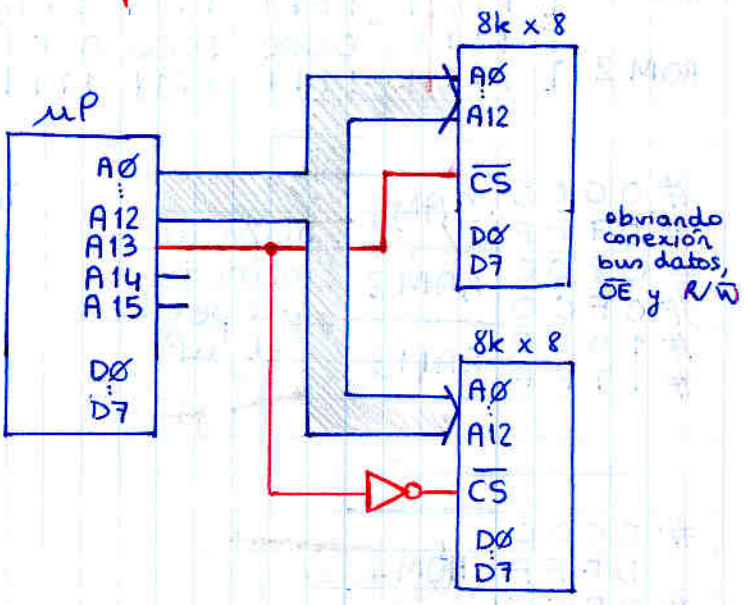
$2k = 2^{11}$
$4k = 2^{12}$
$8k = 2^{13}$

memoria de 8k x 8 tiene 13 entradas de dirección (A12-A0) -1!!

• ampliación del número de palabras

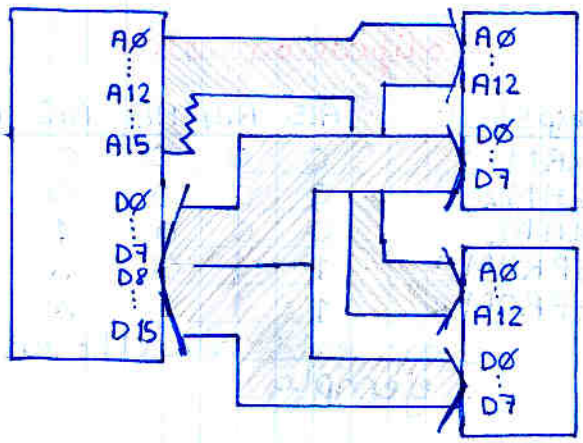
b 0000 0000 0000 0000
b 0001 1111 1111 1111
b 0010 0000 0000 0000
b 0011 1111 1111 1111

chip select — entradas a cada chip (13 = 8k)



• ampliación del número de bits por palabra

Cogemos 8 bits de un chip y 8 bits de otro.
Cada palabra debe tener una única dirección.



10 - 1k
11 - 2k
12 - 4k
13 - 8k
14 - 16k
15 - 32k
16 - 64k
17 - 128k
18 - 256k
19 - 512k
20 - 1M
21 - 2M
:

ejercicio: Disponemos:
- µP: 8 bits palabra datos, 16 bits direcciones
- EPROM: 8k x 8
- RAM: 2k x 8

Requisitos:
- 16 kB de ROM al final del mapa
- 6 kB de RAM al principio del mapa

RAM: 2k → 11 bits dirección

RAM1 { 0000 0000 0000 0000
0000 0111 1111 1111 1111

RAM2 { 0000 1000 0000 0000
0000 1111 1111 1111 1111

RAM3 { 0001 0000 0000 0000
0001 0111 1111 1111 1111

chip select — bits de dirección a cada chip

$\overline{CS}_{RAM_1} = A15 + A14 + A13 + A12 + A11$

$\overline{CS}_{RAM_2} = A15 + A14 + A13 + A12 + \overline{A11}$

$\overline{CS}_{RAM_3} = A15 + A14 + A13 + \overline{A12} + A11$

recuerda como hacerlo

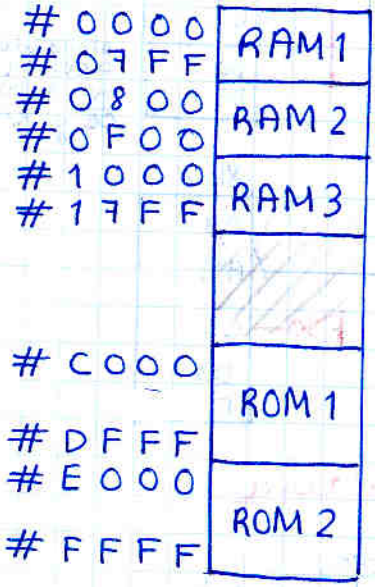
ROM : 8k → 13 bits dirección

ROM 1 { 110 | 0 0000 0000 0000
110 | 1 1111 1111 1111

ROM 2 { 111 | 0 0000 0000 0000
111 | 1 1111 1111 1111

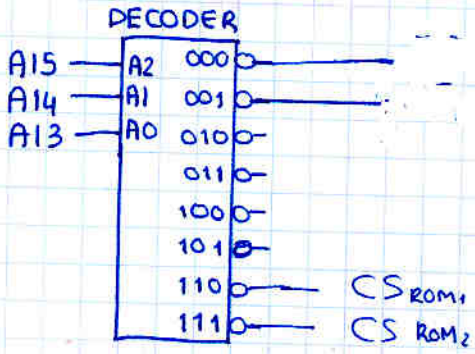
$\overline{CS}_{ROM_1} = \overline{A15} + \overline{A14} + A13$

$\overline{CS}_{ROM_2} = \overline{A15} + \overline{A14} + \overline{A13}$



Mapa de memoria que percibe el µP

La mejor forma de implementar los chip select es con un decodificador



· Decodificación Total y Parcial

en el ejemplo anterior tenemos

	A15	A14	A13	A12	A11	A10 ... A0
RAM1	0	0	0	0	0	X.....X
RAM2	0	0	0	0	1	X.....X
RAM3	0	0	0	1	0	X.....X
EPROM1	1	1	0	X	X	X... X
EPROM2	1	1	1	X	X	X... X

· Decodificación Total

Chip select	A15	A14	A13	A12	A11
RAM1	0	0	0	0	0
RAM2	0	0	0	0	1
RAM3	0	0	0	1	0
EPROM 1	1	1	0	X	X
EPROM 2	1	1	1	X	X

como está resuelto en el ej.

· Decodificación Parcial:

RAM { A15 = 0

	A12	A11
RAM1 :	0	0
RAM2 :	0	1
RAM3 :	1	0

Eliminamos las linear de dirección redundantes

$\overline{CS_{RAM1}} = A_{15} + A_{12} + A_{11}$

$\overline{CS_{RAM2}} = A_{15} + A_{12} + \overline{A_{11}}$

$\overline{CS_{RAM3}} = A_{15} + \overline{A_{12}} + A_{11}$

EPROM { 1

	A13
EPROM1	0
EPROM2	1

$\overline{CS_{E1}} = \overline{A_{15}} + A_{13}$

$\overline{CS_{E2}} = \overline{A_{15}} + \overline{A_{13}}$

- Ahorramos niveles de puertas lógicas: mayor velocidad
- El mapa de memoria tendrá chips replicados: no importa
- No podemos ampliar memoria: no es usual hacerlo.

↓

se prepara cualquier posible ampliación diseñando la lógica y poniendo zócalos, no tiene sentido ampliarlo sin estar previsto.

ejercicio 2 EPROM 2k x 8 µP: - datos: 8 líneas
RAM 8k x 8 - dir: 16 líneas (64 K)

Se pide: 4KB programa al principio
16 KB memoria a continuación

→ 1) Necesitamos 2 chips EPROM y 2 chips RAM
→ 2) EPROM : 2k → 11 líneas
RAM : 8k → 13 líneas

→ 3) Asignar direcciones: Truco
Calcular la capacidad de memoria en hexadecimal
2^{11} : 0000 1000 0000 0000 = 0800
2^{13} : 0010 0000 0000 0000 = 2000

Dirección Final = Direccion Inicial + [capacidad - 1]

EP_1 { 0000
0000 + 07FF = 07FF
EP_2 { 0800 ← ¡No 07FF!
0800 + 07FF = 0FFF
R_1 { 1000
1000 + 1FFF = 2FFF
R_2 { 3000
3000 + 1FFF = 4FFF

CS es negado por tanto todo al revés
Y es +
O es •

→ 4) Decidir chip selects

→ EPROM 11 líneas dirección

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E_1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	1	1	1	1	1	...		...			
E_2	0	0	0	0	1	0	0	0	0	0	...					
	0	0	0	0	1	1	1	1	1	1	...					
R_1	0	0	0	1	0	0	0	0	0	0	...					
	0	0	1	0	1	1	1	1	1	1	...					
R_2	0	0	1	1	0	0	0	0	0	0	...					
	0	1	0	0	1	1	1	1	1	1	...					

↳ Columna extraña

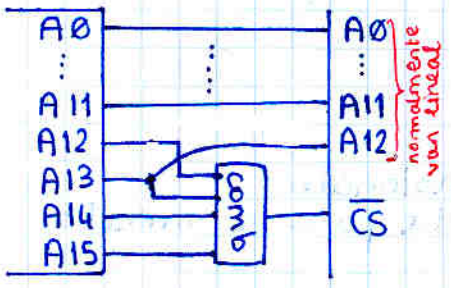
RAM
11 líneas dir

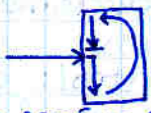
$CS_{E_1} = A_{15} + A_{14} + A_{13} + A_{12} + A_{11}$

$CS_{E_2} = A_{15} + A_{14} + A_{13} + A_{12} + A_{11}$

Las líneas que normalmente van directamente a la dirección de la memoria suelen ir desde 00000... hasta 11111... recomiendo así la memoria

Ahora no es el caso: → Si conecto A12, el µP direcciona la parte alta de la memoria como parte baja
↳ no pasa nada (mientras se tenga en cuenta en el caso por ej de memoria ya grabada)





↳ si conectamos A13 todo será como siempre
↳ (en la práctica se conectaría A12 por patillaje)
cuestiones de

$\overline{CS_{R_1}} = A_{15} + A_{14} + ((A_{13} + \overline{A_{12}}) \cdot (\overline{A_{13}} + A_{12}))$
$\overline{CS_{R_2}} = A_{15} + ((A_{14} + \overline{A_{13}} + \overline{A_{12}}) \cdot (A_{14} + \overline{A_{13}} + \overline{A_{12}}))$

Decodificación parcial: ¿Qué linea podemos quitar?

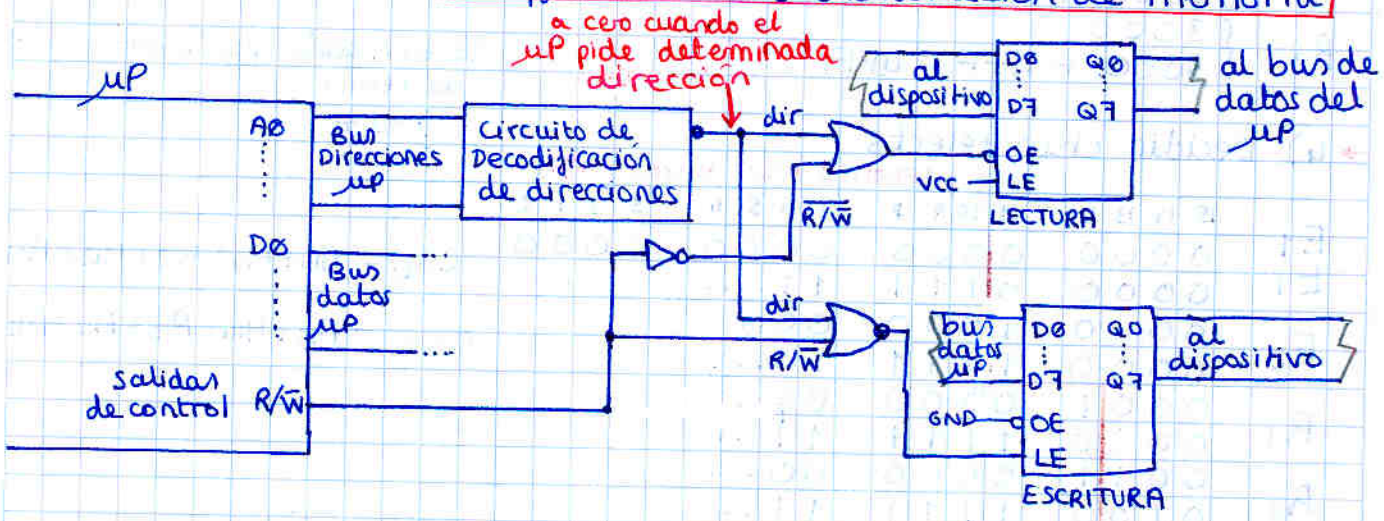
Podemos quitar A15
no podemos quitar más: truco
si 16 lineas → 64k
15 lineas → 32k > 20k
14 lineas → 16k < 20k !! no se puede

¿Y no podemos quitar A14 unicamente en la eprom, similarmente al ejemplo 1? No; en el ejercicio 1 habia un bit que distinguia claramente la RAM de la EPROM; nos permitia quitar lineas diferentes.
Supongamos que quitamos A14.
la dirección A11 — 0 seleccionaria tanto E1 como R2
A12 — 0 ↳ conflicto
A13 — 0

Interfase de entrada salida

El µP no puede conectarse directamente al bus a la entrada-salida. El bus se conecta a unos buffers que guardan los datos de entrada salida para ponerlos en el bus cuando se le pida (o leer del bus y ponerlo en la e-s)

El µP accede a estos buffers mediante una dirección de memoria



LE: latch enable
Qn: las salidas son buffers tresestado con la salida ya preparadita para que nada más se active sacarla

Lineas de conexión

3 tipos de buses:
- Direcciones: unidireccional
- Datos
- Control: conjunto de lineas bidimensionales
  el µP pone la información necesaria para el correcto funcionamiento del sistema
  - ej. linea R/W̄
  - ej. linea de validación de direcciones

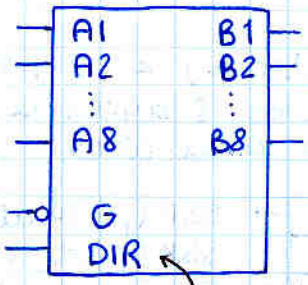
Precauciones con buses:

- minimización de acoplo capacitivo y electromagnético
  ej. condensador de desacoplo
- comprobación de FAN-IN y FAN-OUT, para controlarlo se pueden utilizar buffers

BUFFER UNIDIRECCIONAL

deciden si se activa la zona 1 o 2

BUFFER BIDIRECCIONAL



dirección de A a B o de B a A

Pila

Bloque de posiciones de memoria sucesivas a las que se accede mediante un sistema LIFO (last in first out)
Se consigue mediante predecremento y postincremento del puntero de pila

Se utiliza:

- Subrutinas (almacena PC antes de saltar a subrutinas)
- Interrupciones. Excepciones de emergencia. (rutinas de interrupción)
- Paso de parámetros

Permite el anidamiento de subrutinas.

Interrupción

- Provoca la detención de la ejecución del programa. Realiza un nuevo programa asociado a la interrupción
- Permite controlar procesos en tiempo real
- Se activa mediante las líneas de interrupción del procesador.
- En el 68000 hay 7 niveles de prioridad

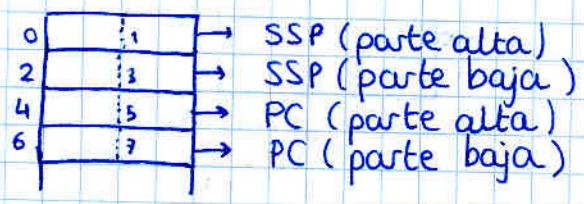
Inicialización del procesador

- Reset Frio: el µP inicialmente sin alimentación
- Reset Caliente: mediante un pulso de reset

Un circuito se encarga de preparar todo tras un reset.

- En el 8086:
  Se pone PC a $ FFFF0
  (normalmente ahí hay un JMP)

- En el 68000:
  - SR (registro sistema) = 2700
  - Lee los primeros 8 bytes de memoria para cargar los registros siguientes



nota:
SSP: puntero de 'pila de supervisión'
(pila a la cual sólo se tiene acceso en modo supervisor)

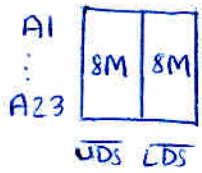
Si falla leyendo esos 8 bytes el µP se bloquea

Diseño de un sistema completo. Especificaciones

Consideraciones Previas

- El 68000 tiene 24 líneas de dirección → 16MB

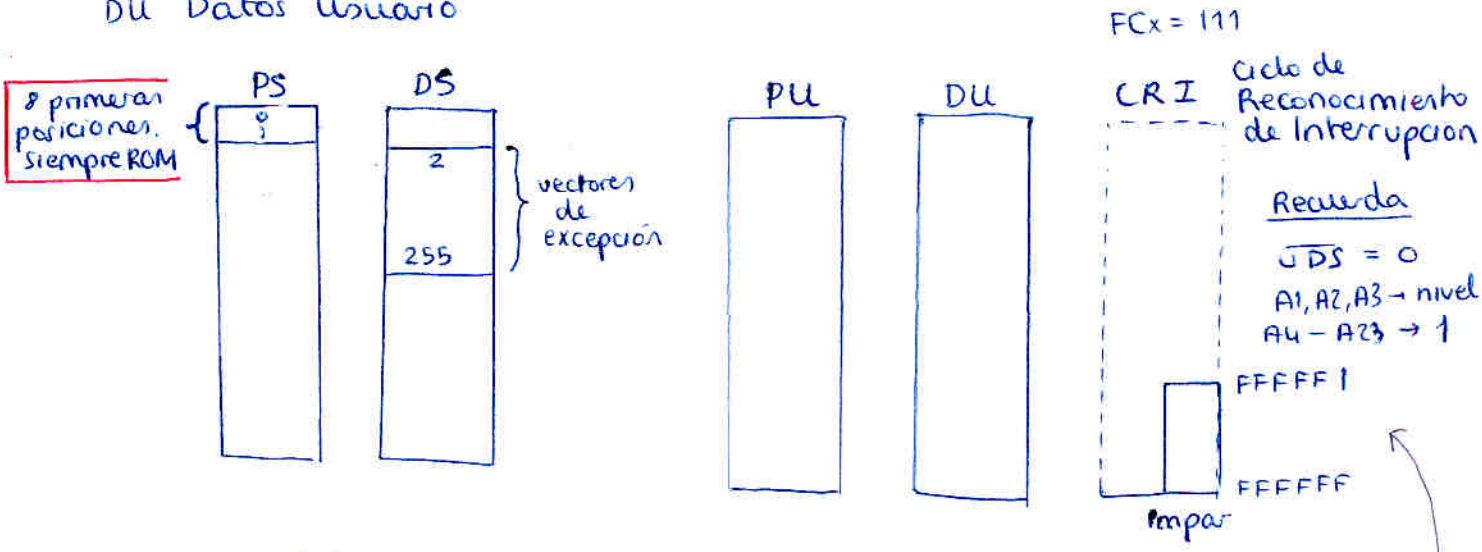
8MB par
8MB impar



- Además tenemos líneas FCX

PS Programa supervisor } Al hacer un FETCH
PU Programa Usuario

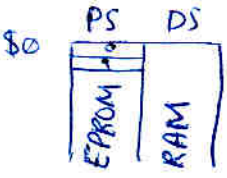
DS Datos Supervisor
DU Datos Usuario



- Sistema autónomo: programa en ROM → programa definitivo
  ↳ no tienen sentido cosas como BERR
- Evaluadores / Instructores: programa en ROM: sistema operativo que permite copiar programas en RAM a los que puede saltar y perder el control

> Recuerda que en realidad el CRI es como un ciclo de lectura a direcciones FFFFF1 → FFFFFF, si ahí hay memoria no funcionará:
> - el dispositivo que interrumpe respondería
> - el dispositivo direccionado interrumpe
>
> Siempre asegurarse de que no hay nada, o separar bancos de memoria.

Nota: en un examen podían poner EPROM y RAM ambas a partir de dirección 0: Había que deducir: poner EPROM en PS y RAM en DS



Los vectores de excepción estarán vacíos. El programa de arranque deberá rellenarlos (permite ir variandolos → se suele hacer)

TEMA 2: Modelo de programación del MC 68000

Introducción

El 68000 nace por la necesidad de cálculo más potente para utilizar software más potente.

Nuevas habilidades del 68000

- Bus de datos de 16 bits.
- 16 MB de memoria (frente a 64 kB)
- Frecuencia 16 MHz
- Mas modos de direccionamiento (14 modos !!)
- Mas eficaz multitarea, tratamiento mas eficaz de interrupciones
- Seguridad: modo supervisor
- Abierto: facilmente ampliable

Estructura Hw interna. Características generales

Registros:

- 2 bloques:
  - 8 registros de 32 bits (D0, D1, ..., D7) (reg. de datos)
  - 8 registros de 32 bits (A0, A1, ..., A7)
    - del A0 al A6: reg. de direcciones
    - A7 es el registro de pila.
      - Hay dos registros A7
        - USP (user stack pointer) se accede en modo usuario
        - SSP (supervisor Stack Pointer) se accede en modo super

- Registro PC (program counter) 32 bits
- Registro de estado SR
- Registro de instrucciones IR

16 bits

byte de sistema — codigos de condición: acarreo, zero, desbord, ext. signo, de la ultima operación

Operaciones:

- 56 tipos
- Cada una con diferente tipo de datos
  - .B Byte
  - .W word
  - .L LongWord

  tambien hay algun caso con Bit o BCD

  NOTA: El µP sólo puede utilizar el bus con 8 o 16 bits así que, si p. ej modificamos 1 bit, lo que hará será leer el byte entero, alterar el bit, y guardar el byte entero.
- 14 modos de direccionamiento

Buses asíncronos

Tiene una línea de reconocimiento para esperar a que contesten antes de quitar los bits y continuar. optimiza el tiempo.

Buses no multiplexados

Un bus de direcciones y otro de datos

Bus síncrono (8086)

A ciclos de reloj

1. Pongo dirección
2. Valido dirección
3. Pongo datos
4. Quito datos

si no los pillas no me importa

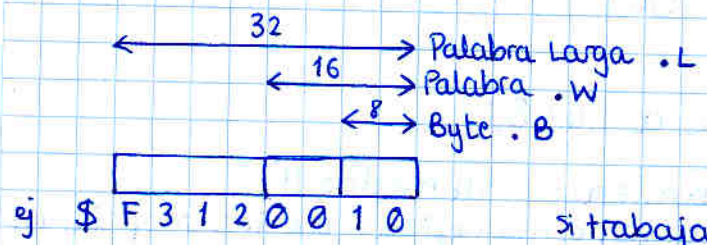
Buses multiplex. (8086)

En unos ciclos es de datos y en otros de direcciones

sistemas multiprocesadores: fáciles de implementar
ej. compartir buses facilmente.

Detección de errores: tanto software como hardware.
es MUY, muy, muy robusto.

Registros de Datos D0 - D7



Las operaciones sobre D0 - D7 modifican los bits del CCR (registro codigos condicion)

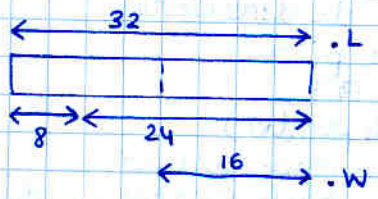
si trabajas sólo con .W o .B no modificas el resto:

ADD.W D0, D1

D0	3 F 2 0	8 0 0 0
D1	4 3 1 0	8 0 0 0
↓		
D1	4 3 1 0	0 0 0 0

C = 1
V = 1 } Hacemos caso de uno u otro segun interpretemos enteros sin o con signo

Registros de Direcciones A0 - A7



Permiten trabajar en .L o .W, como el 68000 tiene 24 lineas de dirección, trabajemos siempre en .L (y se malgastan 8 bits)

NOTA. Cuando se trabaja en .W el registro destino realiza extensión de signo

Usar A0 para referirse al contenido del registro y (A0) para referirse al contenido de la dirección apuntada

A0 - A7 no modifican los bits del CCR al realizarse operaciones con ellos.

Puntero de Pila. Stack Pointer. A8 o SP

- Son en realidad 2: USP (en modo usuario) y SSP (modo supervisor) solo está uno activo segun el modo.
- Apunta siempre al ultimo dato almacenado en la pila, modificado por el sistema de forma automática. se predecrementa para guardar datos y se postincrementa para sacarlos, tanto como sea el tamaño del dato.

Unica diferencia con A0 - A7 de SP(A8)

- .L [MOVE.L #0, -(An) / MOVE.L (An)+, #0] Predecrementa / Postincrementa 4 posiciones
- .W predecrementa/postincrementa 2 posiciones
- .B → A0 - A7: predecrementa/postincrementa 1 posicion
  → A8 o SP: predecrementa/postincrementa 2 posiciones

Para que siempre sea par, por si hay una interrupción

- Del A0 al A7 sigue pre/post incrementar 1 al usar .B, pero se supone que tu lo tienes en cuenta.
- Como el A8 se modifica automaticamente (p.ej si hay interrupción) el programador no puede controlarlo, por seguridad debe estar en posición par → incrementa 2 al usar .B

Contador de Programa

- Pasos que sigue un µP

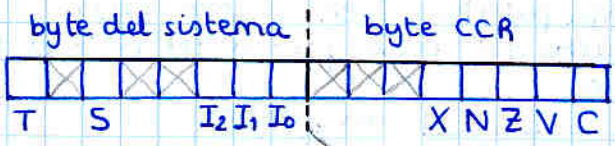
Tras el reset.

1. Mira su PC y hace búsqueda de instrucción FETCH. Almacena la instrucción en IR (instruction registry)
2. La lógica interna del µP interpreta los bits en IR y ejecuta la instrucción (en los ciclos que necesite para traer los datos, hacer algún cálculo y guardar los datos)
3. Incrementa el PC. Ya que las instrucciones ocupan como mínimo una palabra, el contador de programa se incrementa en dos unidades. Siempre deberá tener valor par.

De los 32 bits, PC sólo tiene activos 24 (líneas del 68000)

Registro de Estado (SR)

2 Bytes (16 bits)



Byte del Sistema Arranca con $ 27

0	0	1	0	0	1	1	1
T		S			I_2	I_1	I_0

T : bit de traza; permite ejecución paso a paso

S : 1 = supervisor
0 = usuario

I_2 I_1 I_0	Máscara de interrupciones
000	sin Prioridad
001	prioridad 1 : acepta interrupciones de nivel > 1
⋮	⋮
	prioridad N : acepta interrupciones de nivel > N
⋮	⋮
110	prioridad 6 :
111	No enmascarable : sólo acepta interrupciones nivel 7 (igual que 110) suelen ser fallos de alimentación.

C : Bit de acarreo: El resultado no cabe en los bits disponibles

V : Overflow / desbordamiento : al sumar 2 números con mismo MSB (most significant bit = bit de signo) el resultado tiene distinto MSB. Sirve para operaciones con signo. (resultado fuera de intervalo)

Z : Zero : el resultado es todo 0 (sin contar acarreo)

N : Bit de signo : indica resultado negativo (toma el valor de MSB)

X : Bit de extensión : para operaciones multiprecisión Normalmente X=C

ej:

$$\begin{array}{r} 1000 \\ +\ 1000 \\ \hline 10000 \end{array} \rightarrow \begin{array}{l} C = 1 \\ V = 1 \\ Z = 1 \end{array}$$

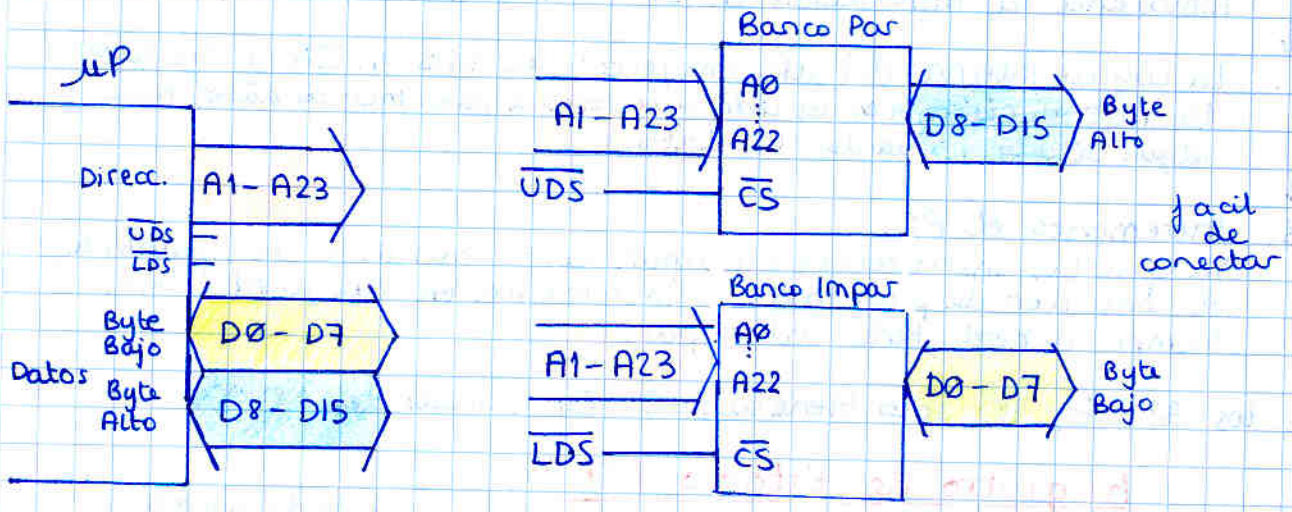
Organización de datos en memoria

En realidad 68000 no tiene 24 lineas de direcciones; le falta A0.

A0 en realidad sólo indica si una dirección es par o impar.
Trabajando con .L y .W A0 siempre 0 (numero par)
con .B A0 puede ser par o impar.

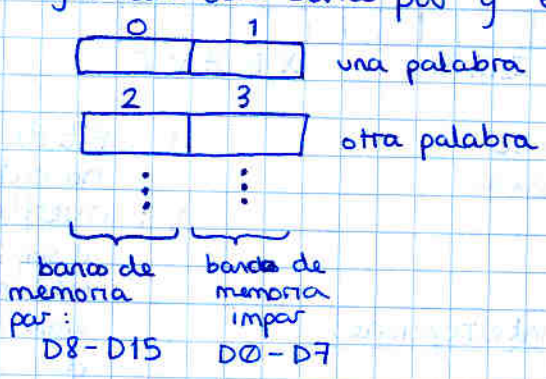
En lugar de A0 se utilizan dos lineas: $\overline{UDS}$ (Upper Data Stream) y $\overline{LDS}$ (Lower Data Stream)

$\overline{UDS} = 0$ $\overline{LDS} = 1$ → Direcciona Byte Par
$\overline{UDS} = 1$ $\overline{LDS} = 0$ → Direccionar Byte Impar
$\overline{UDS} = 0$ $\overline{LDS} = 0$ → Direcciona Palabra (necesariamente par)



facil de conectar

En efecto, UDS permite recibir el byte alto de datos y LDS el byte bajo. Para recibir una palabra (en un solo ciclo) activamos ambos; coge el byte alto del banco par y el bajo del impar

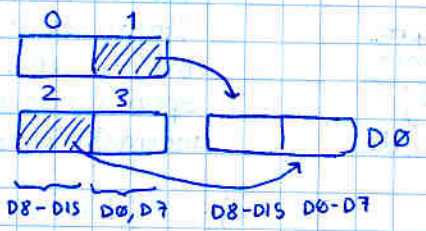


Los long word se cargan cargando dos palabras

Para trabajar en modo byte no hay mas que activar el byte par o impar (UDS y LDS: uno de ellos activos)

Si tratasemos de coger una palabra empezando en una posición impar

ej

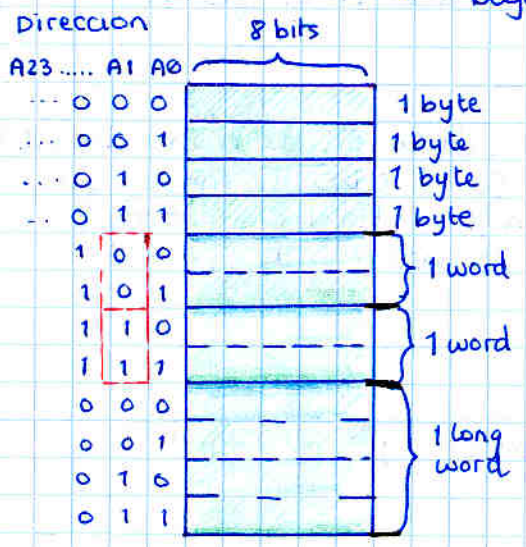


a parte de tener que hacer swaps, necesitariamos dos ciclos ya que la linea A1 cambia entre los dos Bytes, no se pueden coger a la vez.

⇒ .L y .W siempre en direcciones pares
si no: ERROR del µP

Alineamiento de Palabra:

En la memoria (formato Big Endian = Parte alta en posiciones bajas y parte baja en posiciones altas)

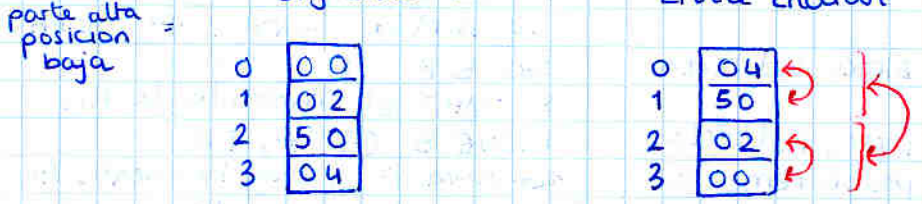


NOTAS:

- Las direcciones pares e impares; juntas desde un punto de vista 'lógico' estan en realidad en distintos bancos de memoria (el par y el impar)
- En cada word se mantiene el bit A1; permite coger los 16 bits en un solo ciclo por las 16 lineas de datos
- Siempre se deben reservar numeros pares de bytes si luego se quiere reservar words o longs (para que éstos esten en posición par)

Diferencia Big Endian (ej 68000) y Little Endian ej (8086)

para almacenar un long : ej: $00025004



Formato de instrucciones

ADD <ea>, Dn
Dn, <ea>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	1	Register			opmode			Effective Mode			Address Register		

código de ADD

especifica uno de los 8 registros de datos

opmode	Byte	Word	Long
<ea> + Dn → Dn	000	001	010
Dn + <ea> → <ea>	100	101	110

Effective Address (6 bits) <ea>
Indica el modo de direccionam.

Casi todas las operaciones son muy parecidas a ADD

MOVE <ea>, <ea> Es la orden más utilizada, se codifica con sólo 2 bits por lo que permite la gran flexibilidad de <ea> <ea> (6 bits por cada uno). Es la única instrucción que hace esto

0 0	Size	Destination <ea>	Source <ea>

01 - Byte
11 - word
10 - Long

campo de effective address

6 bits que aparecen en muchas instrucciones.
Indican el modo direccionamiento y si por ejemplo hay que hacer mas fetch

Effective Address	
Mode	Register

En el caso de que éste campo indique una dirección de memoria (.L) ésta estará en las dos words siguientes. En el caso de move, que tiene dos campos <ea>, esto puede significar hasta 4 words adicionales a la instrucción, haciendo un máximo de 5 words.

Modos de direccionamiento

Campo de effective Address		Mode de Direccionamiento	Ejemplo y/o Descripción
Modo	Registro		
000	num reg.	Registro Datos Directo	MOVE.L D0, D1
001	num.reg	Registro Direcc. Directo	MOVE.L D0, A1 No afecta al CCR Hace extensión de signo en .W! Por ello el nombre en el ensamblador se cambia: ADDA, SUBA, COMPA...
010	num.reg	Reg. Direcc. Indirecto	MOVE.B (A1), D0 se refiere al contenido de An
011	num reg	Reg. Direcc Indirecto con postincremento	MOVE.B (A1)+, D0 hace operación y luego incrementa An según el tamaño usado (cuidado: A7 = SP incrementa 2 en .B, el resto sólo 1)
100	num reg.	Reg. Direcc. Indirecto con predecremento	MOVE.W D0, -(A1) igual que anterior pero DEcrementa ANTES
101	num.reg.	Reg. Direcc. Indirecto con Desplazamiento	MOVE.B $8000 (A0), D0 D16 (ES) 16 bits y extensión de signo Suma un número a (A0) ej. para usar vectores
110	num.reg	Reg. Direcc. Indirecto con Indice	MOVE.W $30 (A0, D0.W), D2 D8 (ES) Xn.Z (ES) cualquier registro y cualq. tamaño suma $30 + A0 + D0.W y obtiene de ahí el dato
111	000	Absoluto Corto	MOVE.W D0, $8300.W E.S. y direccion 16 bits → absurdo
111	001	Absoluto Largo $25000	MOVE.B D0, $25301 Indica directamente direcc de mem Requiere 2 palabras + en la instruccion
111	010	Contador de Prog con Desplaz	MOVE.B $8(PC), D0 .....
111	011	Contador de Prog con Indice	
111	100	Inmediato # $30	MOVE.W # $43F8, D0 indica nº de fetch que debe hacer

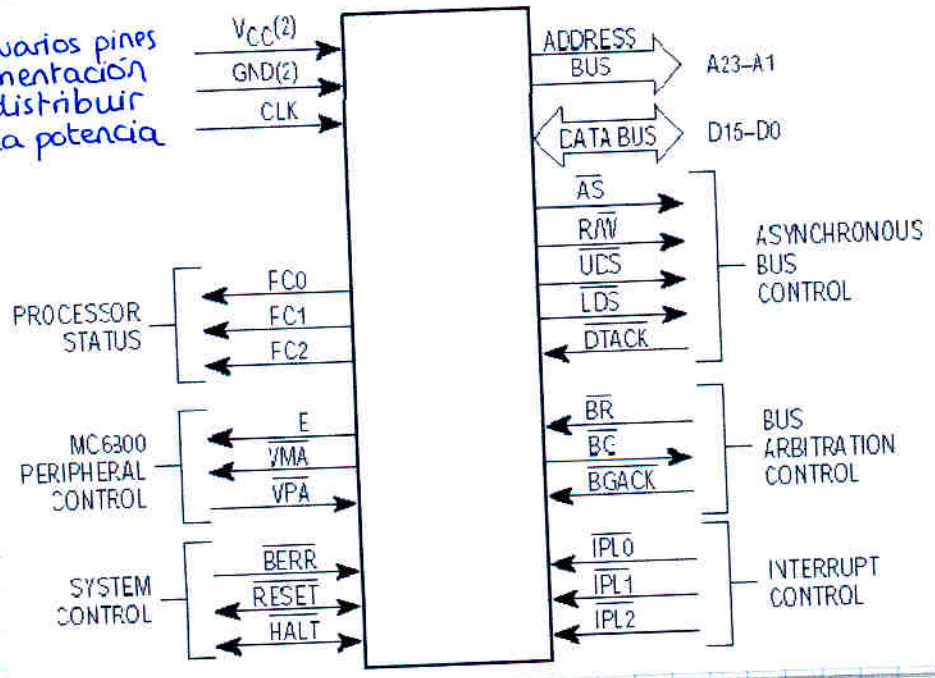
Se apoyan en registros .... (requieren nº reg. 3 bits)

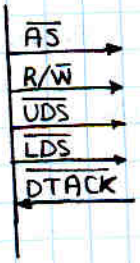
No confundir

No se codifican:
- Inmediato Rápido: ADDQ, MOVEQ : El valor cabe en 3 u 8 bits según la instrucción
- Implícito: ej JMP ⇒ PC

TEMA 3. LINEAS Y OPERACIONES DE BUS DEL MC68000

tiene varios pines de alimentación para distribuir bien la potencia





$\overline{AS}$: Address Strobe
Validación de direcciones
Indica que A23 - A1, $\overline{UDS}$ y $\overline{LDS}$ tienen una dirección fija y correcta → el ciclo está activo

R/$\overline{W}$: 1: µP lee, 0: µP escribe, Alta impedancia

$\overline{DTACK}$: Data acknowledge.
El sistema indica al µP que se ha completado transferencia de información → fin del ciclo
← comportamiento asíncrono ← handshake ← robustez

Bus arbitration control:

ej. la µP tardaría mucho en copiar grandes cantidades de información de una memoria a otra (leer instrucción, leer datos, escribir datos...)
Se utilizan controladores de memoria DMA
Hacen transferencias entre memorias 'a saco'
Necesitan el control del bus; el µP debe dejar el bus en paz.

← $\overline{BR}$: "oye me das el bus?"
→ $\overline{BG}$: "vale"
← $\overline{BGACK}$: se mantiene 'pulsado' mientras se utiliza el bus; se 'suelta' cuando ya han acabado y el µP debe volver al trabajo.

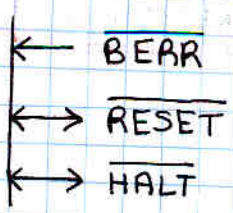
Interrupt Control

← $\overline{IPL0}$
← $\overline{IPL1}$
← $\overline{IPL2}$

Indica una interrupción y el nivel de ésta

Si el µP ve que la interrupción es de mayor nivel que su máscara la atenderá

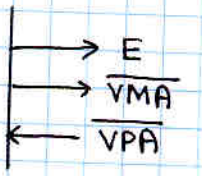
System Control



BERR: Bus ERRor - se activa tras cierta temporización
Para que µP no espera indefinidamente un DTACK que no llega por alguna transferencia errónea.
El µP termina el ciclo e intenta corregirlo

RESET

HALT: Permite ir parando el µP (depuración paso a paso)

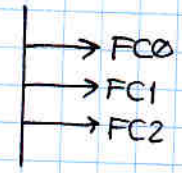


MC 6800 Peripheral Control

E: Reloj a 1/10 de la frec del 68000 y con ciclo de trabajo 40%
Para compatibilizar el 68000 con todos los periféricos del antiguo 6800 que funcionaban muy bien

VPA: Funciona como el DTACK

VMA: Indica q. E está sincronizado. Usaremos VMA para la ecuación de los dispositivos (tipo CS)



Procesor Status

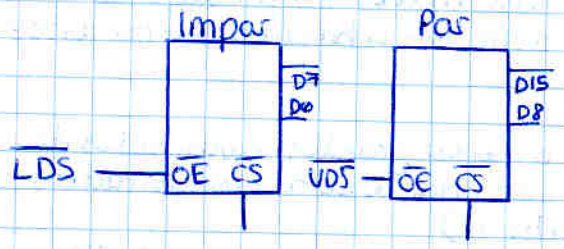
Indica cosas como:

- Usuario / Supervisor (FC2 = 0 / FC2 = 1)
- memoria Programa / memoria Datos (FC1, FC0 = 10/01)
  i.e. Fetch / move, add, ...
- Ciclo de reconocimiento de interrupción (FC3, FC2, FC1 = 111)

Puedo utilizar estas líneas para extender el mapa de memoria (mem de programa / datos, ...)

Decodificación de memoria

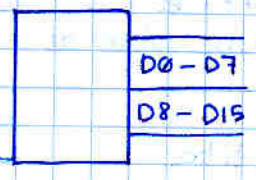
ej. Decodificación EPROM



$\overline{CS} = f(A_1 \ldots A_n, \overline{AS}, R/\overline{W})$
puede ser el mismo para ambas

Memoria 16 bits

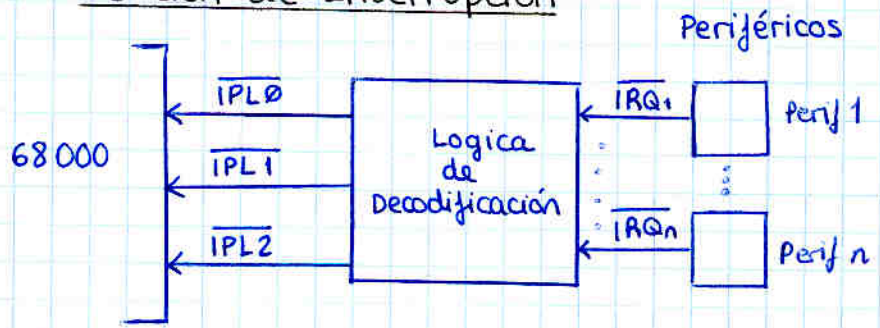
Una memoria de 16 bits se puede incluir sin ningun problema siempre que realizemos unicamente lecturas: i.e. EPROM



El µP coge sólo los bits que se le pidan (byte alto, byte bajo o word) aunque la memoria coloque siempre en el bus todo el word.

Una RAM no funcionaría a la hora de escribir en ella un único byte, ya que ésta cogería todo el word, poniendo la basura del bus encima del byte que no se quería tocar.

Petición de Interrupción



ejemplo: 3 dispositivos

IRQA (nivel prioridad 2)
IRQB (nivel prioridad 6)
IRQc (nivel prioridad 6)

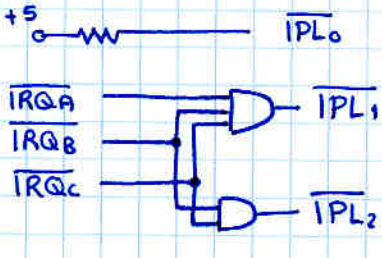
· A mano (no recomendado a menos que sea inmediato)

$\overline{IRQA}$	$\overline{IRQB}$	$\overline{IRQc}$	Nivel	$\overline{IPL_2}$	$\overline{IPL_1}$	$\overline{IPL_0}$
0	0	0	6	0	0	1
0	0	1	6	0	0	1
0	1	0	6	0	0	1
0	1	1	2	1	0	1
1	0	0	6	0	0	1
1	0	1	6	0	0	1
1	1	0	6	0	0	1
1	1	1	0	1	1	1

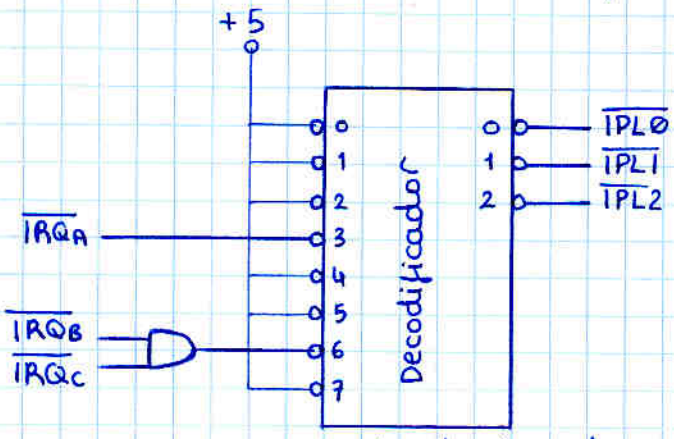
$\overline{IPL_0} = 1$

$\overline{IPL_1} = \overline{IRQ_A} \cdot \overline{IRQ_B} \cdot \overline{IRQ_c}$

$\overline{IPL_2} = \overline{IRQ_B} \cdot \overline{IRQ_c}$



· Con C.I. Decodificador con Prioridad 74148
Saca por su sálida de 3 bits el número de la entrada más alta que esté activa (a cero)

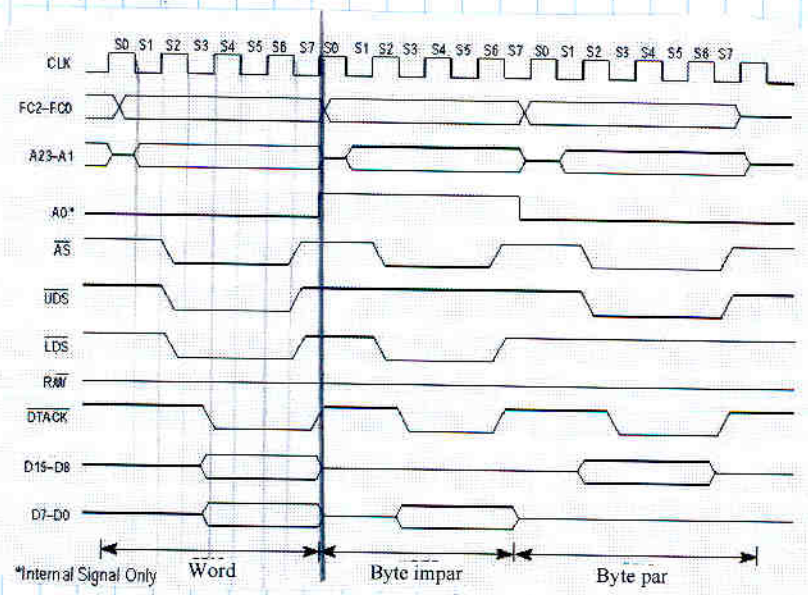


Retardo igual a una única puerta lógica.

Error de direccionamiento

Has accedido en .W a posicion impar

Ciclo de lectura



El ciclo ideal de lectura son 4 ciclos. Cada semiciclo corresponde a un estado (internam. tiene un reloj doblado)

Los dos estados (semiciclos) de un mismo ciclo van siempre juntos e inseparables

ej. veremos que si en S4 no detecta DTACK, se ve obligado a volver a S3 (y no puede a S4) ya que está en un flanco de bajada, que corresponde a S3.

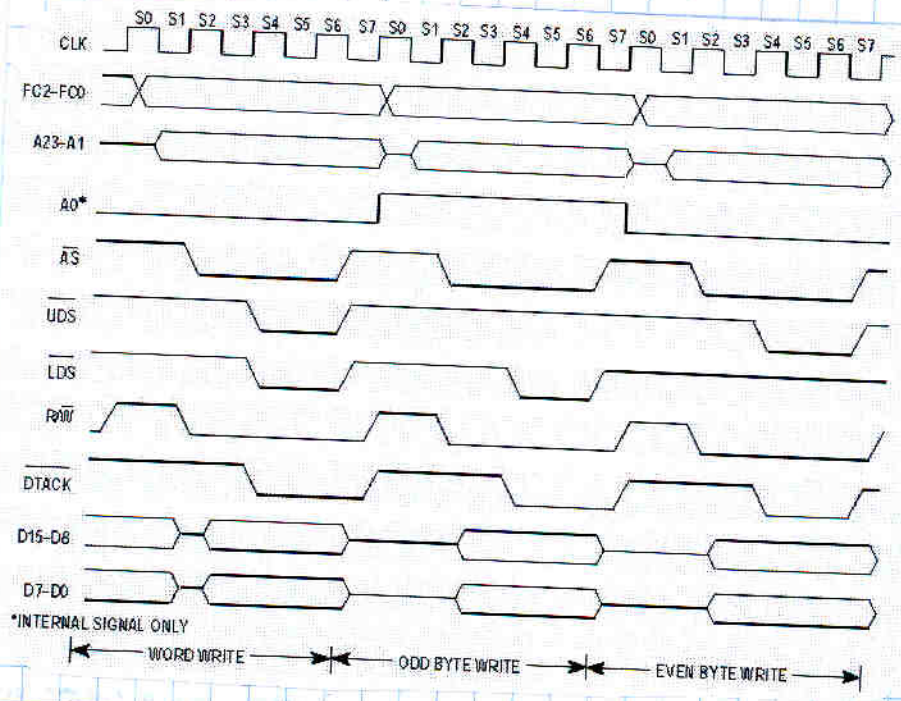
- S0 :- pone valores de FC (instruccion/datos, usuario/supervisor)
  - pone R/$\overline{W}$ a 1 (aunque el ciclo fuera de escritura)
- S1 - pone direcciones válidas en A23-A1
- S2 - valida direcciones (Address Strobe = $\overline{AS}$ = 0)
  - indica tamaño con $\overline{UDS}$ y $\overline{LDS}$ (byte par/impar o word)
- S3 El sistema tiene lo que necesita. Es un estado de espera para que las señales se propagen hacia el sistema ($\overline{OE}$'s y $\overline{CS}$'s)
- S4 - El µP comprueba $\overline{DTACK}$ en el flanco de bajada. Si está activa continúa; sino vuelve a S3.
- S5 - muestrea bus de datos
- flanco S6-S7 : - Desactiva $\overline{UDS}$, $\overline{LDS}$, $\overline{AS}$
- flanco S7-S0 : - quita las direcciones

NOTAS:

Como mínimo cualquier operación de bus dura 4 ciclos. Se consigue esta duración mínima si el sistema logra responder $\overline{DTACK}$ antes del flanco de bajada de S4.

Muchas memorias no tienen $\overline{DTACK}$ y éste lo genera el diseñador a partir de los $\overline{CE}$ y $\overline{OE}$

Ciclo de escritura



- S0 - pone FC's
  - R/$\overline{W}$ a 1
- S1 - pone A23-A1
- S2 - R/$\overline{W}$ a 0
  - $\overline{AS}$ activa (valida direcciones)

¿Porqué aún no ha puesto $\overline{UDS}$ y $\overline{LDS}$?

Porque además de hacer de A0, $\overline{UDS}$ y $\overline{LDS}$ sirven para validar el bus de datos (equiv. a $\overline{AS}$ en direcciones)

- S3 - pone datos en bus de datos
- S4 - $\overline{UDS}$ y $\overline{LDS}$ ⇒ valida datos e indica el tamaño

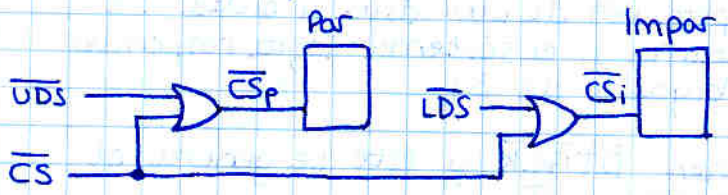
El sistema ya lo tiene todo. Una vez ha capturado los datos envia DTACK

- S5 (flanco de bajada) - El sistema comprueba $\overline{DTACK}$ muestreando la linea
- S6 → S7 : va quitando las señales

NOTA

Si el sistema no tiene $\overline{DTACK}$ el diseñador puede utilizar validación anticipada.

Cuidado; en este caso $\overline{UDS}$ y $\overline{LDS}$ se activan poco antes de muestrear DTACK, por lo que es peligroso generar el DTACK utilizando estas señales: por el tiempo de propagación y el de hold nos pasamos.



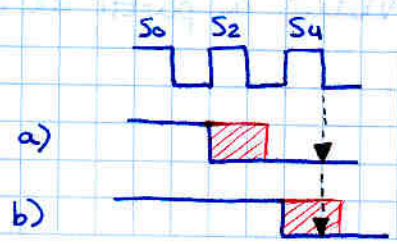
$\overline{DTACK} = \overline{CS_p} \cdot \overline{CS_i}$
$= (\overline{CS}+\overline{UDS}) \cdot (\overline{CS}+\overline{LDS})$

2 opciones para generar $\overline{DTACK}$

ⓐ $\overline{DTACK} = \overline{CS}$

ⓑ $\overline{DTACK} = (\overline{CS}+\overline{UDS}) \cdot (\overline{CS}+\overline{LDS})$

Desde un punto de vista lógico, las dos opciones son la misma. Desde un punto de vista temporal, la opción ⓑ ademas de requerir más puertas no llega a tiempo



Ciclo de Lectura-Modificación-Escritura

Es un ciclo especial para un caso muy específico.

Accede a una dirección en la cual:
- Lee
- modifica internamente lo leido
- escribe

Es exactamente igual a hacer los 3 ciclos uno detras de otro, excepto porque $\overline{AS}$ se mantiene a nivel bajo

Se utiliza unicamente con la instrucción TAS (Test And Set) para la:

Implementación de semáforos

En sistemas multiprocesador con recursos de memoria compartidos; para no estar utilizando varios µP una misma memoria a la vez se usan unas banderas llamadas semáforos (bit 7 de una posición)

- Cuando µP entra: semáforo a 0
- Cuando µP acaba: semáforo a 1

Antes de entrar el µP comprueba el semáforo

Podriamos hacer: BSET.B #7, $25000

① Ciclo Lectura $25000
② $\overline{b7} \rightarrow Z$, $1 \rightarrow b7$
③ Ciclo escritura $25000

BEQ PROCESO

Problema; puede llegar una interupción, un bus grant, otro µP ponerse con la memoria... entre el ciclo de lectura y el de escritura.

En este instante el semáforo podría estar a 1 (libre) aún cuando el µP ya está pensando en poner un 0 y utilizar el recurso compartido.

Sistema podria pensar que el recurso está libre → si ahora llegara un bus grant, el µP se quedaría esperando para hacer la escritura del semáforo cuando acabe el bus grant, lo cual podria ser catastrófico.

Solución:

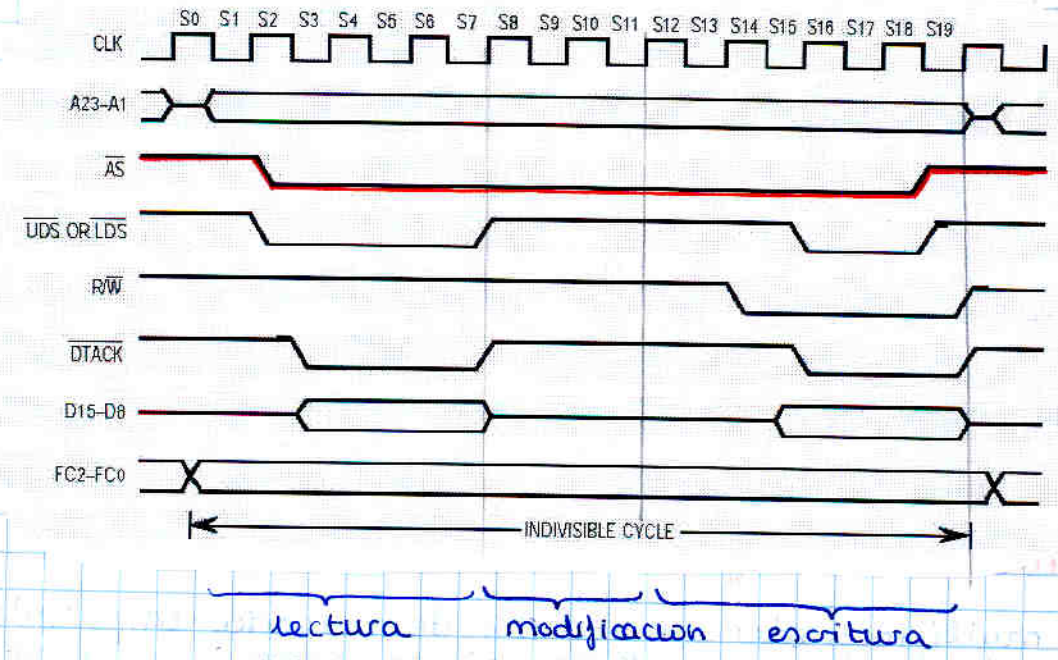
TAS.B $25000 ≡ BSET.B #7, $25000

igual excepto que $\overline{AS}$ se mantiene a nivel bajo

↓ Utiliza ciclo de lectura-modificación-escritura Pensado únicamente para este propósito.

→ En tiempo tarda igual
→ No te pueden coger el bus a mitad del ciclo

Ciclo de lectura-modificación-escritura

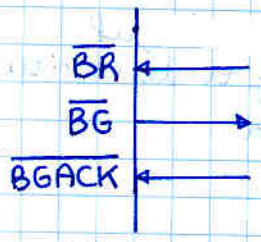


control de la posesión del bus

Pensados para DMA (dispositivos capaces de transferencias masivas de información a alta velocidad)

Estos dispositivos deben usar el bus, y para ello deben pedir al µP que ponga el bus en alta impedancia

Como los sistemas que piden el bus los maneja (supuestamente) el propio µP, éste SIEMPRE accede a los Bus Request ($\overline{BR}$)

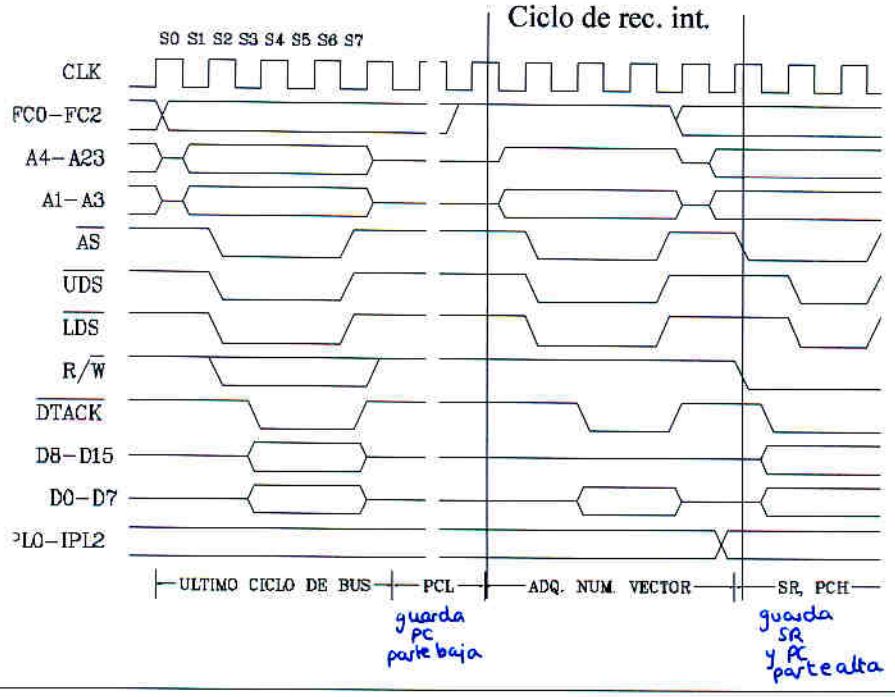


- Llega un Bus Request ($\overline{BR}$) del sistema
- Bus grant ($\overline{BG}$) se activa, indicando que: el µP cederá el bus cuando acabe el ciclo
- El sistema espera a que $\overline{AS} = 1$ (fin ciclo) (he ahí utilidad del lectura-modificación-escritura) El sistema 'pisa' (activa) la señal Bus Grant Acknowledged (BGACK) y la mantiene 'pisada' hasta que acaba
- Al quedar $\overline{BGACK}$ inactivo, el µP sigue con su secuencia normal

Ciclo de reconocimiento de interrupciones

- Se indica colocando las líneas FCx a 1.
- Surge como respuesta a una petición de interrupción aceptada
  llega $\overline{IPLX} \neq 111$
- El nivel de interrupción que se está reconociendo se muestra en las líneas A1, A2 y A3.
- Las líneas A4-A23 se ponen a 1.
- Es un ciclo de lectura byte especial ya que el uP únicamente captura los datos de D0-D7 (adquisición del número de vector) estando /UDS y /LDS a 0.
- En el proceso de excepción correspondiente a una interrupción externa se realizan más ciclos que incluyen operaciones con la pila, saltos y cálculos internos

Ciclo de reconocimiento de interrupciones



Ciclo de Reconocimiento de Interrupciones

FCx :
- programa(fetch) / datos (R/W)
- usuario / supervisor
- Ciclo de reconocimiento de interrupción
  FC0 = FC1 = FC2 = 1

- Entra una interrupción de nivel n (ha llegado $\overline{IPLx} \neq 111$)
  Si la máscara < nivel, el µP la atiende
- µP reconoce la interrupción de la interupción de mayor nivel que haya
  FCx : indica ciclo de rec de interrupción
  A1, A2, A3 : indican el nivel de interrupción que reconoce
  (el resto se ponen a uno)
- El ciclo será como de lectura
  El µP captura el byte $\underbrace{D0 - D7}_{\text{se llama número de vector}}$ (aunque $\overline{UDS}$ y $\overline{LDS}$ están ambas a cero)
  (al recibir el DTACK)

- Una vez capturado el número de vector, el µP entra a la rutina de interrupción saltando a una dirección de memoria

Hay 2 tipos de dispositivos que interrumpen :
a) Dan nº de vector (familia del 68000)
b) No dan nº de vector

a) Familia 68K
Todos poseen linea $\overline{IACK}$ y $\overline{DTACK}$
con $\overline{IACK}$ activa, el dispositivo activa $\overline{DTACK}$ y ponen el número de vector por sus D0 - D7
(nota: deben de estar conectados a parte baja del bus de datos = posición impar)

b) El dispositivo no tiene capacidad de dar $\overline{DTACK}$ ni dar vector por D0 - D7
Para indicarlo activa el $\overline{VPA}$ del µP (no necesariamente el dispositivo es del 6800) y ante esto el µP le asigna un número automaticamente.
Generamos

$$\overline{IACK} = \overline{FC0} + \overline{FC1} + \overline{FC2} + \overline{AS} + \underbrace{f(A1-A3)}_{\text{que sea mi nivel}} + \overline{IRQ}$$

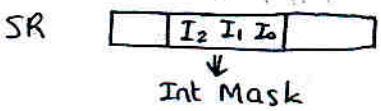
$\overline{VPA}$ = lo mismo

Por tanto para conectar 2 de estos de un mismo nivel hay que priorizar por hardware

Ciclo de reconocimiento de interrupciones

- Llega $\overline{IPLX} \neq 111$ interrupción al µP.

  Si el nivel definido por IPLX es mayor que el de la máscara, el µP atiende la interrupción.



- Para saber quién es el dispositivo que interrumpe:
  El µP necesita un vector de 8 bits 64-255 las vectorizadas
  25-32 las autovectorizadas

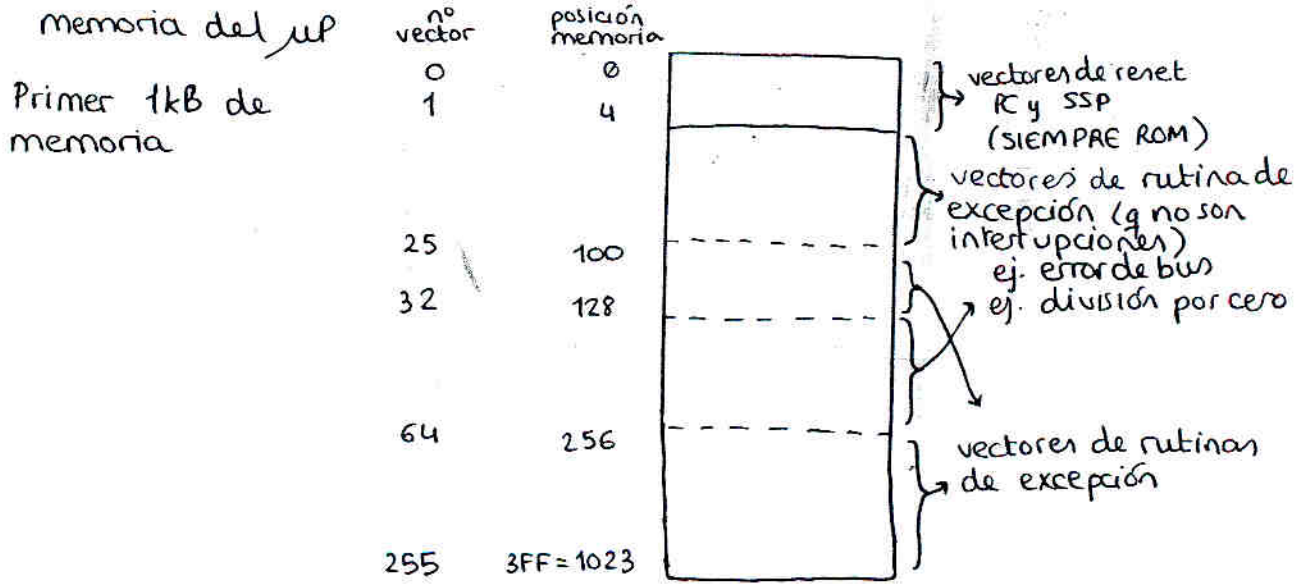
  El µP hará nº vector · 4 = vector interrupción
  = dirección de memoria del primer PC de la rutina de excepción

  para que sean posiciones de tipo palabra larga

¿Cómo obtiene el número de vector una vez se acepta la interrupción?

Ciclo reconocimiento de interrupciones.
FCx a 1
A1-A3 indica el nivel de interrupción. A4-A23 a 1. $\overline{UDS} = \overline{LDS} = 0$
El µP lee el número de vector en DØ-D7

El sistema debe proporcionar un dato de 8 bits



¿Cómo devuelve el sistema el nº de vector ante el ciclo de int?

1- Que el dispositivo esté diseñado para ello (devolver vector (64-255), DTACK, ...)

2- Para dispositivos muy básicos ej: pulsador
   El sistema envia VPA
   El µP asocia el nº de vector 24 + Nivel Interrupción

3- $\overline{BERR}$. El dispositivo no responde (ej: ya no hay interrupción)
   no hay $\overline{DTACK}$ ni $\overline{VPA}$ → $\overline{BERR}$

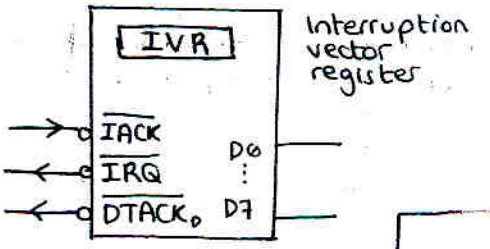
Se llama interrupción espúrea.
- nº vector = 24

El µP hará lo que el diseñador haya programado

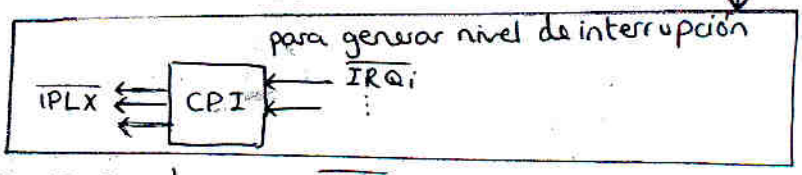
4. Nº de vector 15 : (fuera del rango)
   Rutina de atención a dispositivo no inicializado.
   ↳ Los dispositivos preparados para devolver nº vector tienen un 15 por defecto si no les programas nada

Dos tipos de dispositivo

- Responden con $\overline{DTACK}$:



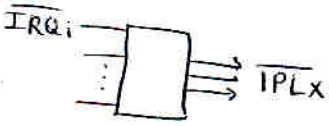
→ Dispositivo manda $\overline{IRQ}$ cuando tiene una interrupción.
↳ CRC : ciclo reconocim. interrupción
- Llega $\overline{IACK}$ (generado a partir de FCx=1, A1-A3=nivel)
- Copio IVR a D0-D7
- $\overline{DTACK}$ = 0



- Responden con $\overline{VPA}$
  El sistema únicamente responde con $\overline{VPA}$

Resumen Interrupción

— Dispositivo manda $\overline{IRQ}$ cuando tiene interrupción



interrupción al µP

— Comienza CRI

µP pone FCx = 111
A1-A3 = nivel interrup
A23-A4 = 1
} cuidado que no direccione a ningún dispositivo

↳ si el dispositivo tiene nº vector, le hacemos llegar IACK con estas señales, el cual copia su nº de vector a D0-D7 ⇒ Posición Impar y devuelve DTACK

↳ Si el dispositivo no tiene nº de vector
Generamos con estas señales y el $\overline{IRQ_i}$ que generó al principio una $\overline{VPA_i}$ que entra al µP

El µP continuará entonces con su ciclo de lectura largo (i.e. activando $\overline{VMA}$ 2 ciclos antes de que suba E) pero sin leer ningún dato; en vez de ello se generará él mismo su vector como nivel de interrupción + 24

Ejercicio examen

Periferico	¿Puede generar num vector?	Nivel
A	No	2
B	No	6
C	Si	6

Ya hicimos:

$$\overline{IPL_0} = 1$$
$$\overline{IPL_1} = \overline{IRQ_A} \cdot \overline{IRQ_B} \cdot \overline{IRQ_C}$$
$$\overline{IPL_2} = \overline{IRQ_B} \cdot \overline{IRQ_C}$$

Circuito de Petición de Interrupción (CPI)

A la hora de reconocer la int, necesitamos saber cual de los dos (B o C) es mas prioritario DATO: C es más prioritario

NOTA: No conviene poner dos dispositivos sin capacidad de generar número de vector en el mismo nivel.
(A no ser que en la propia rutina de interupción se distinga por software un dispositivo u otro)

$$\overline{VPA_{INT_2}} = \underbrace{\overline{FC0} + \overline{FC1} + \overline{FC2}}_{\text{esto ya implica } A4-A23=1} + \overline{AS} + \underbrace{A1 + \overline{A2} + A3}_{\text{nivel de interrupción}} + \underbrace{\overline{IRQ_A}}_{\text{que yo haya interrumpido.}}$$

No hace falta incluirlo igual que $\overline{UDS} = \overline{LDS} = 0$

En teoria, como A es el unico disp con nivel 2, las lineas $A1 + \overline{A2} + A3$ implican $\overline{IRQ_A}$
Pero conviene ponerlo por si se producen interrupciones espureas

$$\overline{VPA_{INT_6}} = \overline{FC0} + \overline{FC1} + \overline{FC2} + \overline{AS} + A1 + \overline{A2} + \overline{A3} + \overline{IRQ_B} + \underbrace{\overline{\overline{IRQ_C}}}_{\text{para priorizar el C}}$$

Para el C tenemos que generarle $\overline{IACK}$ (no confundir con DTACK)

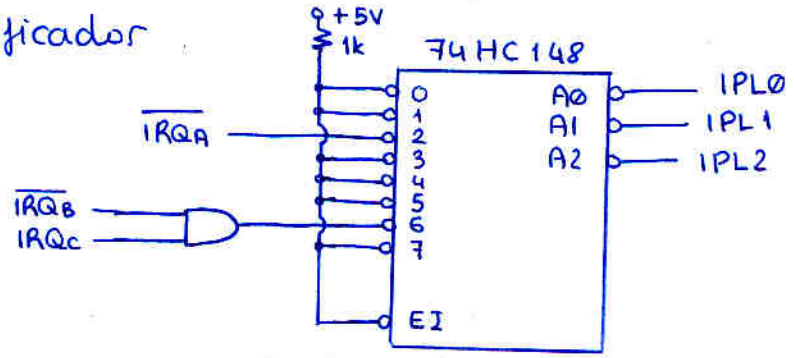
$$\overline{IACK} = \overline{FC0} + \overline{FC1} + \overline{FC2} + \overline{AS} + A1 + \overline{A2} + \overline{A3} + \underbrace{\overline{IRQ_C}}_{\text{opcional}}$$

opcional, ya que si el dispositivo recibe $\overline{IACK}$ sin tener interrupción, no responde con $\overline{DTACK}$

$$\overline{DTACK_{\mu P}} = \overline{DTACK_C} \cdot \overline{DTACK_{\text{resto circ}}}$$

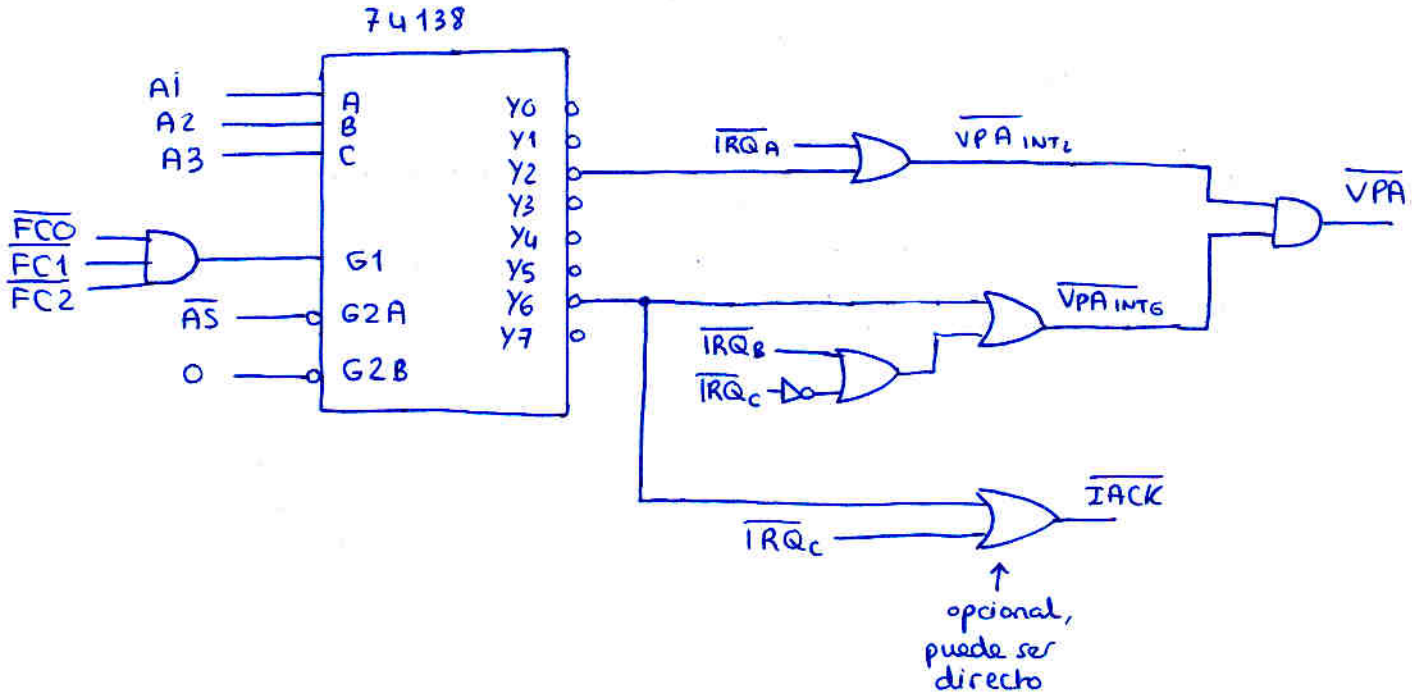
Integrado usado para petición interrupción

usar un codificador



Integrado usado para reconocimiento de interrupción

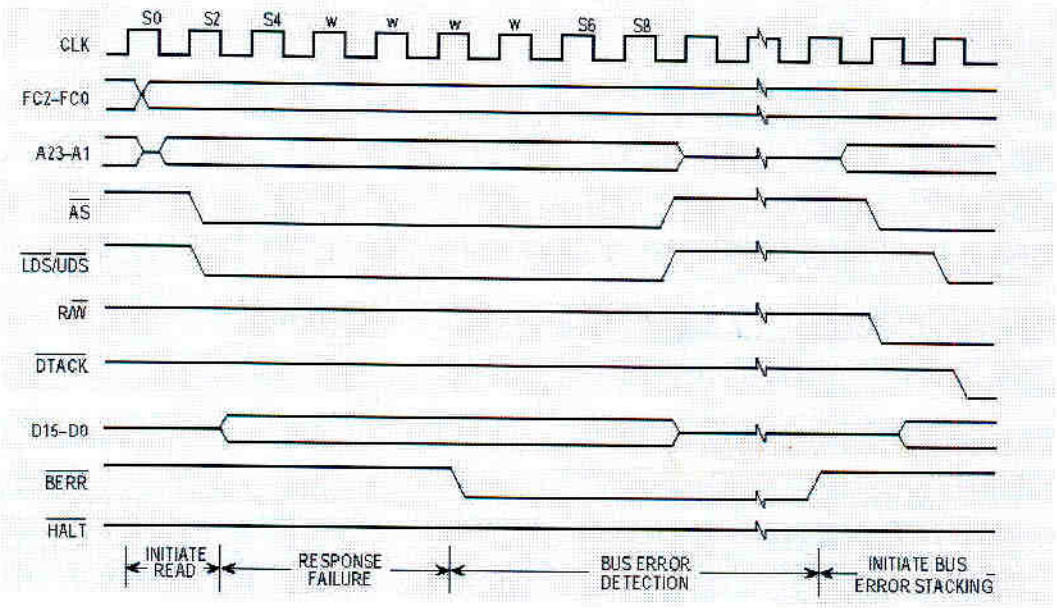
usar un decodificador



Ciclos especiales. Error de Bus

- La señal /BERR de entrada al uP, sirve para terminar un ciclo en situación anómala (no responde el dispositivo).
- Esta señal se deberá activar cuando /AS lleve un tiempo considerablemente largo activada.
- Se produce una excepción de error de bus cuando: /DTACK=1, /HALT=1 y /BERR=0.
- Cuando se reconoce el error de bus, el ciclo termina en el estado S9 (dos más), pasando los buses de datos y de direcciones a Z mientras /BERR=0.
- Una vez abortado el ciclo y /BERR=1, se entra en un proceso de excepción en el que se guarda información en la pila.
- La circuitería de error de bus se tiene que realizar externamente.
- Se produce un doble error de bus cuando en el transcurso del proceso de excepción se produce otro error de bus. Entonces el microprocesador para y pone la línea /HALT a 0.

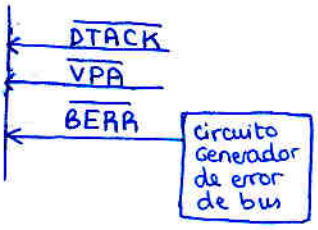
Ciclo típico de error de bus



Error de bus ($\overline{BERR}$)

cuando el $\overline{DTACK}$ (o $\overline{VPA}$) no llega nunca se produce un cuelge indefinido a causa de fallo software o hardware.

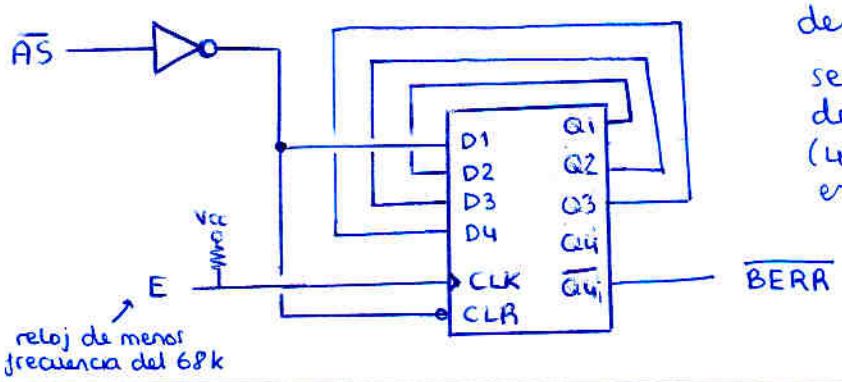
mediante un hardware adicional (circuito de generacion de error de bus) puedo sacar al µP del cuelgue activando $\overline{BERR}$.



mientras el µP esté en estado de espera nunca llegará al ciclo S6 donde pone Address Strobe a uno, por tanto podemos hacer el circuito gen de error a partir de un contador de ciclos de $\overline{AS}$ a nivel bajo.

No se podría hacer con $\overline{DTACK}$ / $\overline{VPA}$ ya que podria activarse $\overline{BERR}$ en momentos en que el µP ni siquiera esté esperando datos
- ej. el µP está multiplicando
- ej. al µP le roban el bus
- ej. el µP está en stop esperando interupción

Circuito gen de error de bus



Es un registro de desplazamiento.
se activa $\overline{BERR}$ al cabo de 4 ciclos de E (40 ciclos del 68k) estando $\overline{AS} = 0$

Tras $\overline{BERR}$, el µP entra en un estado de excepción

Parada ($\overline{HALT}$)

Al llegar $\overline{HALT}$, el µP termina el ciclo y luego se para (no se para a mitad de ciclo)
El µP continua con el siguiente cido cuando la linea $\overline{HALT}$ pasa a 1

Ciclo de Relanzamiento ($\overline{BERR}$ y $\overline{HALT}$)

Si se reciben (antes de $\overline{DTACK}$) $\overline{BERR}$ y $\overline{HALT}$ pueden pasar dos cosas:

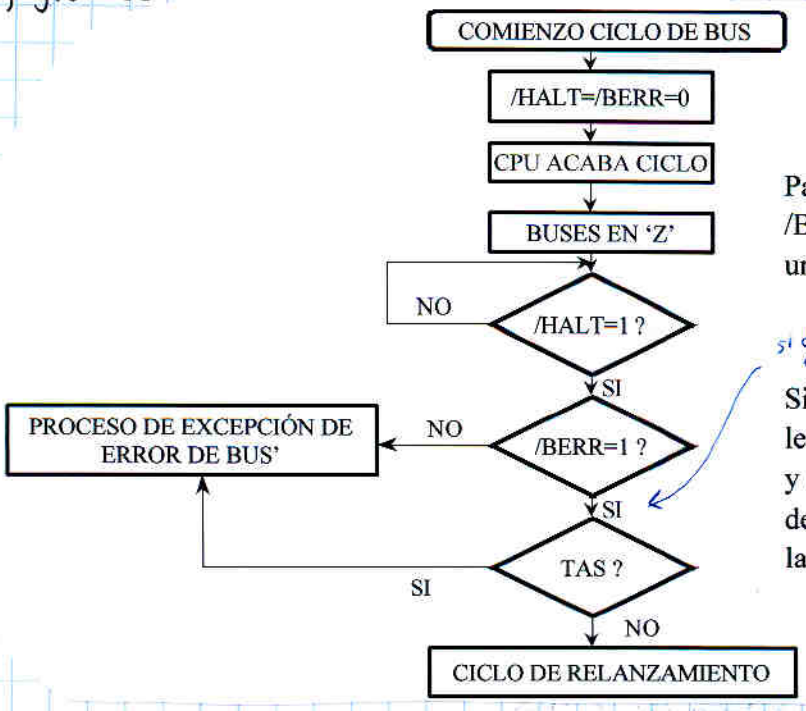
- Si $\overline{BERR}$ sube DESPUES de $\overline{HALT}$ (i.e. BERR está mas tiempo activa) → Proceso de excepción de error de bus ↑ (excepto si el ciclo es TAS)
- Si $\overline{BERR}$ ha subido antes de $\overline{HALT}$ (i.e. BERR está menos tiempo activa) ↓ Ciclo de relanzamiento (reintenta el ciclo anterior)

es decir, el que aguanta más es el que decide

El ciclo de relanzamiento puede ser util en, por ejemplo, un sistema autónomo.
Un sistema autónomo que funcione bien no tiene porqué tener un error de bus (no tiene sentido), y si lo tiene debe deberse a algo como por ejemplo ruido externo, en ese caso conviene que se reintente el ciclo anterior (en lugar de hacer proceso de excepción)

El diseñador puede diseñar por hardware el circuito generador de bus para que active a la vez $\overline{BERR}$ y $\overline{HALT}$ y que mantenga $\overline{HALT}$ un ciclo más (por ej)
(sería un buen diseño para preguntar en examen)

Flujograma



Para un ciclo de relanzamiento /BERR=1 antes (por lo menos un ciclo) que /HALT=1

si dice que SI significa que BERR ya estaba a 1 antes de que HALT lo estuviera

Si estamos en un ciclo de lectura-modificación-escritura y /BERR=0 siempre será error de bus independientemente de la línea /HALT.

Resumen terminaciones de ciclo

- $\overline{DTACK}=0$, $\overline{BERR}=\overline{HALT}=1$: Normal
- $\overline{HALT}=0$ antes o a la vez que $\overline{DTACK}$, $\overline{BERR}=1$: Parada
- $\overline{BERR}=0$ antes o a la vez que $\overline{DTACK}$, $\overline{HALT}=1$: Proceso Excepción error de bus
- $\overline{BERR}=\overline{HALT}=0$ antes o a la vez que $\overline{DTACK}$: (ver flujograma de arriba)

Operación de RESET

Mantenidos $\overline{RESET}$ y $\overline{HALT}$ > 100 ms tras alcanzar Vcc los 5 V

① Arranque interno (inicializa flip-flops, registros, ...)

② Acceso tipo word : SSP Parte Alta
dirección Ø

③ Acceso tipo word : SSP Parte Baja
dirección 2

④ Acceso tipo word : PC Parte Alta
dirección 4

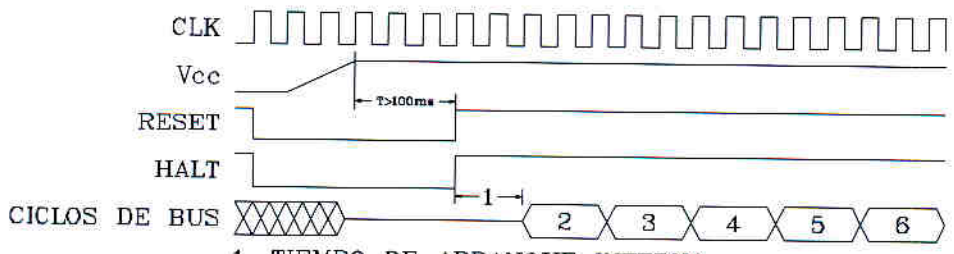
⑤ Acceso tipo word : PC Parte Baja
dirección 6

⑥ Lectura de la 1ª Instrucción

Memoria ROM de supervisor

Ø	SSP H
2	SSP L
4	PC H
6	PC L

- RESET FRIO.- El uP necesita las líneas /RESET y /HALT a nivel bajo durante 100ms en la primera inicialización.
- RESET CALIENTE.- Se reinicializa el sistema si las líneas /RESET y /HALT están a nivel bajo al menos 10 ciclos de reloj.



1. TIEMPO DE ARRANQUE INTERNO
2. LECTURA DEL SSP PARTE ALTA
3. LECTURA DEL SSP PARTE BAJA
4. LECTURA DEL PC PARTE ALTA
5. LECTURA DEL PC PARTE BAJA
6. LECTURA DE LA PRIMERA INSTRUCCION

NOTAS:

- Las primeras 6 direcciones deben ser SIEMPRE ROM y tener SSP y PC
  - El valor de SSP suele ser la ultima posición PAR de memoria RAM existente
- Las siguientes posiciones no se deben usar (recomendado dejar > 400 posiciones), son para los vectores de excepción
  - El valor de PC suele ser > 400

NOTA 2: Si en uno de los 4 accesos se produce un error de bus
→ PARADA IRREVERSIBLE

Ciclos Reales (retardos, tiempos propagación, ...)

Operación asíncrona de transferencia de datos

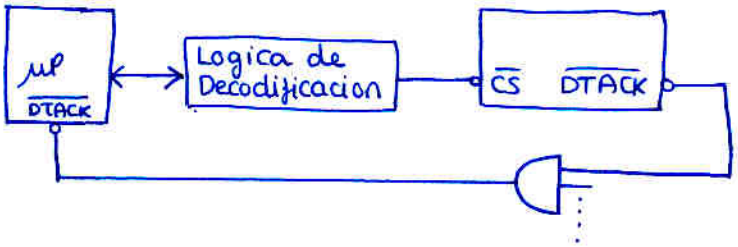
Para dispositivos pensados para el 68000; disponen de DTACK

Cumple los criterios que buscamos:

① Transferencia correcto

② lo más rápido posible ⟺ mínimos recursos hardware

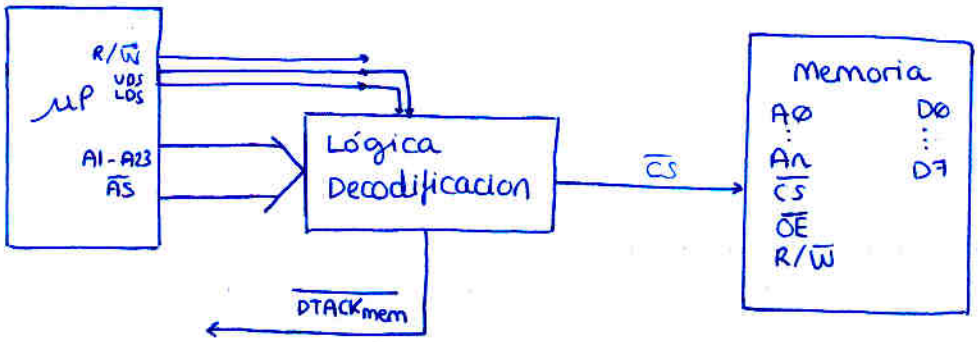
ej 68681 : DUART



Operación síncrona de transferencia de datos

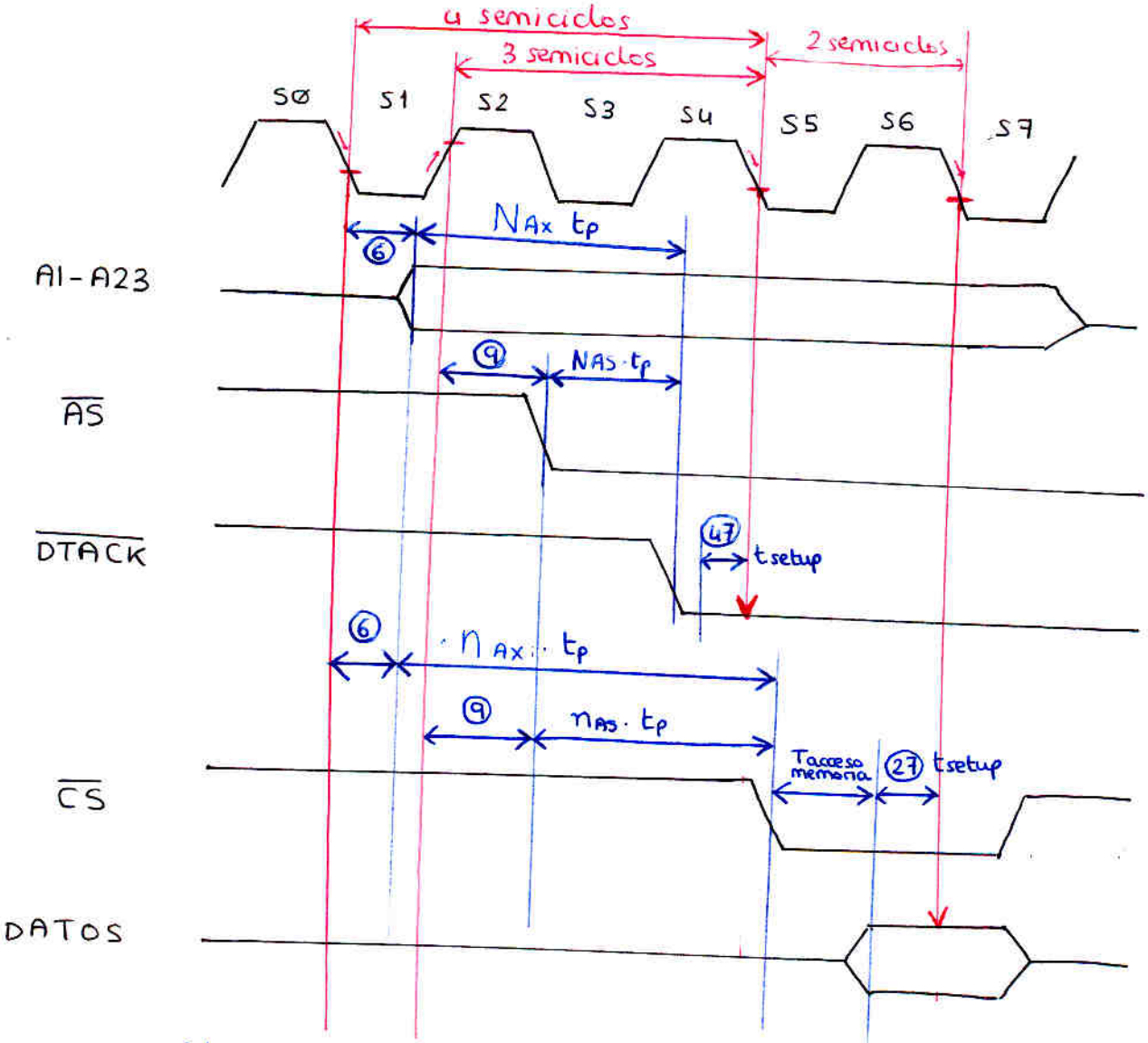
- La mayoria de memorias no tienen pin $\overline{DTACK}$
- Generamos DTACK a partir de las propias señales del µP

Ciclo de lectura



$\overline{DTACK}$ se genera a partir de $\overline{AS}$ y de las direcciones Ax.
DTACK debe llegar antes del flanco de bajada de S4 para no generar estados de espera innecesarios.
En general funciona generar DTACK antes de tiempo, pero habrá que comprobar que luego los datos estén en el bus a tiempo, si no lo estan, hay que retrasar DTACK

$\overline{CS}$ se genera a partir de $\overline{AS}$, de las direcciones Ax y, a diferencia de DTACK, tambien a partir de $\overline{UDS}$ y $\overline{LDS}$
(por si hay escritura que escriba en el byte correcto)
$\overline{CS}$ debe ser lo más rápido posible para que los datos estén en el bus cuanto antes



N_{Ax} : nº puertas desde A1-A23 hasta generar $\overline{DTACK}$
N_{AS} : " " $\overline{AS}$ " " $\overline{DTACK}$
n_{Ax} : " " A1-A23 " " $\overline{CS}$
n_{AS} : " " $\overline{AS}$ " " $\overline{CS}$
t_p : tiempo de propagación máximo (peor caso)

- $\overline{DTACK}$: En principio no deben aparecer estados de espera
  ↳ $\overline{DTACK}$ activo antes de flanco bajada S4

desde AX : $⑥ + N_{Ax} \cdot t_p + ㊼ < 4$ semiciclos

desde $\overline{AS}$: $⑨ + N_{AS} \cdot t_p + ㊼ < 3$ semiciclos

SUELE ser $\overline{AS}$ el más limitante ya que dispone de un semiciclo menos (62'5 ns) y sólo un nivel de puertas menos que N_{Ax} ($t_p \simeq 27$ ns). Además ⑥ ≃ ⑨ PERO a veces cambia

- Datos: Comprobas si, con el DTACK tan rápido, los datos estarán:

desde AX : $⑥ + n_{Ax} \cdot t_p + T_{acceso\ memoria} + ㉗ < 6$ semiciclos

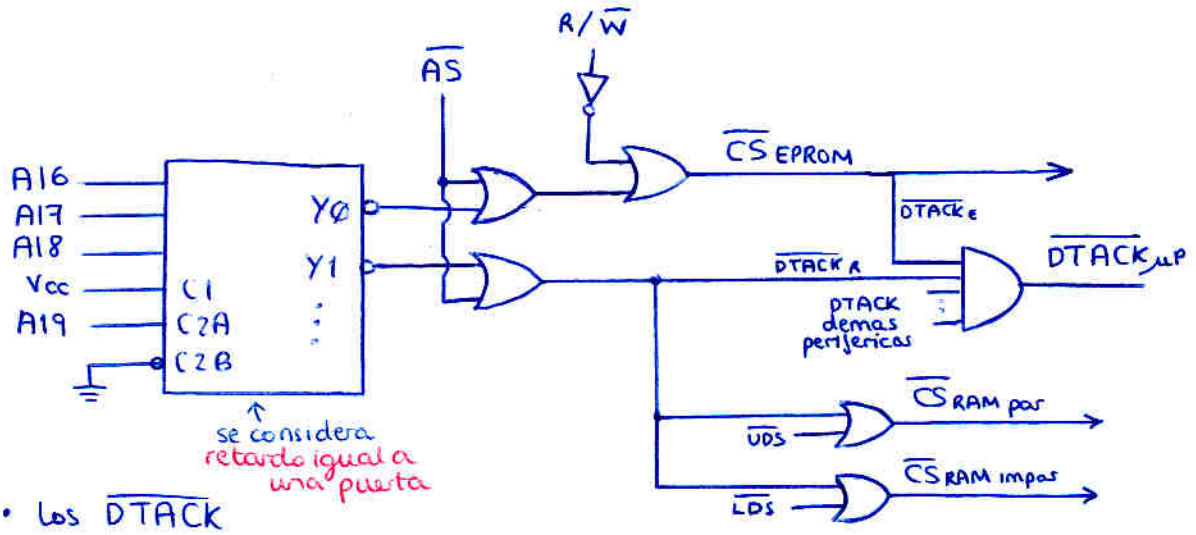
desde $\overline{AS}$: $\underbrace{⑨ + n_{AS} \cdot t_p}_{\text{lo que tarda } \overline{CS}} + T_{acceso\ memoria} + ㉗ < 5$ semiciclos

- Si no se cumple suficiente velocidad en los datos, entonces habrá que retardar DTACK tantas veces (ciclos) como sea necesario

ejemplo: ciclo de escritura con RAM y EPROM

$Tacc_R = 100ns$ $\quad t_{pmax} = 27\ ns$

$Tacc_E = 100ns$



- Los $\overline{DTACK}$

$$\overline{DTACK}_E = A19 + A18 + A17 + A16 + \overline{R/W} + \overline{AS}$$

la EPROM sólo debe activarse para lectura

$$\overline{DTACK}_R = A19 + A18 + A17 + \overline{A16} + \overline{AS}$$

← como ya vimos, aquí no incluimos UDS y LDS ya que generaría estados de espera y además no aporta nada

$$\overline{DTACK}_{\mu P} = \prod_{i=1}^{n} \overline{DTACK}_i$$

- En cuanto a los $\overline{CS}$ (y por tanto lo pronto que estarán datos en bus)

$$\overline{CS}_E = \overline{DTACK}_E$$

$\overline{CS}_{Ram}$

$$\overline{CS}_{R\,par} = A19 + A18 + A17 + \overline{A16} + \overline{AS} + \overline{UDS}$$

$$CS_{R\,impar} = \underbrace{A19 + A18 + A17 + \overline{A16} + \overline{AS}}_{\overline{DTACK}_R} + \overline{LDS}$$

← aquí HAY que incluir $\overline{UDS}$ y $\overline{LDS}$ por si hay escritura. Ponerlos siempre en la ultima puerta.

- Evitar estados de espera i.e. DTACK antes de bajada S4

Desde AS

$$\#9 + N_{AS} \cdot t_{pmax} + \#47 < 3 \cdot \text{semiciclo}$$

$$60ns + 3 \cdot 27ns + 10ns = 151\ ns < 3 \cdot 62'5 \checkmark$$

NOTA:

$NAS_{RAM} = 2$

$NAS_{EPROM} = 3$

tomamos peor caso

Niveles de puertas entre $\overline{AS}$ y $\overline{DTACK}$

$NAX_{RAM} = 3$

$NAX_{EPROM} = 4$ ← como sólo hay 1 nivel más, en general ese t_p es menor que el semiciclo que se gana, y como #6 ≃ #9, no hace falta comprobar AX una vez comprobado AS

AX

desde AX

$$\#6 + N_{AX}\, t_{pmax} + \#47 = 4 \cdot \text{semiciclo} \checkmark$$

- Asegurar los datos:

desde AS $\quad \#9 + n_{AS} \cdot t_{pmax} + t_{acc} + \#27 < 5 \cdot \text{semiciclo} \checkmark$

desde AX $\quad \#6 + n_{AX} \cdot t_{pmax} + t_{acc} + \#27 < 6 \cdot \text{semiciclo} \checkmark$

$n_{AS_R} = 2 \qquad n_{AX_R} = 3$

$n_{AS_E} = 2 \qquad n_{AX_E} = 3$

← esta vez el $\overline{UDS}$ y $\overline{LDS}$ equipara los niveles de la RAM y la EPROM que tenía el $R/\overline{W}$ de más.

Supongamos ahora que Tacc = 400 ns

Desde (AS) 60ns + 2·27ns + 400ns + 10ns = 524 ns $\not<$ 312'5ns
no lo cumple!

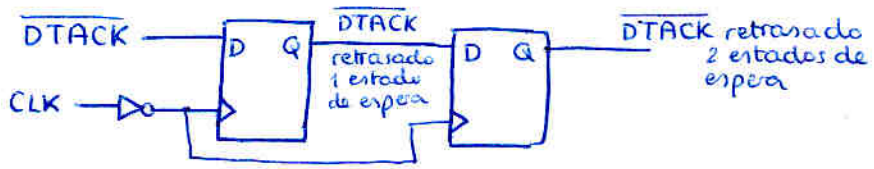
Habrá que insertar estados de espera hasta que se cumpla, sabiendo que cada estado de espera son 2 semiciclos mas.

2S + 5S > 524 ns ? no

2S + 2S + 5S > 524 ns ? si

NOTA: $S = \frac{1}{8MHz \cdot 2} = 62'5ns$
por ser semiciclos

Habrá que introducir 2 estados de espera
¿Cómo?

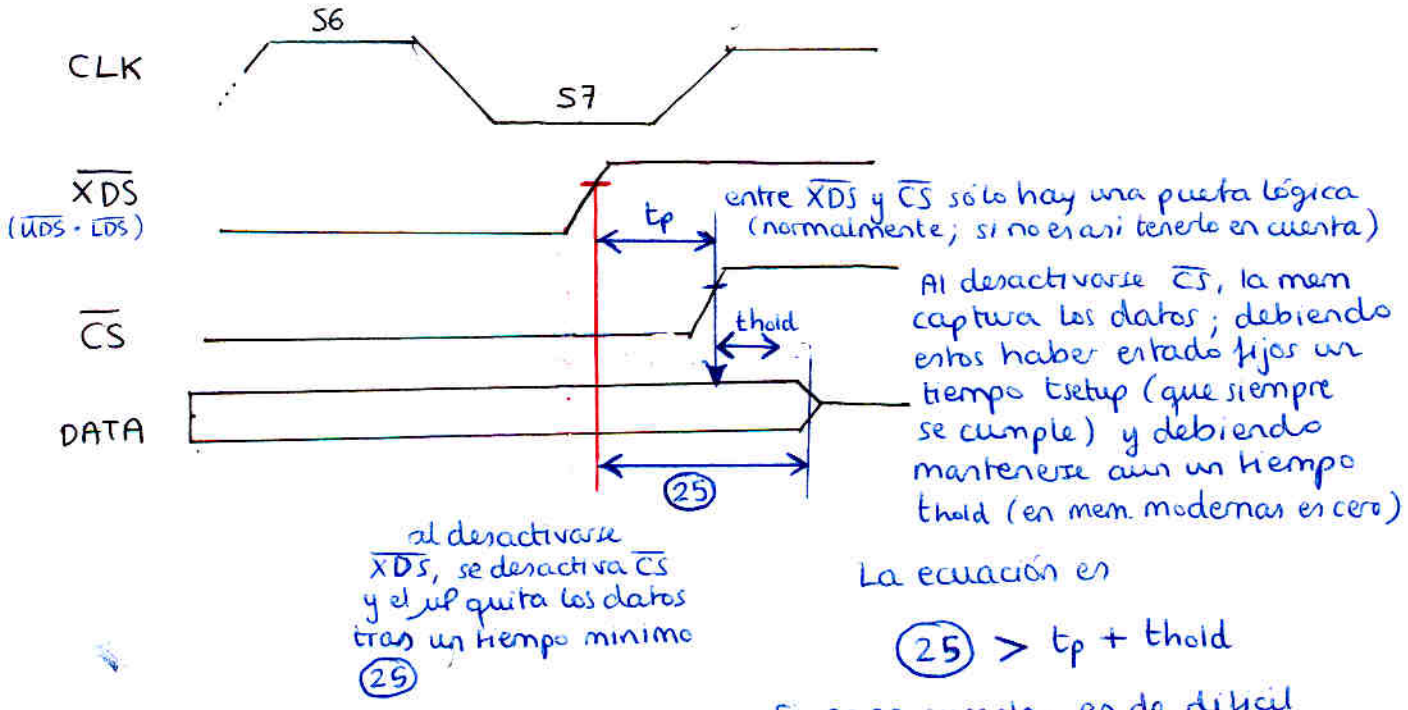


Cuidado: No los coloquemos en el DTACK$_{\mu P}$, porque retardaríamos a TODOS los dispositivos.

Colocar flip-flops en el DTACK del dispositivo en cuestión
i.e. $\overline{DTACK}_{RAM}$ y $\overline{DTACK}_{eprom}$ en el ejemplo, antes de pasar la última AND

Ciclo de escritura

La memoria escribe / captura un dato cuando se desactiva $\overline{CS}$.
Puesto que $\overline{CS}$ depende de forma inmediata de $\overline{UDS}$ o $\overline{LDS}$ ($\overline{XDS}$) (ya que estan en la ultima puerta de CS) es sencillo ver si se cumplen los tiempos



La ecuación es

$$25 > t_p + t_{hold}$$

Si no se cumple, es de difícil solución

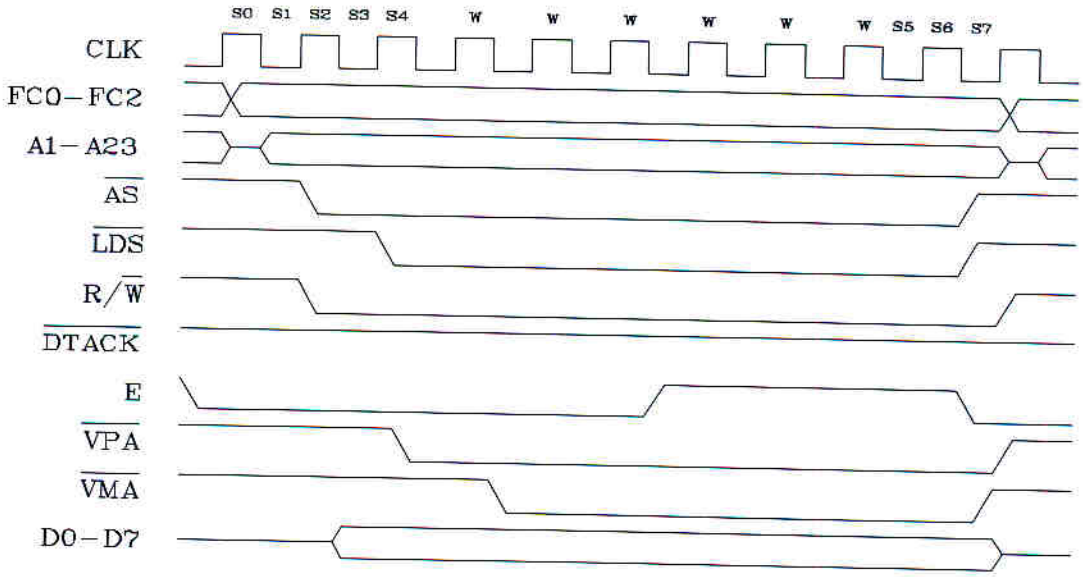
Conexión Interfaces 6800 (VPA)

Si el sistema, en lugar de con $\overline{DTACK}$ responde con $\overline{VPA}$, significa que el dispositivo funciona con un 'ciclo largo'.
Funciona con el reloj E (Ho llamado Ø) que va a $f_{\mu P}/10$ y $\delta = 40\%$

Una vez el dispositivo ha enviado $\overline{VPA}$, necesita que el µP le envie la señal $\overline{VMA}$ con un cierto desfase de E (concretamente 2 ciclos antes de que E suba) para poder funcionar con sincronismo

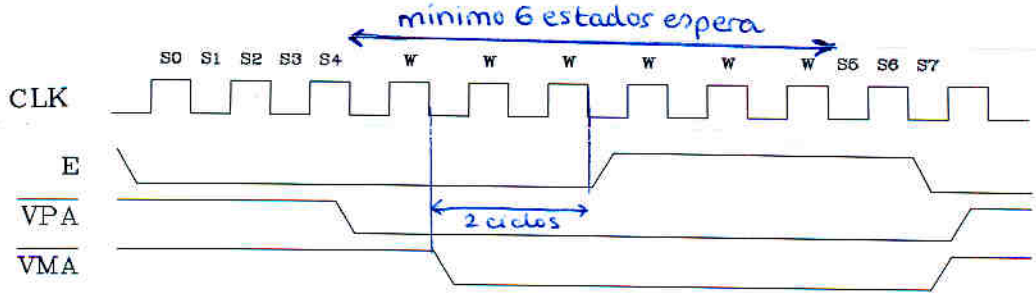
Es decir: Al detectar $\overline{VPA}$, el µP activa $\overline{VMA}$ 2 ciclos antes de que suba E, (según el estado de E cuando ha llegado $\overline{VPA}$, la espera de $\overline{VMA}$ puede ser mayor o menor)

Caso Mejor:



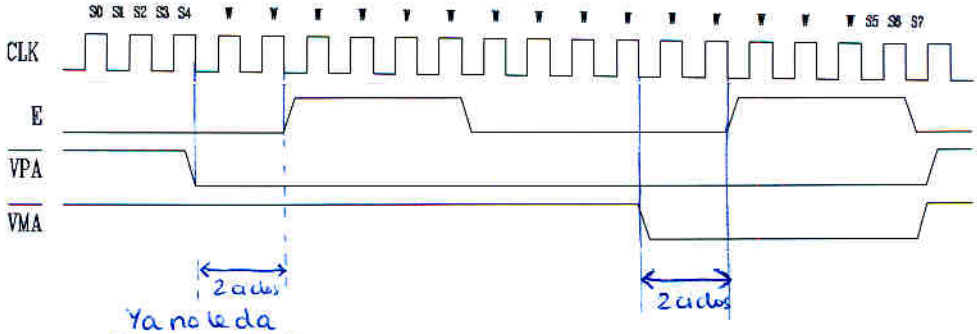
La parte importante

Caso mejor



$\overline{VPA}$ llega justo para que $\overline{VMA}$ se ponga a cero en el proximo flanco de bajada 2 ciclos antes que E

Caso peor:



Ya no le da tiempo a VMA activarse y tiene que estarse esperando todo el ciclo de E

Ciclo de reconocimiento de interrupciones autovectorizadas

Cuando en un ciclo de reconocimiento de interrupciones el sistema devuelve $\overline{VPA}$, el µP hace un ciclo largo de lectura donde no se lee ningun dato del bus de datos ¡y el µP se genera su autovector.

TEMA 4. Procesos de excepción del MC68000

Estados del 68000

- Normal: ejecución secuencial de instrucciones
  Caso especial: Parada Reversible : STOP
- Excepción: Estado inusual. Por programación, por error o por situaciones de tiempo real (interrupción exterior)
  ej: interrupción
  ej: instruccion TRAP
- Parada Irreversible: Fallo hardware catastrófico, el µP no se puede recuperar
  ej: error de bus al inicializarse
  ej: error de bus doble (error de bus estando atendiendo un error de bus)
  Sólo se le saca de esta situación mediante reset externo.

Modos de privilegio

Según el bit S del SR. En el arranque $2700 → SR

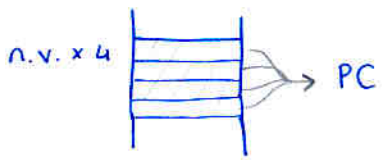
T = 0
S = 1
$I_2 I_1 I_0 = 7$
CCR = ∅ (X, Z, N, C, V)

- Supervisor: S = 1
  Instrucciones privilegiadas:
  STOP, RESET, cambiar A7 (SSP), cambiar SR, ...
  no a sí mismo, sino al sistema
  y tambien USP (MOVE.L A0, USP)
  Permite pasar a estado usuario (cambiar SR)
- Usuario: S = 0
  - Juego de instrucciones restringido
  - Algunos registros restringidos

  La unica forma para volver a supervisor es mediante un proceso de excepción.
  ejemplo: TRAP #5 ; salta directamente al sistema operativo (esperando comandos por el puerto)
  devuelve el control al SO, que lo perdió tras recibir 'g' (ejecución)

Procesos de excepción. Vectores de excepción

- Bit supervisor = 1
  Traza = 0
- obtención del vector de excepción → metodos internos o externos (número de 8 bits)
- µP guarda el contexto actual
- Nuevo contexto: (nº de vector · 4) (ya que son 'Long')



carga en el PC el contenido de la posición nºvector. 4

i.e. JMP (n°v · 4)
contenido de la dirección

Procesos de excepción

Externos: Error de Bus, RESET (aunq este ultimo un poco especial)

Internos: Instruccion (ej: div por cero, TRAP, violación de privilegio, ...
error de dirección (mover WORD a impar)

Prioridad:

Cual se atiende si se producen varias simultaneamente

Grupo 0: Reset, Error de dirección, error de bus
El proceso de excepción comienza 2 ciclos despues

Grupo 1: Traza, Interrupción, Ilegal, Privilegiada
Comienza al acabarse la ejecución de la instrucción actual

Grupo 2: Trap, división por cero
Se inicia con la ejecución normal de la instrucción

Tabla de vectores

Todos en datos de supervisor
EXCEPTO los 2 primeros (Reset - SSP, Reset - PC)

Número de vector	Dirección	Espacio	Asignación
0	$000	P.S.	Reset, SSP inicial
1	$004	P.S.	Reset, PC inicial
2	$008	D.S.	Error de bus
3	$00C	D.S.	Error de dirección
4	$010	D.S.	Instrucción ilegal
5	$014	D.S.	División por cero
6	$018	D.S.	Instrucción CHK
7	$01C	D.S.	Instrucción TRAPV
8	$020	D.S.	Violación de privilegio
9	$024	D.S.	Traza
10	$028	D.S.	Emulación 1010
11	$02C	D.S.	Emulación 1111
12	$030	D.S.	No asignado
13	$034	D.S.	No asignado
14	$038	D.S.	No asignado
15	$03C	D.S.	Vector de interrupción no inicializado
16 - 23	$040-$05C	D.S.	No asignados
24	$060	D.S.	Interrupción espúrea
25	$064	D.S.	Autovector de interrupciones nivel 1
26	$068	D.S.	Autovector de interrupciones nivel 2
27	$06C	D.S.	Autovector de interrupciones nivel 3
28	$070	D.S.	Autovector de interrupciones nivel 4
29	$074	D.S.	Autovector de interrupciones nivel 5
30	$078	D.S.	Autovector de interrupciones nivel 6
31	$07C	D.S.	Autovector de interrupciones nivel 7
32 - 47	$080-$0BC	D.S.	Vectores de la instrucción TRAP#dato
48 - 63	$0C0-$0FF	D.S.	No asignados
64 - 255	$100-$3FC	D.S.	Vectores de interrupción de usuario

→ comprobar a lo largo de un programa que las operaciones matemáticas se hacen correctamente.

Permiten generarnos instrucciones personalizadas
← empieza por $A
← empieza por $F

ej: ORG $25000
⋮
DC.W $ A038 → parametros
↑ instruccion
nº vector 10

en la rutina de atención hacemos lo que queramos interpretando los parametros que queramos

Pila de excepción

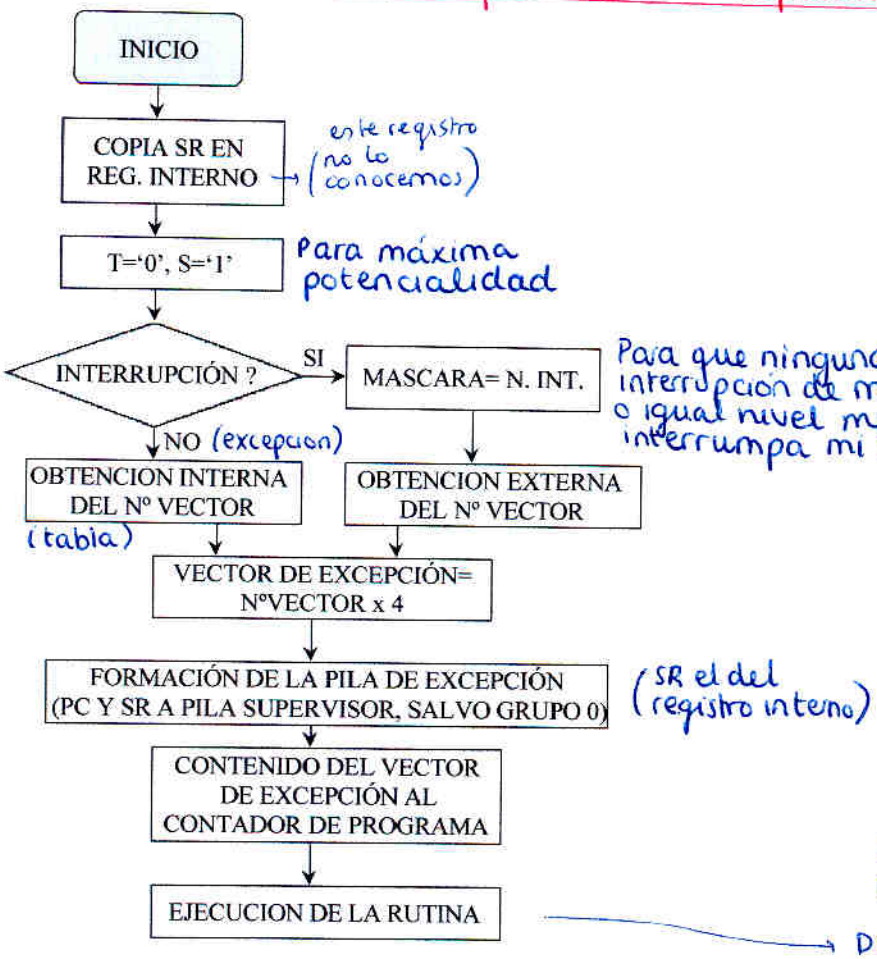
El proceso de excepción guarda lo que sea necesario del contexto actual en la pila SSP

ej: RESET no guarda nada

SSP → | SR | PC (de la sig instruc) |

- Los demás guardan SR y PC
- El error de bus guarda ademas la dirección errónea e instrucción causante (para informar al usuario) (ocurre en excepciones grupo 0)

Secuencia de un proceso de excepción genérico



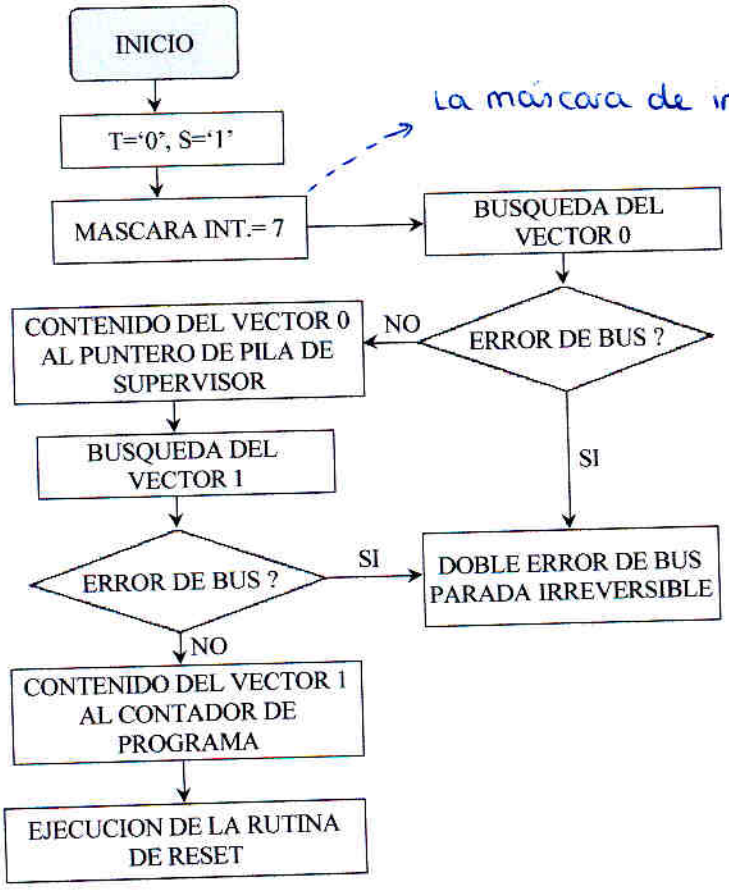
Durante este proceso previo a atender a la interrupción, no se pueden aceptar otras interrupciones.

Es un proceso indivisible salvo int. grupo 0 i.e. RESET, error bus

No se hace caso a ninguna otra excepción / interrupción

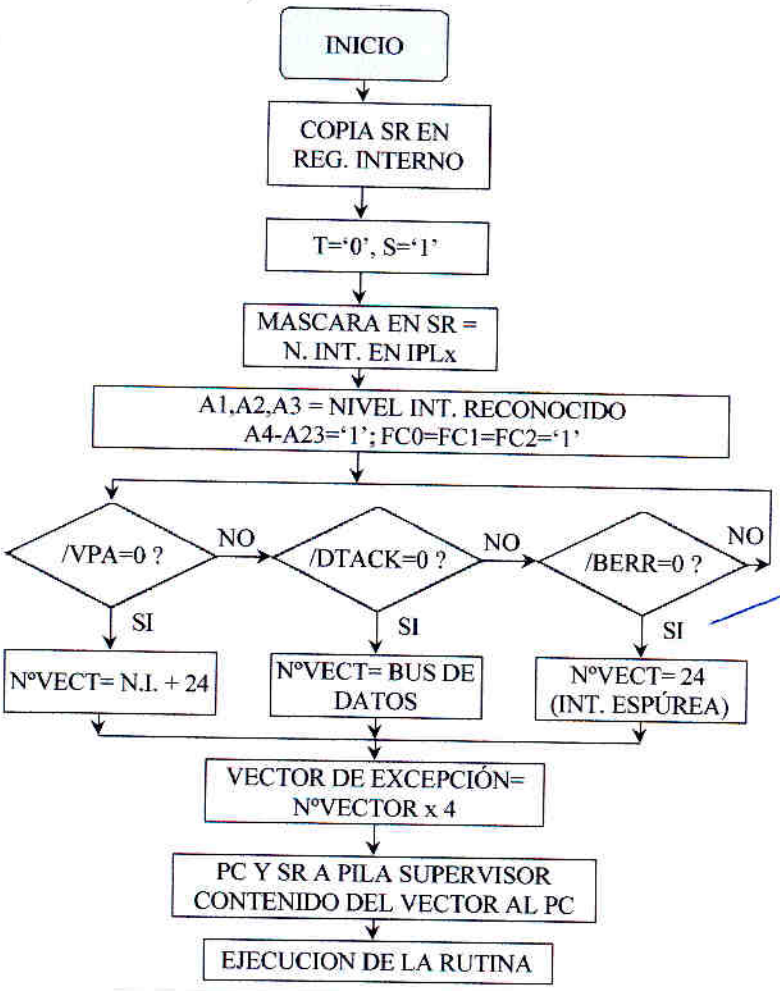
Durante la rutina se atenderán interrupciones que superen la máscara i.e. nivel mayor que el actual.

Secuencia de Reset



La máscara de interrupción esta a 7, por tanto, durante la ejecución de la rutina de reset sólo se atienden interrupciones nivel 7 las cuales suelen ser fallos de alimentación únicamente (a menos que el diseñador esté un poco "loco")

Interrupciones externas



Notar que un BERR durante un reconocimiento de interrupciones no es un fallo serio; es sólo que el dispositivo ya no quiere interrumpir y ya no contesta (interrupción espúrea)

DTACK : nº vector entre 64 y 255
VPA : nº vector entre 25 y 31

Excepciones de TRAP (causadas por instrucción)

Cualquier instrucción que provoque un estado de excepción

- DIVS y DIVU : si se divide por cero
- TRAP #n : instrucción con ese mismo propósito
- TRAPV (control aritmético) y CHK (subindice fuera de límite)

Instrucción ilegal

Instrucciones reservadas por Motorola o para usuario. → da error

Violación de privilegio

Se intenta ejecutar una instrucción privilegiada en modo usuario.

(las que modifican SR
RESET
STOP)

Instrucciones no implementadas

las palabras que empiezan por $A o $F fuerzan excepciones. Sirven para emulación de instrucciones por parte del usuario (similar a TRAP)

Modo Traza

- Bit T del SR a 1
- Provoca una excepción despues de la ejecución de cada instrucción. Recuerda que en la propia excepción se pone T=0, (ya que sería absurdo interrumpir las instrucciones de la propia excepción). Al acabar la excepción el RTE volverá a cargar el SR original con T=1

Depuración de programas:

Podemos activar modo traza a mitad de una ejecución

```
MOVE.W  $A000, SR
```

y desactivarlo

```
MOVE.W  $2000, SR
```

Cuidado: No se debe depurar de esta forma un programa que implique tiempo real

Error de direcciones y error de bus

Error direcciones: acceso .W o .L a dirección impar (ya sea datos o fetch)

Error bus: Un $\overline{BERR}$ (sin HALT asociado; ya que entonces sería un ciclo de relanzamiento, y sin que sea en un ciclo de reconocimiento de interrupciones; ya que entonces sería una interrupcion espurea)

Casi SIEMPRE ocurren por un fallo de programación.

Son IRRECUPERABLES; la única razón de la excepción es poder informar del error, por eso recoge datos:

SSP 14	bit 4: R/$\overline{W}$ | bit 3: I/N | bits 2-0 → codigos de funcion
SSP 12	Direccion a la que se accedía
SSP 8	1ª palabra instrucción en proceso
SSP 6	Registro de estado
SSP 4	Contador de programa (PC)

I/N → No instrucción

En un sistema autónomo no se concibe que ocurran, a no ser que sea por ruido, En ese caso el circuito generador de $\overline{BERR}$ tambien genera $\overline{HALT}$ de forma que tiene lugar un ciclo de relanzamiento.

FAMILIA DE DISPOSITIVOS DE ENTRADA SALIDA DEL MC68000. LA V.I.A Y LA D.U.A.R.T

TEMA 6

Índice

1.- Introducción.
2.- Pastillas de propósito general conectables al bus síncrono del 68000. Estudio detallado de la "V.I.A." 6522.
- 2.1.-Interfase con el microprocesador.
- 2.2.-Interfase con el periférico.
- 2.3.- Funcionamiento de los ports.
  - 2.3.1.- Operaciones simples de E/S.
  - 2.3.2.- Señales de reconocimiento de condición.
  - 2.3.3.- Producción de señales de salida.
- 2.4.- Control de interrupciones.
- 2.5.- Temporizadores. Modos de funcionamiento.

3. Estudio particular de la DUART del 68000. MC68681
- 3.1.- Introducción.
- 3.2.- Descripción del patillaje. Conexión al 68000.
- 3.3.- Diagrama de bloques.
- 3.4.- Modos de funcionamiento.
  - 3.4.1.- Funcionamiento modo emisor.
  - 3.4.2.- Funcionamiento modo receptor.
  - 3.4.3.- Funcionamiento modo multiconexión.

4.- Conexión de otros periféricos al 68000.
- 4.1.- Conexión de un convertidor A/D concreto.
- 4.2.- Conexión de un convertidor D/A concreto.

LA VIA

IFR : Registro de Banderas de Interrupción
asociado a IER : Interruption Enable Register - determina qué sucesos generan interrupción

ACR: Registro Auxiliar → utilizamos sus bits para lo que queramos

T_1, T_2 : T_1 → temporizador contador
T_2 - temporizador } → se cargan de byte en byte

Puertos:

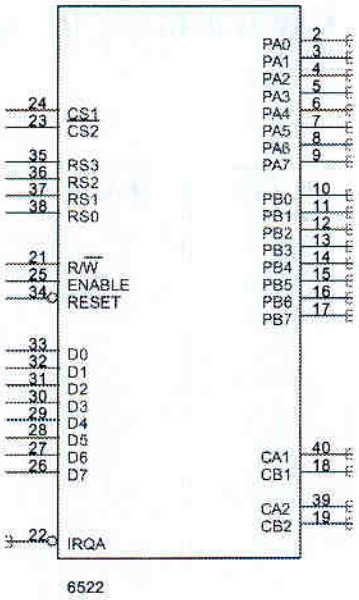
- DDRA / DDRB : configura cada pin/bit de los puertos como entrada o salida
- Puerto A y Puerto B :
  Ligeramente mas corriente

CA1 / CA2 : permiten el handshake

La V.I.A. 6522. E/S paralelo de propósito general

4 líneas de dirección para los 16 registros internos (conectaremos A1, A2, A3, A4)

LDS y LDS en chip select



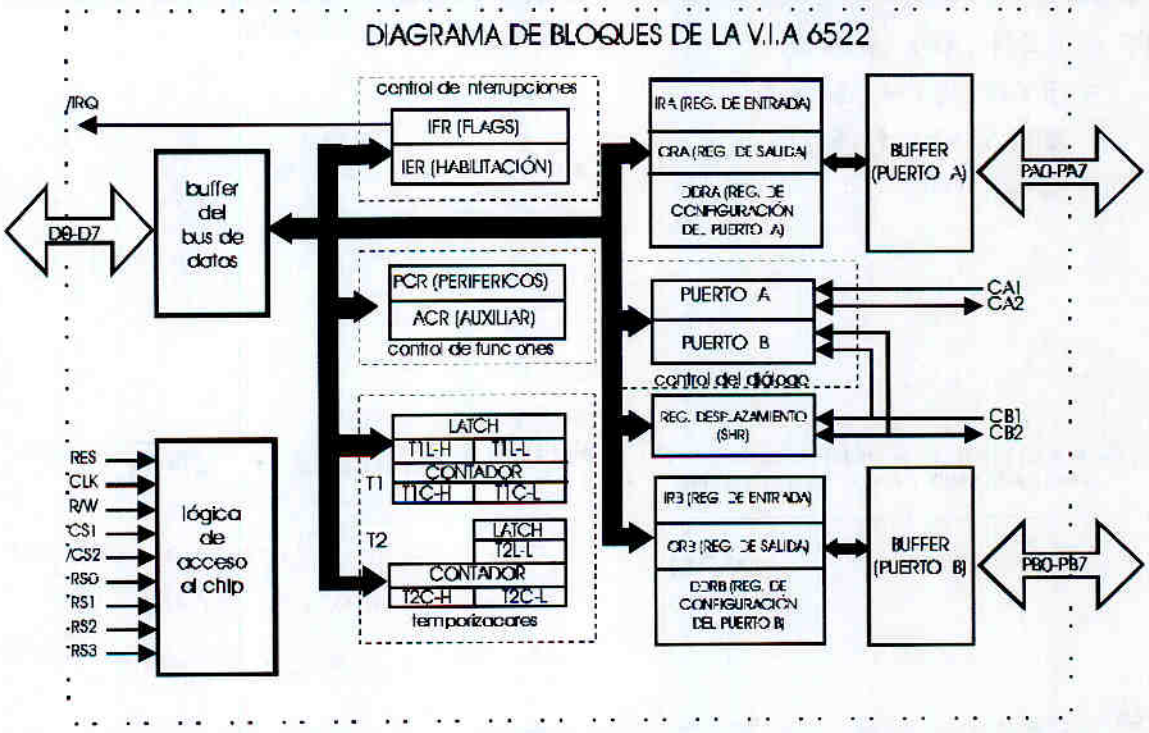
- Es un dispositivo de adaptación de periféricos de propósito general.
- Es compatible con la familia 6800, (ciclo síncrono).
- Posee:
  - 2 Puertos bidireccionales. configurables pin a pin utilizando registros DDR
  - 1 Registro de desplazamiento.
  - 2 Temporizadores contadores.
  - 1 Registro de interrupción. se pone a 1 el bit de la razón de la interrupción en el registro de interrupción

No tiene registro de nº de vector.

No se activará CS hasta que el ciclo este sincronizado: i.e. VMA activo
i.e. VMA debe estar en la ecuación del CS

- No hace falta estar en posición impar (pero o uno o otro, no una mezcla)
- En un dispositivo con nº de vector, el D0-D7 debe estar conectado al D0-D7 del µP
  i.e. DEBE estar en posiciones impares

Diagrama de bloques de la V.I.A



Conexión para la interrupción

Recordemos que VIA no tiene nº de vector; en un ciclo de reconocimiento de interrupción debe devolver $\overline{VPA}$ cuando:

- ciclo reconocimiento: FCX a 1
- Nivel en A3 A2 A1 igual al nivel de interrupción que le demos a la VIA
- Que esté validado $\overline{AS}$
- Que la VIA haya interrumpido! IRQ_{VIA}

$$\overline{VPA}_{INT} = \overline{FC0} + \overline{FC1} + \overline{FC2} + \overline{A3} + A2 + A1 + \overline{AS} + \overline{IRQ_V}$$

Conexión para el mapa de memoria

Conectado como si fuera el único dispositivo conectado →

En la posición 60021

$$PF = PI + (\overset{\text{capacidad memoria}}{CM} - 1) * 2$$
$$= \$60021 + \$F*2$$
$$= \$60021 + \$1E$$
$$= \$6003F$$

la VIA es como una memoria de 16 registros de 8 bits

6	0	0	2	1
0101	0000	0000	0010	0001
0101	0000	0000	0011	1111
6	0	0	3	F

→ 1 → impares

A18, A16 y A5 a 1

$$\overline{CS}_{VIA} = \underbrace{A23+A22+A21+A20+A19+\overline{A18}+A17+\overline{A16}+A15+A14+A13+A12+A10+A9+A8+A7+A6+\overline{A5}}_{\overline{VPA}_{VIA}} + \overline{AS} + \overline{LDS} + \overline{VMA}$$

CUIDADO: ¡¡Nunca hacer que $\overline{VPA}$ dependa de $\overline{VMA}$!!

de hecho siempre se cumple $\overline{VPA} = \overline{CS}$ menos $\overline{VPA}$

Luego será $\overline{VPA}_{\mu P} = \overline{VPA}_{VIA} \cdot \overline{VPA}_{INT} \cdot \overline{VPA}_{resto}$

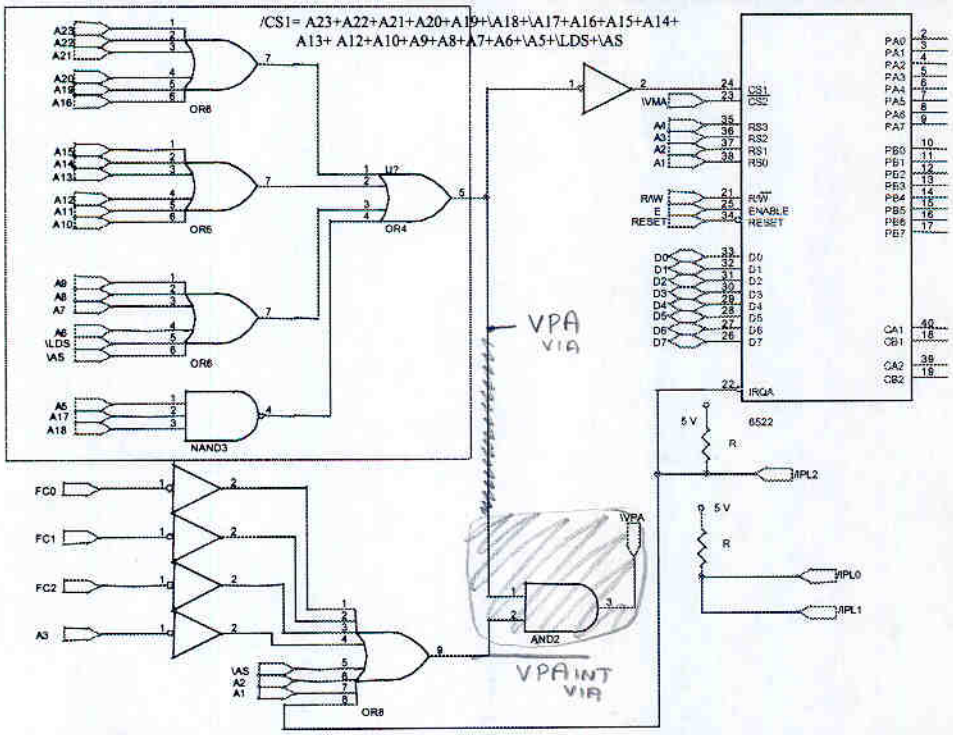
Interfase con el uP de la V.I.A.

- 2 Terminales de selección de chip CS1 y /CS2
- 4 Terminales para la selección de registro: RS0-RS3.
- Transferencia de datos cuando E esté a nivel alto. Base de tiempos para temporizadores y registro de desplazamiento.
- Bus de datos de 8 bits.
- Línea de R/\W y RESET.
- Línea de petición de interrupción.

E conectado a ENABLE

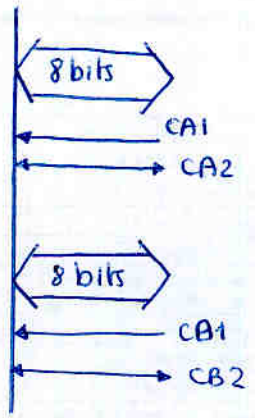
RS3-RS2-RS1-RS0	REGISTRO	DESCRIPCION
0000	PORTB	Puerto de entrada salida B
0001	PORTA	Puerto de entrada salida A
0010	DDRB	Configuración puerto B
0011	DDRA	Configuración puerto A
0100	T1L_L T1C_L	Parte baja del latch timer 1 (escritura) Parte baja contador timer 1 (lectura)
0101	T1C_H	Parte alta contador timer 1
0110	T1L_L	Parte baja del latch timer 1
0111	T1L_H	Parte alta del latch timer 1
1000	T2L_L T2C_L	Parte baja del latch timer 2 (escritura) Parte baja contador timer 2 (lectura)
1001	T2C_H	Parte alta contador timer 2
1010	SHR	Registro de desplazamiento
1011	ACR	Registro de control auxiliar
1100	PCR	Registro de control de periférico
1101	IFR	Registro de los flags de interrupción
1110	IER	Registro de la máscara de interrupción
1111	ORA	Puerto A sin protocolo

Conexión VIA al 68000 en la posición $60021. Nivel de interrupción 4.



mejor las VPA por separado ya que luego tendrán que unirse a otras VPA's de otros dispositivos

Control de los puertos



Modo transparente: los 8 bits van y vuelven según se van poniendo

Modo latch: capturo cuando veo flanco CA1
La VIA indica por CA2 cuando el dispositivo ya puede seguir poniendo otros 8 bits

Interfase con el Periférico. Funcionamiento de los Ports.

- Los puertos A y B, son dos puertos de 8 líneas cada uno que se pueden programar individualmente como E o S según un registro de configuración DDRX.
- Los datos de entrada se almacenan en un registro interno controlado por la línea CX1.
- El puerto B puede suministrar más corriente de salida. PB7 puede ser controlada por uno de los temporizadores y PB6 puede actuar como entrada para el conteo de impulsos por el otro temporizador. (Nos permite contar N flancos)

BIT	7	6	5	4	3	2	1	0
DDRA	0	0	1	1	0	0	0	1
PORTA	E	E	S	S	E	E	E	S

Control de los puertos. PCR.

REGISTRO PCR

7	6	5	4	3	2	1	0
Control de CB2			Control CB1	Control de CA2			Control CA1

- El registro de configuración de periféricos sirve para programar las líneas de control de periféricos CX1 y CX2.
- CX1 es una entrada de interrupción con flanco programable.
- CX2 puede ser entrada programable o salida de protocolo.

CONTROL DE CX2

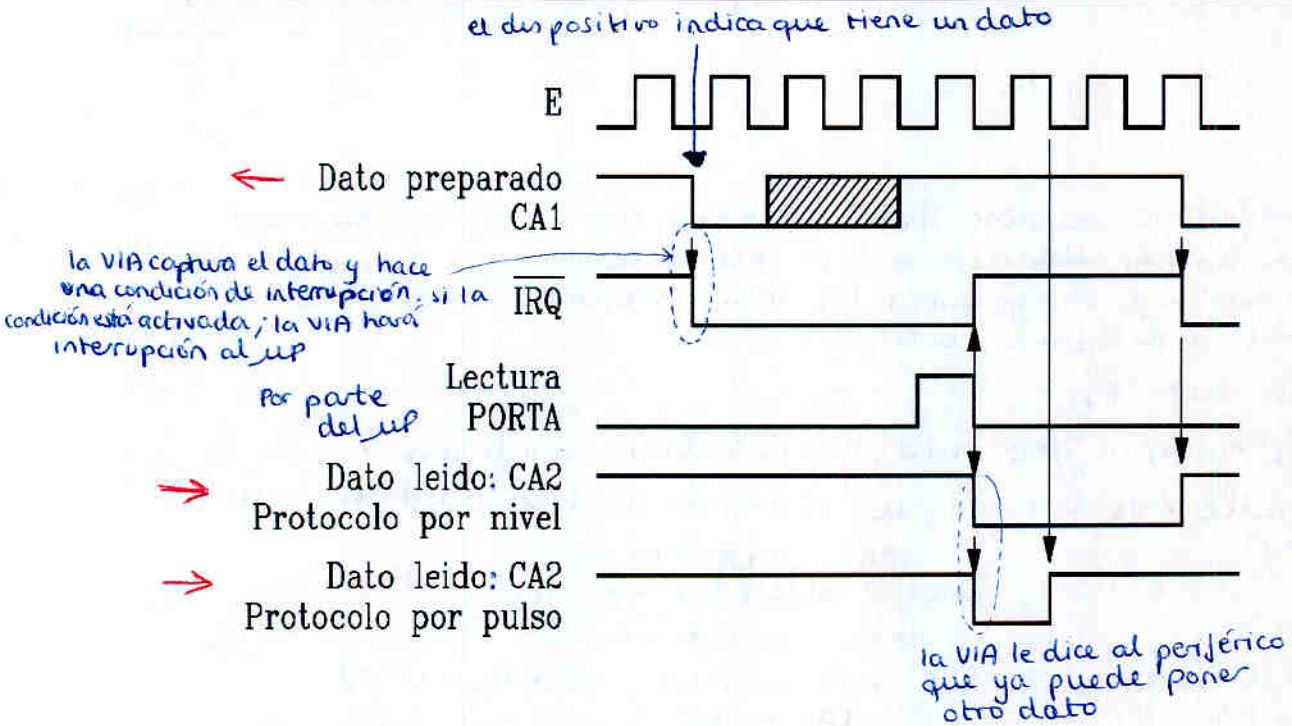
3	2	1	Control de CA2
0			CA2 es entrada de interrupciones
	X	0	Se repone IFR.0 al leer o escribir en puerto A
		1	No se repone IFR.0
	0	X	CA2 sensible al flanco de bajada
	1		CA2 sensible al flanco de subida
			CA2 es salida de protocolo o salida manual
1	0	0	Modo salida de protocolo por nivel
		1	Modo salida de protocolo por pulso
	1	0	Modo salida manual CA2 a 0
		1	Modo salida manual CA2 a 1

cuando el µP lee el dato, CA2 manda flanco automaticamente

lo manejamos manualmente poniendolo a 0 o a 1

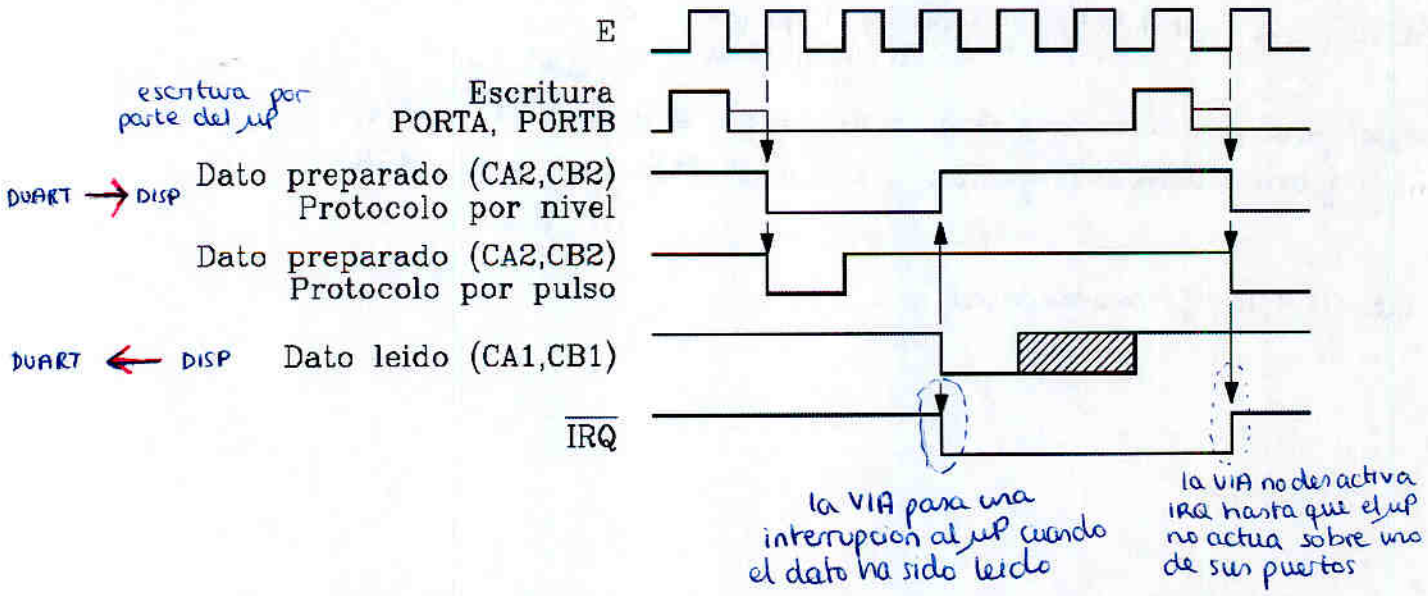
ej. el dispositivo no entiende flanco, sino una determinada secuencia

Control con protocolo en lectura (sólo lo hace el puerto A)



Ccontrol con protocolo en escritura. (lo hacen ambos)

La iniciativa es del µP



IFR

- Algunas condiciones se desactivan solas cuando es lógico que ocurra
  ej: el µP ha escrito un nuevo dato tras haber recibido la interrupción de dato recibido (la vía la recibe por CA2 y activa la condicion), al escribir el nuevo dato la condicion se desactiva (es logico)
- El µP puede leer IFR
- El µP puede modificar IFR SÓLO para desactivar condiciones
  Curiosamente ésto lo consigue escribiendo un 1 en el bit que quiera modificar!

ej

```
IFR      01011100
MOVE.B   #%01000000, IFR
IFR      00011100
```

(Cuidado si haces un bit set → te pone todos los unos a cero)
(piensalo)

IER

En el bit correspondiente a una cierta condición en el IFR
si está a '1' : la condición activa IRQ
Si está a '0' : la condicion afecta a IRQ

- Se maneja desde el µP de una forma extraña:

<u>SET</u> si bit 7 = '1' Todos los '1' pasan a '1'
Los '0' no afectan

<u>RESET</u> si bit 7 = '0' Todos los '1' pasan a '0'
Los '0' no afectan

ejemplo: para ponerlo todo a cero : MOVE.B #%0111 1111, IER (RESET, todos)
y ahora activar algunas : MOVE.B #%1001 0100, IER (SET, estos 2)

- Se lee IER de forma normal

Control de interrupciones

- Consta de dos registros:
  - **IFR:** Registro de flags de interrupción, que refleja la activación de las 7 posibles fuentes de interrupción de la V.I.A. No es posible mediante escritura en este registro poner a '1' los bits, pero si borrarlos. Esto se logra escribiendo un '1' en el bit correspondiente.

7	6	5	4	3	2	1	0
IRQ	T1	T2	CB1	CB2	SHR	CA1	CA2

línea IRQ en sí — temporiz — flanco activo en CB · CA

estos bits se activan cuando ocurre una condición determinada
- se activan solas
- El µP no puede activarlas pero SI desactivarlas

  - **IER:** Registro de validación de interrupción. Para validar una interrupción se escribe un '1' en el bit de control y un '1' en bit de la fuente de interrupción correspondiente. Para invalidar una interrupción se escribe un '0' en el bit de control y un '1' en bit de la fuente de interrupción correspondiente.

7	6	5	4	3	2	1	0
CTRL	T1	T2	CB1	CB2	SHR	CA1	CA2

estos bits indican cuales de las condiciones deben activar a IRQ

IRQ = IER6*IFR6 + IER5*IFR5 + IER4*IFR4 + IER3*IFR3 + IER2*IFR2 + IER1*IFR1 + IER0*IFR0

Temporizadores de la V.I.A

- La V.I.A dispone de 2 temporizadores programables que pueden:
  - Generar un único intervalo de tiempo (monoestable)
  - Contar impulsos en el pin PB6. (contador, sólo T2)
  - Generar contínuamente intervalos de tiempo. (aestable, sólo T1)
  - Generar una única o una serie de impulsos por PB7 (sólo T1)
- El modo de funcionamiento de los temporizadores se programa en el registro de control auxiliar ACR.

ACR

7	6	5	4	3	2	1	0
Control de T1		Control T2	Control de SHR			Control latch ports A y B	

reg. desplaz q no vamos a ver

Temporizadores

Temporizador 2

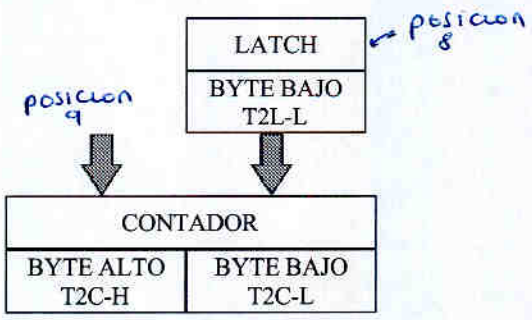
se controla por direcciones 8 y 9

Para cargarle el número:

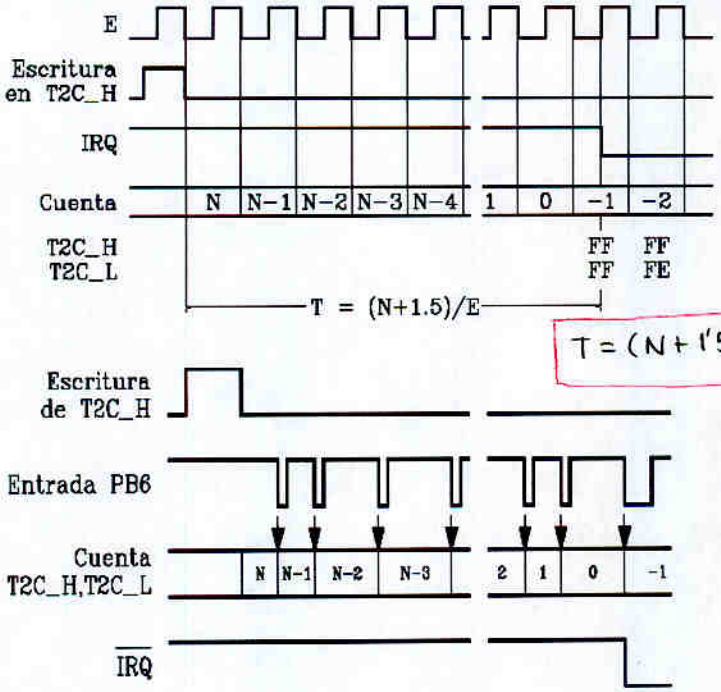
- Solo podemos cargar de 8 en 8 bits asi que el temporizador necesita un LATCH para retener la parte baja hasta que le llegue la parte alta, (que no empiece la cuenta antes de que llegue la parte alta!)
- cuando llega la parte alta ya la copia a la vez que copia el LATCH para que empieze la cuenta

Temporizador 2 de la V.I.A

- Se accede a los registros del temporizador 2 a través de las direcciones 8 y 9 en RSx, 8 (T2L-L), 9 (T2C-H).
- Tiene dos modos de funcionamiento programables en el bit 5 del ACR:
  - ACR(5)='0' Funcionamiento como monoestable (cuenta con flancos del reloj E)
  - ACR(5)='1' Funcionamiento como contador de impulsos por PB6.



Cronogramas del funcionamiento del T2 de la V.I.A



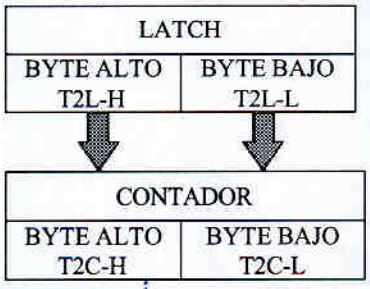
FUNCIONAMIENTO COMO MONOESTABLE

$$T = (N+1'5)\cdot Te = \frac{(N+1'5)}{E} = \frac{(N+1'5)}{1\mu s/10}$$

FUNCIONAMIENTO COMO CONTADOR DE IMPULSOS

Temporizador 1 de la V.I.A

- El T1 es más complejo ya que tiene 4 modos de funcionamiento programables por los bits 7 y 6 del ACR:
  - ACR(7)='1' habilita la salida por PB7
  - ACR(7)='0' inhabilita la salida por PB7
  - ACR(6)='0' Genera un sólo intervalo (monoestable, desactiva la recarga)
  - ACR(6)='1' Genera una serie contínua de impulsos (aestable, recarga la cuenta)
- Se accede a los registros del temporizador 1 a través de las direcciones 4,5,6 y 7 en RSx, 4 (T1C-L), 5 (T1C-H), 6(T1L-L) y 7(T1L-H). Esto nos permite escribir en el latch sin afectar la cuenta.

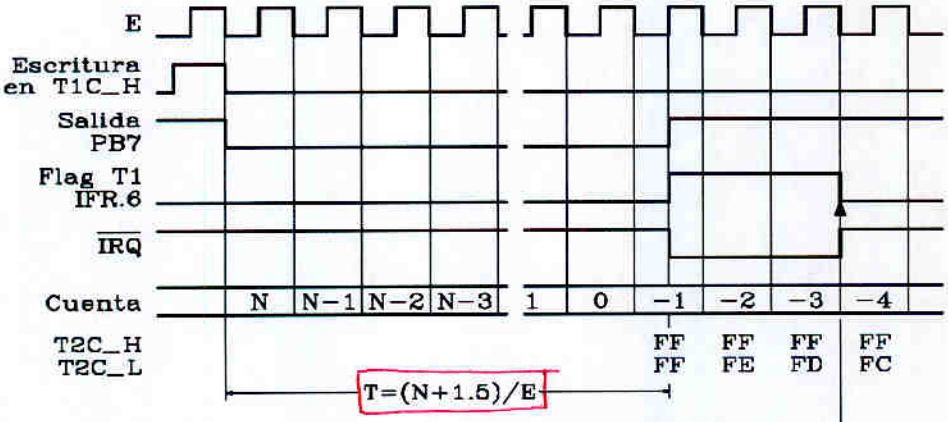


cuando el contador llega a -1 carga el Latch (si esta en aestable)

Al escribir en contador, escribimos tambien en LATCH y se inicia la cuenta (abortando la que hubiera)

contador arranca al escribir en T2C_H

Cronogramas de funcionamiento del T1 de la V.I.A

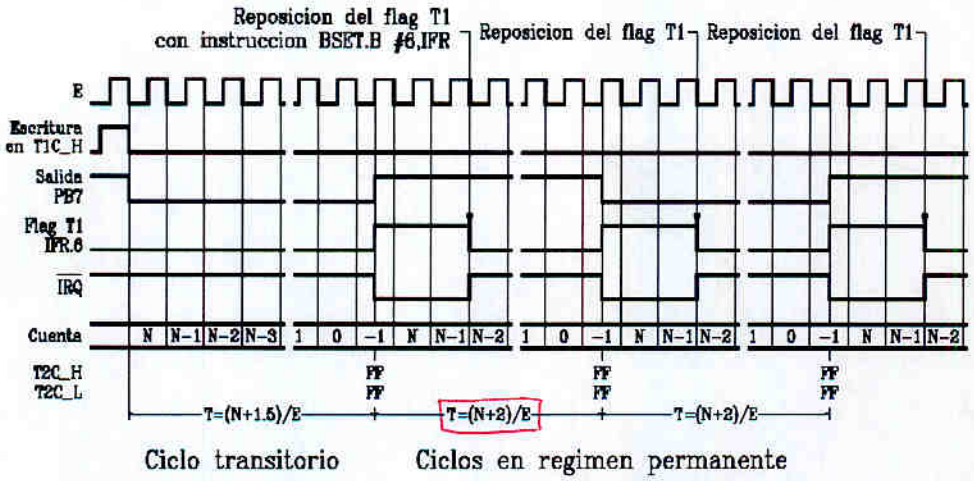


Reposicion del flag T1 con instruccion BSET.B #6,IFR → si es la unica condición

MOVE.B #$40, IFR → si hay mas condiciones para no alterar las demas

FUNCIONAMIENTO COMO MONOESTABLE

Cronogramas de funcionamiento del T1 de la V.I.A (II)



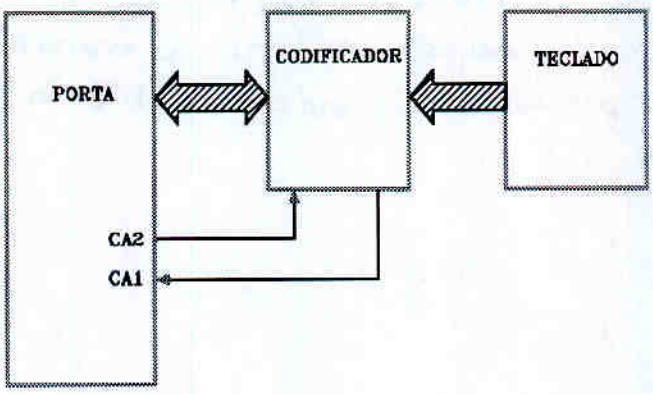
la salida PB.7 hace toggle cada vez que se recarga T1
↓
me genera un reloj divisor del E
$= \frac{E}{\underbrace{2(N+2)}_{\text{un semiciclo}}}$

Frecuencia de salida en PB7 en regimen permanente = E/(2*(N+2))

FUNCIONAMIENTO COMO AESTABLE

Conexión de teclados

- Podemos establecer la conexión de un teclado directamente a la V.I.A, encargándose el uP de realizar la excitación e identificación de la tecla pulsada, o por el contrario incluir un circuito codificador que nos ahorrará tiempo de CPU.



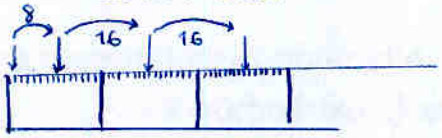
CONEXIÓN DE TECLADO A LA V.I.A MEDIANTE HARDWARE ADICIÓNAL

Transmisión serie:

¿como puede el receptor muestrear correctamente una ristra de bits?

TRUCO: el receptor muestrea más rápido (normalmente 16 veces) que los baudios del emisor

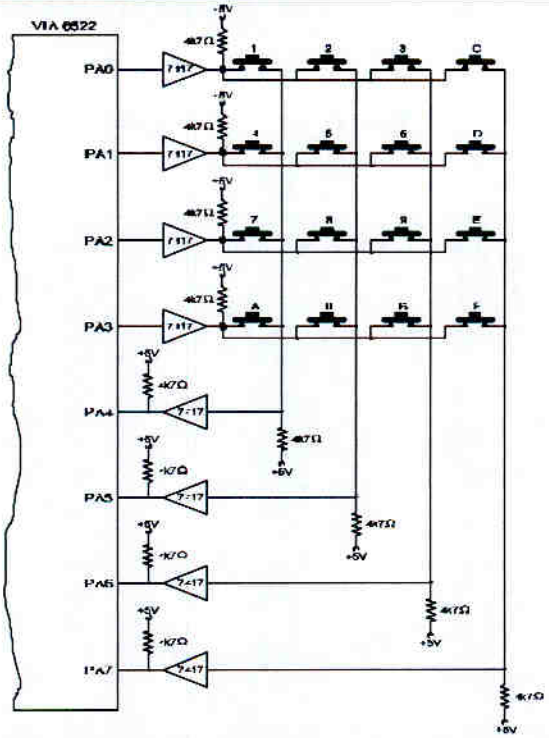
Es decir, cuando se detecta bit de inicio, te esperas 8 muestras para estar a mitad de bit y luego vas muestreando cada 16, cayendo siempre en el centro del bit



Problema: si la frecuencia se desvia un poco (. ej: desviarse 1 ~ 2 %) es posible que en cada bit se vaya yendo un poco y en el ultimo se equivo → hay que usar temporización muy rigida

→ no hay que superar un número máximo de caracteres para que el error que se va sumando bit a bit no acabe por salirse

Conexión de teclados. Teclado matricial.

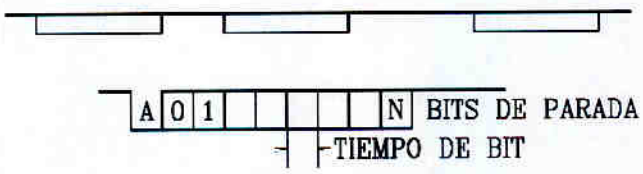


CONEXIÓN DE TECLADO MATRICIAL DIRECTAMENTE A LA V.I.A

LA DUART

La D.U.A.R.T MC68681. E/S Serie.

- Los sistemas electrónicos digitales utilizan normalmente como via de comunicación con el exterior, la transmisión asíncrona via serie. Se usa por la economía en tiempo y en hardware.
- Características de la transmisión asíncrona:
  - Transmisión carácter por carácter.
  - Temporización rígida dentro del carácter.
  - En reposo la línea a '1'.
  - Cada carácter tiene un tiempo de arranque. (bit START)
  - Cada carácter tiene un tiempo de finalización. (tiempo en que la linea debe estar a 1)
  - Número de bits reducido dentro del carácter.



Caramba!
Ta
Sin na que hacer!

CTS: si esta activado, transmite

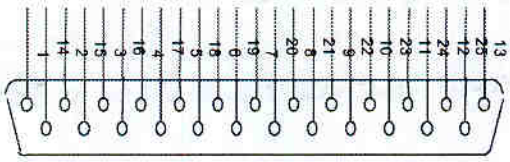
RTS: activado: hay huecos libres en el buffer de recepción
desactivado: tiene el buffer lleno; inhibe al otro extremo

Rapido
Toy
Sin na que hacer

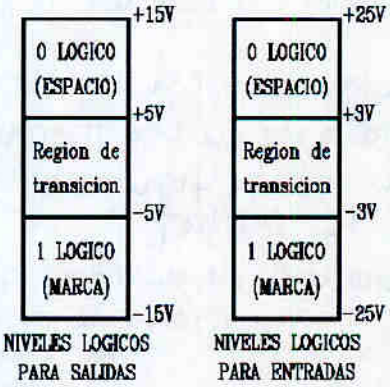
La DUART 68681. Iterfases normalizadas.

- Las conexiones entre DTE (sistema electrónico digital) y DCE (modem) se encuentran normalizadas. adapta la señal al medio
- Como ejemplo de norma de conexión entre DTE y DCE podemos tomar la RS-232-C. También puede ser utilizada para la conexión de dos DTE.

protocolo físico



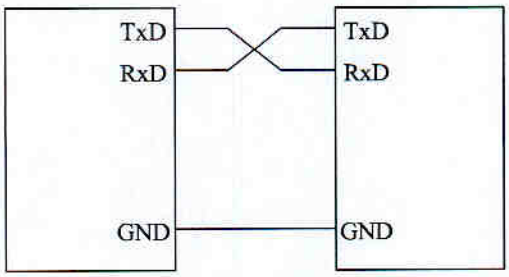
CONNECTOR DB25 RS232



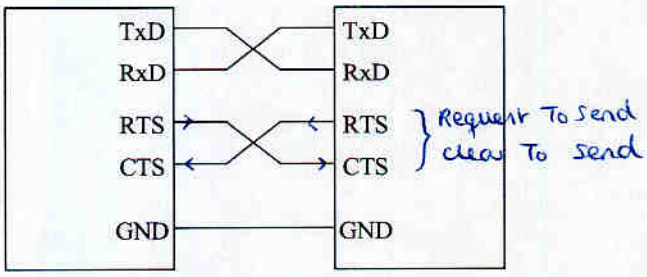
La DUART 68681. Conexión entre dos DTE

1	F-GND	Masa de protección.
2	TxD	Salida de **transmisión de datos** (del DTE al DCE).
3	RxD	Entrada de **recepción de datos** (del DCE al DTE).
4	RTS	Salida de **petición para enviar** (del DTE al DCE).
5	CTS	Entrada de **habilitación para enviar** (del DCE al DTE).
6	DSR	Entrada de **dispositivo de datos listo** para comunicar al DTE que el DCE está encendido y listo (del DCE al DTE).
7	S-GND	Terminal común masa.
8	DCD	Entrada de **detección de portadora de datos**. Es usada por el DCE (modem), para indicar al DTE que esta listo para transmitir datos (del DCE al DTE).
20	DTR	Salida de **terminal de datos listo** para indicar al DCE que el DTE está encendido y listo para funcionar (del DTE al DCE).

DEFINICIÓN DE TERMINALES SEGÚN LA RS232C



CONEXIÓN A 3 HILOS



CONEXIÓN A 5 HILOS

DUART: Mi interfaz con el mundo serie

- Lo programo
- le paso 8 bits paralelo
- genera la trama (bits start, stop, frecuencia...) según lo que le hayamos programado

Características

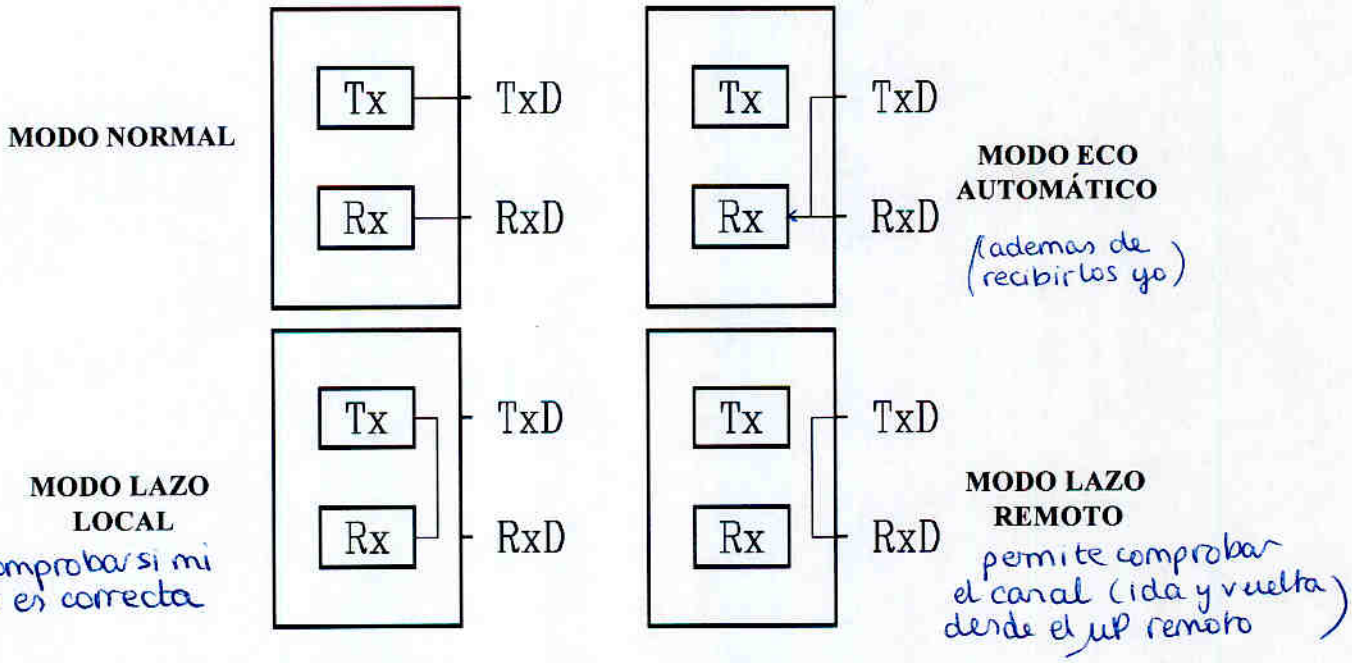
- Es amigo del 68000 (DTACK)
- Es doble: 2 canales Tx Rx Full Duplex
  Tx y Rx al mismo tiempo
- Profundidad registro de recepción: 4 caracteres
- Formato de datos programable
- Velocidad configurable individualmente para los 4 elementos de la DUART
- Capacidad de detectar errores de comunicación y señalizaciones
  i.e. errores de paridad
  de formato de trama
  de exceso (perdamos datos)
- Timer/contador multifunción programable
- Velocidad máxima transmisión: 1Mbit/s

Características generales de la DUART 68681.

- Compatible con el ciclo de bus asíncrono del 68000, incorporando la señal \DTACK.
- Doble Rx/Tx completo, con posibilidad de comunicación full-duplex.
- Profundidad de registros de recepción de 4 caracteres.
- Formato de datos programable.
- Velocidad configurable individualmente para cada transmisor y receptor, con selección de velocidades normalizadas mediante registro.
- Detección de errores de comunicación y señalizaciónes.
- T/C multifunción programable. Timer/contador
- Puertos de entrada y salida multifunción programable.
- Velocidad máxima de comunicación de 1Mbit/s.
- Frecuencias de reloj admitidas de 2 a 4 MHz. Reloj estándar de 3.6864 MHz.
- Modo multiconexión o multidurmiente.

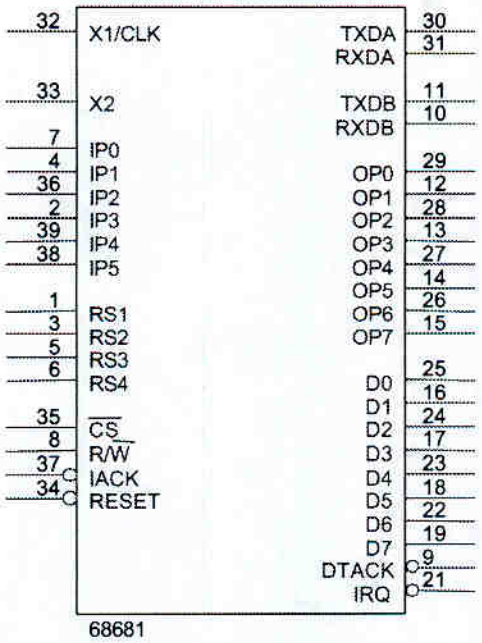
mínimo común múltiplo de todas las frecuencias estándar x16

La DUART 68681. Modos del canal de comunicación.



- DUART se mapea en posiciones impares:
  tiene DTACK, IACK y proporciona <u>número de vector</u>
  └ D0-D7 → impar
- 4 líneas de dirección (16 direcciones)
  pero hay 32 registros (utilizando R/$\overline{W}$)
  y además hay un par de registros extraños que tienen la misma dirección

Descripción del patillaje de la DUART 68681.

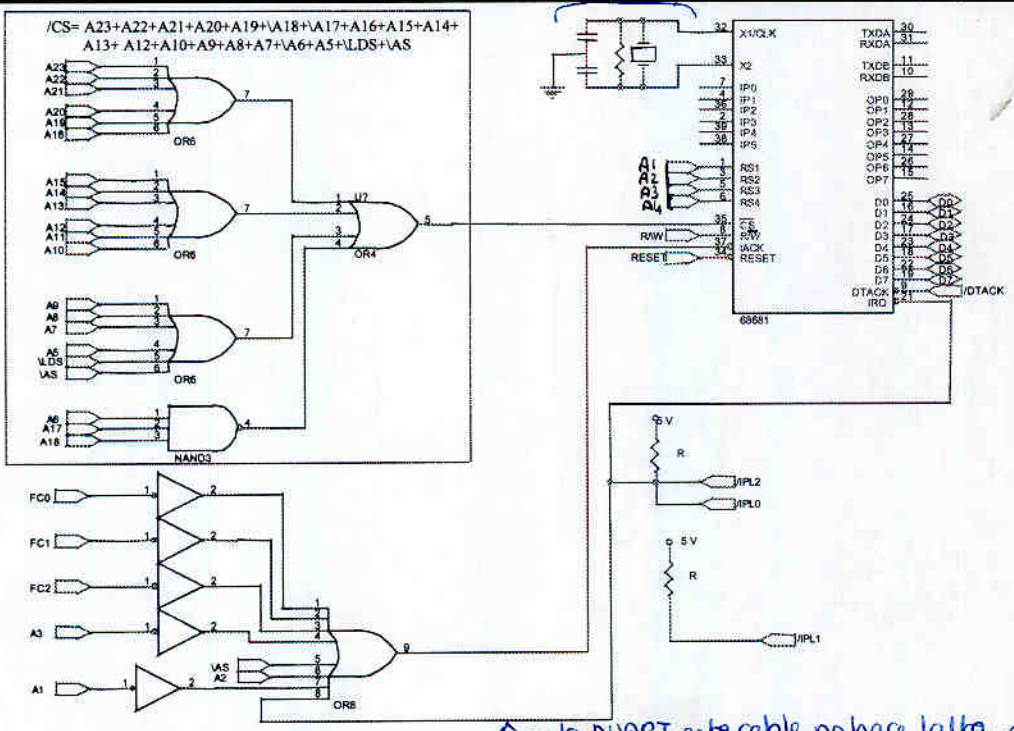


- Bus de datos de 8 bits bidireccional.
- Cuatro entradas de selección de registros internos. 16 posiciones de memoria con capacidad de seleccionar hasta 34 registros.
- Reconocimiento de transferencia de datos.
- Reconocimiento de interrupción.
- Puerto de salida de propósito general o salida de alguna de las condiciones de 8 bits. ej: RTS
- Puerto de entrada de proposito general o entrada de alguna señal específica de 6 bits. ej: CTS
- Salidas de transmisión y entradas de recepción conformando dos estaciones transmisor-receptor independientes.

Conexión de la D.U.A.R.T al 68000 en la posición $60041, nivel de interrupción 5.

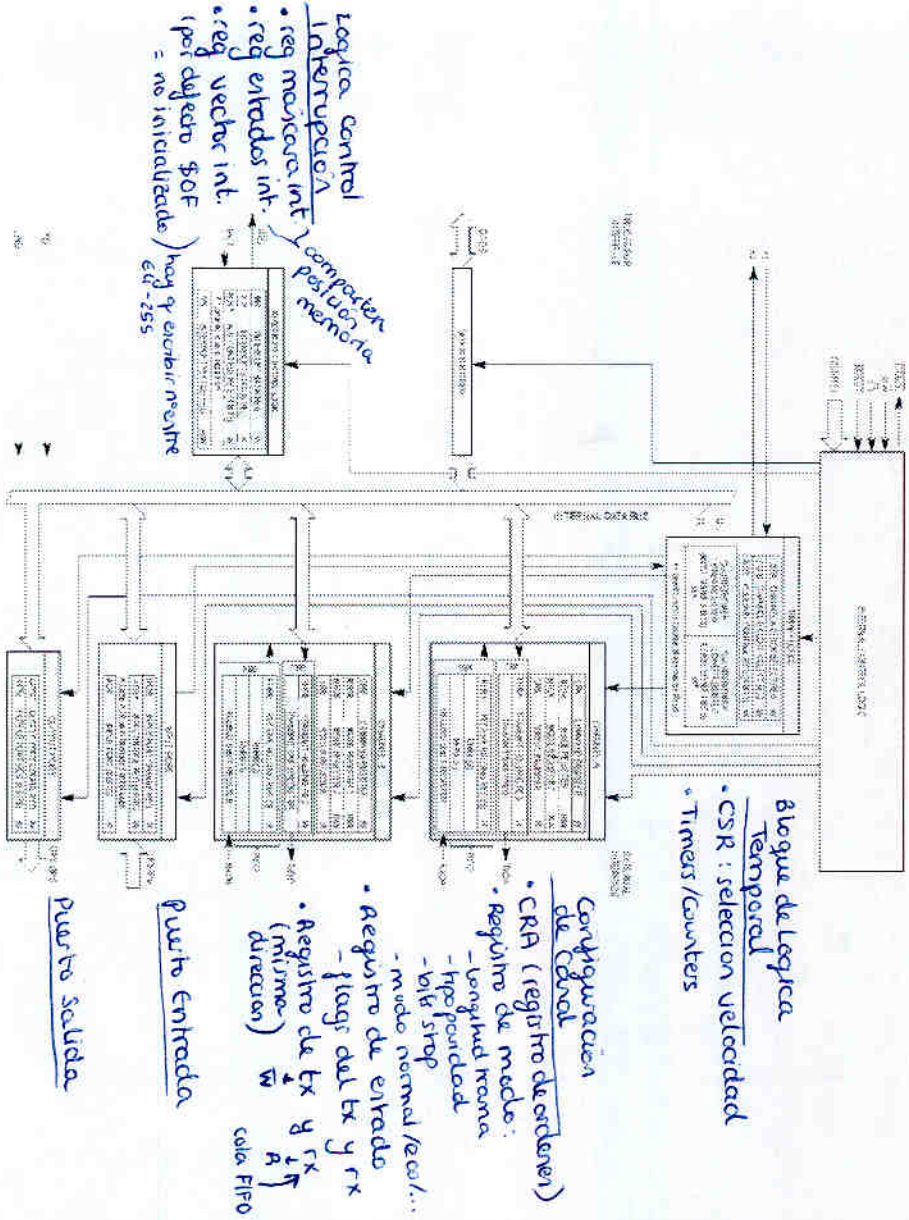
circuito de reloj de 3'6864 MHz

No olvidar en el examen

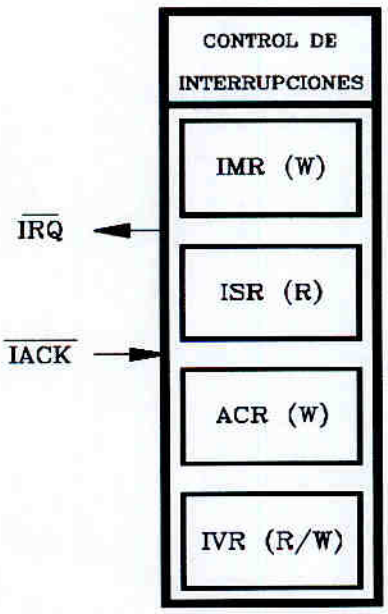


← en la DUART este cable no hace falta porque si llega un IACK sin que tenga interrupción, la DUART no responde con el vector. Por tanto no hay que inhibir IACK cuando la interrupción no esta vigente

Esquema de bloques de la DUART



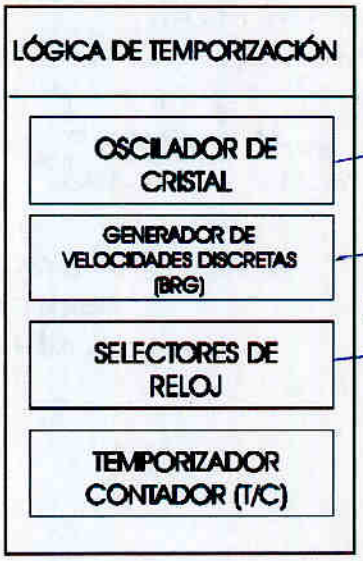
La DUART 68681. Control de interrupciones.



- La DUART tiene una única salida de interrupciones para reflejar los 8 posibles casos de interrupción interna.
- En el ISR se reflejan el estado de las interrupciones internas. (IFR en VIA)
- En el IMR se validan esas interrupciones internas para producir interrupción al uP por \IRQ. (IER en VIA)
- El registro IVR contiene el nº de vector a volcar en el ciclo de reconocimiento de interrupción ante la activación de \IACK. Se inicializa a $0F. = 15 – nº de vector no inicializado (entre 64 y 255)
- El ACR es una ampliación del ISR.

La DUART 68681. Circuitos de temporización.

- Bloque que consta de:
  - Oscilador de cristal.
  - Generador de velocidades discretas (BRG).
  - Un T/C programable
  - 4 selectores de reloj (uno por cada transmisor o receptor).
- El BRG trabaja con el reloj de la DUART y para el reloj estándar genera 18 velocidades entre 50 a 38400 baudios.
- Cada Tx o Rx podrá tomar como fuente de reloj: 16x las vel estándar
  - El BRG. (gen velocidades a partir del reloj de la VIA)
  - Fuente de reloj externa (Ipx).
  - El T/C.

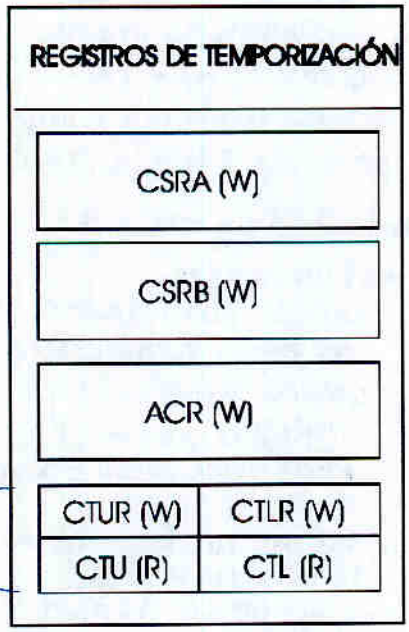


3'6864 MHz

divisor de velocidades programable

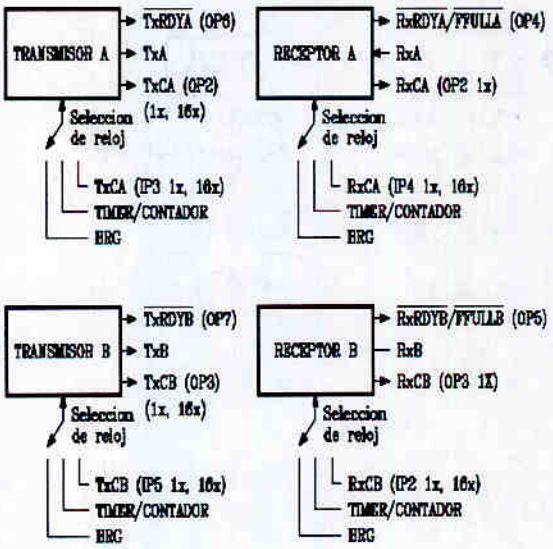
selecciona si el CLK viene del BRG o de uno exterior (del receptor)

La DUART 68681. Circuitos de temporización (II).



No nos lo explican

SELECCIÓN DE RELOJ



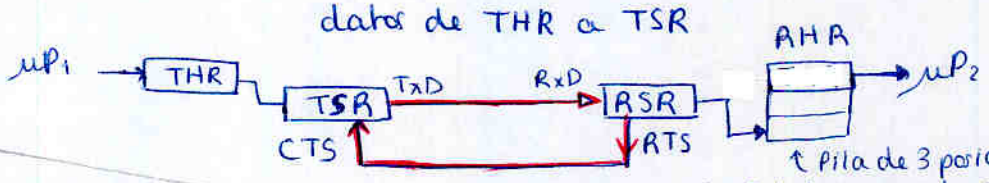
CRA: **Registro ordenes:** activar condiciones de break, iniciar/deshabilitar Rx ó Tx, y ordenes diversas

Registros modo:

MR1A
MR2A
- formato trama
- bits stop
- paridad y tipo de paridad
- tipo de protocolo
- modo de canal

Registro estado: SRA — 8 bits: 2: condiciones receptor
2: condiciones transmisor
4: condiciones error

control de transmisor por $\overline{CTS}$
Si CTS esta desact. se impide el paso de datos de THR a TSR

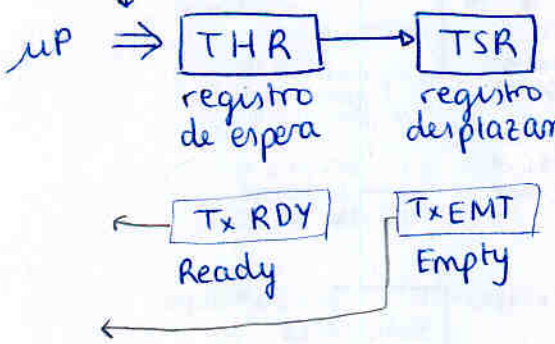


↑ Pila de 3 posiciones
el 4° dato se queda en RSR y se desactiva RTS (si esta programado)

↓ para evitar que le envien mas datos.

Transmisor

- Transmite cuando está habilitado
- se le pueden escribir datos en el registro de espera (si está habilitado)



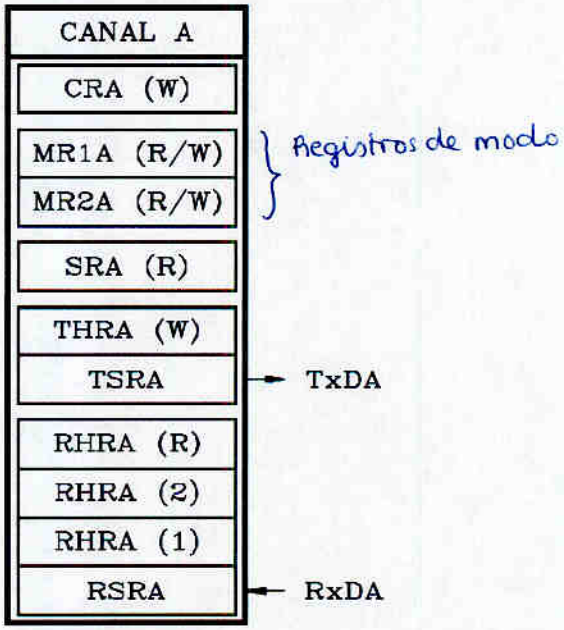
- deshabilitado impido la escritura (pero no me cargo lo que habia) e impido paso de THR a TSR
- Habilitado y activado:
  - → TxRDY activo,
    - cuando paso dato a registro de espera se desactiva TxRDY
    - cuando éste se carga a TSR, TxRDY se vuelve a activar para que pase otro dato (al cabo de un cierto tiempo)

Si TSR ha acabado la tx y THR no tiene nada
→ se activa TxEMT

condición de Break: linea a cero sin respetar formato de trama

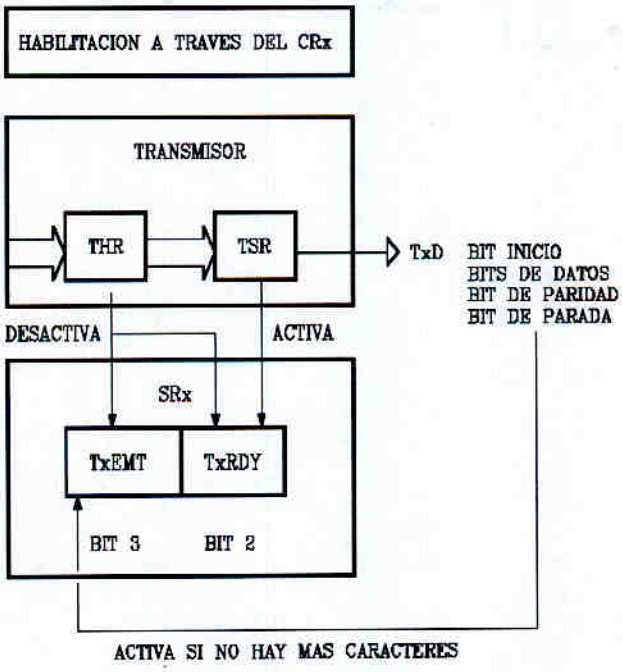
Canales de comunicación de la DUART.

- Cada canal consta de un receptor y un transmisor.
- Frecuencia de operación seleccionable independientemente.
- Emisor: convertidor paralelo-serie con el formato de trama programado
  - Acepta datos en paralelo y los convierte en una trama serie a través del pin TxD.
  - Inserta el bit de arranque, de parada y paridad.
  - Dispone de dos registros, uno de espera y otro de desplazamiento.
- Receptor:
  - Acepta bits serie a través de RxD y los paraleliza. (a una cola FIFO) ← Para que el µP la lea cuando quiera
  - Detecta condiciones de error.
  - Dispone de un registro de desplazamiento y otros tres de espera.



Permite tener hasta 3 datos en espera
↳ el 4° ya se carga al 1°
↳ flag de sobreexceso

El Transmisor de la DUART (preguntable)

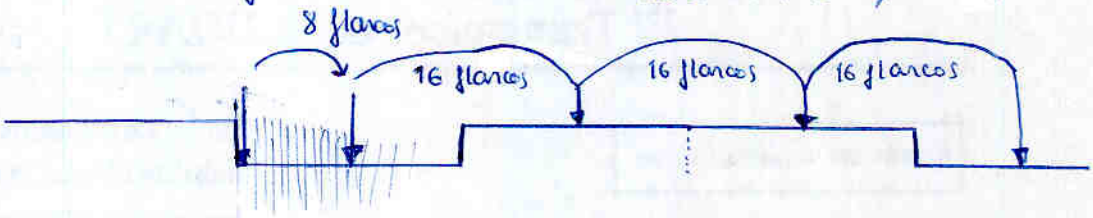


- La DUART transmite cuando se habilita el transmisor (CRx).
- Indica a la CPU que está listo activando la señal TxRDY(SRx).
- Se desactiva cuando se carga el carácter en el THR
- Este carácter se transfiere al TSR para ser transmitido por TxD.
- La señal TxEMT indica THR y TSR vacíos.
- Posibilidad de inicialización.
- Posible control del transmisor por parte de \CTS.
- Modo autoreset.
- Posibilidad de BREAK.

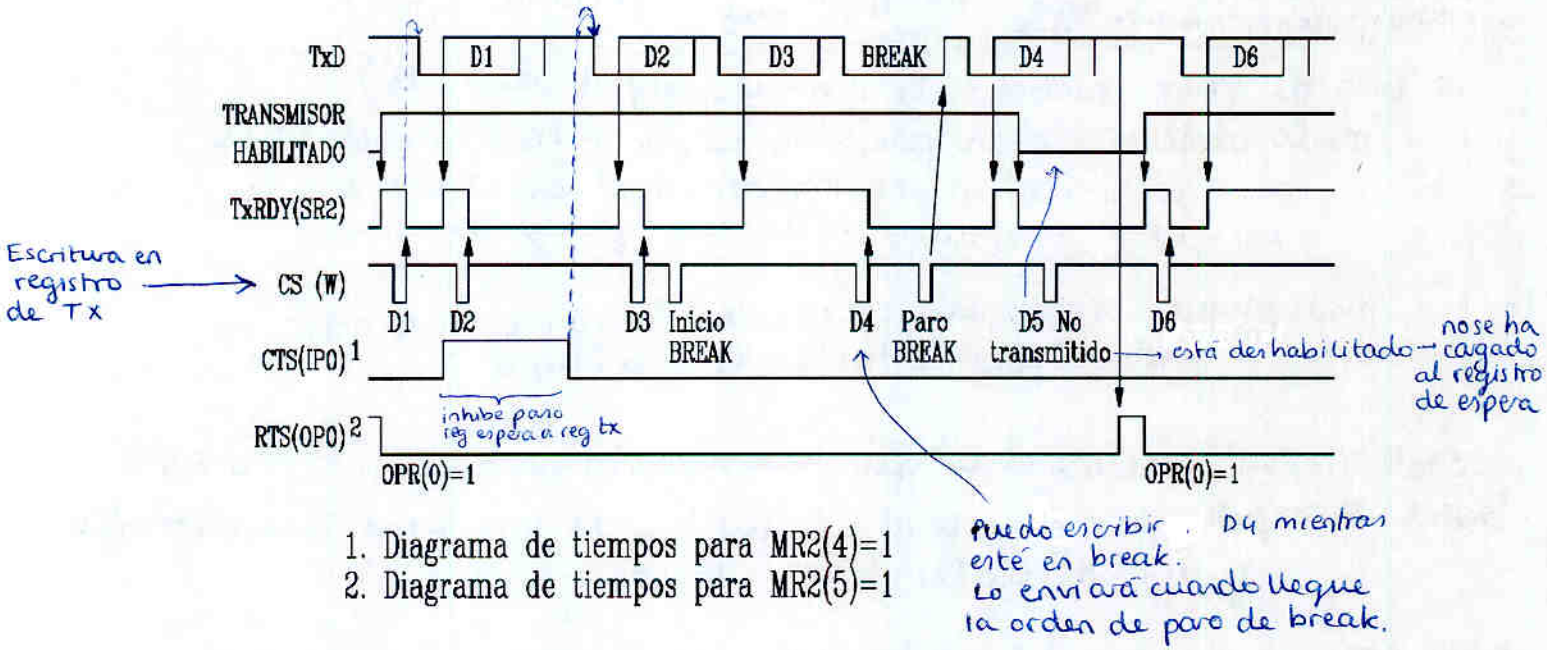
Transmisión asíncrona: los relojes en tx y rx no están inicialmente en fase (no sabemos cuando llegará un paquete) por ello el reloj en rx debe ir a mayor frecuencia (en este caso 16 veces).

cuando rx detecta un cero, espera 8 muestreos y si muestrea otro cero, interpreta un inicio de trama, y a partir de ahí muestrea cada 16 flancos (en el 48% → 52% del bit)

evita falsos inicios de trama



Cronograma de la Transmisión de la DUART.



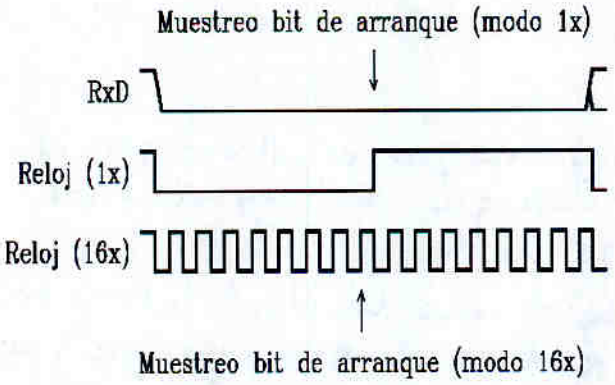
1. Diagrama de tiempos para MR2(4)=1
2. Diagrama de tiempos para MR2(5)=1

Puedo escribir D4 mientras esté en break. Lo enviará cuando llegue la orden de paro de break.

Una pregunta de examen podría ser este cronograma sin TxRDY ni TxD

El Receptor de la DUART

DETECCION DE INICIO DE TRAMA



- El reloj de recepción funciona a 16x la frecuencia de la comunicación salvo cuando utilizamos reloj externo. — misma en rx y tx
- La detección de inicio de trama la realiza mediante un doble muestreo.
- Posteriormente muestrea una vez cada bit.
- Nos permite realizar un muestreo seguro ante posibles variaciones en la frecuencia entre el Tx y Rx.

- Error de trama

- Error de break: error de trama cuando la trama es todo ceros provocado por tx o por rotura de medio físico

Bits de error:

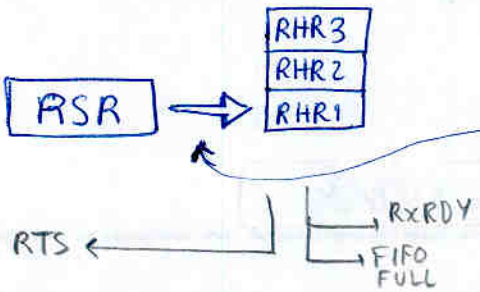
- Error de exceso: Pila llena, registro rx lleno, y aun así llega quinto dato (no se da nunca si uso protocolo CTS y RTS) se han perdido datos
- Error de Paridad

Modo de error

La pila de errores funciona de 2 modos (registro de modo)

- Modo carácter: cada posición de la pila FIFO (un caracter) tiene asociado su registro del error. Para utilizar bien el µP 1ero compruebo reg de errores y luego leo datos
- Modo bloque (trama): es como una OR del error. Basta que un caracter sea erróneo para deshechar todo el bloque

- Señal RxRDY: decirle al µP que hay al menos un dato en la pila FIFO
- Señal FIFO full: para decirle al µP que la pila FIFO tiene 3 caracteres, (llena) el µP puede leerlos todos a la vez
- Señal RTS: se conecta al CTS del emisor. Se desactiva cuando tiene la pila llena, y tiene ocupado el registro de rx.



Cuando la pila está llena; se desactiva el trasvase RSR → RHR1 si llegan nuevos caracteres sin que leamos, el nuevo caracter se carga el anterior (que está en RSR), pero se mantienen los de la pila

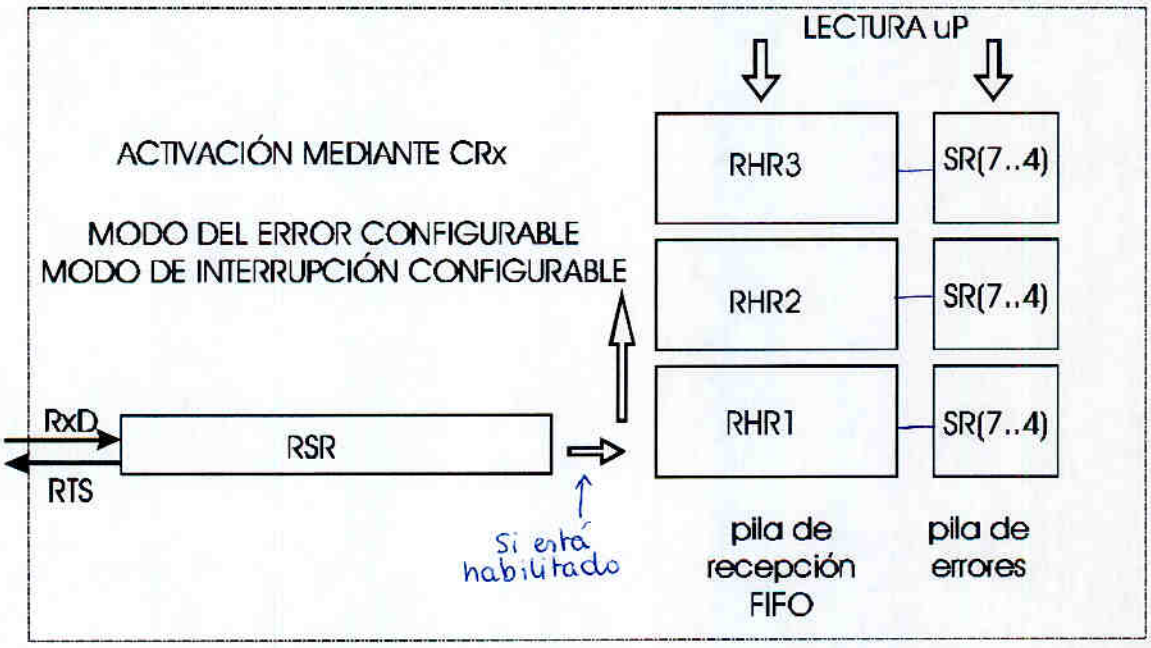
Para leer carácter desde el µP:

- Primero leemos estado (SR)
- Luego leemos RHR (la DUART se encarga de que el µP vea sólo el registro que contiene el primer dato que ha llegado → los otros registros se llaman registros shadow.

Al leer el dato, ya cambia el registro que lee el µP (al siguiente dato) por eso hay que leer SR ANTES de RHR, ya que si lo hacemos al reves, leeriamos el SR del siguiente.

El receptor de la DUART (II).

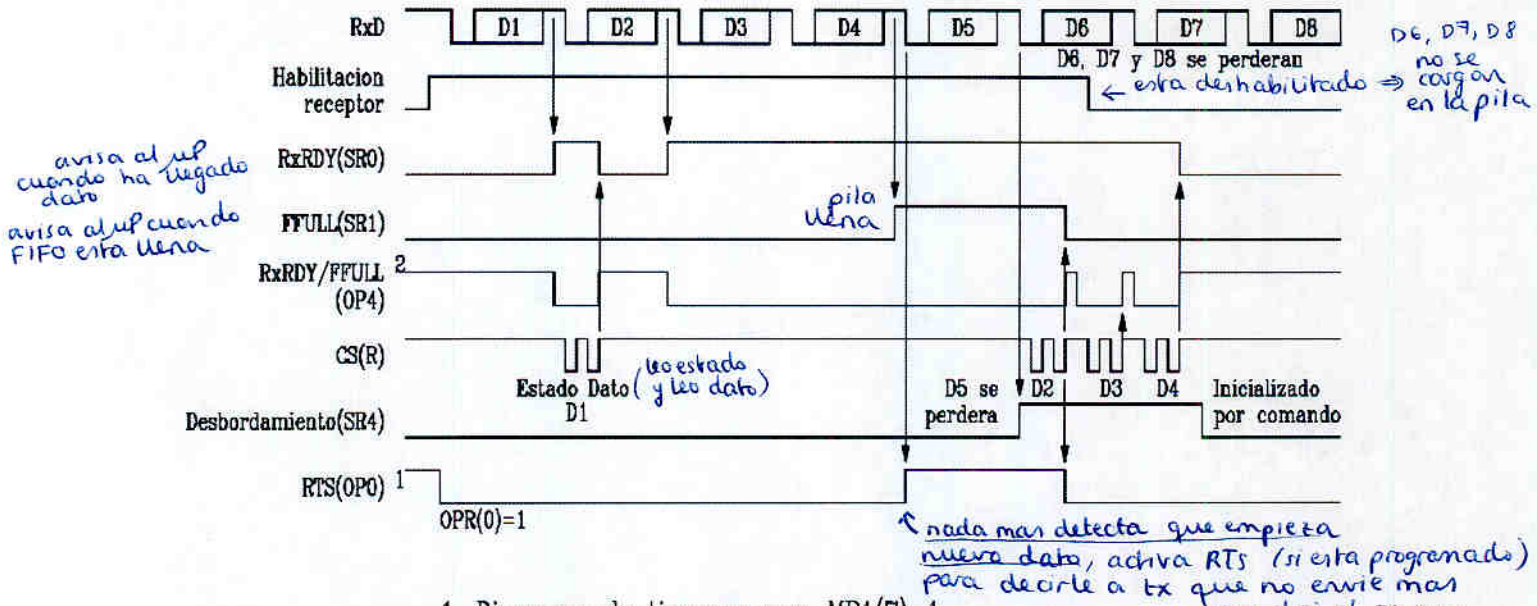
ESQUEMA DEL RECEPTOR



Funcionamiento como receptor.

- Detección de bit de arranque, y captura del dato en el RSR.
- Detección de errores de paridad, trama, condición de BREAK y exceso.
- Modo de error programable mediante el MR1x:
  - CARÁCTER: Los bits de estado se generan carácter a carácter y están asociados al primer dato que llegó y no ha sido leído.
  - BLOQUE: Los bits de estado se generan mediante la OR lógica de todos las condiciones de error de los caracteres que pasan por la primera posición de la FIFO desde la 'inicialización del estado del error' en el CRx.
- Modo de interrupción programable:
  - RxRDY: Existe al menos un carácter en la FIFO de recepción.
  - FFUL: Las tres posiciones de la FIFO están ocupadas.
- El receptor puede controlar la desactivación de \RTS si se ha programado en el MR1x.
- La inhabilitación del receptor no impide la lectura de los caracteres guardados en la FIFO, pero si la llegada de nuevos caracteres. La inicialización implica el borrado de todos los caracteres existentes en la FIFO.

Cronograma de la recepción.

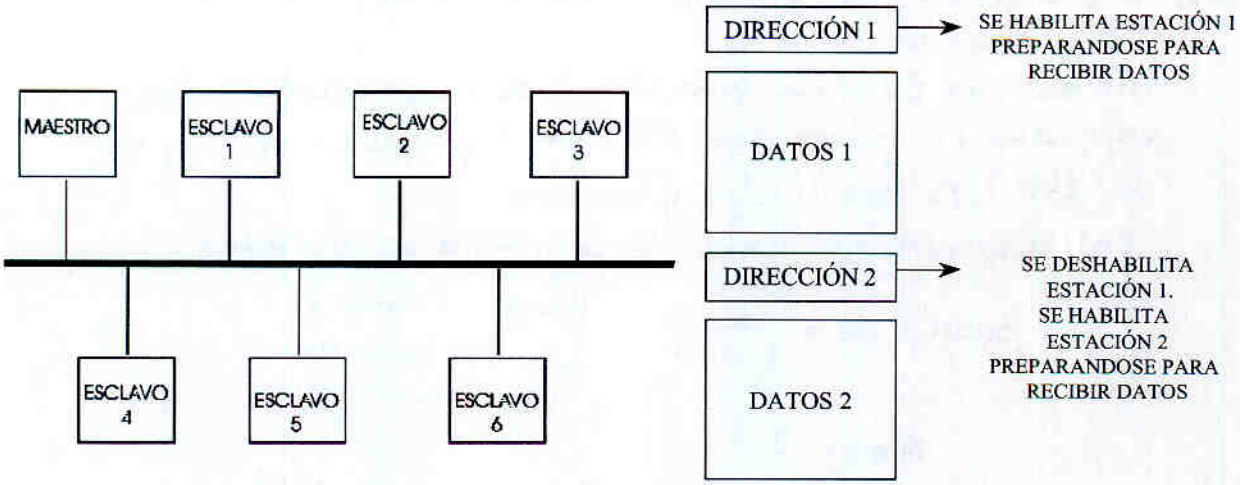


1. Diagrama de tiempos para MR1(7)=1
2. Diagrama de tiempos para OPCR(4)=0 y MR1(6)=0

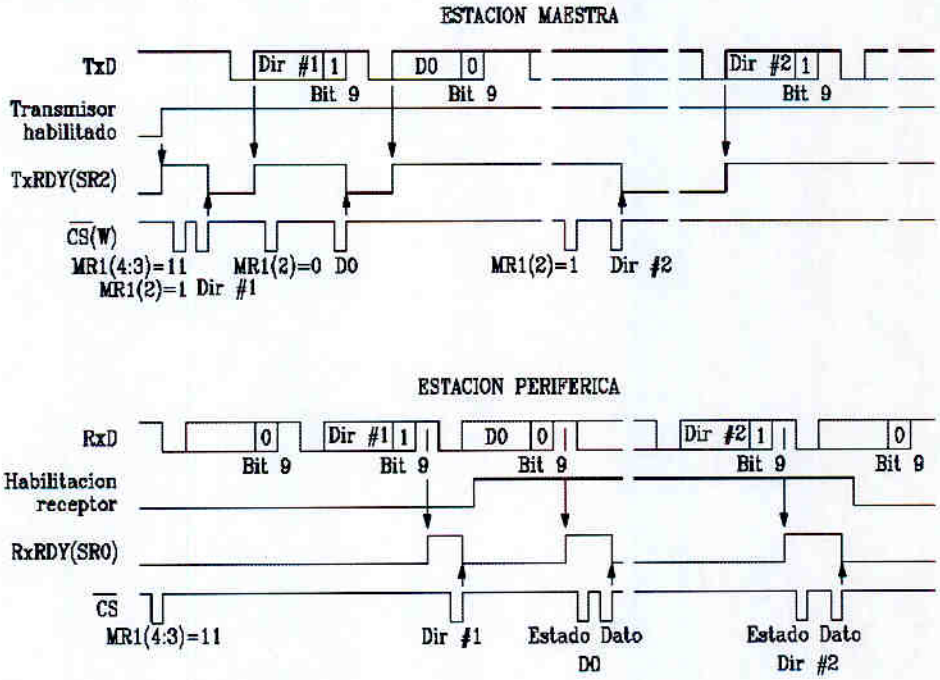
Modo Multiconexión.

- Se selecciona con MR1x (4:3) a '11'.
- Un maestro transmite un carácter de dirección seguido por caracteres de datos.
- Las estaciones esclavas deben de estar con los receptores inhabilitados. En el modo multiconexión la DUART interrumpe al uP ante la llegada de un carácter de dirección aunque el receptor esté inhabilitado.
- El tipo del carácter se refleja en el bit de paridad y se programa en el MR1x(2).
- El receptor reflejara si el carácter es de datos o de dirección en el SR(5).

Funcionamiento del modo Multiconexión



Cronogramas del modo multiconexión



TIMER/CONTADOR

- Generar velocidades de funcionamiento/comunicación

ej: queremos frecuencia no estándar

ej: si en lugar de reloj estandar 3'6864 MHz utilizamos otro (se permiten de 2 a 4MHz)

entonces, el BRG (que generaba todas las velocidades estándar a partir de 3'6864MHz (ya que son múltiplos de ella)) se descuadra y no sirve.

Entonces podemos generarlo con el timer/contador

$$\text{Baud Rate} = \frac{f}{2 \cdot N \cdot 16}$$

ya que el timer es x16 la velocidad de comunicación

$N_{min} = 2$

Escogemos el T/C con CSRx (clock select registry)

Nota: existe modo asíncrono con reloj

Registros de la DUART.

A4	A3	A2	A1	LECTURA (R/\W=1)	ESCRITURA (R/\W=0)
0	0	0	0	Reg. de modo canal A (MR1A, MR2A)	Reg. de modo canal A (MR1A, MR2A)
0	0	0	1	Reg. de estado canal A (SRA)	Reg. de selección reloj canal A (CSRA)
0	0	1	0	Reservado	Reg. comandos canal A (CRA)
0	0	1	1	Reg. espera receptor canal A (RHRA)	Reg. espera transmisor canal A (THRA)
0	1	0	0	Reg. cambios del puerto de entrada (IPCR)	Reg. control auxiliar (ACR)
0	1	0	1	Reg. del estado de las interrupciones (ISR)	Reg. máscara de interrupciones (IMR)
0	1	1	0	Reg. del C/T parte alta contador (CTU)	Reg. del C/T parte alta del latch (CTUR)
0	1	1	1	Reg. del C/T parte baja contador (CTU)	Reg. del C/T parte baja del latch (CTLR)
1	0	0	0	Reg. de modo canal B (MR1B, MR2B)	Reg. de modo canal B (MR1B, MR2B)
1	0	0	1	Reg. estado canal B (SRB)	Reg. de selección reloj canal B (CSRB)
1	0	1	0	Reservado	Reg. ordenes canal B (CRB)
1	0	1	1	Reg. espera receptor canal B (RHRB)	Reg. espera transmisor canal B (THRB)
1	1	0	0	Reg. del vector de interrupciones (IVR)	Reg. del vector de interrupciones (IVR)
1	1	0	1	Reg. del puerto de entrada (IPR)	Reg. configuración puerto salida (OPCR)
1	1	1	0	Orden de inicio del contador	Reg. activación bits puerto salida (OPRS)
1	1	1	1	Orden de paro del contador	Reg. inactivación bits puerto salida (OPRR)

El Timer/Contador.

- Se configura en el ACR, y se programa la cuenta escribiendo en CTUR y CTUL.
- Se utiliza para poder generar velocidades de comunicación no recogidas por el BRG o cuando la DUART no tiene el reloj estándar.
- Para arrancarlo hay que leer en la posición $E y para pararlo en la $F.
- Cuando se utiliza como reloj, es el reloj 16x la velocidad de comunicación siendo la cuenta mínima 2.
- La velocidad de comunicación alcanzada cuando lo programamos con el reloj de la DUART 1x, es:

$$Baud\ rate = F_{DUART}/(2*16*N)$$

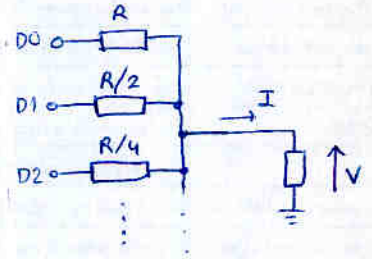
ENTRADA-SALIDA ANALOGICA

mundo real continuo ⟷ n bits
↳ Los n bits no sufrirán distorsión

Resolución: mínimo incremento detectable

Frecuencia de muestreo: Teorema de Nyquist para no tener aliasing

Convertidor D/A



convertidor D/A
Cada bit se suma (ponderado) para sacar una corriente que será la de cada bit ponderado.
La corriente de salida es proporcional al valor digital

- Necesito algo que me mantenga el dato en D0-D7 en el tiempo entre escrituras: (no podemos conectarlo directamente al bus) Hay que poner un Latch.

- Necesitamos señales de referencia (valores máximo y mínimo)

La salida son en realidad pequeños escaloncitos; se quitan poniendo un filtro.

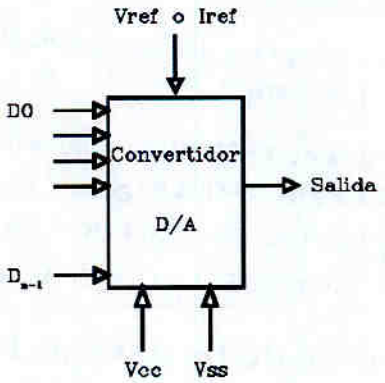
Entrada Salida Analógica.

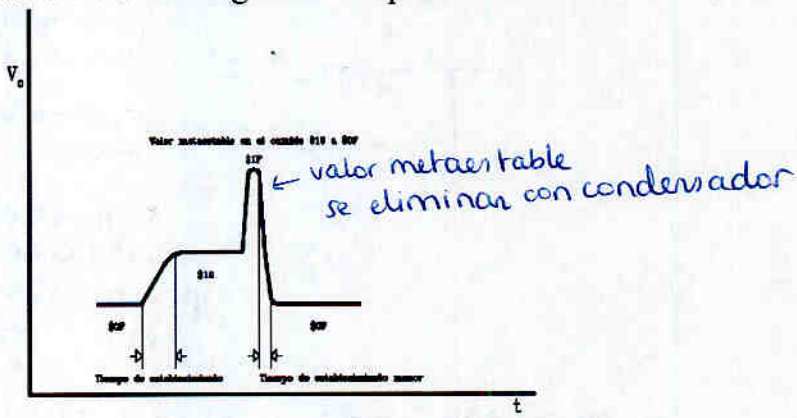
- Mucha parte del antiguo procesado analógico de las señales hoy en dia se realiza digitalmente.
- La principal ventaja es que una vez digitalizada la señal, ésta no va a sufrir ninguna distorsión.
- Los circuitos de conversión utilizados para poder realizar este procesado son los convertidores A/D y D/A.
- Como aspectos a tener en cuenta para el diseño del sistema digital: resolución (mínimo incremento detectable por nuestro sistema) y frecuencia de muestreo (Niquist).

SECUENCIA DE TRATAMIENTO DIGITAL DE SEÑALES

El convertidor D/A

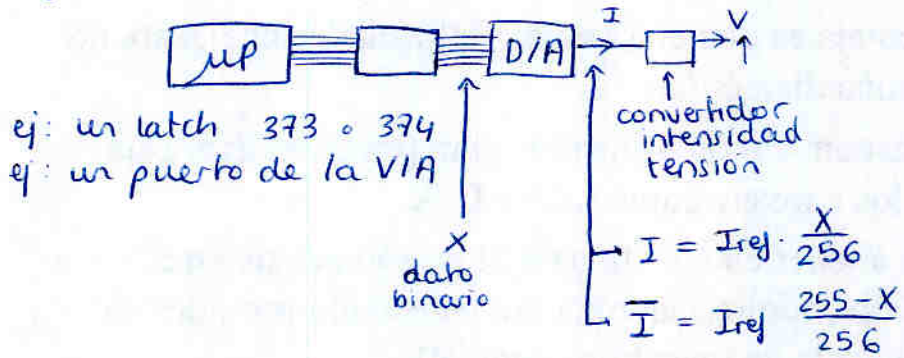
- Se encarga de convertir un dato digital de n bits, en una corriente o tensión analógica (2^n posibles valores de salida).
- Necesidad de incorporar un latch para capturar el dato.
- Señales de referencia (Valores máximo y mínimo de salida)
- Funcionamiento: Se coloca el dato digital en la entrada y al cabo de un tiempo (asentamiento) aparece en la salida el valor analógico correspondiente.





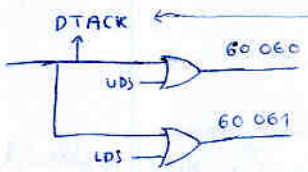
DAC 0800

- No se puede conectar directamente al µP.
- Debe haber una memoria que registre el dato que el µP quiera poner en el D/A.



Nota de este esquema:

- $\overline{DTACK}$ está antes de la ultima puerta para que $\overline{UDS}$ no retarde el ciclo de escritura → Sólo esta bien si hay algo en 60061
- Podriamos tener este circuito (D/A en 60060) y ademas tener el A/D en 60061 que tendría el mismo DTACK, pero la ultima puerta sería con LDS



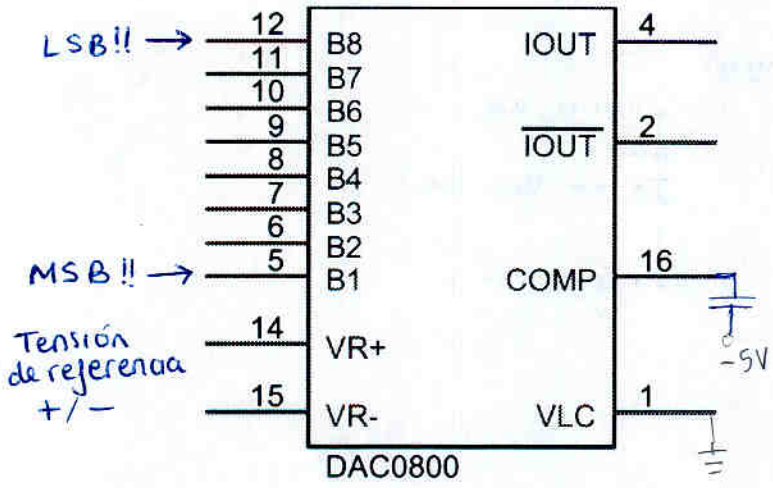
podriamos generar DTACK aquí, ya que o uno o otro se va a activar

Esto era valido siempre y cuando estuvieran AMBOS bancos (par/impar)

Si no están ambas direcciones (60060 y 60061) NO PODEMOS GENERAR DTACK antes de la ultima puerta ya que podría direccionar la 60061 y generar DTACK sin que tuviera sentido.

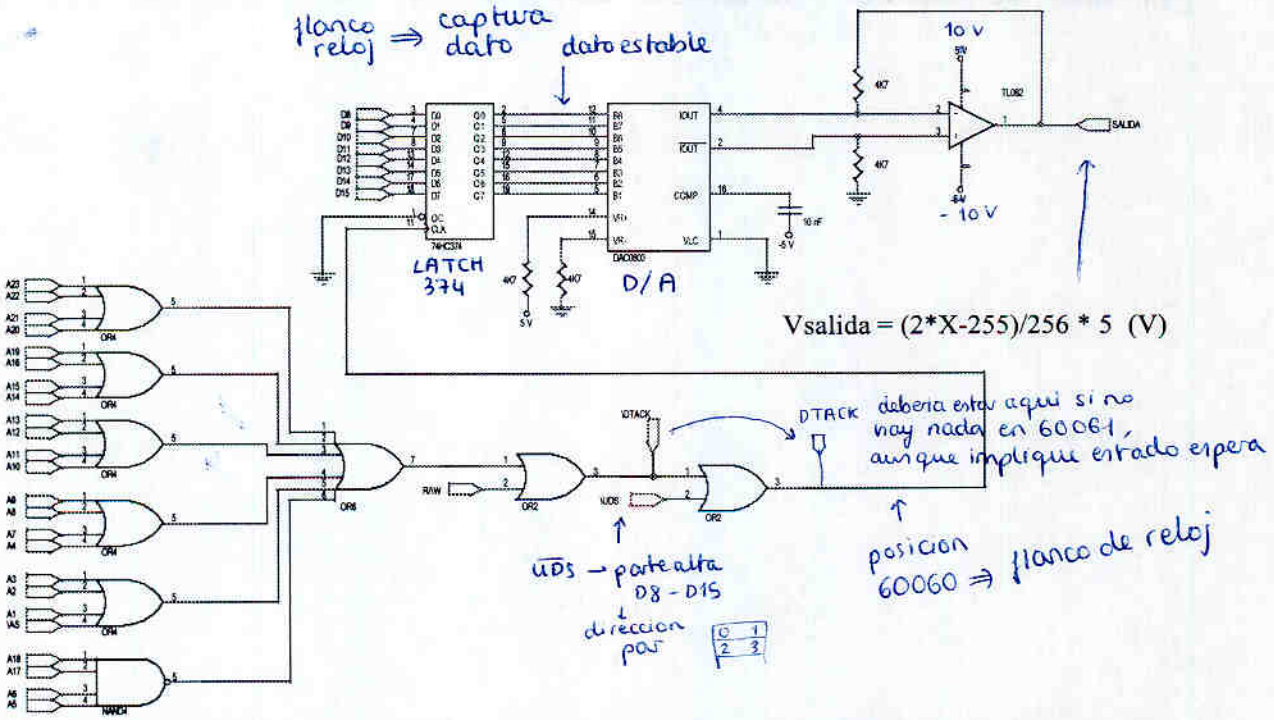
⇒ si no tenemos nada en 60061 el circuito adjunto debería tener el DTACK tras UDS → generará estados de espera →

El convertidor D/A. El DAC0800



- Convertidor Digital Analógico de 8 bits.
- Acepta niveles lógicos tanto TTL, CMOS y de otras familias. Se configuran a través de VLC.
- B1 es el bit más significativo.
- Por Vref se le proporciona una corriente de referencia.
- COMP compensa en frecuencia el OPAM interno de referencia. Conectando un condensador
- $\boxed{I_{out} = I_{ref} * (X/256);}$
- $\boxed{\backslash I_{out} = I_{ref} * ((255-X)/256)}$
- $I_{FS} = I_{ref} * 255/256$

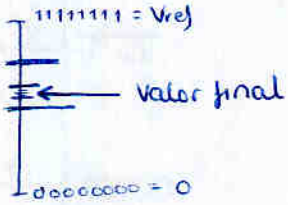
Conexión del DAC0800 al 68000. Posición $60060



CONVERTIDOR A/D

V_{IN} debe ser estable durante el tiempo de conversión (suele ser un muestreo de valores fijos) (sample and hold)

- Convertidor por aproximaciones sucesivas

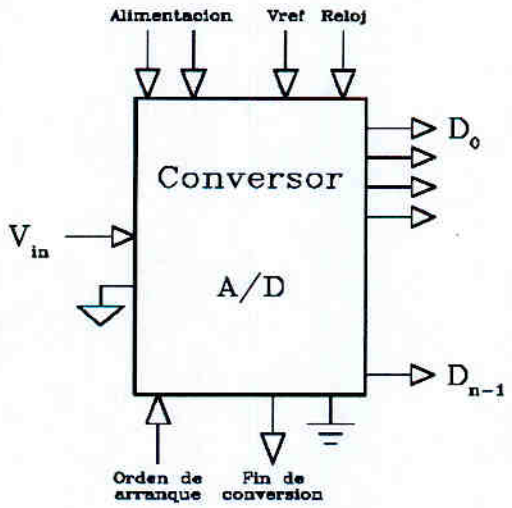


ADC 0808

- Tiene 8 canales de entrada → elegimos cual con A0-A2 y activando ALE (address latch enable)
- Al activar START
  EOC (end of conversion) baja

EOC sube al acabar la conversión

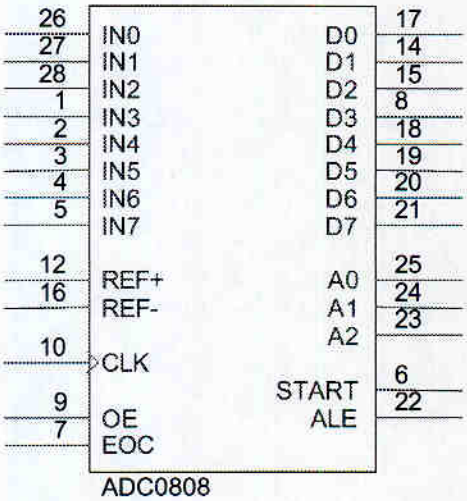
El Convertidor Analógico Digital



- Los convertidores A/D poseen gran variedad de mecanismos de conversión.
- Necesitan generalmente una señal de sincronismo (reloj).
- El dato de salida convertido puede ser serie o paralelo.
- La tensión de referencia debe de ser estable y precisa.
- Precisa de señales de control (inicio de conversión, fin de conversión).
- Algunos llevan incorporado un multiplexor analógico.
- Masas separadas para evitar ruidos.

El convertidor A/D. ADC 0808.

- Es un convertidor A/D de 8 bits con 8 canales analógicos de entrada.
- Funcionamiento:
  - Se activa la señal ALE a nivel alto para capturar el canal a convertir (Inx).
  - Por las líneas A0-A2 se indica dicho canal.
  - Se activa la señal START de incio de conversión.
  - Baja la línea EOC a '0'.
  - Una vez realizada la conversión se activa la línea EOC. a "1"
  - Activaremos la línea OE para capturar el dato por D0-D7.



Tiempo hasta la salida es maximo 250 ns → muy lento

↓

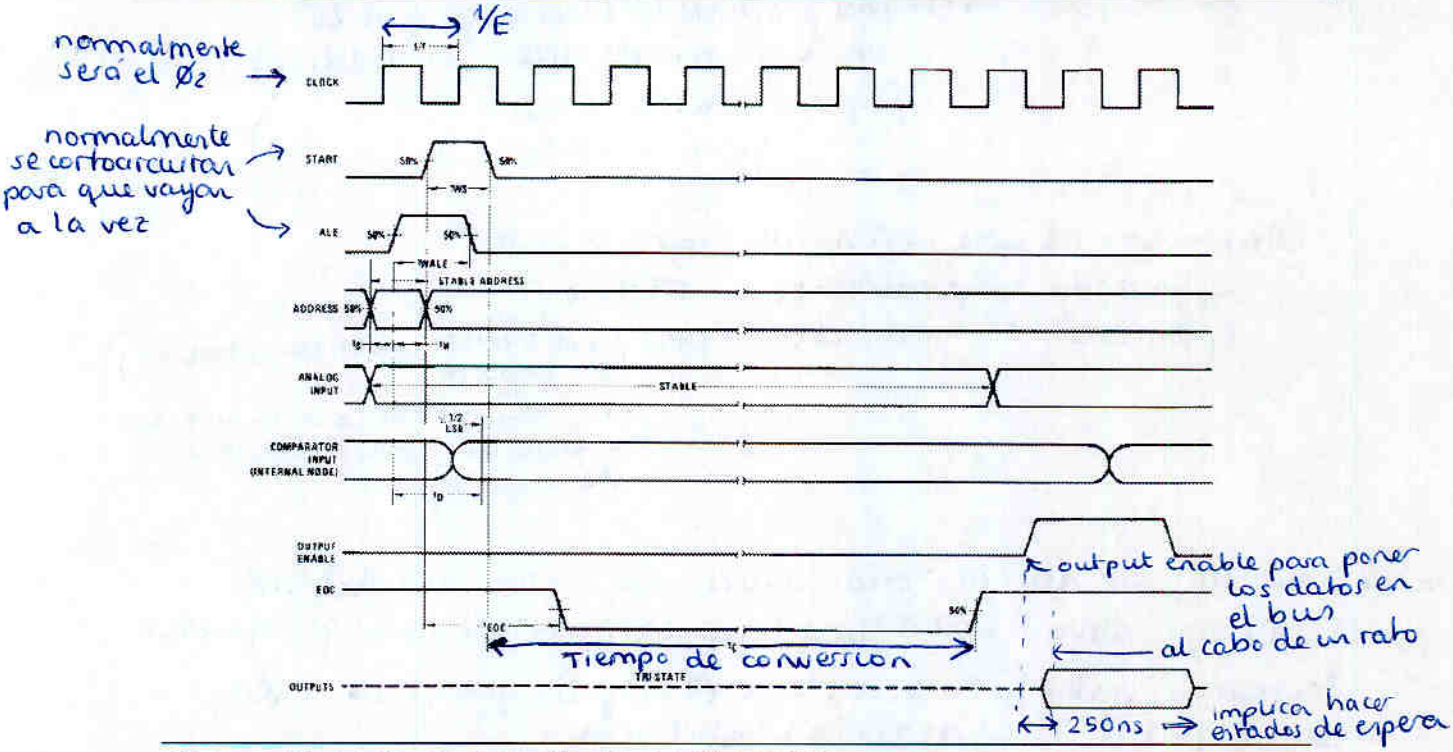
probablemente generará estados de espera.

Habrá que retardar DTACK

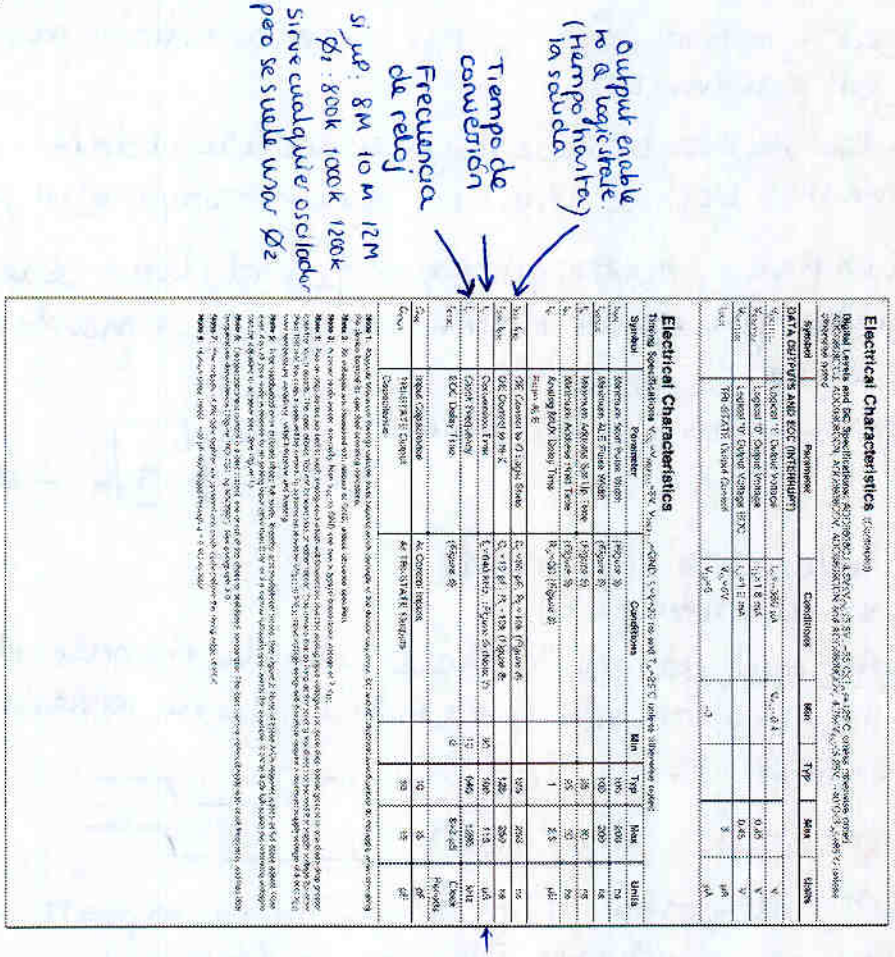
Otra solución: utilizar VPA

(es un dispositivo de acceso poco habitual, así que da igual que sea mas lento)

Cronograma de la conversión.



Hoja técnica del ADC0808



Electrical Characteristics (continued)

Digital Levels and DC Specifications

Symbol	Parameter	Conditions	Min	Typ	Max	Units
DATA OUTPUTS AND EOC (INTERRUPT)						
V_{OUT1}	Logical "1" Output Voltage	$I_{OUT1} = -360\ \mu A$	2.4			V
V_{OUT0}	Logical "0" Output Voltage	$I_{OUT0} = 1.6\ mA$			0.45	V
I_{SOURCE}	Logical "0" Output Voltage EOC	$V_{OUT1} = 5\ V$	4.5		0.45	V
I_{OUT}	TRI-STATE Output Current	$V_{OUT} = 0V$	-3			μA
		$V_{OUT} = 5V$			3	μA

Electrical Characteristics

Timing Specifications $V_{CC} = V_{REF(+)} = 5V$, $V_{REF(-)} = GND$, $T_A = 25°C$ unless otherwise noted

Symbol	Parameter	Conditions	Min	Typ	Max	Units
t_{WS}	Minimum Start Pulse Width	(Figure 3)	100	200		ns
t_{WALE}	Minimum ALE Pulse Width	(Figure 3)	100	200		ns
t_S	Minimum Address Set Up Time	(Figure 3)	25	50		ns
t_H	Minimum Address Hold Time	(Figure 3)	25	50		ns
t_D	Analog MUX Delay Time	$R_S = 0\Omega$ (Figure 2)		1	2.5	μs
t_{H1}, t_{H0}	OE Control to Q Logic State	$C_L = 50\ pF$, $R_L = 10k$ (Figure 5)		125	250	ns
t_{1H}, t_{0H}	OE Control to Hi-Z	$C_L = 10\ pF$, $R_L = 10k$ (Figure 5)		128	250	ns
t_C	Conversion Time	$f_{CLK} = 640\ kHz$ (Figure 6) (Note 7)	90	100	116	μs
f_C	Clock Frequency		10	640	1280	kHz
t_{EOC}	EOC Delay Time	(Figure 6)	0		8+2.5 μs	Clock Periods
C_{IN}	Input Capacitance	At Control Inputs		10	15	pF
C_{OUT}	TRI-STATE Output Capacitance	At TRI-STATE Outputs		10	15	pF

Note 1: Absolute Maximum Ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

Note 2: All voltages are measured with respect to GND, unless otherwise noted.

Note 3: A 2.5kΩ resistor should be placed between the +5V power supply and the $V_{REF(+)}$ pin when the power supply is used as the reference source. Accuracy is dependent on the +5V power supply when its voltage is used as the reference source.

Note 4: Two optional adjustments allow the user to minimize offset error and full-scale error to zero prior to converting. Typical ladder resistance is 2kΩ at full-scale and 0Ω at zero. The ladder resistance is a strong function of temperature and behaves like the rest of the device. Typical drift is 0.5 ppm/°C.

Note 5: Two optional adjustments allow the user to minimize offset error and full-scale error to zero prior to converting. Typical ladder resistance is 2kΩ at full-scale and 0Ω at zero. The ladder resistance is a strong function of temperature and behaves like the rest of the device. Typical drift is 0.5 ppm/°C.

Note 6: Two optional adjustments allow the user to minimize offset error and full-scale error to zero prior to converting. Typical ladder resistance is 2kΩ at full-scale and 0Ω at zero. The ladder resistance is a strong function of temperature and behaves like the rest of the device. Typical drift is 0.5 ppm/°C.

Note 7: The clock duty cycle can generally be adjusted within the range of 40/60 to 60/40 without affecting performance.

Note 8: Conversion time is the time required for the SAR to perform a conversion. Total conversion time is 8 clock periods plus the delay time from the start pulse to the rising edge of EOC.

Output enable ha q registrate (tiempo hasta la salida)

Tiempo de conversión

Frecuencia de reloj

si sup: 8M 10M 12M
Ø2: 800k 1000k 1200k
sirve cualquier oscilador pero se suele usar Ø2

100μs

$f_{max\,señal} = \frac{1}{2} \cdot \frac{1}{100\mu}$

Nyquist

El ADC lo conectamos como queramos

- DTACK (habrá que tener estados de espera (dato tarda 250 ns)
- VPA (nos permitimos la licencia ya que es un dispositivo de acceso esporádico)
  (tarda en convertir 100µs!! como máximo accedemos a el 1 de cada 2000 ciclos)

Este concepto de saber argumentar como conectar:

optimizado temporalmente → DTACK

optimizado en hardware → VPA
- si el DTACK necesita estados de espera
- no hay que hacerlo NUNCA en memorias, sólo dispositivos de raro acceso

Notas: A0, A1 y A2 del ADC están conectados a D0 - D3 del µP
Para elegir canal habrá que hacer escritura del canal que queramos
Podríamos haberlo conectado a A1, A2 y A3 para que cada canal fuera una dirección distinta

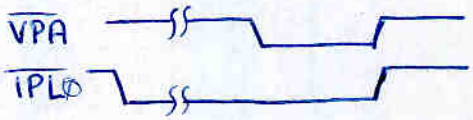
Nota: Escritura del µP → activa START y ALE : D0-D2 indicará canal
Lectura del µP → activa OE

El µP no debe leer hasta que no se haya convertido el dato
i.e. End of Conversion EOC a nivel alto debe interrumpir al µP para decirle que ya puede leer

Nota: el EOC hará un flanco; nosotros queremos que el flanco genere un nivel estable que genere interrupción y que se mantenga hasta que el µP la reconozca.
conversor flanco a nivel → biestable D

Nota: el ciclo al detectar IPL0 hace ciclo de reconocimiento de interrupción.

<u>El circuito está mal</u> ya que al hacer ciclo de reconoc. de interup; en teoría debe activarse VPA (para indicar que no tenemos nº de vector) y luego desactivar IPL

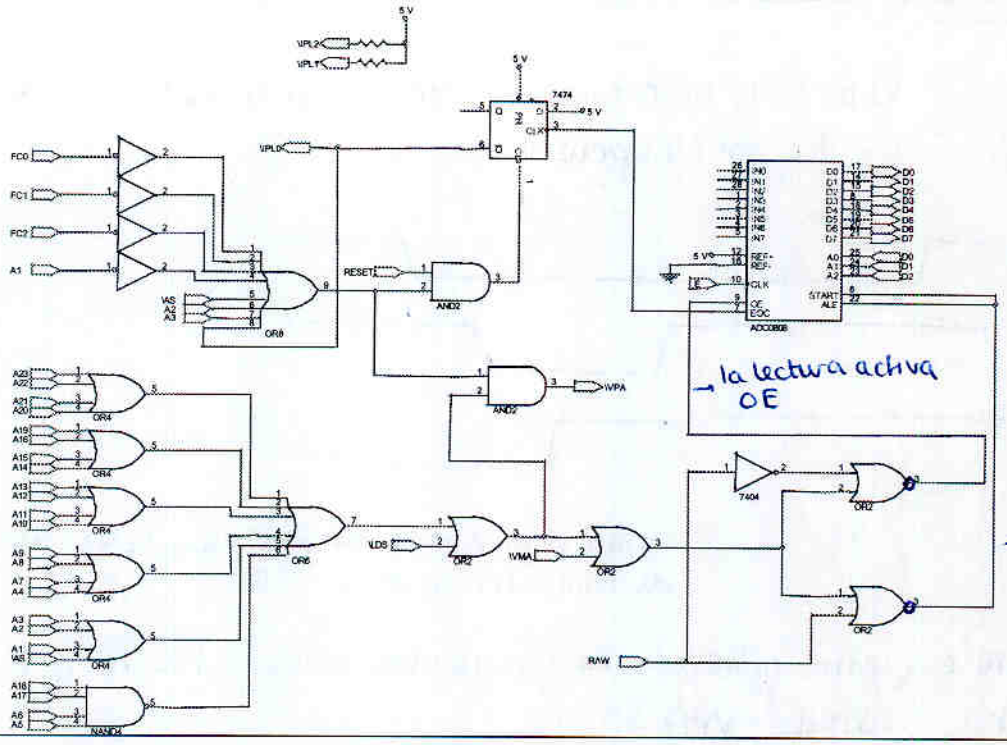


CONDICIÓN DE CARRERA

Pero en el circuito, al ponerse $\overline{IPL0}$, la señal entra en puerta OR que DESACTIVA $\overline{VPA}$. i.e. al activarse $\overline{VPA}$ esta se desactiva

Soluciones:
- cargarme la línea que conecta $\overline{IPL0}$ a la puerta OR que luego genera VPA → la pega es que se generaría VPA aún con interrupciones espúreas
- mejor solución (ver más adelante)

Conexión del ADC0808 al uP. Posición $60061, N.I. 1.



El mecanismo del uPes:

- escribo valor de canal (A0 - A2) en la $ 60061 (se puede conectar perfectamente en posiciones pares)
  (conectados a D0 - D2 del uP → habrá que hacer escritura; i.e. D0-D7)
  ↳ hace además que se active start y ALE
  (la escritura necesitará su DTACK o en este caso VPA)

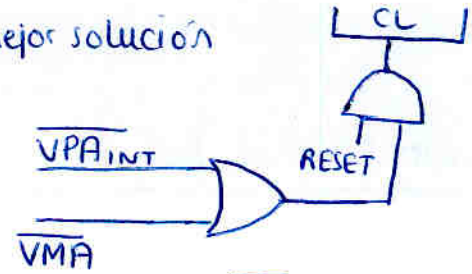
- al cabo de un tiempo hay un flanco en EOC, que con un biestable conversor de flanco a nivel genera el nivel 0 → interrupción
  ↳ activa IPL0
  → activa VPA → (ya que no puede dar nº de vector)
  ↓
  uP se genera su nº vector

- ciclo de reconocimiento de interrupción

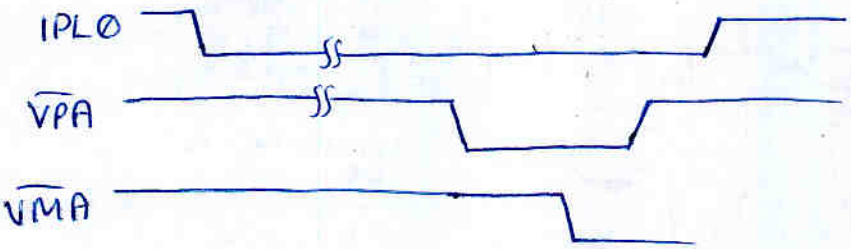
- ciclo de lectura del valor digital
  ↳ activa OE
  y se activa el correspondiente

Hay muchas variaciones : ej: usar DTACK y/o usar numero de vector determinado por el usuario

• mejor solución



$\overline{VMA}$ sólo baja cuando VA ha sido reconocida la interrupción.



VMA baja 2 semiciclos después (caso mejor) de haber reconocido $\overline{VPA}$

Por tanto con esta solución no generamos el clear hasta que el µP no ha 'recibido' $\overline{VPA}$

recuerda que si en un CRI se devuelve $\overline{VPA}$ al µP este se calcula su autovector mientras hace un ciclo de lectura largo sin leer ningún dato, y recuerda que en estos ciclos $\overline{VMA}$ se activa después de $\overline{VPA}$ (un nº entero de ciclos después) y 2 semiciclos antes de subir E

Problemas SED

Problema 5 del libro

- MC 68000 a 10 MHz

- 256 kB EPROM mem programa dir 0
- 256 kB RAM mem datos dir 0
- 2 VIA's (niveles 2 y 3)
- 2 DUART's (niveles 4 y 5)
- ADC 0808 (nivel 1)
- DAC 0800
- Pulsador Abort (nivel 6)

disponemos de RAM 32k x 8
y EPROM 32k x 8

codificadores 3a8 y 8a3

puertas

1 Definición del mapa de memoria:

256kB = 8 x 32kB

D15-D8 Par (UDS)	A0 D7-D0 Impar (LDS)	A18, A17, A16
mem	mem	0 0 0
mem	mem	0 0 1
mem	mem	0 1 0
mem	mem	0 1 1
VIA1	VIA2	1 0 0
	DUART1	1 0 1
	DUART2	1 1 0
DAC	ADC	1 1 1 ←

32k direcciones A1-A15

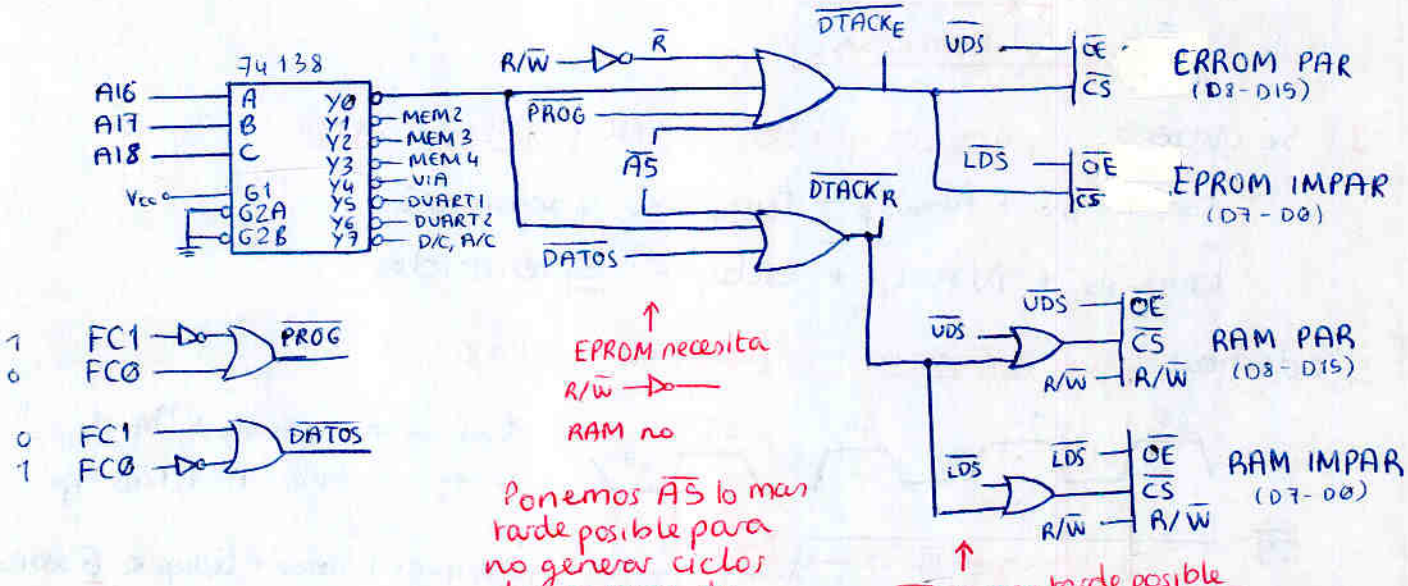
¡cuidado! Se verán afectados por el CRI (A23-A4 = 1)

10 - 1k
11 - 2k
12 - 4k
13 - 8k
14 - 16k
15 - 32k
16 - 64k
17 - 128k
18 - 256k
19 - 512k
20 - 1M

Todos con $\overline{DTACK}$ menos VIA y ADC (ver razones pag 46)

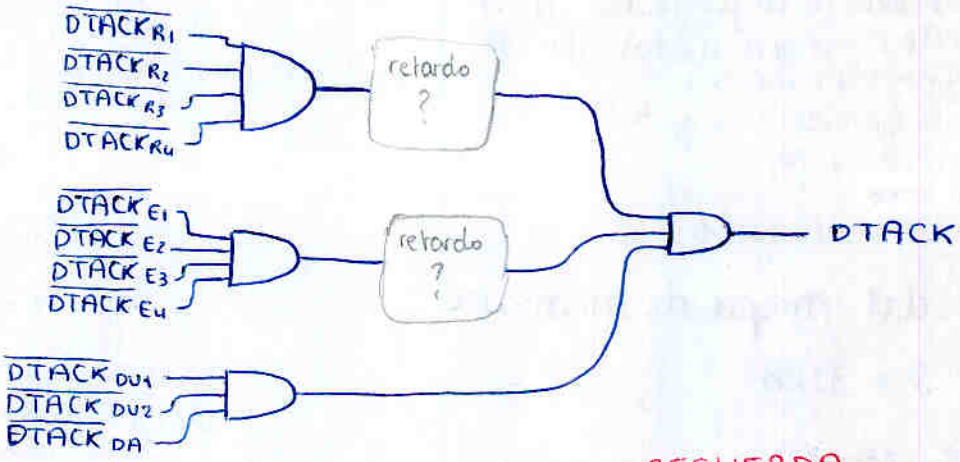
EPROM (programa)
RAM (datos)

2. Circuito decodificación RAM y EPROM



Las entradas de direcciones de las memorias A0 - A14, van conectadas a A1 - A15

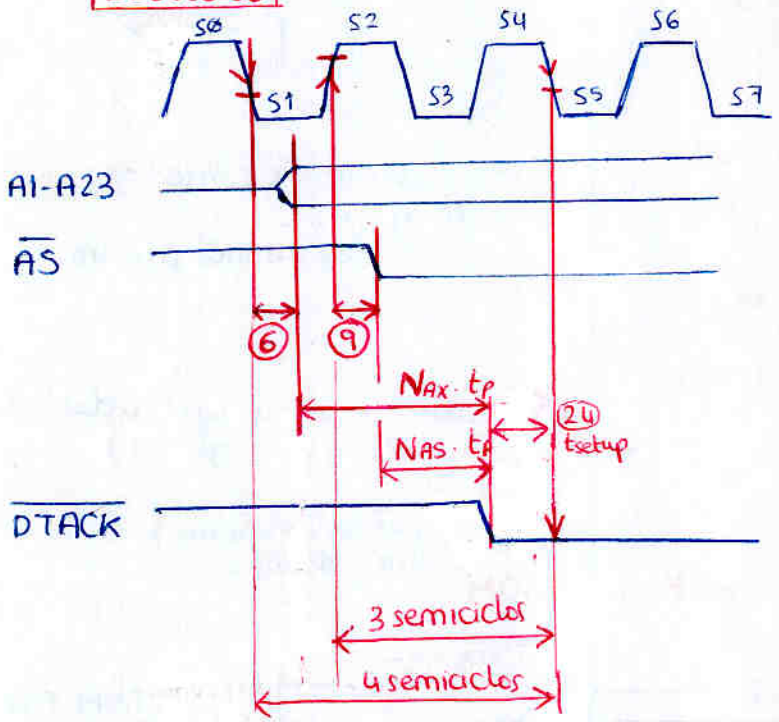
Y los $\overline{DTACK}$ serian



RECUERDA

3. Verificación temporal (hay 3 cosas que verificar, y cada una con $\overline{AS}$ y $\overline{AX}$ TOTAL 6 cosas)

• Lectura



LECTURA
1 → ¿Genera estados de espera?
2 → ¿Le da tiempo a poner datos? i.e. ¿Necesita estados de espera?
ESCRITURA
3 → ¿Se respeta thold al inic CS?

6 = $t_{S0/S1 \to Ax}$

9 = $t_{S1/S2 \to AS}$

$N_{Ax} \cdot t_p$ = retardo entre Ax y generar DTACK

$N_{AS} \cdot t_p$ = retardo entre AS y generar DTACK

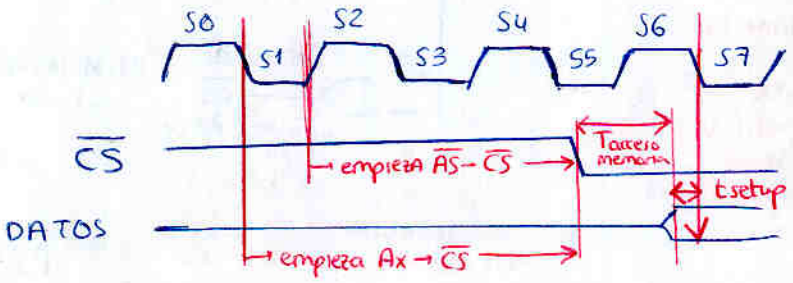
24 = tsetup al muestrear DTACK

1. Se obtiene: para comprobar DTACK al bajar S4/S5

$$t_{S0/S1 \to Ax} + N_{Ax} \cdot t_p + t_{setup} < \underline{4} \text{ semiciclos}$$

$$t_{S1/S2 \to AS} + N_{AS} \cdot t_p + t_{setup} < \underline{3} \text{ semiciclos}$$

2. además para comprobar DATOS en bus al bajar S6/S7



$$t_{S0/S1 - \overline{CS}} = t_{S0/S1 - Ax} + N_{Ax} \cdot t_p$$

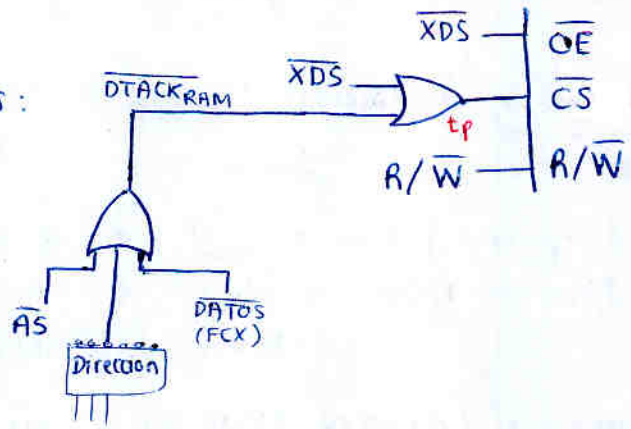
$$t_{S1/S2 - \overline{CS}} = t_{S1/S2 - AS} + N_{AS} \cdot t_p$$

$$t_{S0/S1 - \overline{CS}} + T_{acceso} + t_{setup} < \underline{6} \text{ semiciclos}$$

$$t_{S1/S2 - \overline{CS}} + T_{acceso} + t_{setup} < \underline{5} \text{ semiciclos}$$

• **Escritura**

Si recordamos:

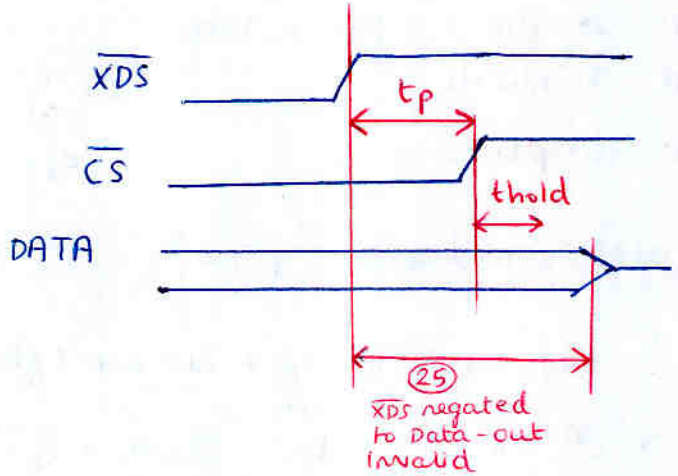


Los datos en bus de datos deben mantenerse en el bus un tsetup antes y un thold despues de desactivarse $\overline{CS}$

↓ se cumple SIEMPRE porque los datos llevan ya mucho tiempo en el bus

↓
- los datos se desactivan un tiempo despues de $\overline{AS}$ y $\overline{XDS}$
- vemos que $\overline{XDS}$ es claramente el que desactivará $\overline{CS}$ mucho antes

(25) = $\overline{AS}$ / $\overline{XDS}$ negated to Data-Out invalid



3.

$$(25) > t_p + t_{hold\ memoria}$$

3. Verificación temporal

Lectura EPROM (¿Genera estados de espera?)

desde Ax: $4 \text{ semiciclos} > t_{S0/S1-Ax} + N_{Ax} \cdot t_p + t_{setup\ DTACK}$

$4 \cdot \left(\frac{1}{2} \cdot \frac{1}{10\,MHz}\right) > (6) + (t_{decod} + 3t_p) + (47)$

$4 \cdot 50\,ns > 50\,ns + (4 \cdot 20\,ns) + 10\,ns$

$200\,ns > 140\,ns$

aqui INCLUIR las 2 puertas que generan DTACK para partir de DTACKE !!!!!

desde AS: $3 \text{ semiciclos} > t_{S1/S2-AS} + N_{AS} \cdot t_p + t_{setup\ DTACK}$

$3 \cdot 50\,ns > (9)(=50\,ns) + 3 \cdot 20\,ns + 10\,ns$

$150\,ns > 120\,ns$

Hemos comprobado que no genera estados de espera

Habrá que comprobar si requiere generarlos

Lectura EPROM ¿requiere estados de espera?)

desde $\overline{AS}$: $\underbrace{t_{S1/S2-AS} + n_{AS} \cdot t_p}_{t_{S1/S2-CS}} + T_{acceso\ EPROM} + t_{setup\ DATA} < 5$ semiciclos

↖ del enunciado

$$9 + 1 \cdot t_p + 120\,ns + 27 < 5 \cdot 50\,ns$$
$$50\,ns + 20\,ns + 120\,ns + 10\,ns < 250\,ns$$
$$200\,ns < 250\,ns$$

desde $\overline{AX}$: también lo cumple (dispone de un semiciclo más (50ns) y sólo una puerta más lo retrasa 20ns)

Lectura RAM (¿Genera estados de espera?)

desde $\overline{AS}$: $4 \text{ semiciclos} > \underset{t_{S1/S2-AS}}{9} + N_{AS} \cdot t_p + \underset{47}{t_{setup\ DTACK}}$

$$4 \cdot 50\,ns > 50\,ns + 3 \cdot 20\,ns + 10\,ns$$
$$200\,ns > 120\,ns$$

Desde $\overline{AX}$: también lo cumplirá

⇒ No genera estados de espera, ¿pero los requiere?

Lectura RAM ¿Requiere estados de espera?

desde $\overline{AS}$: $5 \text{ semiciclos} > t_{S1/S2-AS} + n_{AS} \cdot t_p + T_{acceso\ RAM} + t_{setup\ DATA}$

$$5 \cdot 50\,ns > 9 = 50\,ns + 2 \cdot \overset{20ns}{t_p} + 300\,ns + 27 = 10\,ns$$

↑ en la RAM hay una puerta mas entre AS y CS (la de XDS)

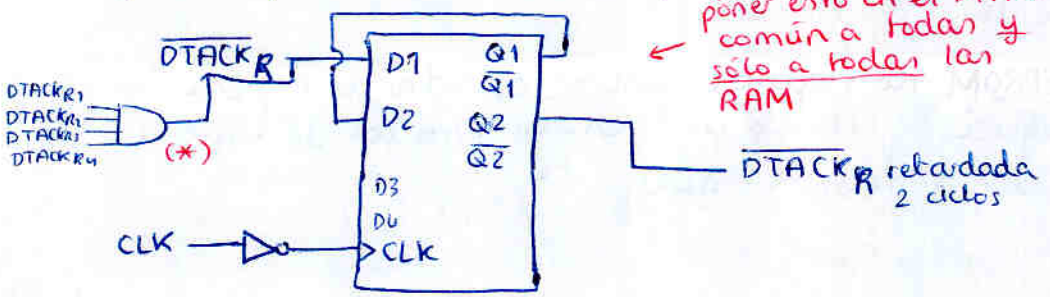
↖ del enunciado

$$250\,ns > 400\,ns$$

No lo cumple! La RAM requiere estados de espera

$$250 + 100 \cdot N^{o}_{estados\ espera} > 400\,ns \Rightarrow N^{o}_{estados\ espera} = 2$$

Habrá que añadir 2 flip flops D a partir del flanco de bajada S3 (flanco bajado → invertir CLK)



(*) poner esto en el DTACK común a todas y sólo a todas las RAM

Escritura RAM

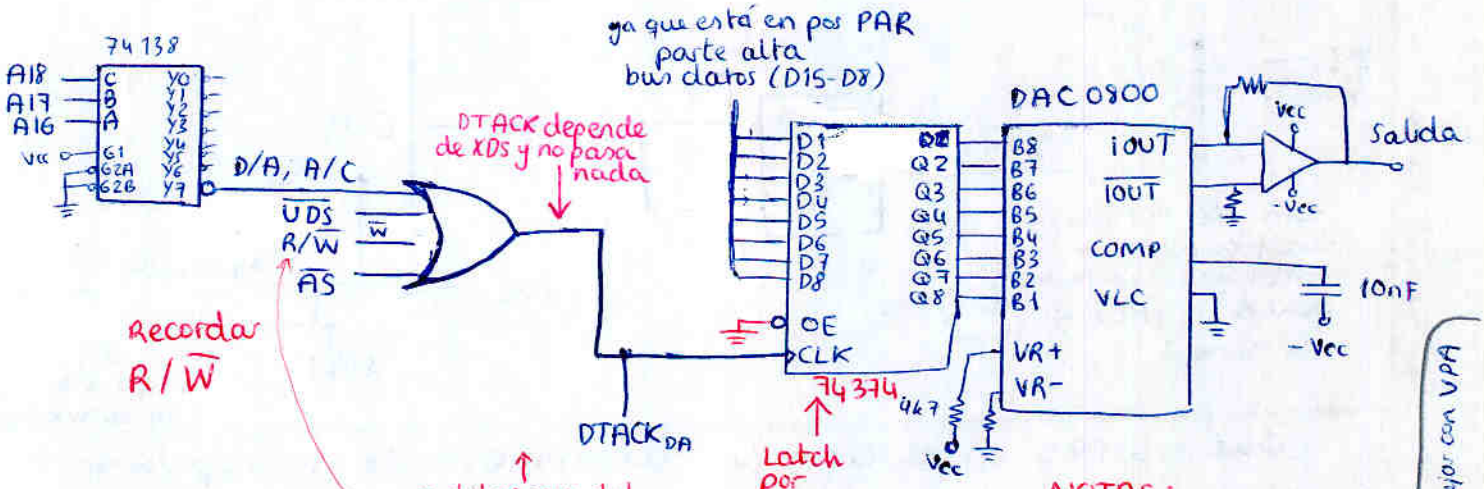
$t_{XDS-Data\ Not\ Valid} > t_p + t_{hold\ DATA\ RAM}$ ↑ del enunciado

$$25 > t_p + 5\,ns$$
$$30\,ns > 20\,ns + 5\,ns$$

SE CUMPLE

4. Conexión de los convertidores A/D y D/A al µP

- D/A: funciona con DTACK ya que usa un LATCH



Recordar $R/\overline{W}$

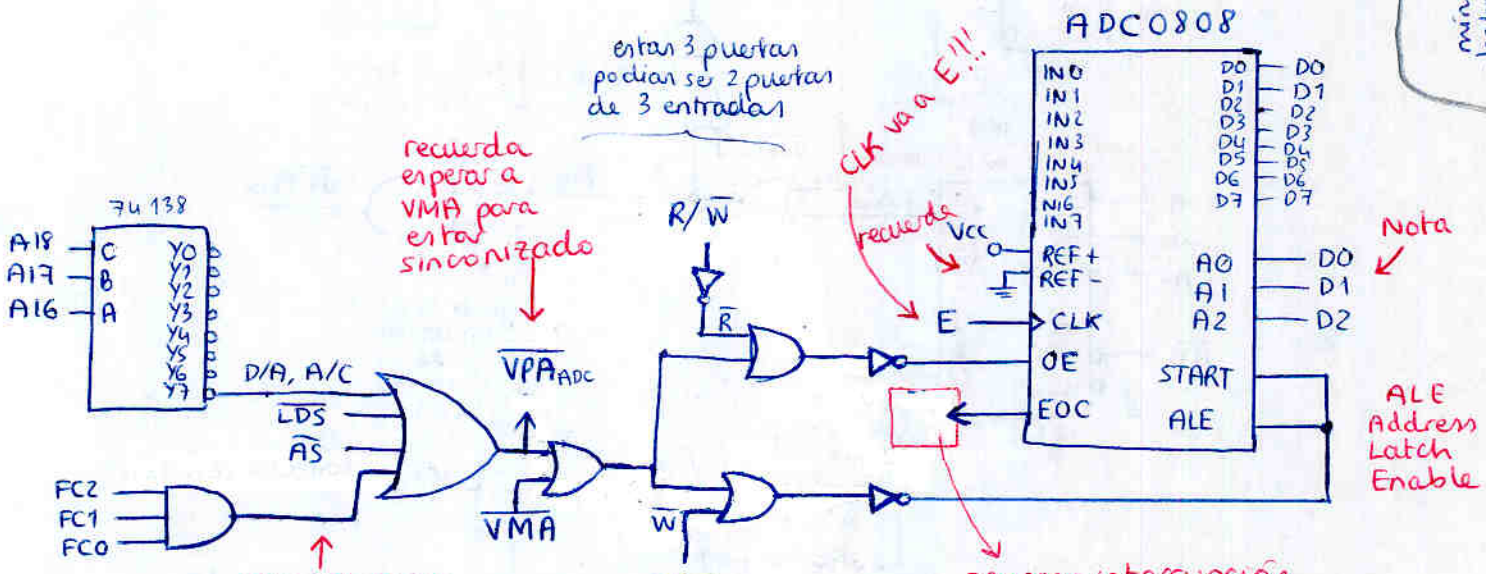
a diferencia del A/C, éste sólo se usa como escritura. No tendrá problemas en el "pseudociclo" de lectura del ICR (que recordemos direcciona A23-A4 = 1 es decir, que afecta a las posiciones D/A, A/C)

Latch por flanco con CLK directo a DTACK

NOTAS:

- B8 es LSB! B1 es MSB!
- VR+ a Vcc y VR- a GND (con 4k7)
- VLC a GND
  COMP a -Vcc con ⫫ 10nF
- Recordar conexión del A0 a la salida

- A/D funciona con VPA — para evitar problemas de estados de espera. En algunos exámenes piden máxima eficiencia y hay que usar DTACK y hacer el estudio temporal → en ocasiones piden mínima hardware: habrá que ver si con DTACK requiere espera, si no requiere, será mejor con DTACK, si requiere, mejor con VPA
  - recuerda, al generar $\overline{VPA}$ debe esperarse a que se active VMA



estas 3 puertas podrían ser 2 puertas de 3 entradas

recuerda esperar a VMA para estar sincronizado

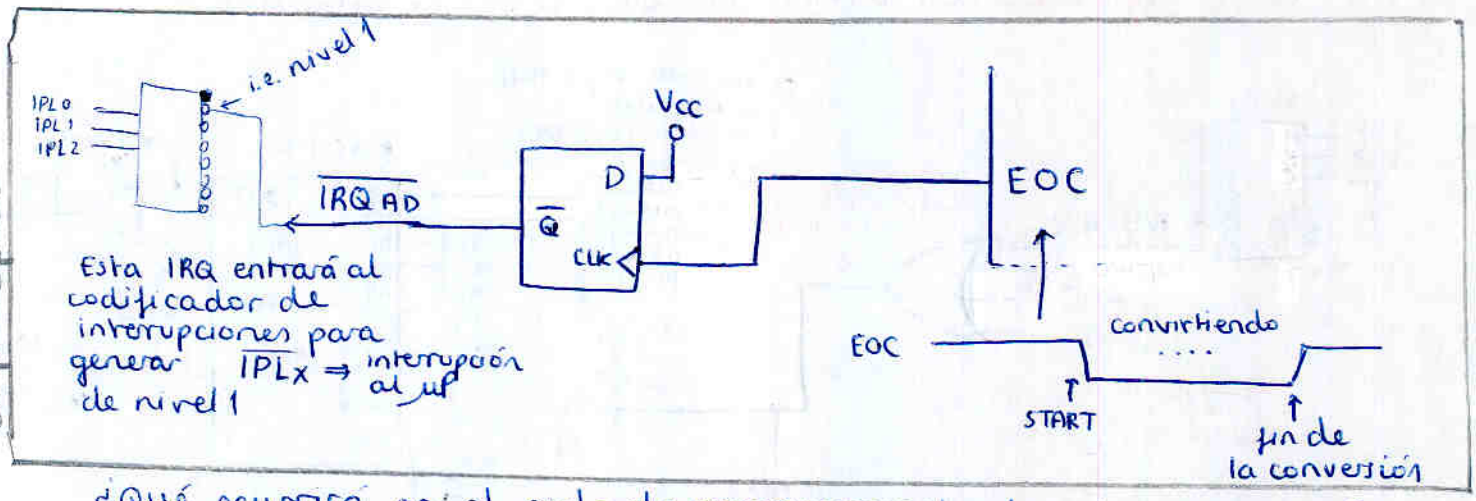
para asegurarnos que no se active en el ICR (se activará cuando FCX ≠ 111)

según leamos o escribamos hacemos una cosa u otra (por eso D0, D1, D2 y A0, A1, A2 del ADC están conectados a D0, D1, D2 del µP a la vez)

generar interrupción nivel 1 a partir del "flanco" de EOC — START — ya ha acabado convirtiendo

Interrupción del A/D

(ver pag III-6 para recordar el funcionamiento de las interrupciones)

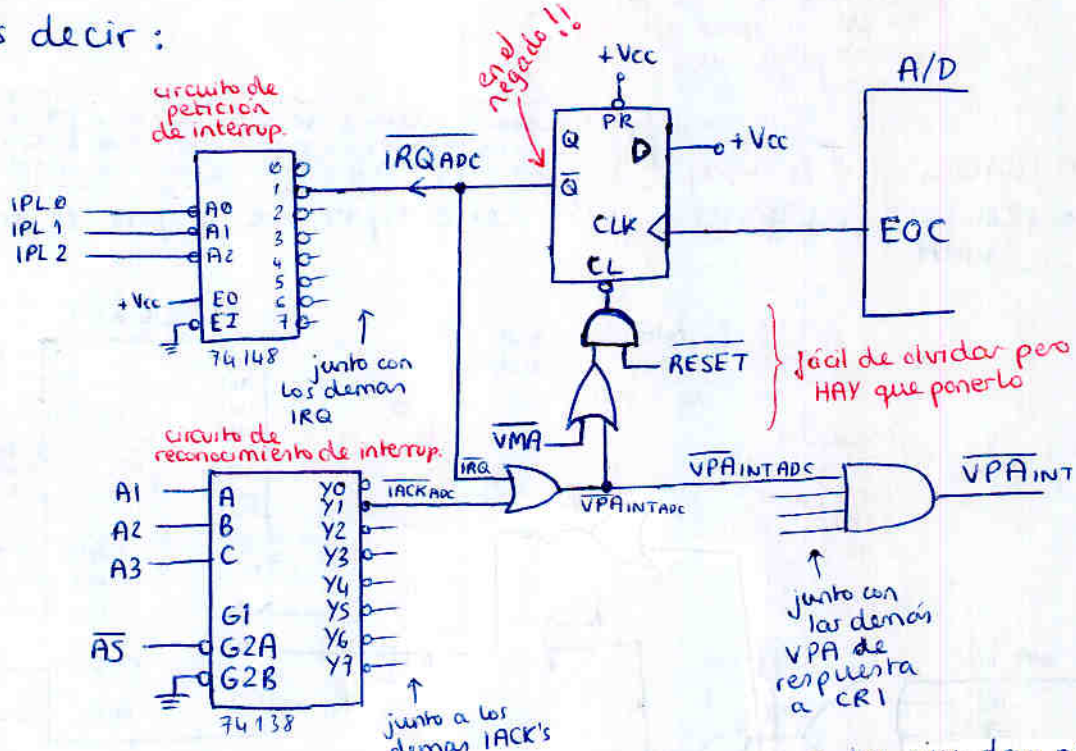


¿Qué ocurrirá en el ciclo de reconocimiento de interrupciones?

FCX = 1
A1 - A3 = nivel
$\overline{AS}$ = 0
A4 - A23 = 1

→ $\overline{IRQ_{AD}}$ → Generar $VPA_{INT\,ADC}$ para q el µP sepa que debe generarse él mismo su nº vector como nivel int + 24 → El µP continuará su ciclo de lectura largo (activando VMA 2 ciclos antes de subir E) mientras se genera su nº de vector → que esto resetee $\overline{IRQ_{AD}}$

Es decir:



- alguno directamente a las IACK del dispositivo } los que dan nº de vector
- otros se juntan con el correspondiente IRQ: para generar el VPA_{INT} (y en casos como éste generar el reset de IRQ) } los autovectorizados

⇓

Utilizar VMA para ello, para evitar condición de carrera.

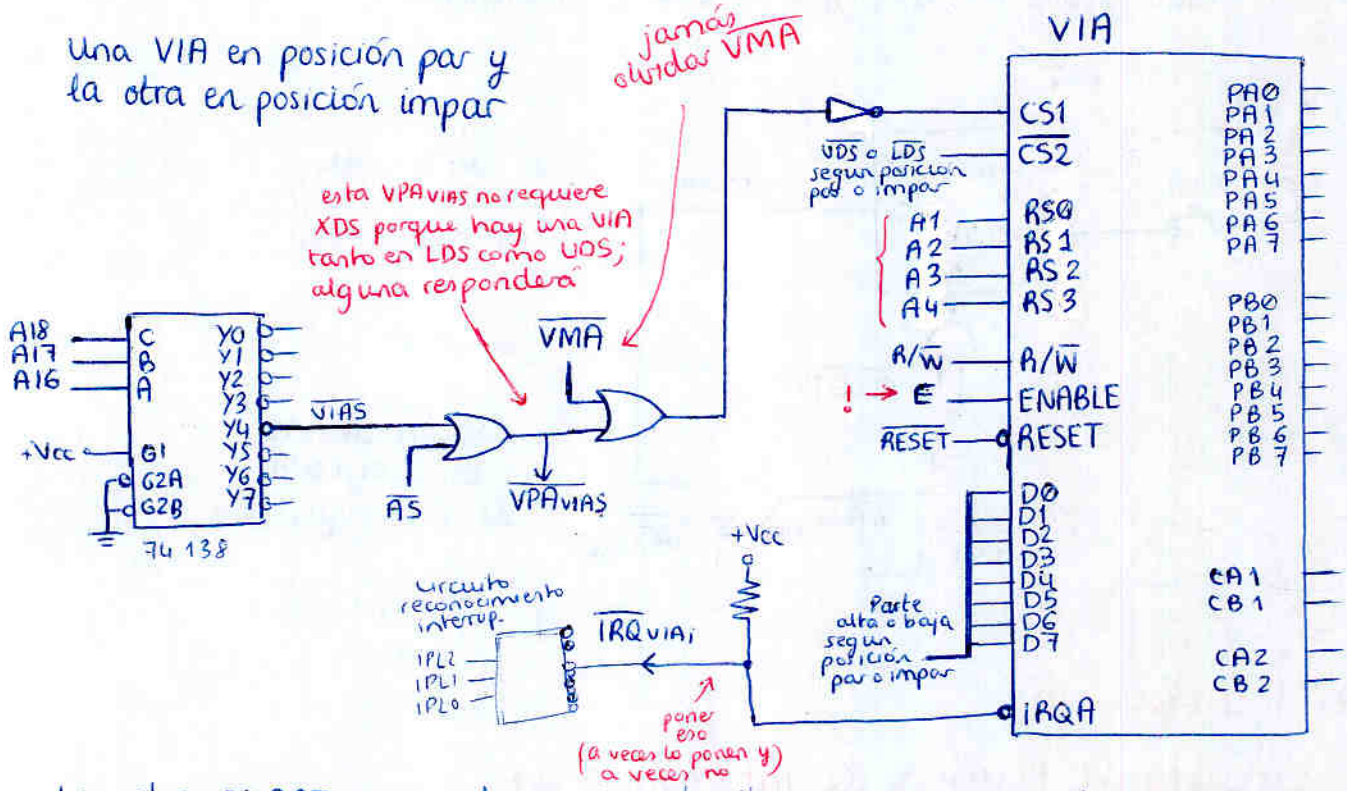
Incluimos IRQ para evitar responder a una interrupción espúrea.

5. Conexión de VIAs y DUARTs

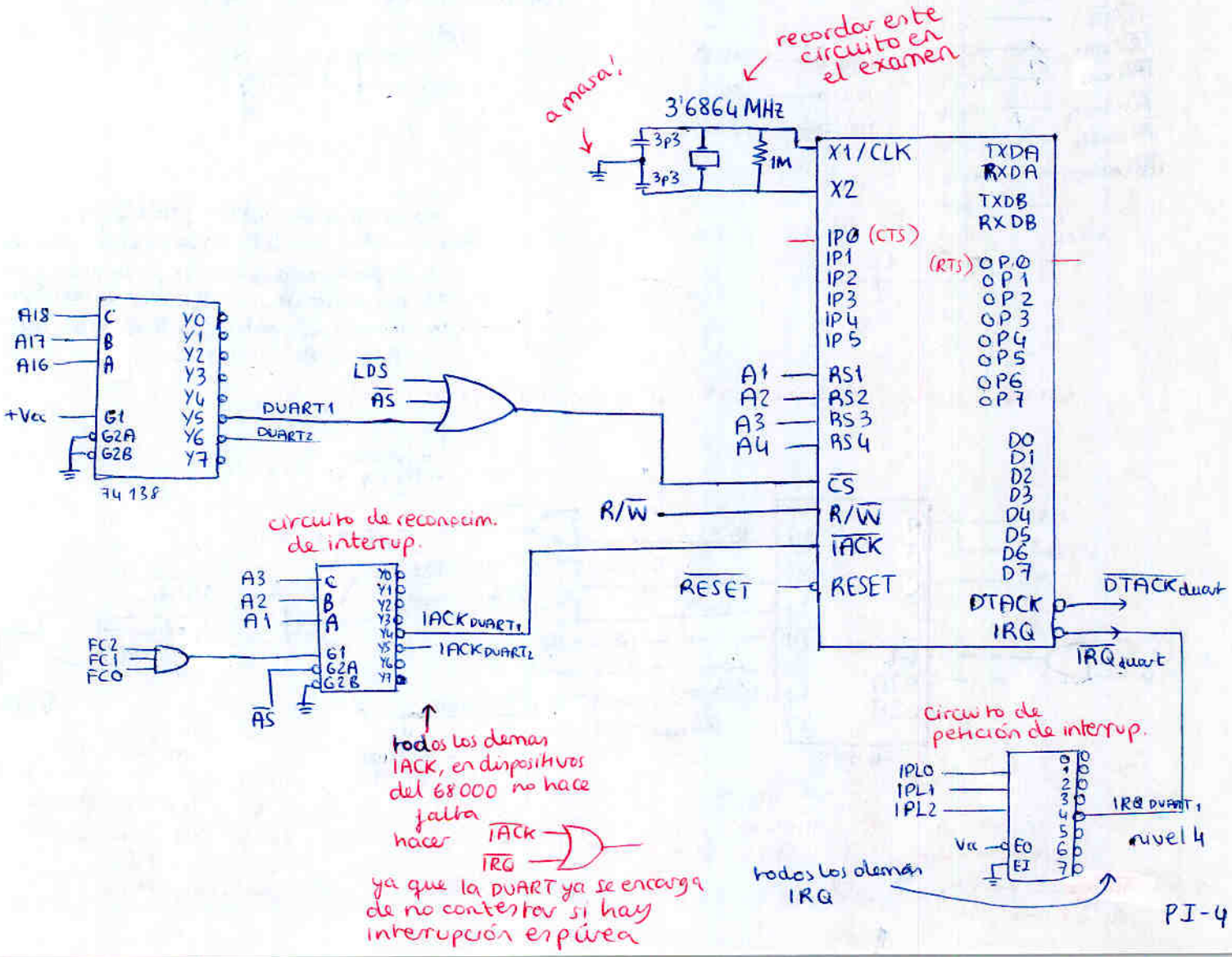
Una VIA en posición par y la otra en posición impar

jamás olvidar VMA

esta VPAVIAS no requiere XDS porque hay una VIA tanto en LDS como UDS; alguna responderá

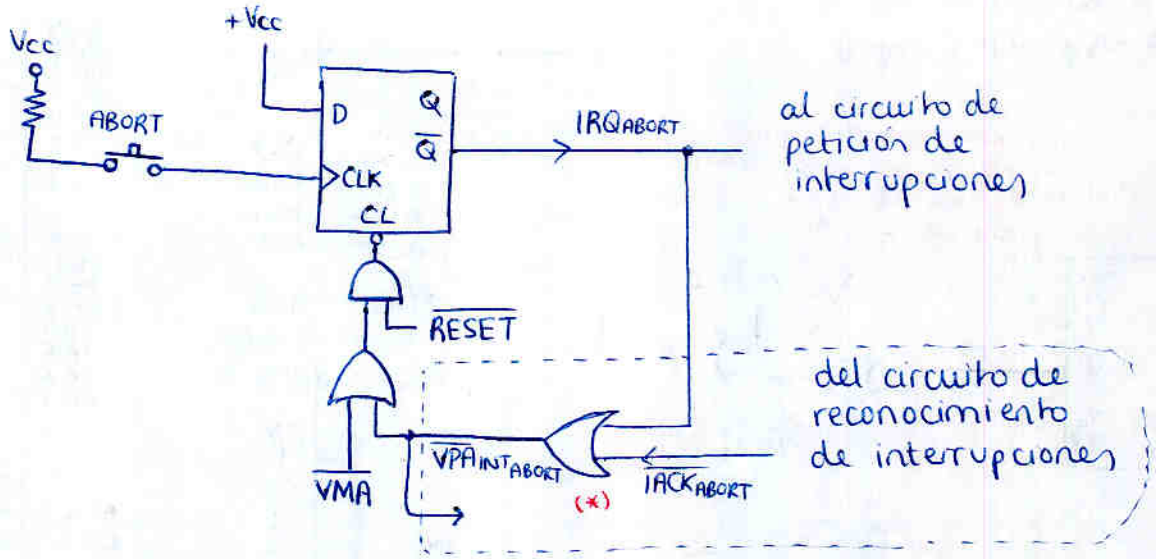


Las dos DUART se conectaran igual sólo que una saliendo de Y5 y la otra saliendo de Y6



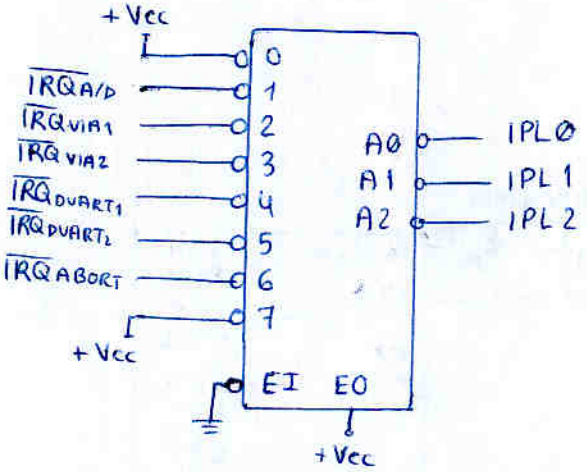
6. Pulsador ABORT

(muy similar al A/C)(la parte de la interrupción)

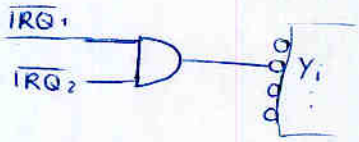


7. Interrupciones

Circuito de Petición de Interrupciones



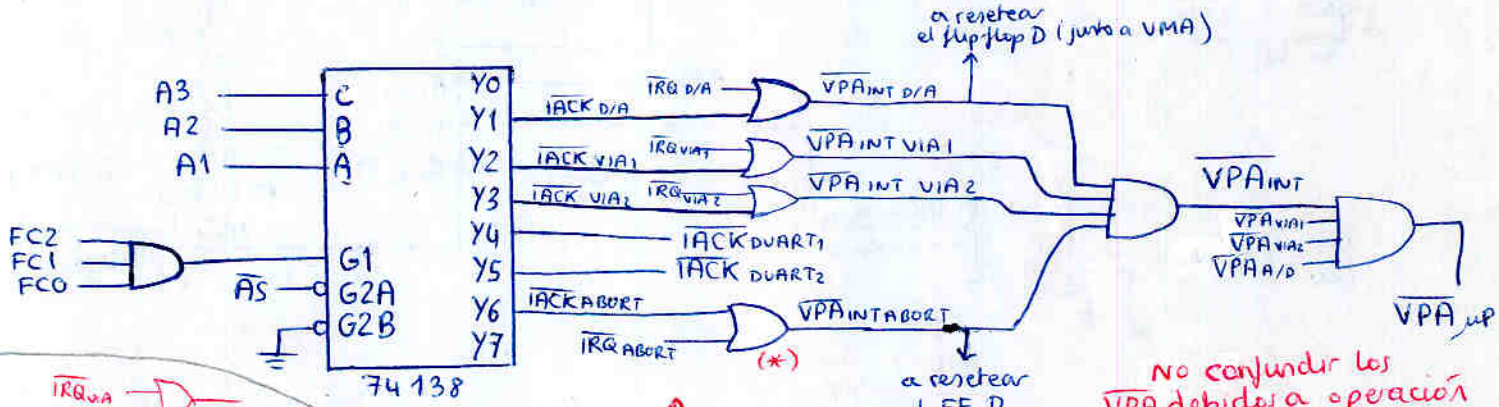
si 2 dispositivos compartieran el mismo nivel, habria que hacer

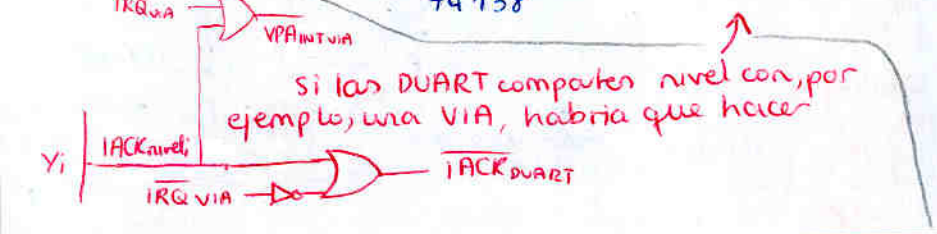


en el caso de interrupción espúrea debe haber un $\overline{BERR}$ (que nadie conteste)

- Los dispositivos del 68000 ya se encargan de no contestar DTACK si no interrumpieron
- Los demas dispositivos deberan tener un OR de IACK con IRQ

Circuito de reconocimiento de interrupciones





si las DUART compartes nivel con, por ejemplo, una VIA, habria que hacer

No confundir los $\overline{VPA}$ debidos a operación $R/\overline{W}$ y los debidos a dispositivo autovectorizado en el CRI (llamamos $\overline{VPA}_{INT}$)

Problema 8 del libro

68000 a 10 MHz

- 4 MB EPROM en dir $0 (partillas de 1MB)
- 8 MB RAM memoria datos supervisor
  8 MB RAM memoria datos usuario } al final del mapa (partillas de 1MB)
- VIA (interrupciones nivel 3)
- Convertidor A/D ADC0808 (interrupciones nivel 4)
- DUART (interrupciones nivel 5)
- Pulsador ABORT (interrupción nivel 6)
- Registro de escritura en la ultima posicion del mapa de memoria

Implementación:

→ mínimo numero de elementos utilizados
— máxima velocidad
en ese orden.

$t_p = 21\,ns$

$T_{acceso\,RAM} = 265\,ns$

$T_{acceso\,EPROM} = 120\,ns$

$t_{hold\ datos\ memoria} = 5\,ns$

1. Mapa de memoria: Que dispositivos usan $\overline{DTACK}$ y cuales $\overline{VPA}$

4 MB EPROM =

Programa Supervisor (110) (o tb programa usuario 010)

UDS	LDS
1M×8	1M×8
1M×8	1M×8

	A21	A20 — A1
	000	000 ... 0
		111 ... 1
	001	000 ... 0
		111 ... 1

10 - 1k
11 - 2k
12 - 4k
13 - 8k
14 - 16k
15 - 32k
16 - 64k
17 - 128k
18 - 256k
19 - 512k
20 - 1M
21 - 2M
22 - 4M
23 - 8M
24 - 16M

16 MB RAM

Datos Supervisor (101)		Datos Usuario (001)		A23	A22	A21
1M RAM 5	1M	1M RAM 1	1M	1	0	0
1M RAM 6	1M	1M RAM 2	1M	1	0	1
1M RAM 7	1M	1M RAM 3	1M	1	1	0
1M RAM 8	1M	1M RAM 4	1M	1	1	1

la ultima posición tendrá un latch →

VIA, A/D y DUART (DUART siempre impar)

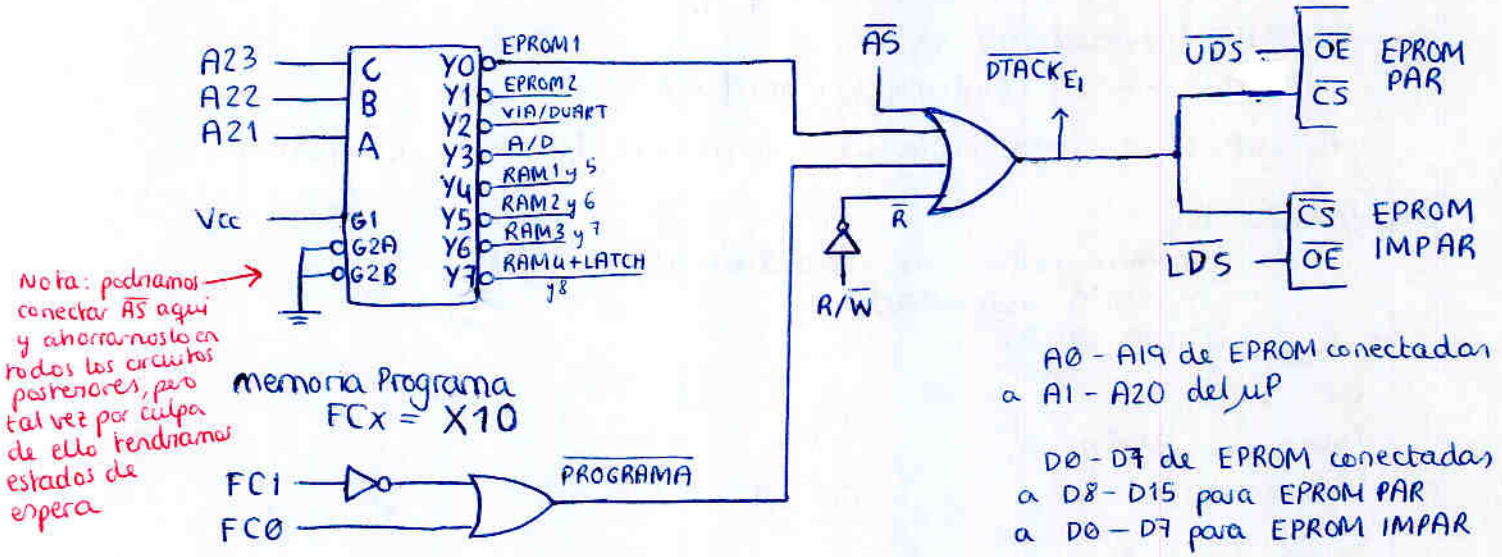
en PS, PU, DS, DU

		A23	A22	A21
VIA	DUART	0	1	0
	A/D	0	1	1

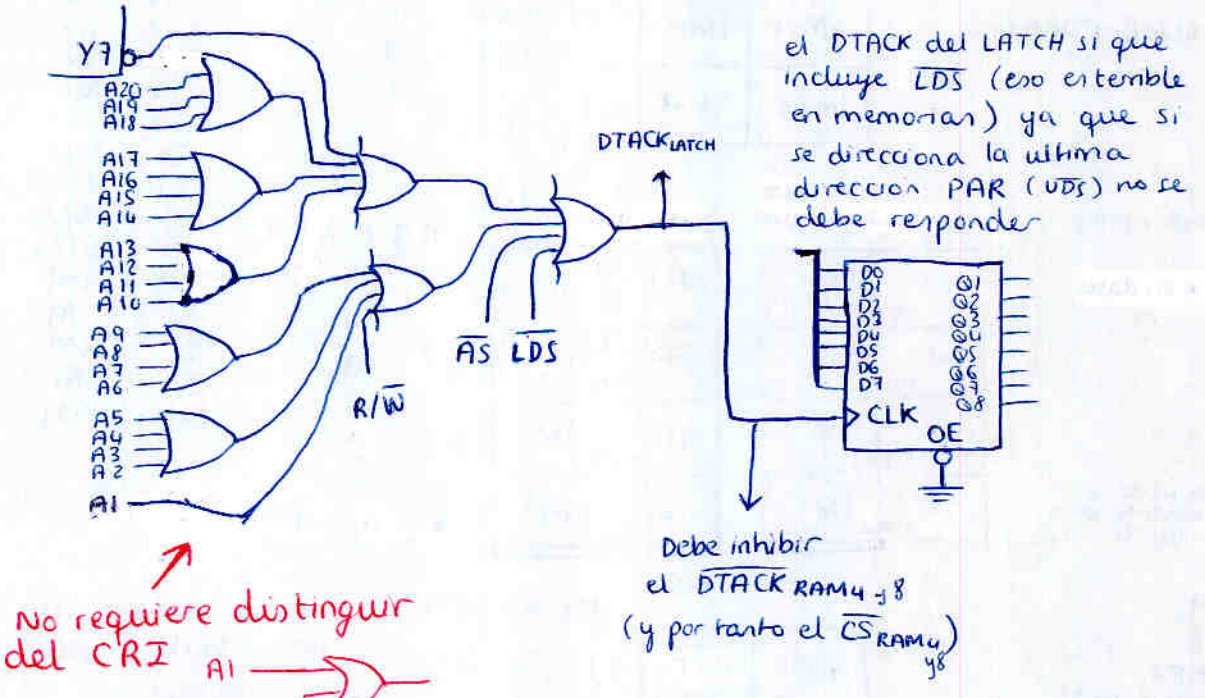
Haciendo esto nos ha cabido perfectamente en 8 bloques
⇒ ¡convertidor 3 a 8!

- La RAM y la EPROM responden con $\overline{DTACK}$ (acceso continuo y común)
- La VIA con $\overline{VPA}$ (por ser periférico del 6800)
- convertidor A/D (da igual en principio; habrá que comprobar con que solución emplea menos hardware)
- La DUART con $\overline{DTACK}$ (en familia del 68000)

2. Diseño del circuito de decodificación y control de la EPROM (ubicada como memoria de programa)



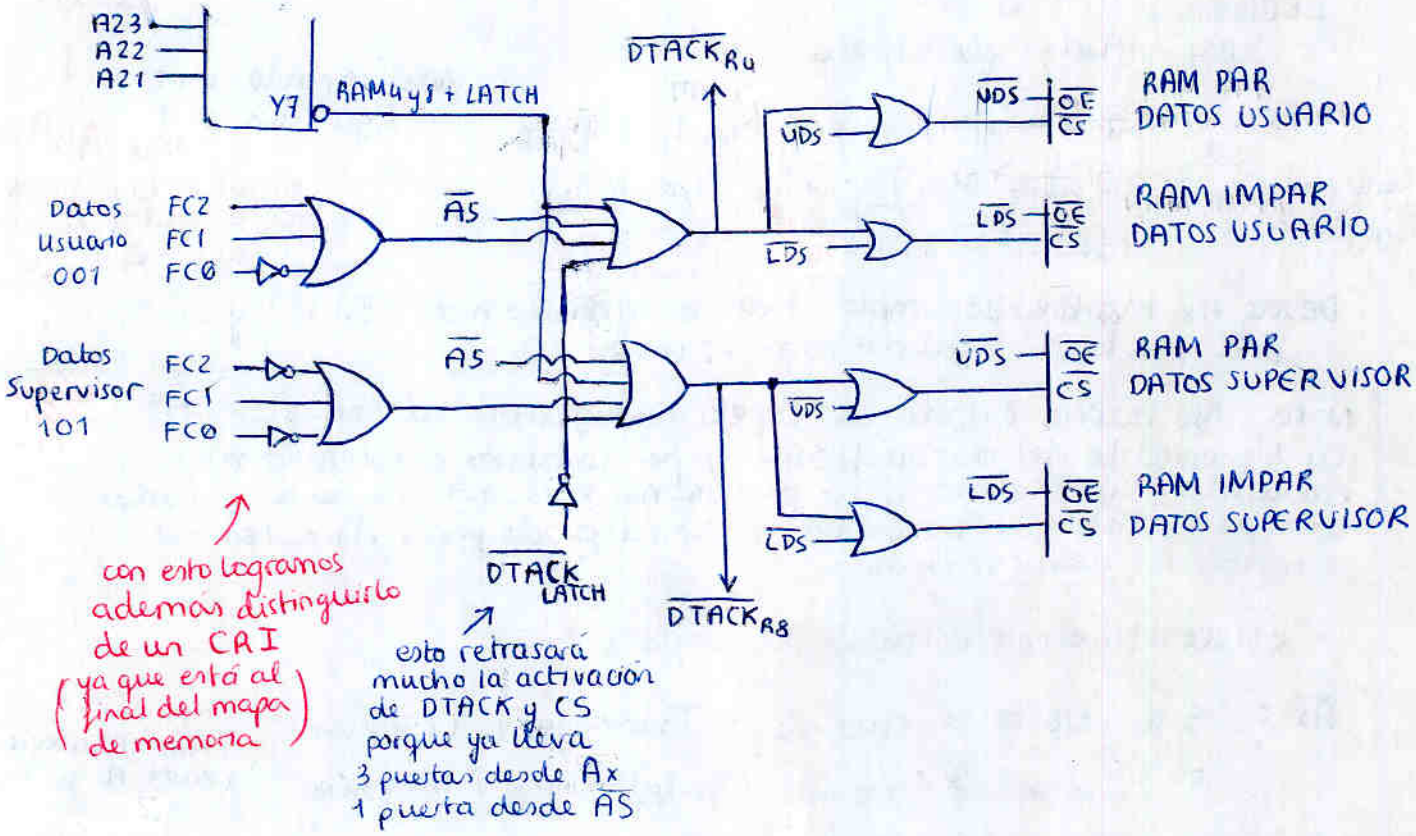
3. Diseño del circuito de codificación y control del registro
   - utilizaremos LATCH 74374 (utilizará flanco de subida de CS)
   - deberá inhibir al $\overline{CS}$ de la RAM 4



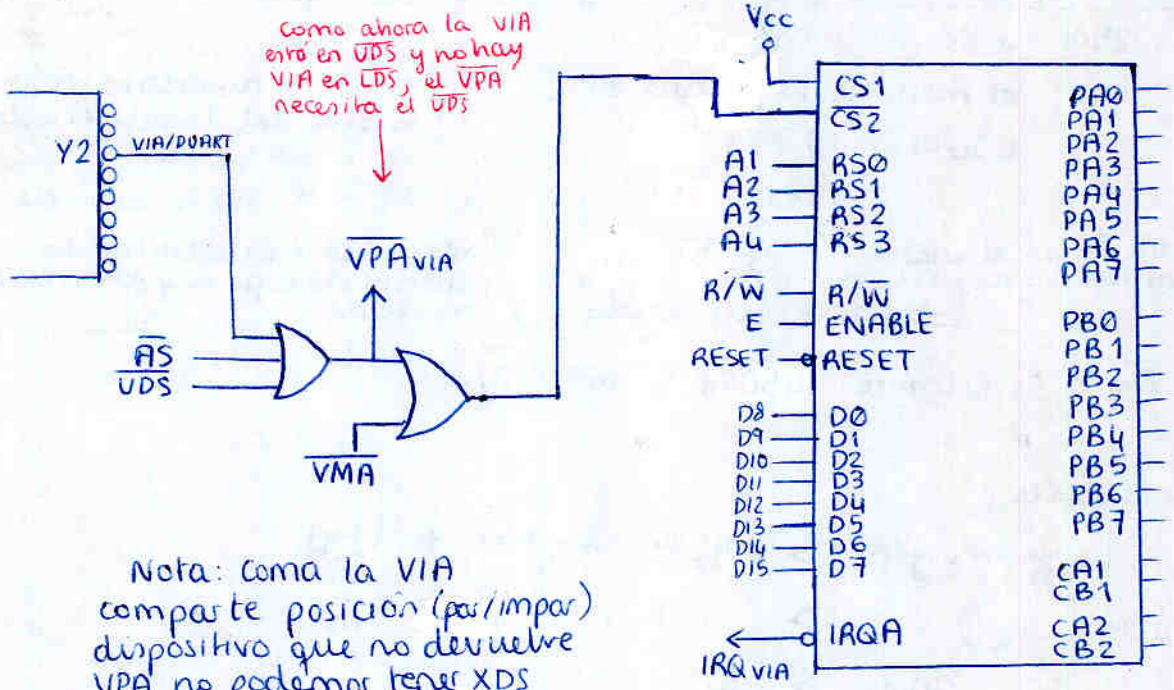
No requiere distinguir del CRI

porque el CRI es ciclo de lectura, y este registro sólo funciona en escritura

4. Diseño del circuito de decodificación y control de los 2MB ultimos de la RAM (tanto datos de usuario como de supervisor)



5. Diseño del circuito de decodificación y conexión de todas las patillas de la VIA



Nota: Como la VIA comparte posición (par/impar) dispositivo que no devuelve VPA no podemos tener XDS directamente en CS1 y hacer un VPA común (como hacemos en el problema 5) con dos VIA's

Aquí nos vemos obligados a incluir UDS en el VPA

6. Verificación temporal en lectura y escritura de memoria RAM (suponer que no existe el registro por simplicidad)

- Lectura:
  - ¿Hay estados de espera?

$\overline{AS}$: 3 semiciclos > $t_{S1/S2-\overline{AS}} + N_{AS} \cdot t_p + t_{setup\,DTACK}$

contarlos en el cronograma

$3 \cdot \left(\frac{1}{2} \cdot \frac{1}{10MHz}\right) > 9_{ns} + 1 \cdot 21ns + 47$

$150\ ns > \cancel{81}\ ns$ 3ª 123 ns (144 ns si $\overline{AS}$ en enable de decod.)

N_{AS} = nº puertas entre $\overline{AS}$ y $\overline{DTACK}$ = 1

ver circuito pag siguiente

¡¡¡¡FAAALSO!!!!

Incluir todas las AND que unen los DTACK!!!!!

N_{AS} = 3

optimizar ese circuito

Desde Ax tambien se cumple: tiene un semiciclo más (50 ns) y sólo una puerta de retardo más (21 ns)

Nota: No habría estados de espera ni siquiera al conectar $\overline{AS}$ en los enable del decodificador; por lo tanto ahorraríamos cableado haciéndolo y no perderíamos hardware generando estados de espera. Faltaría comprobar si por culpa de poner $\overline{AS}$ antes no estarían los datos a tiempo.

- ¿Necesitaremos estados de espera?

$\overline{AS}$: 5 semiciclos > $t_{S1/S2-\overline{CS}} + T_{acceso\,RAM} + t_{setup\,DATOS}$

$5 \cdot \left(\frac{1}{2} \cdot \frac{1}{10MHz}\right) > (t_{S1/S2-\overline{AS}} + N_{\overline{AS}} \cdot t_p) + T_{acceso\,RAM} + t_{setup\,DATOS}$

$250\ ns > (9 + 2 \cdot 21ns) + 265\ ns + 27$

$250\ ns > \cancel{367}\ ns$

(N_{AS}: nº niveles entre AS y CS) = 2

¡Aquí es falso que sea falso! ¡cuidado no corregir demasiado seguido!! que ya estaba yo sumando 3

No se cumple! Habrá que insertar N estados de espera

$250 + N \cdot 100 > \cancel{367}\ ns$

el mínimo N posible es 2

$450\ ns > \cancel{367}\ ns$

$> \cancel{388}\ ns$

poniendo $\overline{AS}$ en el enable del decodificador no perdemos nada
↳ Habría que ponerlo

Nota: Si tuviéramos $\overline{AS}$ en el enable del decodificador sería un t_p más, es decir 388 en lugar de 367

seguiríamos necesitando 2 estados de espera y ahorraríamos cableado.

Desde $\overline{Ax}$: se cumple 500 ns > 367 + 21 ns

- Escritura:

$t_{desactivar\,XDS \to quitar\,datos} > t_{desactivar\,XDS \to desactivar\,CS} + t_{hold\,datos\,RAM}$

$25 > 1 \cdot t_p + 5\ ns$

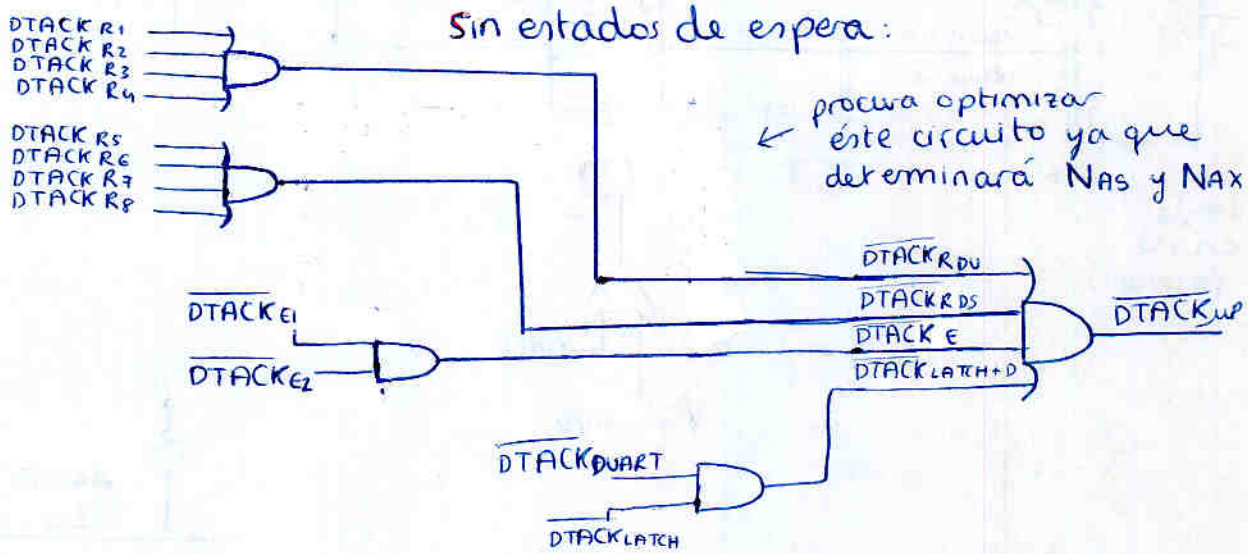
$30\ ns > 26\ ns$

Se cumplen los tiempos de escritura

Nota: no piden el calculo en la EPROM, pero si que se cumple, incluso poniendo $\overline{AS}$ en la entrada ENABLE del decodificador.

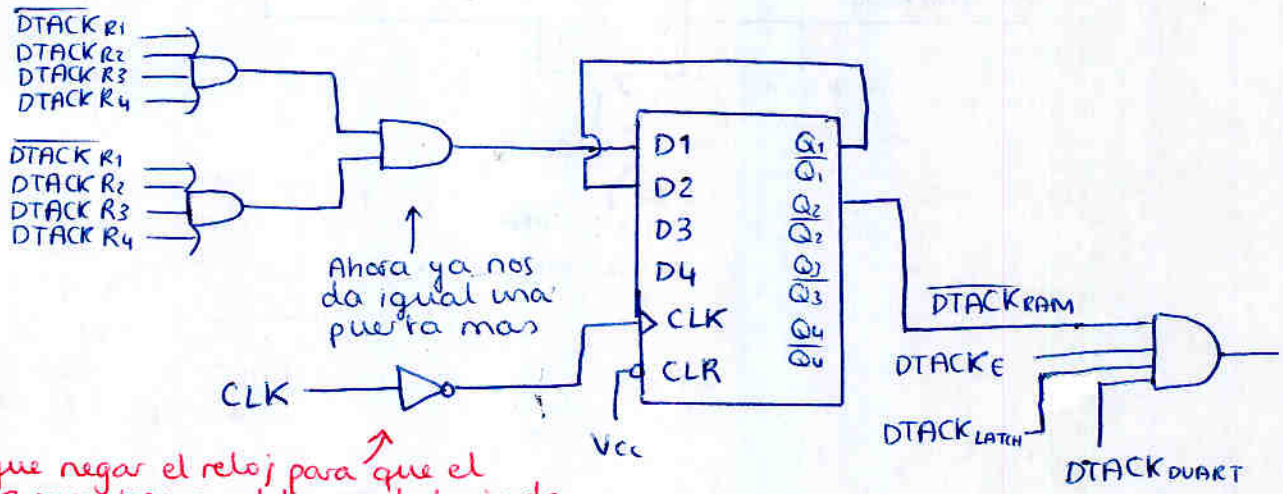
Para implementar esos 2 estados de espera utilizamos 2 flip-flop D que funcionen con el flanco de BAJADA del reloj y que afecten unicamente al DTACK común de la RAM

Sin estados de espera:



procura optimizar éste circuito ya que determinará NAS y NAX

Generar estados de espera a DTACK común de la RAM



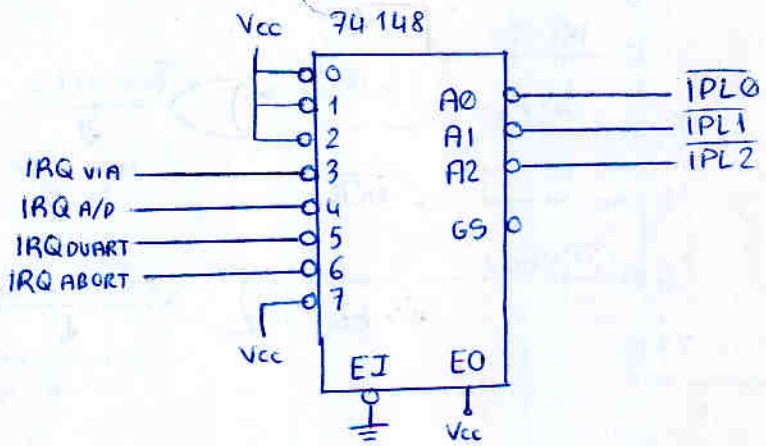
Ahora ya nos da igual una puerta mas

Hay que negar el reloj para que el flip-flop muestree en el flanco de bajada, a la vez que el uP muestrea DTACK

7. Circuito de petición y reconocimiento de interrupciones así como ecuación resultante de $\overline{DTACK}$ y $\overline{VPA}$

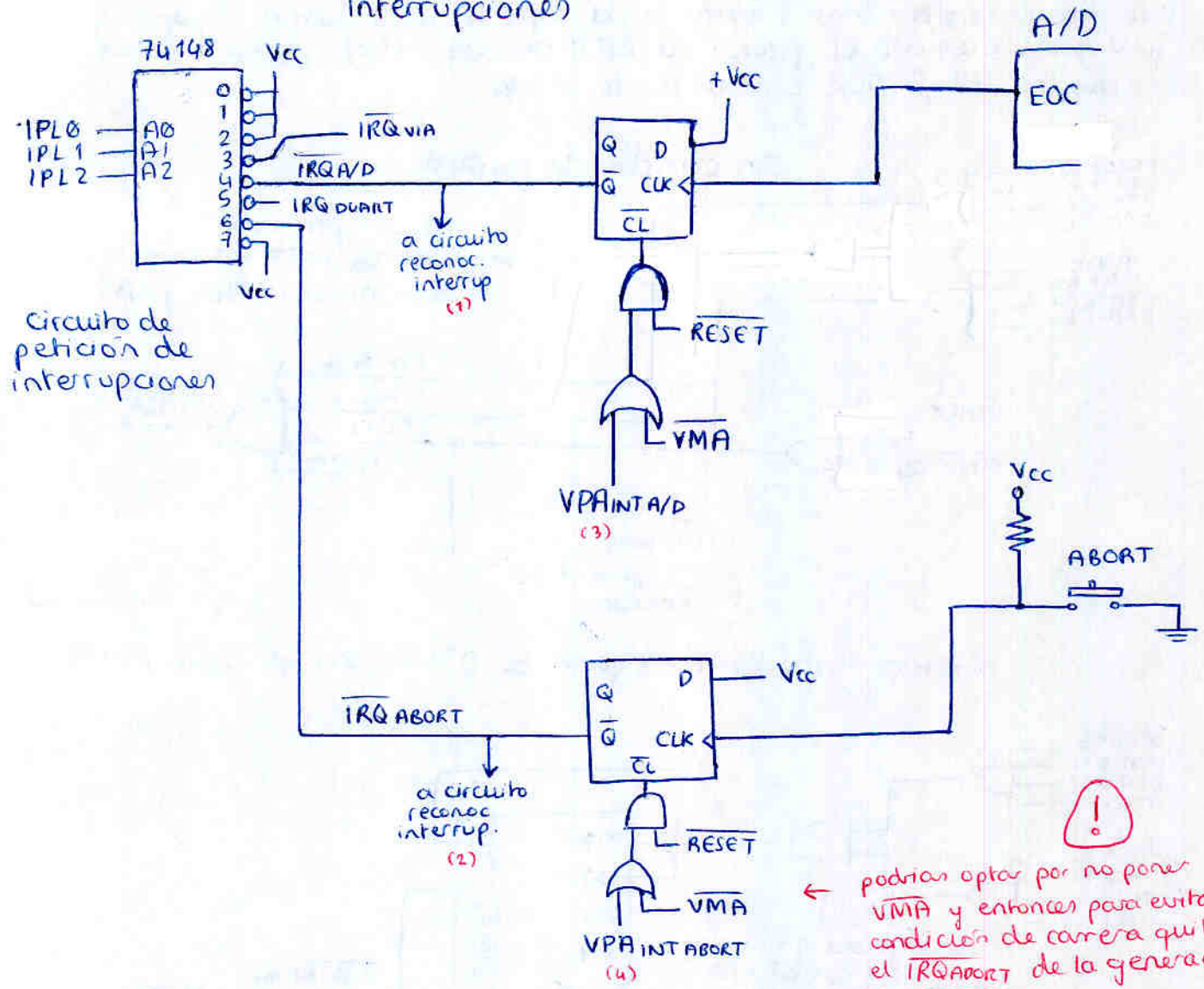
Circuito de petición de interupciones:

Donde IRQ A/D y IRQ ABORT requieren un convertidor flanco a nivel que se borre en el final del CRI



$$\overline{DTACK} = \overline{DTACK_{R1}} + \overline{DTACK_{R2}} + \overline{DTACK_{R3}} + \overline{DTACK_{R4}} + \overline{DTACK_{R5}} + \overline{DTACK_{R6}} + \overline{DTACK_{R7}} + \overline{DTACK_{R8}} + \overline{DTACK_{DUART}} + \overline{DTACK_{LATCH}} + \overline{DTACK_{E1}} + \overline{DTACK_{E2}}$$

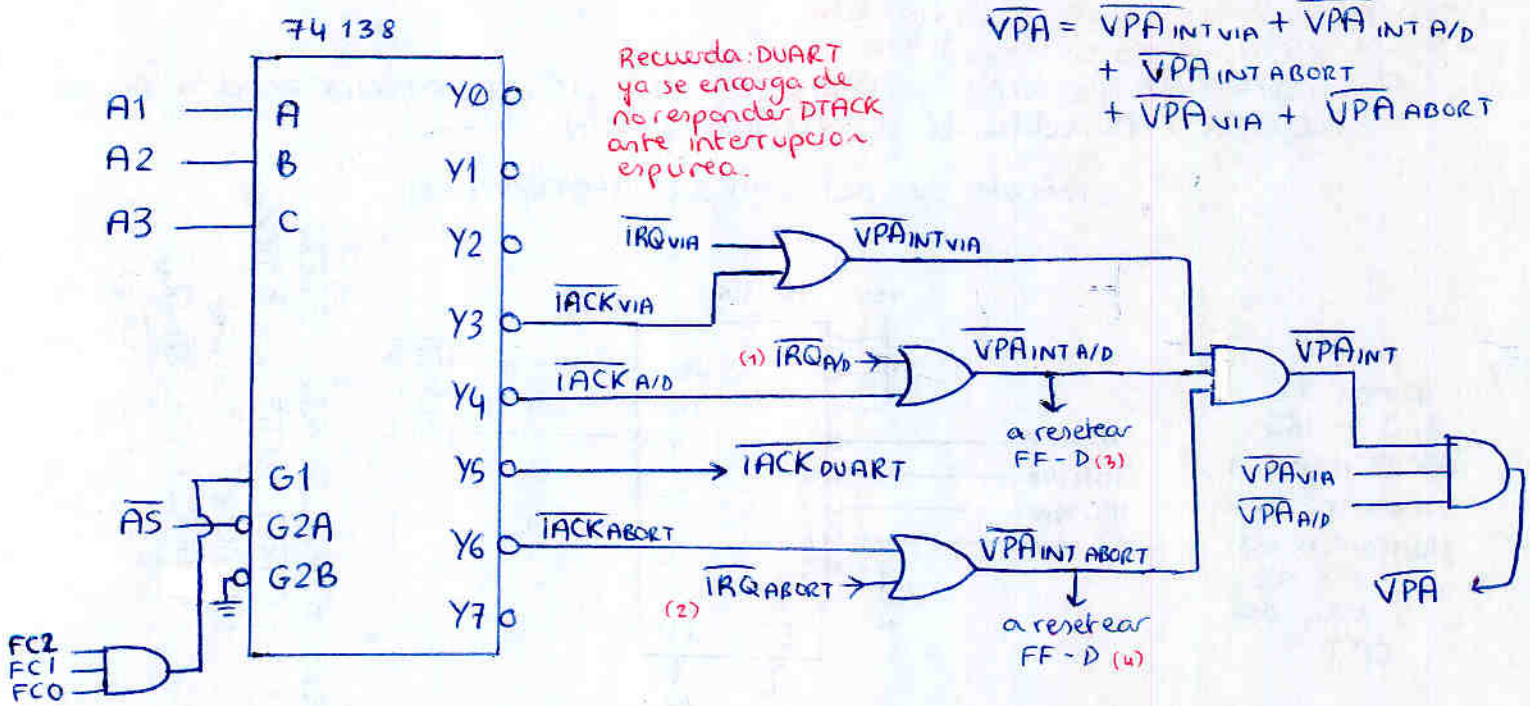
Intentaré dibujar el circuito completo de las interrupciones



podrías optar por no poner $\overline{VMA}$ y entonces para evitar condición de carrera quitar el $\overline{IRQ_{ABORT}}$ de la generación de $\overline{VPA_{INTABORT}}$

↳ Desventaja: respondería a interrupciones espúreas

$$\overline{VPA} = \overline{VPA_{INT\,VIA}} + \overline{VPA_{INT\,A/D}} + \overline{VPA_{INT\,ABORT}} + \overline{VPA_{VIA}} + \overline{VPA_{ABORT}}$$



Problema SED – Bañera Hidromasaje — Junio 2005

- 68000 a 10 MHz
- El caudal que surte de los jets de la bañera lo proporciona una bomba de agua.

Debe gestionar:

- Calentador de agua (1 entrada digital on/off)
- 2 sensores de temperatura del agua (0V – 5V)
- 2 sensores de nivel de agua (salida digital: uno se activa si agua excede el máximo)
- 10 electroválvulas controlan flujo de agua a los jets (entrada digital 8 bits)
- 3 dispensadores de esencias aromáticas (1 entrada digital on/off)
- 1 electroválvula de desagüe (1 entrada digital on/off)
- 1 teclado de al menos 15 teclas
- 2 visualizadores 7 segmentos
- Puerto RS-232 serie

La solución debe ser lo más optimizada posible

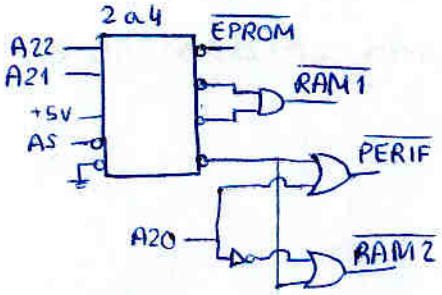
Problema 1.

- 2 MB memoria EPROM a partir de $0
- 4 MB memoria RAM a partir de $200000 (RAM1)
- 1 MB memoria RAM al final del mapa (RAM 2)

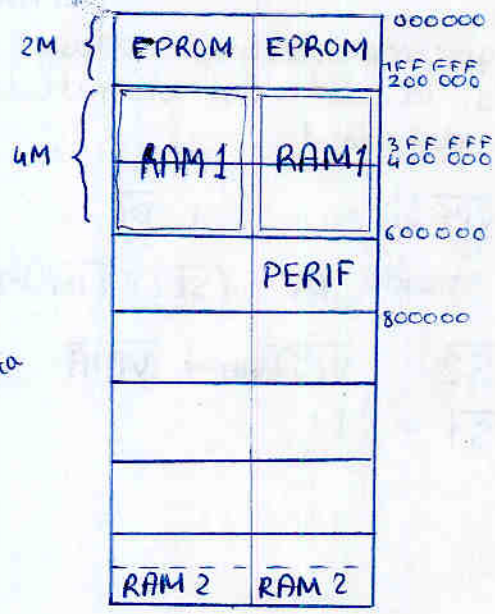
Mínimo número pastillas: i.e. 1MB EPROM
2 MB RAM1
512 kB RAM 2

¿qué decodificación utilizamos? Total 16MB

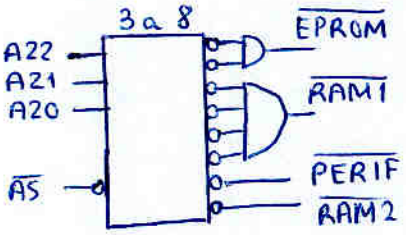
ej: 16 MB
8 MB ← ~~A23~~

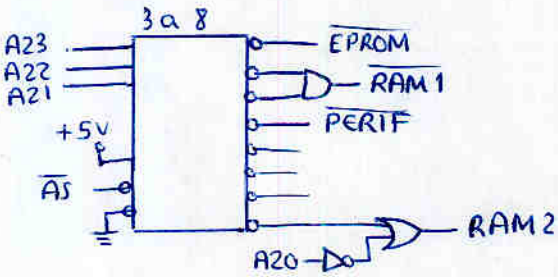


ej: decod 3 a 8



tambien





Escogemos la 2ª opción:

$\overline{CS_E} = A22 + A21 + R/\overline{W} + \overline{AS}$

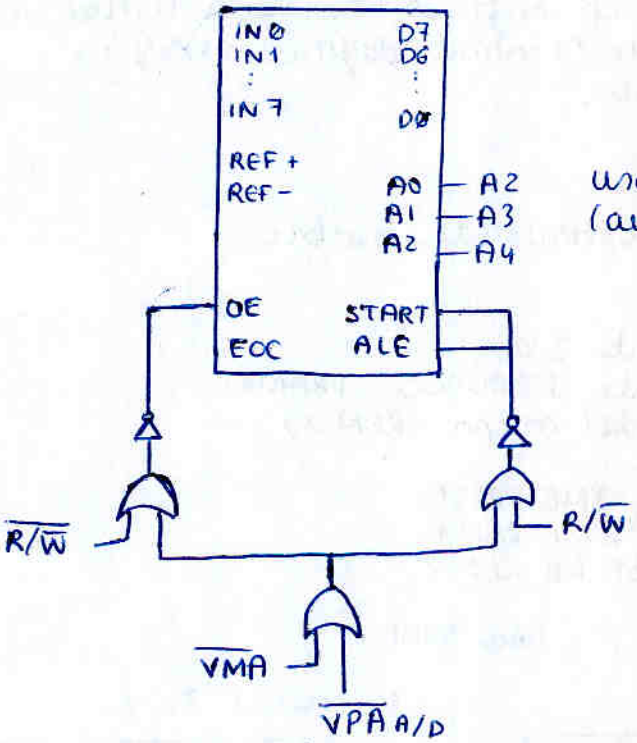
$\overline{CS_{RAM_1}} = (A22 + \overline{A21}) \cdot (\overline{A22} + A21) + \overline{AS} + \overline{XDS}$

$\overline{CS_{RAM_2}} = \overline{A22} + \overline{A21} + \overline{A20} + \overline{AS} + FC0 \cdot FC1 \cdot FC2$

$\overline{PERIF} = \overline{A22} + \overline{A21} + A20 + \overline{AS} + \overline{LDS}$

Problema 2

a) Mínimo número de direcciones de memoria que necesita este dispositivo



usa 8 posiciones de memoria (aunque estarán cada 4)

Indique mecanismo hardware que permita cambiar (mediante conmutador) la VIA en posiciones pares o impares

b) Si la salida del conmutador SEL vale 0 o 1 ¿Funcionaría correctamente?

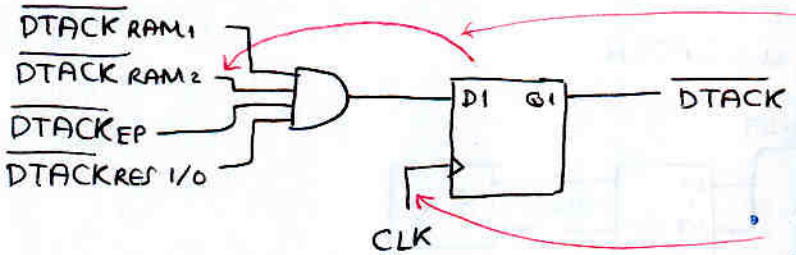
$\overline{VPA_{VIA}} = \ldots + \overline{DS}$

siendo $\overline{DS} = (\overline{SEL} + \overline{LDS}) \cdot (\overline{SEL} + \overline{UDS})$

$\overline{CS2} = \overline{VPA_{VIA}} + \overline{VMA}$

$\overline{CS1} = '1'$

c) Suponiendo que hay que insertar estados de espera en la RAM 2 ¿Que hay incorrecto en este circuito?



- El retardo debe afectar unicamente a la RAM2, no a todos
- El reloj debe retardar con el flanco de bajada. Debe negarse antes (Para que vaya justo al flanco de muestreo del μP)

CLK —▷o—

Problema 3

Activación sensores — interrupción nivel 4 y 5
Teclado — interrupción nivel 5

Implementar con una o varias VIA's
- Calentador (entrada on/off)
- 2 sensores nivel (salida true/false)
- 3 dispensadores de esencias
- La válvula de desagüe
- El teclado de 15 teclas

Haremos VIA1 (nivel 4): calentador PA.0
sensor de nivel CA1 ← para que pueda interrumpir y ademas por flanco
dispensadores de esencia PA.1 – PA.3
electroválvula de desagüe PA.4

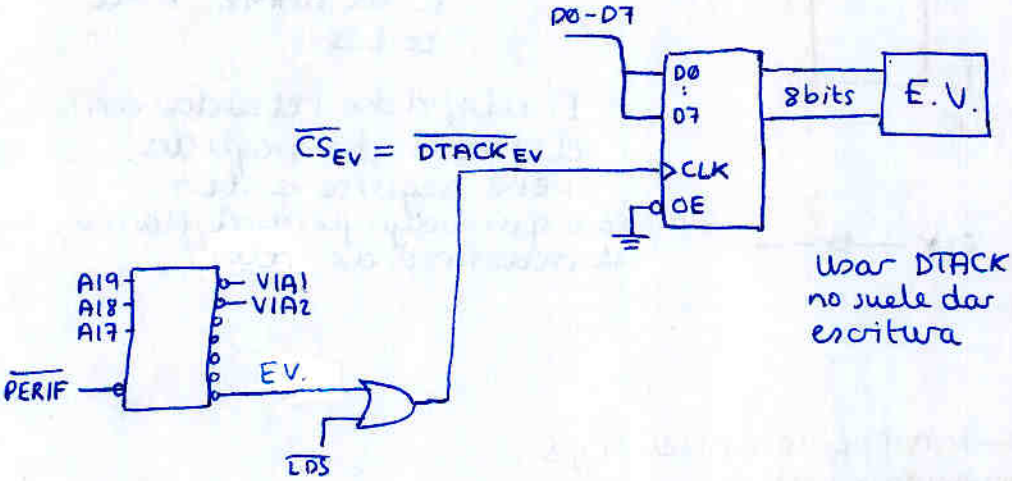
VIA 2 (nivel 5): sensor de nivel CA.1
teclado 4×4 PB0 – PB3 filas como salida
PB4 – PB7 columnas como entrada (con pull-up)
CB1 interrupción del teclado

Planteamiento: puedo tener N VIAs, cada una con dos puertos de 8 bits y ademas CA1, CB1 } entradas a la VIA
CA2, CB2 } según PCR → salidas de protocolo (i.e. el disp avisa datos con CA.1 la VIA indica que ya esta con CA2) ó entradas de propósito general

Pero cada VIA debe tener un nivel de interrupción único, por tanto habrá que usar una VIA en cada nivel

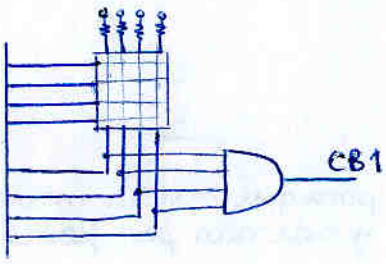
Los 2 visualizadores LCD
si tenemos convertidor BCD - 7 segmentos
necesitamos 4 puertos para cada uno

b) La electroválvula necesita un LATCH



Usar DTACK ya que un LATCH no suele dar problema en escritura

c) Petición de interrupción del teclado



NOTA: En sistema definitivo

- BERR → RESET software:

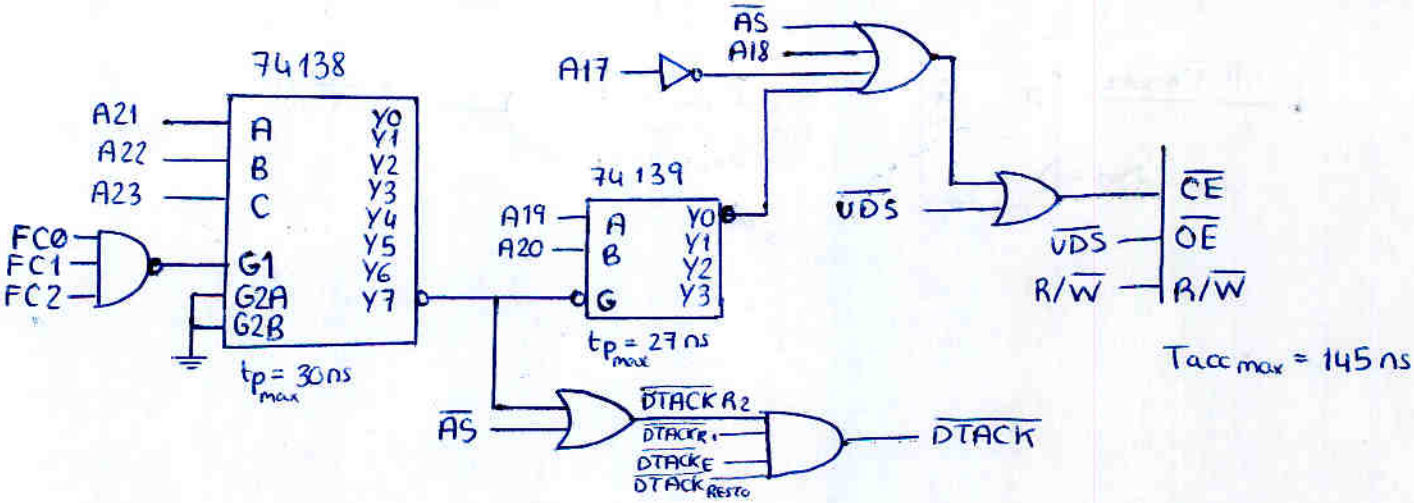
```
MOVE.L  $0, SSP
CLR.L   D0, D1, ..., D7
MOVE.L  0 , A0
        :
        0 , A7
MOVE.W  #$2700, SR
JMP     $4  (primer PC)
```

- Ciclo de Reintento (BERR + HALT)
- Nada
  Tener un watchdog

Problema 15

Estudio temporal

puertas $t_{p_{max}} = 20\,ns$ inversores $t_{p_{max}} = 18\,ns$



→ Lectura

- ¿Genera estados de espera?

Desde Ax

$$\underset{S0/S1 \to S4/S5}{4S} > t_{S0/S1-Ax} + t_{Ax \to DTACK} + t_{setup\,DTACK}$$

$$4 \cdot \tfrac{1}{2}\left(\tfrac{1}{10M}\right) > 6 + t_{p_{74138}} + 2 \cdot t_p + 47$$

$$200\,ns > 50\,ns + 30\,ns + 2 \cdot 20\,ns + 10\,ns$$

$$200\,ns > 130\,ns$$

Desde AS

$$\underset{S1/S2 \to S4/S5}{3S} > t_{S1/S2} + t_{AS \to DTACK} + t_{setup\,DTACK}$$

$$150\,ns > 9 + 2 \cdot t_p + 47$$

$$150\,ns > 50\,ns + 2 \cdot 20\,ns + 10\,ns$$

$$150\,ns > 100\,ns$$

- ¿Datos a tiempo? → ¿Necesita estados de espera?

Desde Ax

$$\underset{S0/S1 \to S6/S7}{6S} > t_{S0/S1-Ax} + t_{Ax-\overline{CS}} + Tacc + t_{setup\,DATA}$$

$$6 \cdot 50\,ns > 6 + t_{74138} + t_{74139} + 2t_p + Tacc + 27$$

$$300\,ns > 50\,ns + 97\,ns + 145\,ns + 10\,ns$$

$$300\,ns > 302\,ns$$

↓

¡No se cumple!
Requiere estados de espera

$$300 + N \cdot 100 > 302$$

$$N_{minimo} = 1$$

Desde AS

$$\underset{S1/S2 \to S6/S7}{5 \cdot S} > t_{S1/S2 \to \overline{AS}} + t_{\overline{AS}-\overline{CS}} + Tacc + T_{setup\,DATA}$$

$$250\,ns > 9 + 2 \cdot t_p + Tacc + 27$$

$$250\,ns > 50\,ns + 2 \cdot 20\,ns + 145\,ns + 10\,ns$$

$$250\,ns > 245\,ns$$

→ Escritura

$$t_{\overline{XDS}\,inactive \to Data\,Invalid} > t_{\overline{XDS}\,inactive \to \overline{CS}\,inactive} + t_{hold\,DATA}$$

$$25_{min} > t_p + t_{hold\,DATA}$$

$$30\,ns > 20\,ns + t_{hold\,DATA} \rightarrow \text{se cumple si } t_{hold} \leq 10\,ns$$

Para insertar el estado de espera:

Cuestión 3. Junio 2005

Se desea
- 32 kbytes EPROM en $0
- 16 kbytes RAM en $40000

Todas las pastillas son de 8kB

- El ultimo kB de la RAM debe ser pila de supervisor

Se pide:

1. Lógical del $\overline{CS}$ de la RAM

Nota
10 - 1k
11 - 2k
12 - 4k
13 - 8k
14 - 16k
15 - 32k
16 - 64k

RAM

dirección inicial: 0000 0100 0000 0000 0000 0000 (A19 A18 A17 A16)

16kBytes → 14 lineas dirección

00 0000 0000 0000
⋮
11 1111 1111 1111

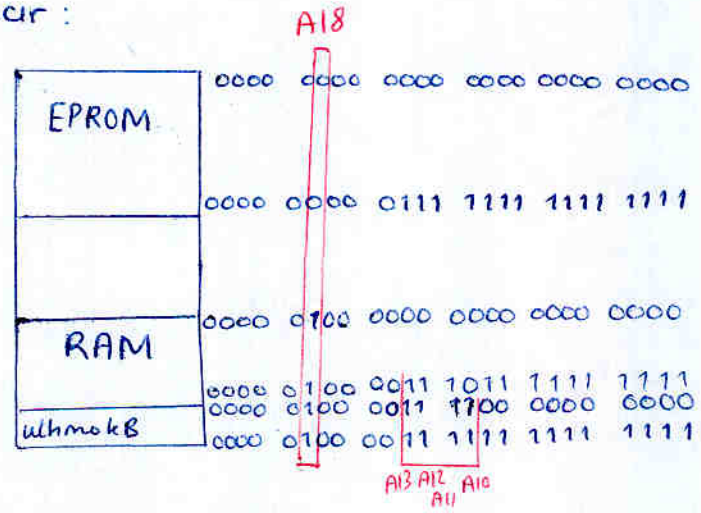
dirección final: 0000 0100 0011 1111 1111 1111

1kByte → 10 lineas dirección

ultimo kB inicio: 11 1100 0000 0000
ultimo kB final: 11 1111 1111 1111

EPROM Inicial: 0000 0000 0000 0000 0000 0000
Final 0000 0000 0111 1111 1111 1111

Es decir:



$$\overline{RAM} = \overline{A18}$$

$$\overline{PILA} = \overline{RAM} + (\overline{A13} + \overline{A12} + \overline{A11} + \overline{A10})$$

$$\overline{DTACK_{RAM}} = \overline{A18} + \overline{AS}$$

$$\overline{CS_{RAMpar}} = \overline{DTACK_{RAM}} + \overline{UDS}$$

$$\overline{CS_{RAMimpar}} = \overline{DTACK_{RAM}} + \overline{LDS}$$

Circuito que detecte accesos indebidos a la pila por parte del usuario y genere error de bus

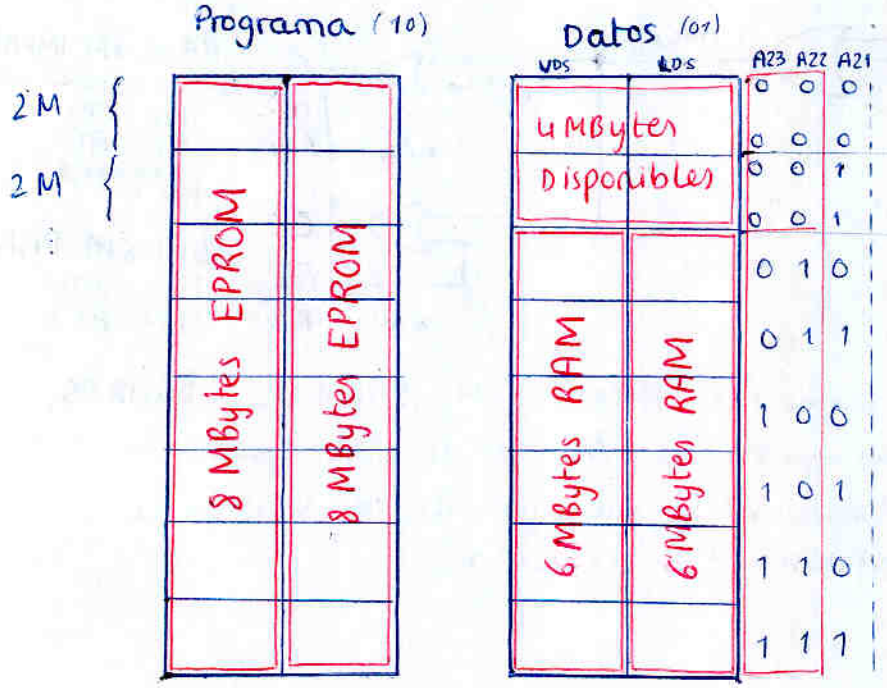
$$\overline{BERR} = \overline{PILA} + \overline{AS} + \overline{USUARIO}$$

siendo $\overline{USUARIO} = FC2$

Problema 1 - 13 Enero 2003

- MC68000 a 10 MHz
- máxima cantidad de memoria no volátil de programa
- 12 MB memoria de almacenamiento
- memoria no volátil para los vectores de excepción
- 4 VIA's (niveles 1, 2, 3 y 5) (deben tener nº vector configurable)
- 3 DUART's (niveles 3, 4 y 5) despues de las VIAs
- Convertidor A/D ADC0808 (nivel 1) } despues de las DUARTS
- Convertidor D/A DAC0800

- Disponemos de pastillas memoria 8 MBytes
- tp = 18 ns, Tacc = 198 ns



- la memoria ocupa casi todo el mapa.
- Cada pastilla de 8MBytes de RAM tendrá usados sólo 6MB
- Los 4 MBytes disponibles están arriba ya que debe ser arriba donde se almacenar los vectores de excepción 1kByte

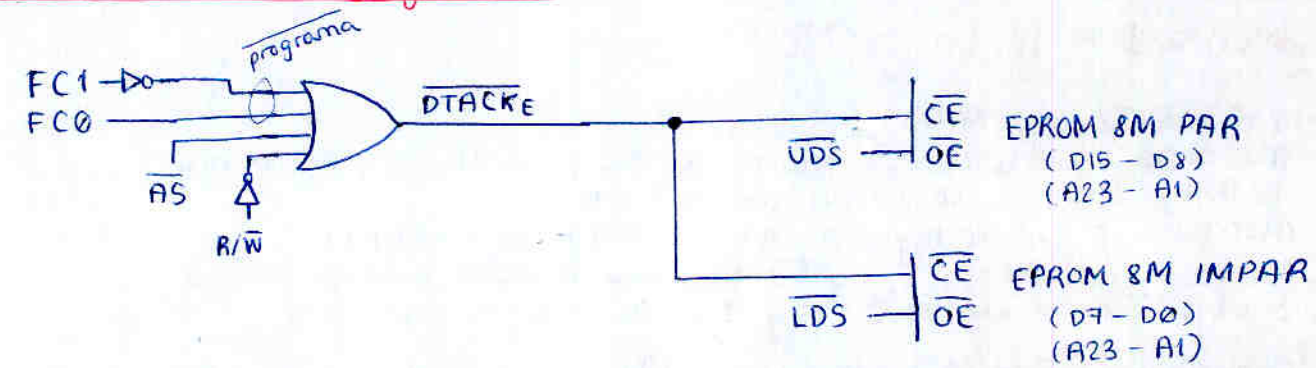
Esos 4MBytes serán:

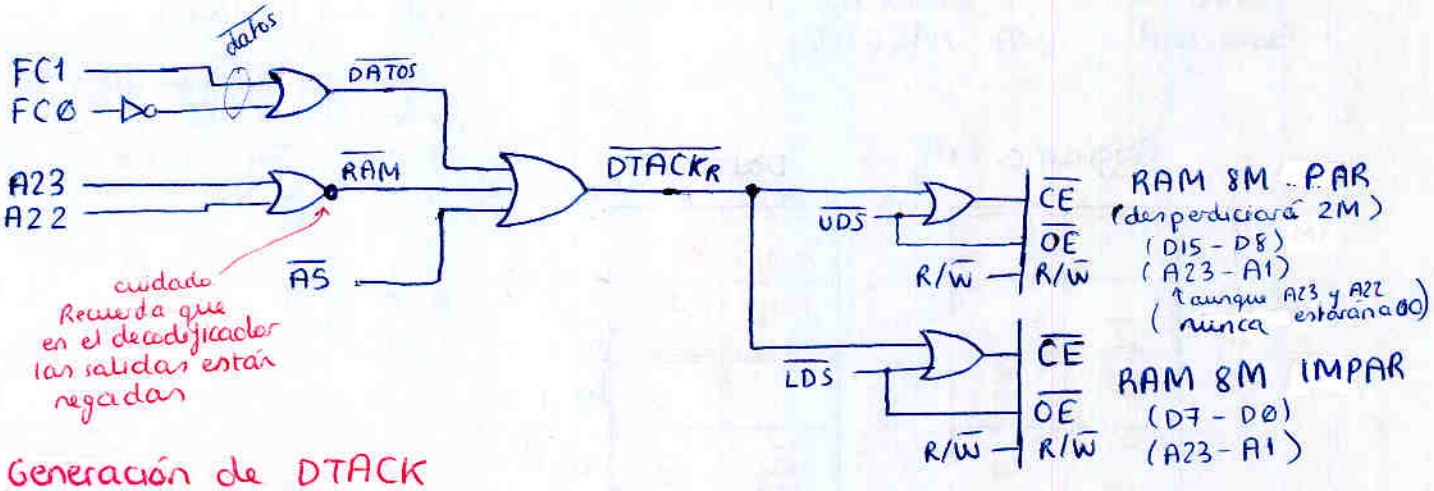
512kB { sólo 1kB de los 512kB

		A23	A22	A21	A20	A19	A18 ... A1
EPROM VECT EXC. 512B	EPROM VECT. EXC. 512B	0	0	0	0	0	0 ... 0 / 1 ... 1
VIA 1 int: 1	VIA 2 int: 2	0	0	0	0	1	
VIA 3 int: 3	VIA 4 int: 4	0	0	0	1	0	
	DUART 1 int: 3	0	0	0	1	1	
	DUART 2 int: 4	0	0	1	0	0	
	DUART 3 int: 5	0	0	1	0	1	
D/A	A/D int: 1	0	0	1	1	0	
	4 IVR's (4 FFD)	0	0	1	1	1	

los 4 LATCH harán que las VIAS devuelvan nº de vector.
Estarán en esta posición para la escritura. (CLK)
Su $\overline{OE}$ se activará con IACK VIAx y darán DTACK

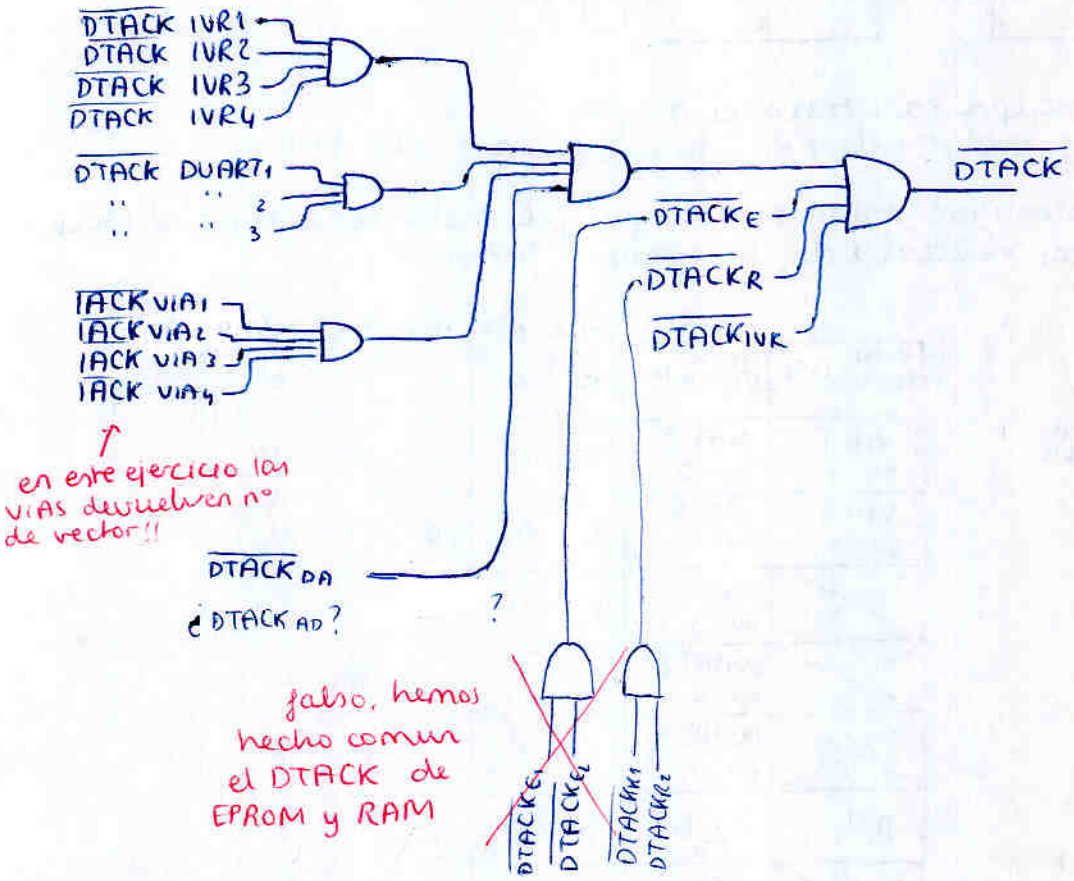
Conexión memorias grandes



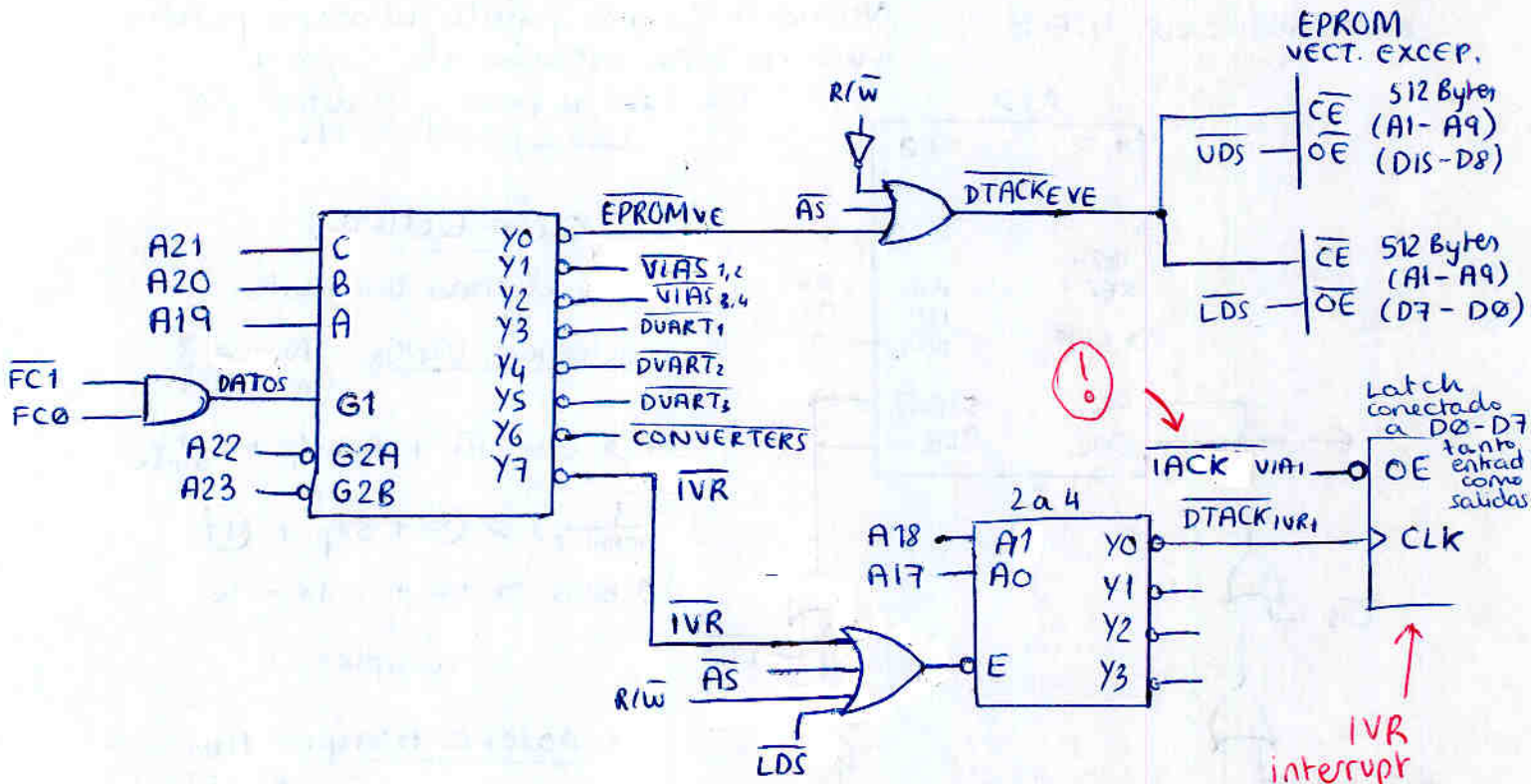


Generación de DTACK

Con DTACK responderán las memorias RAM, EPROM, EPROM U.E, DUARTS, el A/D si no necesita estados espera, el D/A, los 4 IVR (para que respondan ante su ciclo de escritura) y curiosamente las VIAS en el ciclo de ~~excepc~~ reconocimiento de excepción



EPROM V.E y IVRs



Verificación estados de espera de las memorias

RAM

Lectura

- Existencia de estados de espera desde $\overline{AS}$ (mas limitante)

$$35 > 9 + 2 \cdot t_p + 47$$

$150ns > 50 + 2 \cdot 18 + 10$ se cumple

- Datos en el bus a tiempo

$$5 \cdot S > 9 + 2 \cdot t_p + T_{acc} + 27$$

$250 > 50 + 2 \cdot 18 + 198 + 10$

$250 > 294$

requiere 1 estado de espera (100ns)

Escritura

$$25 > \underset{(XDS-CS)}{1 \cdot t_p} + t_{hold_{RAM}}$$

$30 > 18 +$ no lo dicen

suponemos que se cumple ya que de lo contrario sería difícil de solucionar

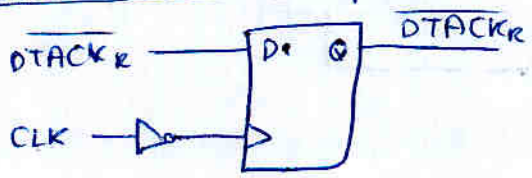
EPROM tiene una puerta menos desde $\overline{AS}$ a $\overline{CS}$; no es suficiente tambien requiere un estado de espera

EPROM V.E. Tiene más puertas; tal vez necesite algún estado más

- Datos en el bus a tiempo: ↳ ah no, tiene las mismas

$$5 \cdot S > 9 + 1 \cdot t_p + 27$$

insertar estados de espera (en $DTACK_R$, $DTACK_E$ y $DTACK_{EVE}$)

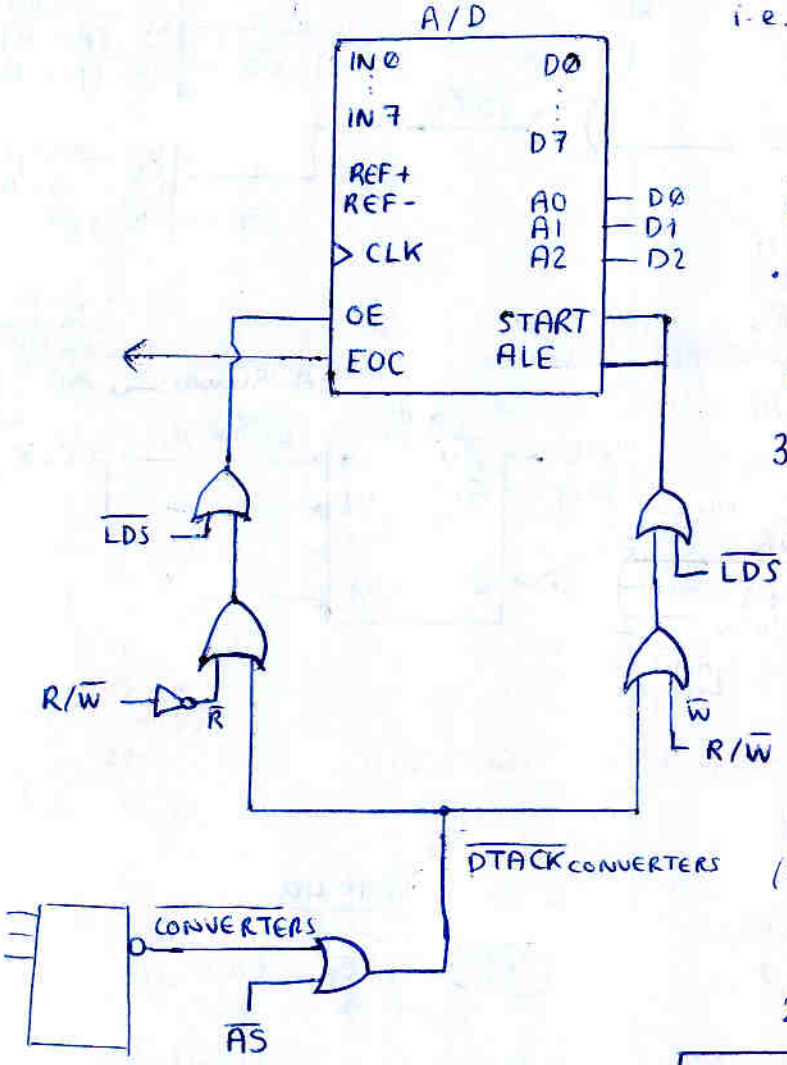


A/D : Con DTACK o con VPA ?

¿Funcionaría con DTACK ?

Vamos a intentar ponerlo lo mejor posible para no tener estados de espera

i.e. $\overline{XDS}$ al final, $\overline{AS}$ detrás de decodificador, etc...



Ciclo lectura

$\overline{AS}$ será el mas limitante:

- Generación DTACK $N_{AS} = 3$ (AS → DTACK)

$$3S > t_{S1/S2 - \overline{AS}} + N_{AS} \cdot t_p + t_{setup\,DTACK}$$

$$3 \cdot \left(\frac{1}{10MHz \cdot 2}\right) > 9 + 3t_p + 47$$

$$3 \cdot 50ns > 50 + 3 \cdot 18 + 10$$

se cumple

- Datos a tiempo $n_{AS} = 3$ (AS - CS)

$$\underset{(\text{i.e. hasta } S6)}{5S} > \underbrace{9 + n_{AS} \cdot t_p}_{S1/S2 \to \overline{OE}} + T_{acc} + t_{setup\,DATOS}$$

$$250\,ns > 50 + 3 \cdot 18 + 250\,ns + 10$$

No se cumple

A partir de ahora recordar:

Datos a tiempo en Lectura

$$5 \cdot S > t_{S1/S2 - \overline{AS}} + n_{AS-CS} \cdot t_p + T_{acc} + t_{setup\,DATOS}$$

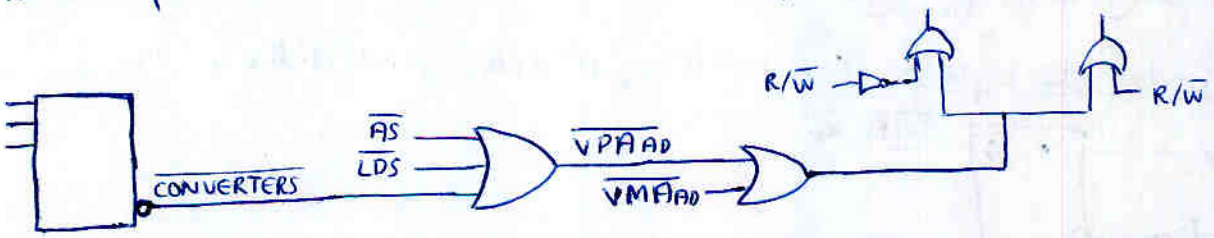
$$5 \cdot 50ns > 9 + n_{AS} \cdot t_p + T_{acc} + 47$$

$$250\,ns > 50 + n_{AS} \cdot t_p + T_{acc} + 10$$

si $T_{acc} > 190\,ns \Longrightarrow$ SEGURO que habrá estados de espera

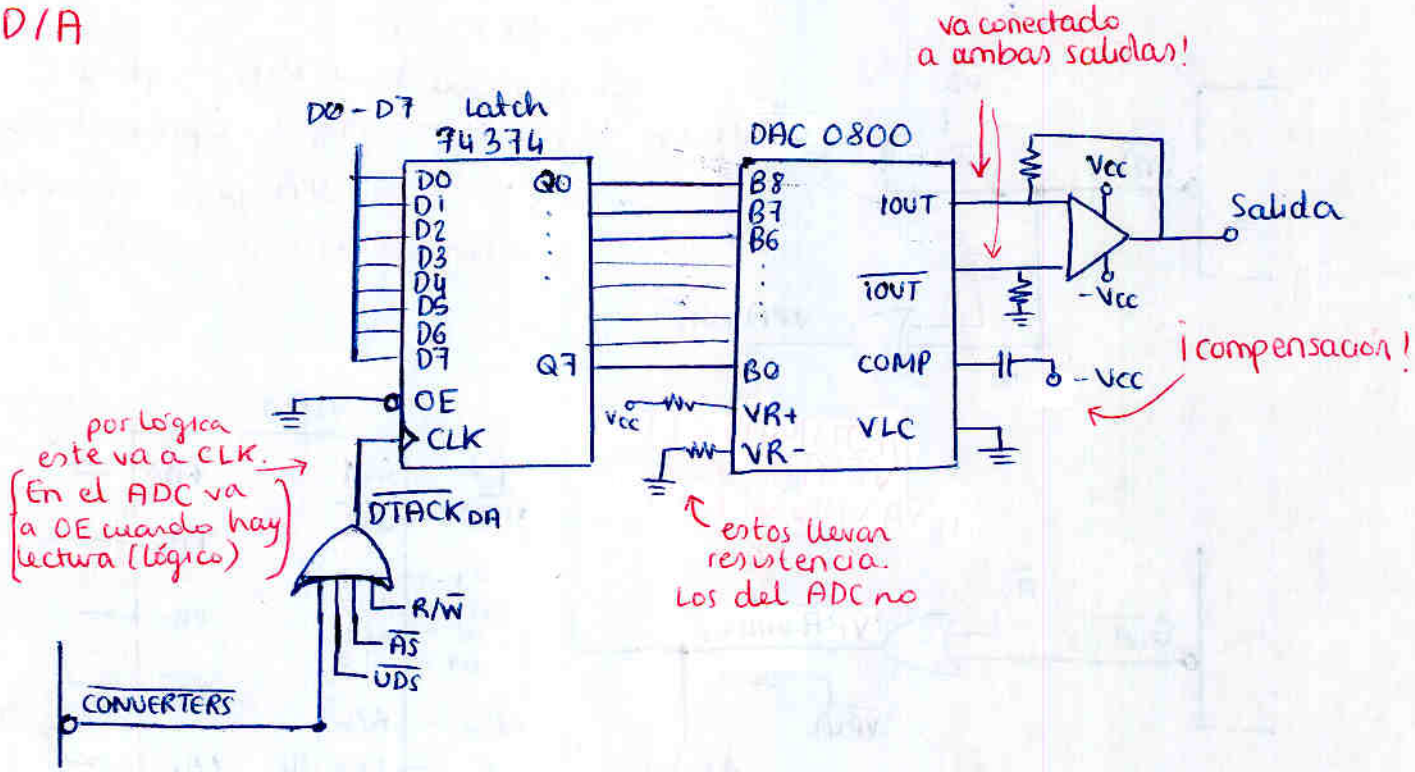
en realidad $190 - n_{AS} \cdot t_p$ → peor aún !!

Habrá que conectar el A/D con $\overline{VPA}$ para optimizar hardware

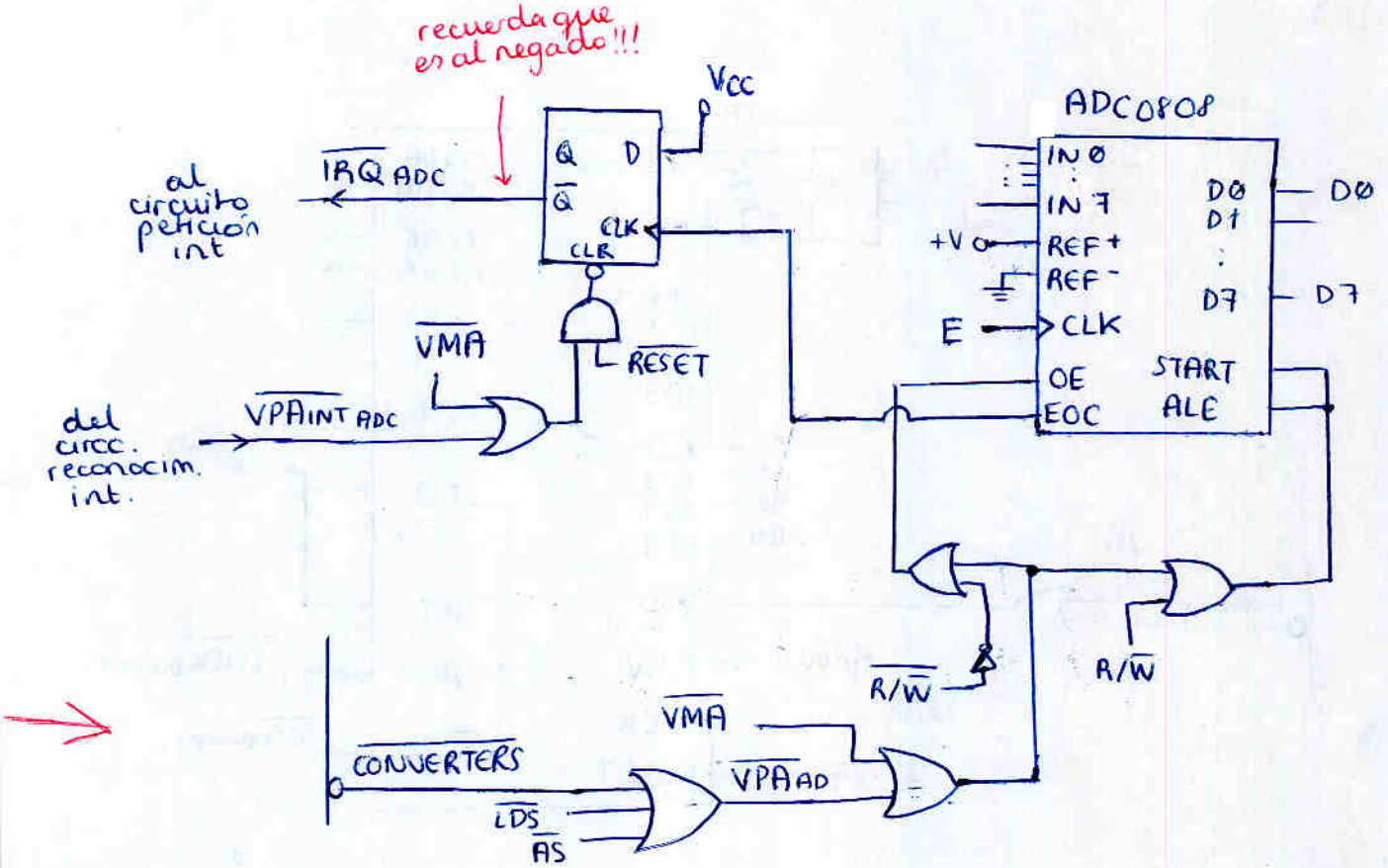


Conexión de los convertidores

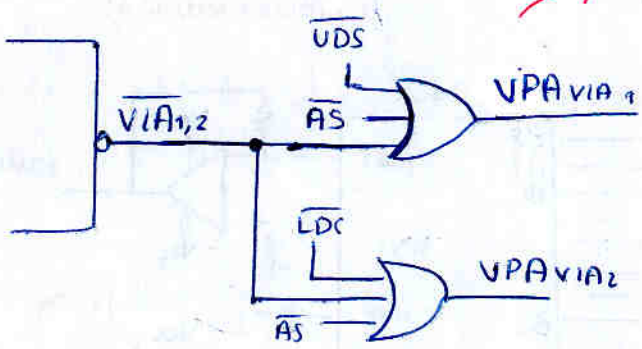
D/A



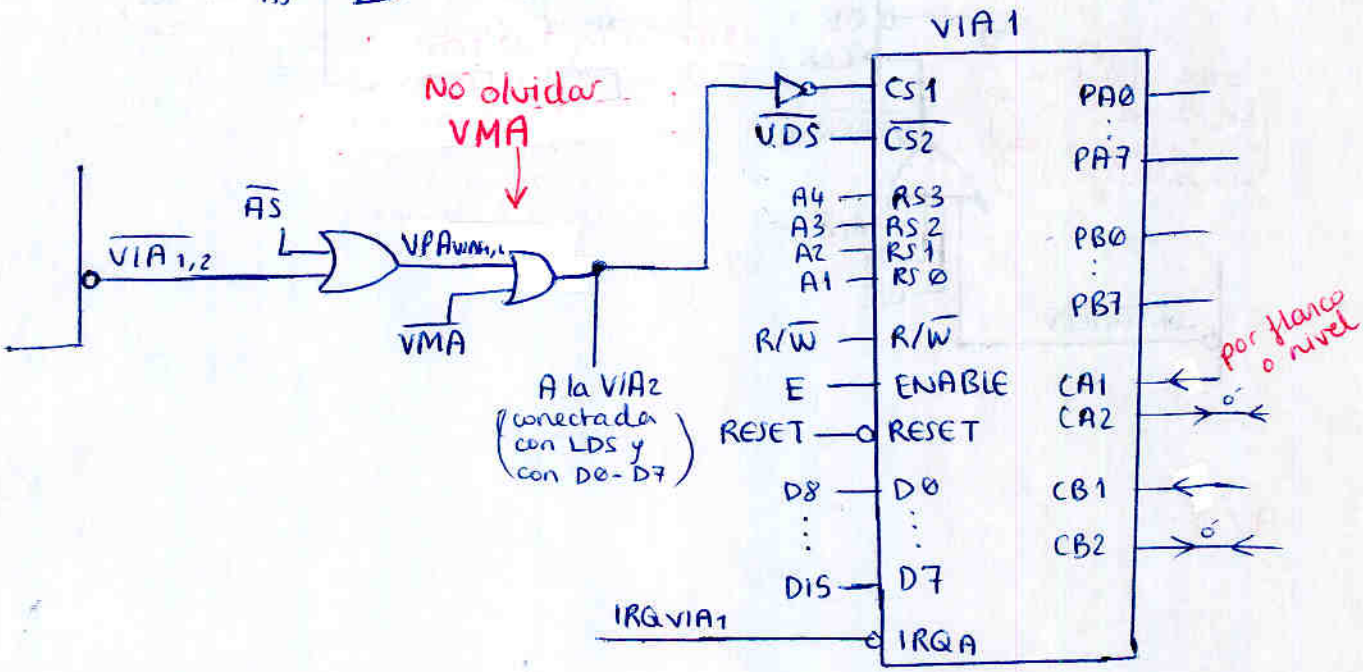
A/D



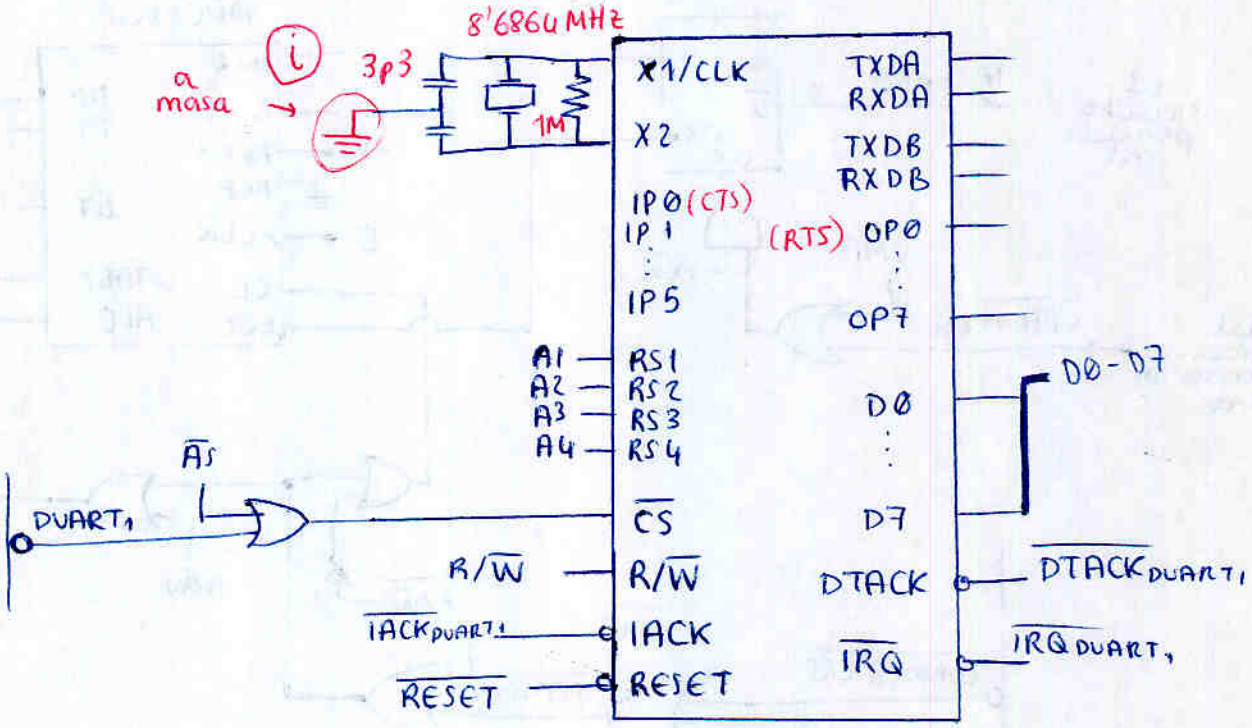
VIA



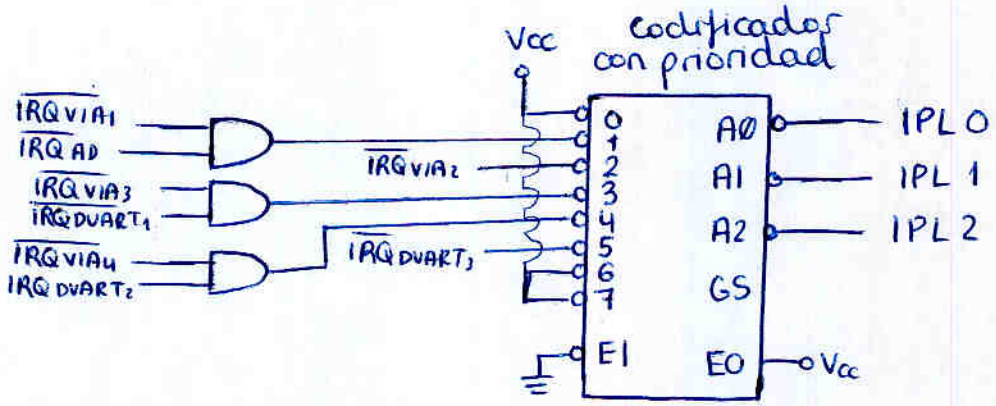
Mal. No es óptimo.
Siendo que hay VIA tanto en par como impar, aprovéchalo y genera VPA VIA1,2, conjunto antes de XDS.



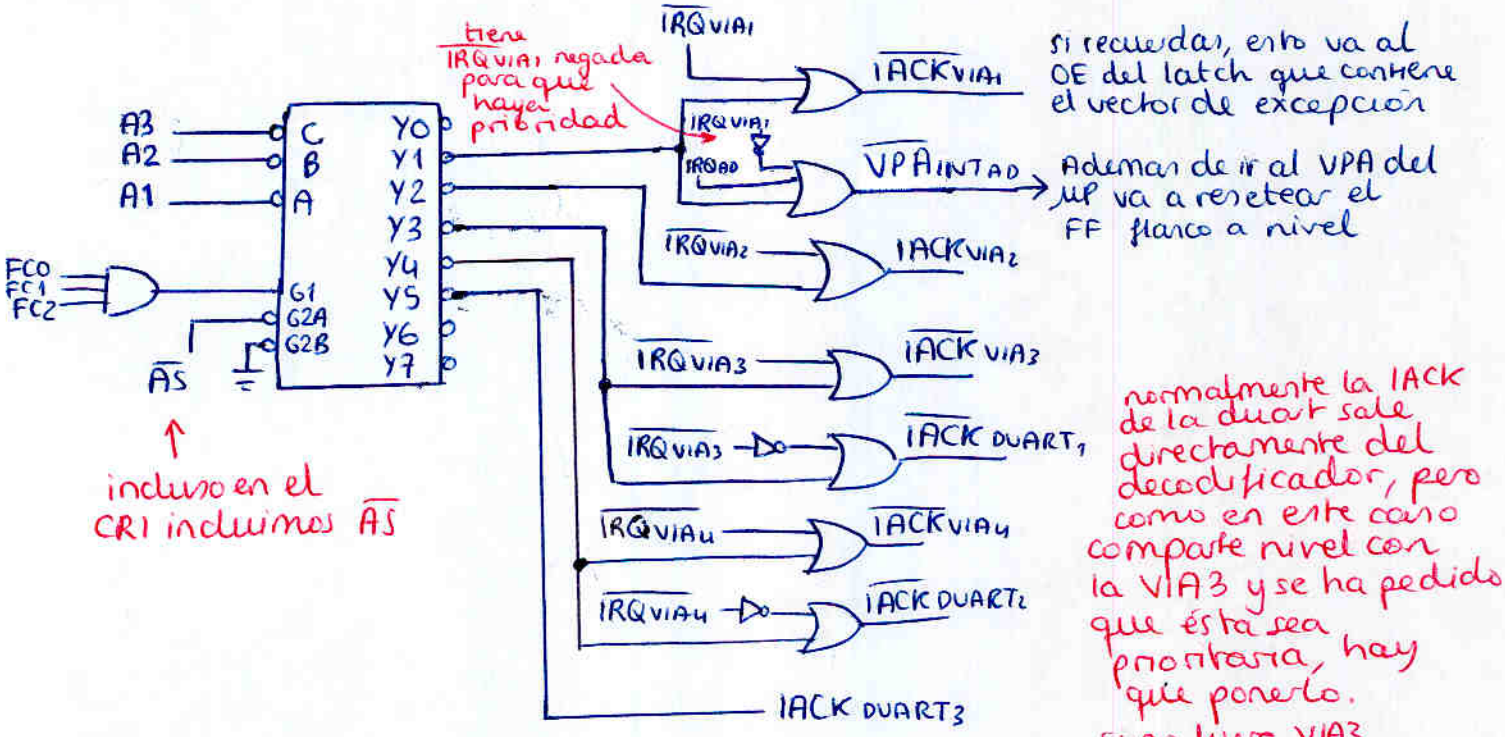
DUART



Circuito de petición de interrupciones



Circuito de reconocimiento de interrupciones



Ecuaciones de VPA y DTACK

$$\overline{VPA} = \overline{VPA_{AD}} \cdot \underbrace{\overline{VPA_{VIAS_{1,2}}}}_{\text{recuerda que compartían VPA}} \cdot \overline{VPA_{VIAS_{3,4}}} \cdot \underbrace{\overline{VPA_{INTAD}}}_{\text{en este ejercicio sólo hay un VPA INT (i.e. solo un dispositivo autovectorizado)}}$$

$$\overline{DTACK} = \overline{DTACK_E} \cdot \overline{DTACK_R} \cdot \overline{DTACK_{EVE}} \cdot \overline{DTACK_{IVR1}} \cdot \overline{DTACK_{IVR2}} \cdot \overline{DTACK_{IVR3}} \cdot \overline{DTACK_{IVR4}} \cdot \overline{DTACK_{DUART1}} \cdot \overline{DTACK_{DUART2}} \cdot \overline{DTACK_{DUART3}} \cdot \overline{IACK_{VIA1}} \cdot \overline{IACK_{VIA2}} \cdot \overline{IACK_{VIA3}} \cdot \overline{IACK_{VIA4}}$$

estos eran para la ESCRITURA en IVR

estos son para cuando IVR devuelve nº de vector

Problema Enero 2005

MC 68000 a 10 MHz para medir velocidad y sentido del viento:

- Veleta: – posición angular (6 bits paralelo)
  (señal validación de datos flanco de bajada)
  - El 68000 debe ser capaz de leer los 6 bits mediante lectura en $ 70 0000
- Anemómetro – Pulso a 0 durante 10ms cada vez que completa un giro
- Visualizador LCD – contraste entrada 2'5 V
- Mínimo 128 Kbytes RAM (partillas 128K x 6 cuyo patillaje se detalla más adelante)
- Puerto serie asíncrono RS-232
- Mínimo 20 Kbytes EPROM (partillas 32k x 8)

Si eliges utilizar VIA(s) o DUART(s) utiliza VIA nivel 4
DUART nivel 5
anemómetro nivel 6

LCD

RS: instrucción (0)
datos (1)

Vo: contraste 0-5V

R/$\overline{W}$: escritura (0)
lectura (1)

Enable: habilita (1) operaciones lectura/escritura

DB0-DB7: Bus datos

$\overline{DTACK}$: valida lectura/escritura

RAM

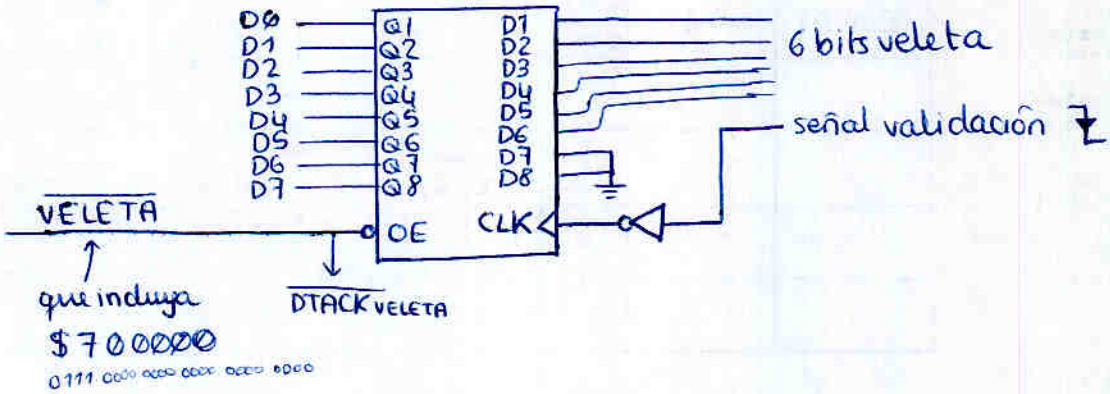
A0 - A16
D1 - D16 ← !! 16 bits
$\overline{CS}$

$\overline{WE}$ Write enable
$\overline{OE}$
$\overline{LB}$ Lower byte
$\overline{UB}$ Upper byte

se dan además los parámetros temporales.

Problema 9.1 listado completo de los elementos

- Para la veleta un latch 74 374



- El anemómetro generará la interrupción con un convertidor flanco a nivel.
- Visualizar LCD conectado sin usar dispositivo intermedio (ya que viene con todo bien preparado) → $\overline{DTACK}$
- Una pastilla RAM 128K x 16 (ya serán 256 kB) → $\overline{DTACK}$
- Para el puerto serie RS-232 utilizamos una DUART que está preparada para transmisión asíncrona. → $\overline{DTACK}$
- Dos pastillas ROM 32k x 8 (una par y otra impar) → $\overline{DTACK}$

2. Diseñe el mapa de memoria y el decodificador que lo implementa (sólo modo supervisor)

32 k posiciones: 64 kB

UDS	LDS	A18	A17	A16	A15	A11	A7	A3	A0
ROM 32k8	ROM 32k8	0	0	0	0000	0000	0000	0000	0
		0	0	0	1111	1111	1111	1111	1
RAM 128k x 16		0	0	1					
		0	1	0					
		0	1	1					
		1	0	0					
	DUART	1	0	1					
	LCD	1	1	0					
	VELETA	1	1	1	← 70000				

valores que usaría la RAM

Me he equivocado, me había equivocado, creía que era $70000 y no $700000

La veleta no está en $700000. Habrá que hacer otra solución

1M x 16

ROM 32k8	ROM 32k8	000	x	xxxx	xxxx	0000	0000	0000
		000	x	xxxx	xxxx	1111	1111	1111
RAM 128k x 16		001	x	xxxx	xx00	0000	0000	0000
		001				11	1111	1111 1111
VIA	DUART	010						0000 0 cada 2
		010						1111 1
VELETA	LCD	011						00
		011						11

si quisiéramos que la veleta interrumpa podríamos usar VIA y puerto B ⇒ RS ≠ 0000

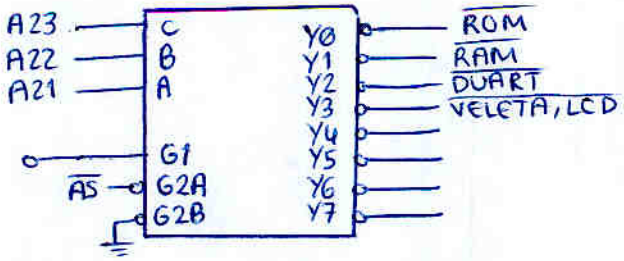
ROM 32k x 16
64k x 8 → 16 líneas (A0 - A15)

RAM 128k x 16
256k x 8 → 18 líneas (A0 - A17)

DUART → 4 líneas + LDS (A0 - A4)

LCD → 1 línea + XDS (A0 - A1)

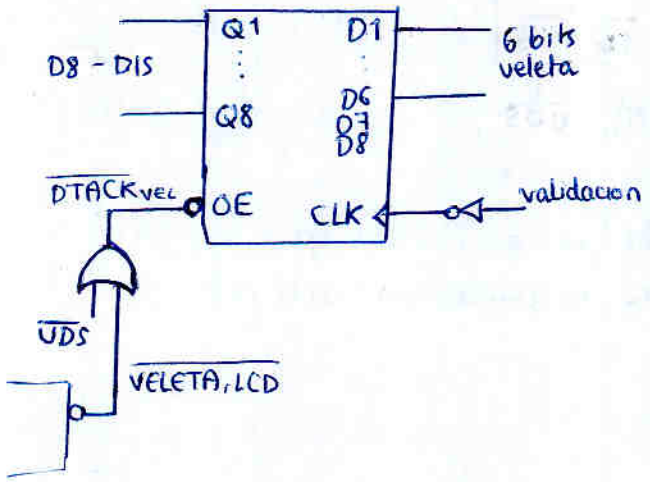
al usar un 3a8 no hay que preocuparse del CR1 ya que incluye A23 = 0



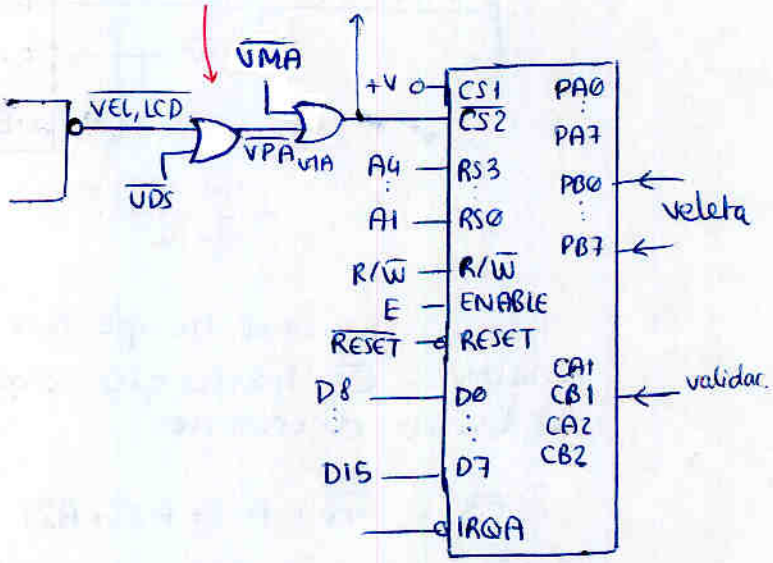
Problema 10

1. Conexión de la veleta al sistema

2 opciones

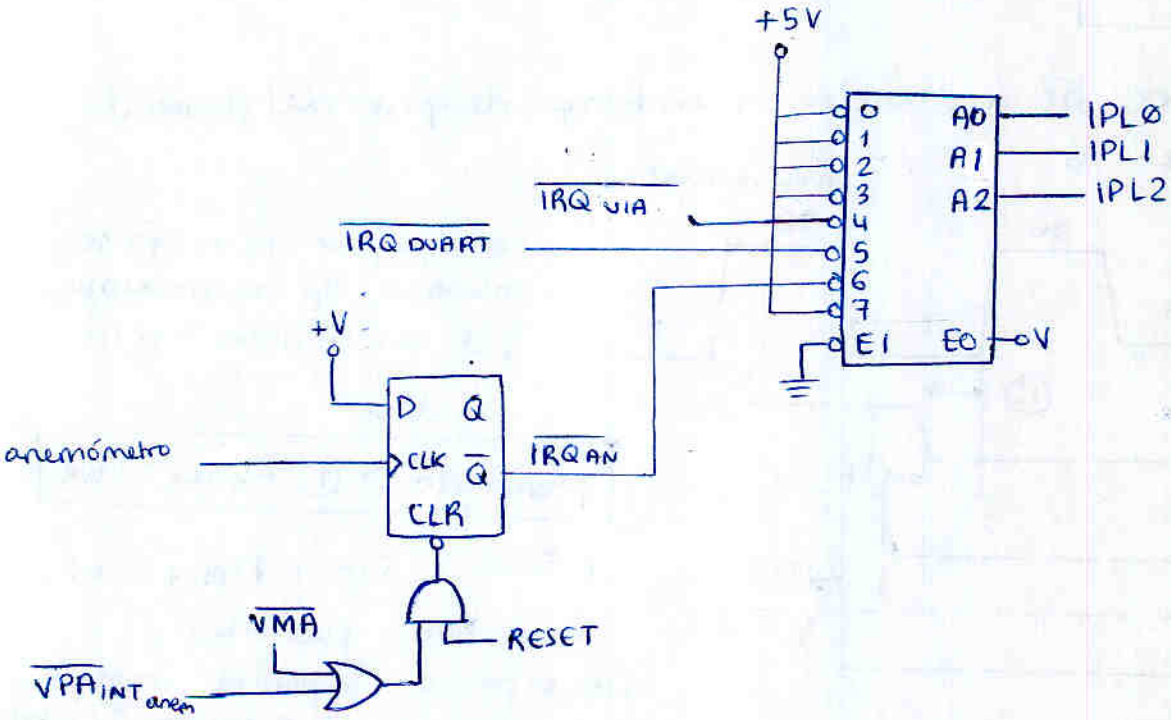


No podemos tener VPA antes de XDS ya que comparte con el LCD, que devuelve DTACK

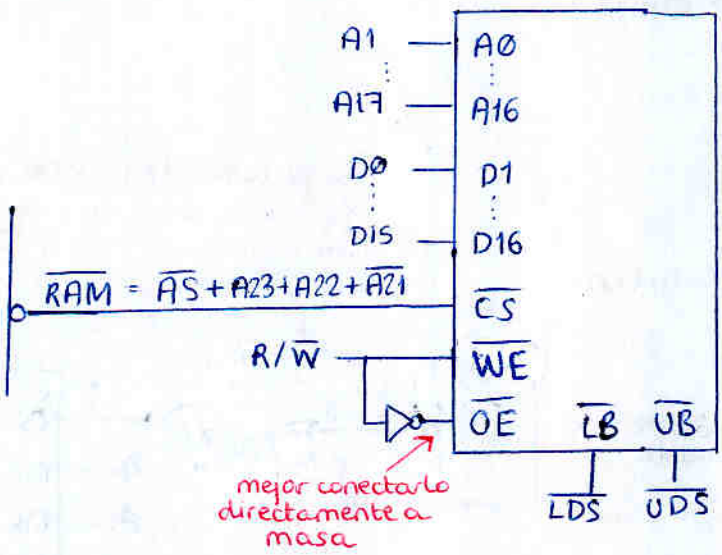


En puerto B porque es la dirección 0000.
Habrá que configurar CB1 para responder por flanco

2. Circuito de petición de interrupciones



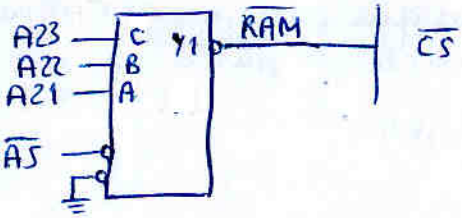
3. Conexión de la pastilla de memoria



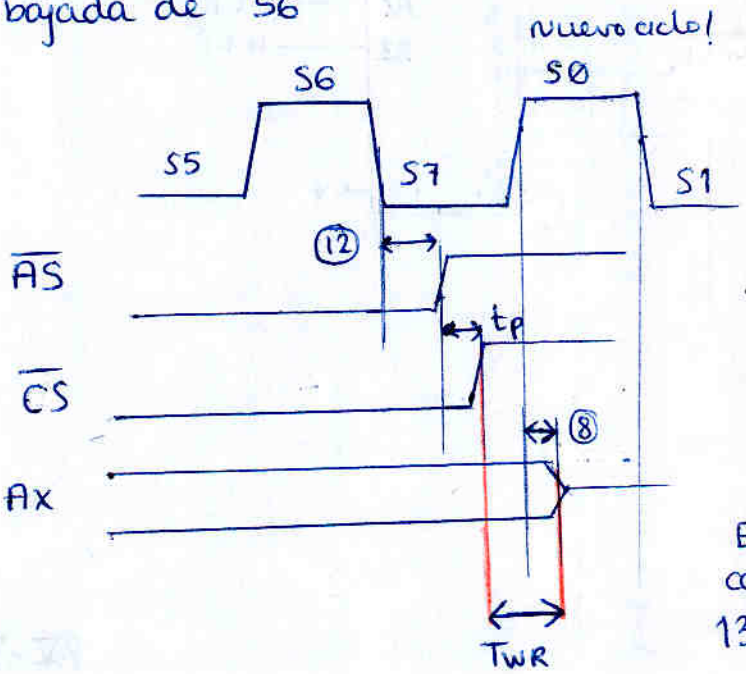
T_{WR} mínimo es el tiempo que debe pasar desde que se desactiva $\overline{CS}$ hasta que se quita la dirección actual del bus de direcciones

$$\overline{CS} = \overline{AS} + A23 + A22 + \overline{A21}$$

$\overline{CS}$ se desactivará un tiempo de propagación después de $\overline{AS}$ ya que



y a su vez, $\overline{AS}$ se desactivará un tiempo después del flanco de bajada de S6



el caso peor es que (12) sea máximo, tp sea máximo, y (8) sea mínimo (= 0 ns)

Es decir

$$\boxed{1 \text{ Semiciclo} > (12) + t_{p\,max} + T_{WR}}$$

$1 \cdot 50\text{ ns} > 50\text{ ns} + 13\text{ ns} + 0\text{ ns}$

No podemos garantizarlo

Esto es porque $\overline{CS}$ podría - en el peor caso - llegar a desactivarse hasta 13 ns después.

Julio 2004

Enunciado común a los problemas 5–8

En un concurso de TV cada participante tiene a su disposición un panel controlado por un 68000 a 10 MHz con los siguientes componentes:

1. Un visualizador de cristal líquido (LCD) en el que puede leer las preguntas.
2. Un teclado de 16 teclas (matriz 4x4) que le permite introducir las respuestas.
3. Un pulsador para las pruebas de velocidad de respuesta.
4. 4 visualizadores 7-segmentos que le permiten ver su puntuación.

Para implementar el panel se dispone de:

1. Memoria ROM para código de programa y tablas de datos.
2. Memoria RAM para datos, en posiciones de memoria a continuación de la ROM.
3. Dos zócalos libres para una posible ampliación de la RAM. La nueva memoria ocuparía posiciones inmediatamente superiores a las ya existentes.
4. Una VIA 6522 que simplifica la conexión del teclado 4x4, el pulsador y los visualizadores 7-segmentos. Se dispone para ello de un único conversor BCD a 7-segmentos 74LS48.

5. Dos VIAs 6522 que controlan las luces en el entorno de ese concursante, así como sensores que garantizan que el concursante tiene las manos correctamente ubicadas hasta que finaliza completamente la exposición oral de las preguntas.

6. Una DUART 68681 que permite comunicar el panel de concursante con el sistema centralizado que gestiona todos los paneles.

Otras características del sistema son:

- Todas las VIAs pueden solicitar interrupciones, de niveles 2, 3 y 4 respectivamente. Las interrupciones de las VIAs emplearán los 3 primeros números de vector reservados para el usuario: $40, $41 y $42. La DUART 68681 también solicita interrupciones.

- El código de programa ocupa actualmente 60 kbytes, mientras que las tablas de datos disponibles en ROM ocupan 4 kbytes. Estos valores no incluyen otros datos que el sistema pueda necesitar para un correcto funcionamiento.

- Se estima que los datos variables del programa diseñado pueden llegar a ocupar como mucho 32 kbytes.

- El contraste del visualizador LCD debe ser configurable por *software*.

- El diseño debe hacerse lo más optimizado posible, indicando en caso de conflicto cuál ha sido el criterio elegido.

Para implementar el sistema se dispone de memorias RAM de 32k x 8, memorias EPROM de 32k x 8, puertas AND y OR de 2, 3 y 4 entradas, puertas inversoras, codificadores 8 a 3 y decodificadores 3 a 8. El retardo máximo introducido por las puertas y los decodificadores es de 20 ns, mientras que los tiempos de acceso máximos de las memorias RAM y EPROM son 150 y 100 ns, respectivamente. El tiempo de mantenimiento mínimo de los datos en escritura de la memoria es de 10 ns.

El controlador del LCD es compatible HD44780, siendo su interfaz (conexión asíncrona):

- RS (Entrada): selección de registro de instrucción (0) o datos (1).

- V_o (Entrada): entrada analógica (0–5 V) para el ajuste del contraste.

- R/$\overline{W}$ (Entrada): selección de operación de escritura (0) o lectura (1).

- Enable (Entrada): habilita (1) las operaciones de lectura/escritura.

- DB0–DB7 (Entrada/salida): bus de transferencia de datos y comandos entre el microprocesador y el LCD.

Bus Timing Characteristics

Write Operation

Item	Symbol	Min	Typ	Max	Unit	Test Condition
Enable cycle time	t_{cycE}	500	—	—	ns	Figure 25
Enable pulse width (high level)	PW_{EH}	230	—	—		
Enable rise/fall time	t_{Er}, t_{Ef}	—	—	20		
Address set-up time (RS, R/$\overline{W}$ to E)	t_{AS}	40	—	—		
Address hold time	t_{AH}	10	—	—		
Data set-up time	t_{DSW}	80	—	—		
Data hold time	t_H	10	—	—		

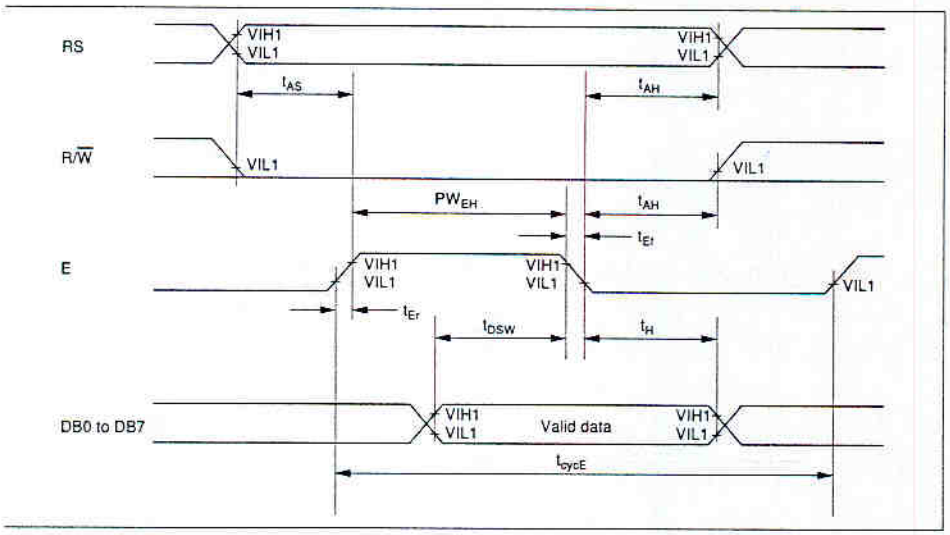
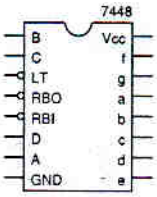


Figure 25 Write Operation

PVI-2



	Entradas						E/S	Salidas						
Función	LT	RBI	D	C	B	A	BI/RBO	a	b	c	d	e	f	g
0	H	H	L	L	L	L	H	H	H	H	H	H	H	L
1	H	X	L	L	L	H	H	L	H	H	L	L	L	L
2	H	X	L	L	H	H	L	H	H	L	H	H	L	H
3	H	X	L	L	H	H	H	H	H	H	H	L	L	H
4	H	X	L	H	L	H	L	L	H	H	L	L	H	H
5	H	X	L	H	L	H	H	H	L	H	H	L	H	H
6	H	X	L	H	H	H	L	L	L	H	H	H	H	H
7	H	X	L	H	H	H	H	H	H	H	L	L	L	L
8	H	X	H	L	L	H	L	H	H	H	H	H	H	H
9	H	X	H	L	L	H	H	H	H	H	L	L	H	H
10	H	X	H	L	H	H	L	L	L	L	H	H	L	H
11	H	X	H	L	H	H	H	L	L	H	H	L	L	H
12	H	X	H	H	L	H	L	L	H	L	L	L	H	H
13	H	X	H	H	L	H	H	H	L	L	H	L	H	H
14	H	X	H	H	H	H	L	L	L	L	H	H	H	H
15	H	X	H	H	H	H	H	L	L	L	L	L	L	L
BI	X	X	X	X	X	L	X	L	L		L	L	L	L
RBI	H	L	L	L	L	L	L	L	L		L	L	L	L
LT	L	X	X	X	X	H	X	H	H	H	H	H	H	H

Problema 5.

- DTACK o VPA

RAM : 2 implementadas y 2 posible ampliación. Responden con $\overline{DTACK}$ para un acceso más rápido en lectura y escritura.

EPROM : 2 pastillas. Responde con $\overline{DTACK}$ - más rápido

VIAS : Responden con $\overline{VPA}$ en ciclos de lectura / escritura, puesto que son dispositivos síncronos de la familia del 6800

Para el CRI deben responder con número de vector; por tanto usarán $\overline{DTACK}$

DUART : con DTACK ya que es periférico del 68000 y está pensada para ello.

LCD : el LCD (y el DAC que usaremos para el contraste) son asíncronos y en principio responderan con DTACK a menos que pudiésemos usar $\overline{VPA}$ para ahorrar hardware de ciclos de espera

Mapa de memoria

usamos bloque mínimo
64 kB = A0 - A15

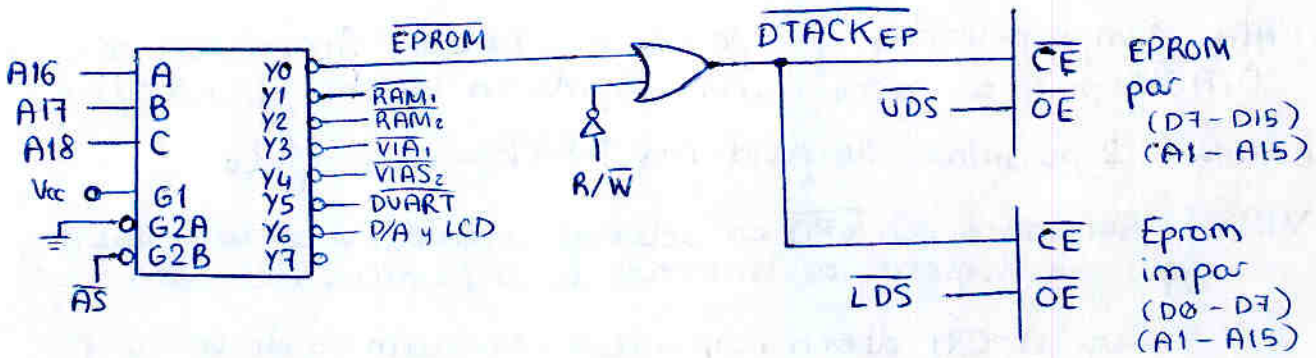
2 pastillas 32k x 8 ROM
4 pastillas 32k x 8 RAM
3 VIAS
1 DUART
1 LCD, 1 DAC
3 latch para nº vector VIA

	UDS Par	LDS Impar	A18 A17 A16	
64kB	EPROM 32k x 8	EPROM 32k x 8	000	0000 0000 0000 0000 0000 0000 / 1111 1111 1111 1111
	RAM 32k x 8	RAM 32k x 8	001	
	slot RAM 32k x 8	slot RAM 32k x 8	010	
	~~3 latch~~	VIA 1 int: 2	011	
	VIA2 int: 3	VIA 3 int: 4	100	
		DUART	101	← DUART siempre en impares para dar nº de vector
	D/A	LCD	110	
			111	← ultimo slot libre para no tener que preocuparnos del CRI

Replicado 32 veces total

los 3 latch no hace falta incluirlos en el mapa, ya que estarán vistos en el circuito de reconocimiento de interrup.

Conexión de ROM



¿que tamaño tendrá la pila del sistema?

suponiendo que los datos ocupan como mucho 32 kB; quedan aún 32 kB libres para la pila (recuerda: la PILA debe estar en RAM)

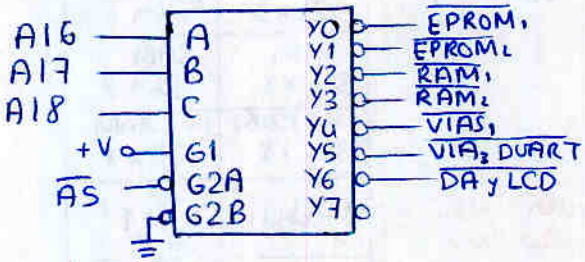
¿cual podría se el tamaño máximo del código del programa?

suponiendo 4kB para las tablas de datos, quedarían 60 kB para el programa.

Si hubieramos 'apretado' un poco el mapa de memoria habrían cabido bien 64 kB más.

X Error: no he tenido en cuenta los 1k para vectores de tablas de excepción !!!
obligado habrá que poner más ROM. Facil:

EPROM 32 k x 8	EPROM 32 k x 8
EPROM 32 k x 8	EPROM 32 k x 8
RAM 32 k x 8	RAM 32 k x 8
zocalo RAM	zocalo RAM
VIA1	VIA2
VIA 3	DUART
D/A	LCD



Nota: si hubieran problemas de generación de estados de espera, convendría poner AS en lógica posterior.

Así tendriamos hueco para un programa de

$$\underbrace{128\,kB}_{\text{Tamaño Total}} - \underbrace{1kB}_{\text{vectores excepcion}} - \underbrace{4kB}_{\text{dato del enunciado}} = 123\,kB$$

Problema 6

1. Suponiendo salida $\overline{CS_{LCD}}$. Conexión del LCD al 68000. No detallar la conexión de Vo

Lineas del LCD

RS : selección de registro : a A1

$R/\overline{W}$: a $R/\overline{W}$

Enable : a $\overline{\overline{CS_{LCD}}}$ (debe incluir la dirección, $\overline{AS}$ y $\overline{LDS}$)

DB0 – DB7 : a D0 – D7

2. Comente, sin incluir esquema alguno, la conexión del panel de participante a la VIA

la VIA tiene conectad:
- teclado 4x4 – 4 pines salida, 4 pines entrada
- pulsador – 1 pin entrada
- 4 visualizadores siete segmentos mediante un único BCD a 7 segmentos

PA0 … PA3 } salidas hacia el teclado

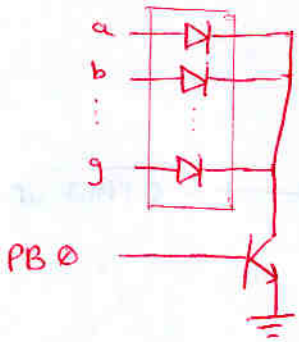
PA4 … PA7 } Entradas desde el teclado

CA1 : Entrada para interrupción del teclado

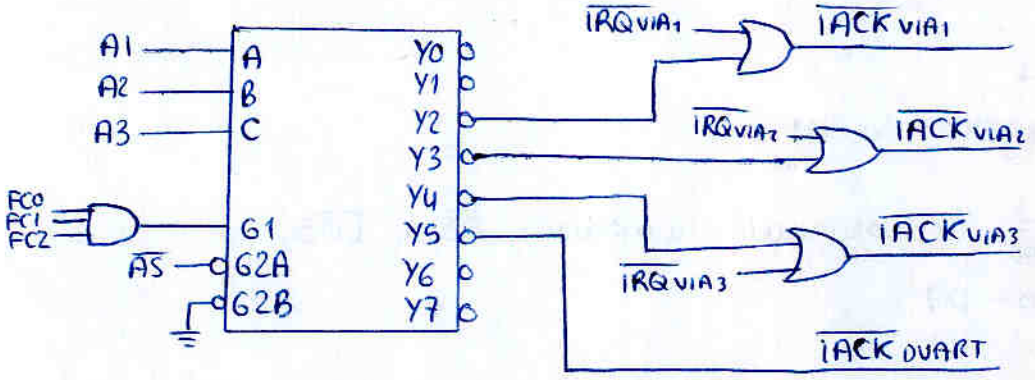
CB1 : Entrada para el pulsador
(o CA2 configurado como entrada)

PB4 … PB7 } conectados a ABCD del convertidor BCD – 7 segmentos

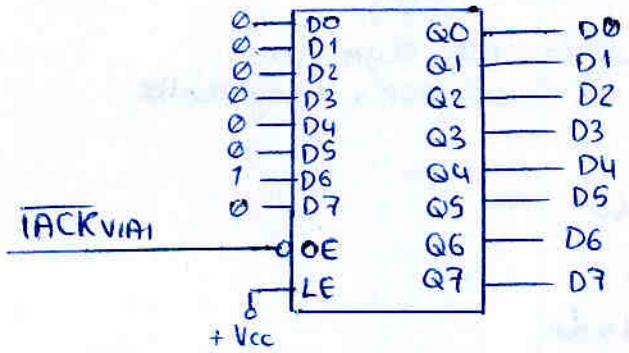
PB0 … PB3 } cada uno conectado para activar uno de los 4 visualizadores. Conectado a la base de un transistor ubicado en el cátodo común de cada uno



3. Circuito de reconocimiento de interrupciones

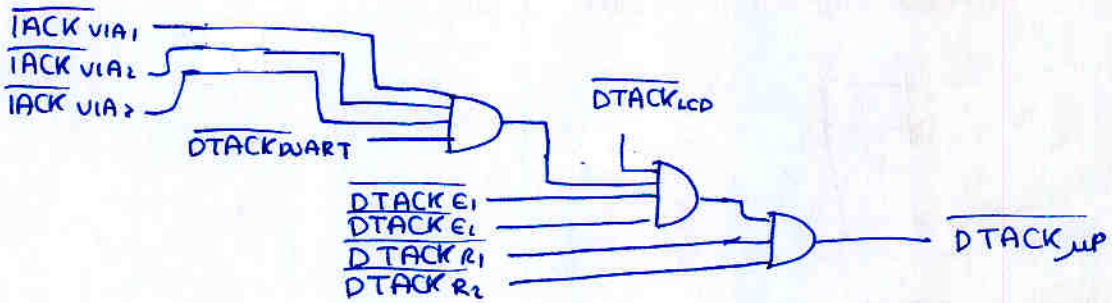


estando, para cada VIA un LATCH con su número de vector
ej. VIA1, nº vector $40



Nota: cuando se quería un número de vector cambiable por software se usaba un latch con OE / ▷CLK en lugar de este.
Y se conectaba a $\overline{CLK}$ el $\overline{DTACK}_{LATCH}$ que estaba mapeado en una dirección de memoria, que sólo debía activarse en ciclo de escritura

Estando los $\overline{IACK}$ de las VIA formando parte del DTACK del µP

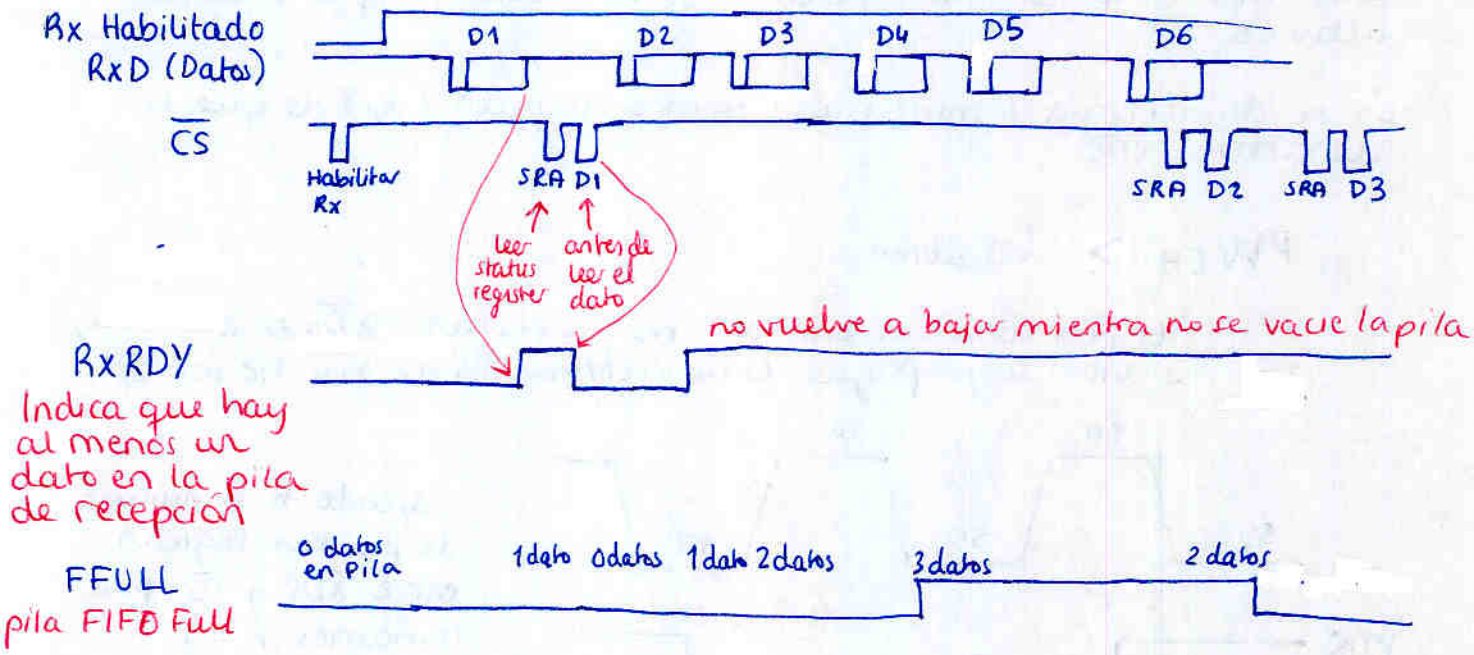


Problema 7

1. Como identificaría el 68000 si la interrupción proviene del pulsador o del 4x4 cuando interrumpa la VIA?

Una vez ha recibido la interrupción, el µP puede comprobar las flags del IFR para ver si la interrupción fué causada por CA1 (teclado) o por CA2 (pulsador)

2 Complete la forma de onda de los bits RxRDY y FFULL



Truco: RxRDY = $\overline{\text{FIFO Empty}}$

Análisis temporal de la conexión al LCD

Garantizar cumplimiento de PW_{EH} en el ciclo de escritura

Ancho mínimo del pulso ENABLE

$$PW_{EHmin} = 230\ ns$$

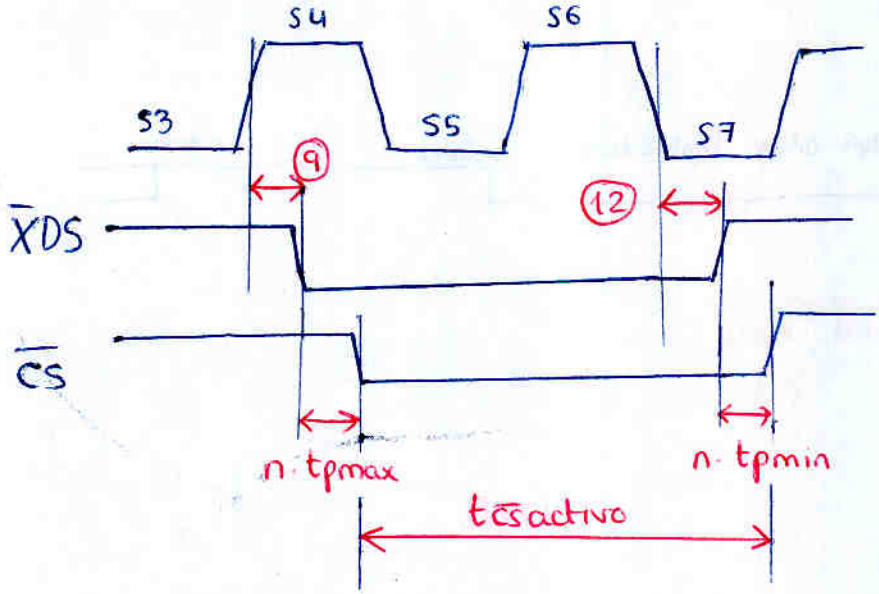
Ya que ENABLE = $\overline{CS}$
hay que estudiar el tiempo en el que $\overline{CS}$ está activo en un ciclo de escritura

- $\overline{CS}$ se activa a partir de Ax, $\overline{AS}$ y $\overline{XDS}$
  Como $\overline{XDS}$ es la última en activarse, será ella la que ~~decidirá~~ activará $\overline{CS}$

- $\overline{CS}$ se desactivará tambien un tiempo despues (t_p) de que se desactive $\overline{XDS}$

$$PW_{EH} < t_{\overline{CS}activo}$$

El peor caso será cuando más se retrase $\overline{XDS}$ en activarse, y antes se propague la desactivación de $\overline{XDS}$ hasta $\overline{CS}$



siendo n el número de puertas lógicas entre $\overline{XDS}$ y $\overline{CS}$ que tomamos n = 1

El peor caso es

$9_{max} = 50\ ns$

$12_{min} = 0\ ns$

$n \cdot t_{pmax} = 20\ ns$

$n \cdot t_{pmin} = 0\ ns$

es decir

$$\boxed{PW_{EH} < 3\ \text{semiciclos} - 9_{max} - n t_{pmax} + 12_{min} + n \cdot t_{pmin}}$$

$$230\ ns < 150\ ns - 50\ ns - 20\ ns + 0\ ns + 0\ ns$$

$$230\ ns < 80\ ns$$

No se cumple.

Habrá que introducir 2 estados de espera (230 ns < 280 ns)

Problema 3 - Enero 2004

- MC68000 a 10 MHz (siempre en modo supervisor)
- 4 MBytes memoria no volátil programa
  + preparado para ampliar la memoria programa al máximo
- 4 MBytes RAM
- 6 VIA (niveles 2, 3, 4, 5, 6, 7)
  Vectores de excepción modificables por el µP escribiendo en RAM

Debe haber RAM en el espacio CRI en las posiciones
111... 11 010
011
100
101
110
111
A3 A2 A1

Recuerda: curiosamente en CRI se hace $\overline{UDS} = \overline{LDS} = 0$ aunque µP sólo lee el byte D0 - D7.

Así que las posiciones que realmente importarán para el nº de vector serán las 6 últimas impares.

- Convertidor A/D (nivel 1) y D/A (conexión optimizada en tiempo)

1. Mapa de memoria

Recuerda: vectores de excepción 1º kByte

vectores 1 y 2: Primeras dos long (SSP y PC) → programa supervisor

vectores 3 a 255: En el resto del 1º kB → usuario supervisar

2M x 8
1M x 16

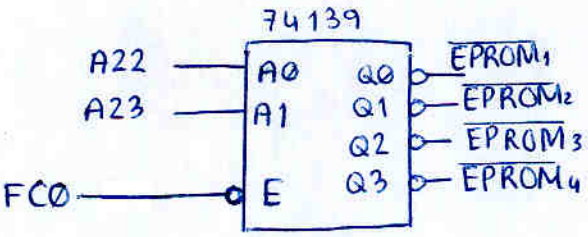
	A23 A22			A23	A22	A21
EPROM1 2M x 8	EPROM1 2M x 8	00	RAM1 1M x 8	RAM1 1M x 8	0	0	0
			VIA 1 int: 2	VIA 2 int: 3	0	0	1
		01	VIA 3 int: 4	VIA 4 int: 5	0	1	0
			VIA 5 int: 6	VIA 6 int: 7	0	1	1
		10	D/A	A/D	1	0	0
					1	0	1
		11			1	1	0
			RAM2 1M x 8	RAM2 1M x 8	1	1	1

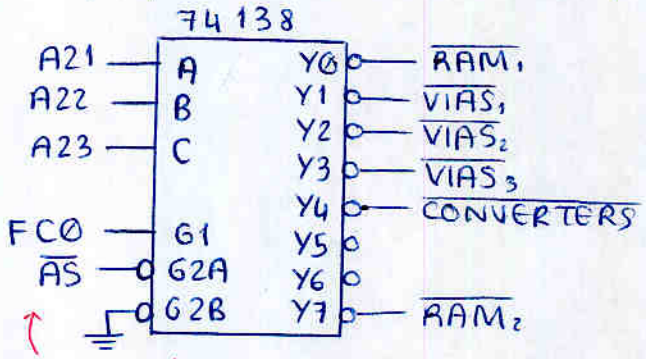
← el primer kB de RAM tendrá vectores de excepción (salvo los 2 primeros)

Programa NO CRI (FC0 = 0)

Datos Y CRI (FC0 = 1)

Obviamos FC2 = 1 ya que el enunciado dice siempre supervisor

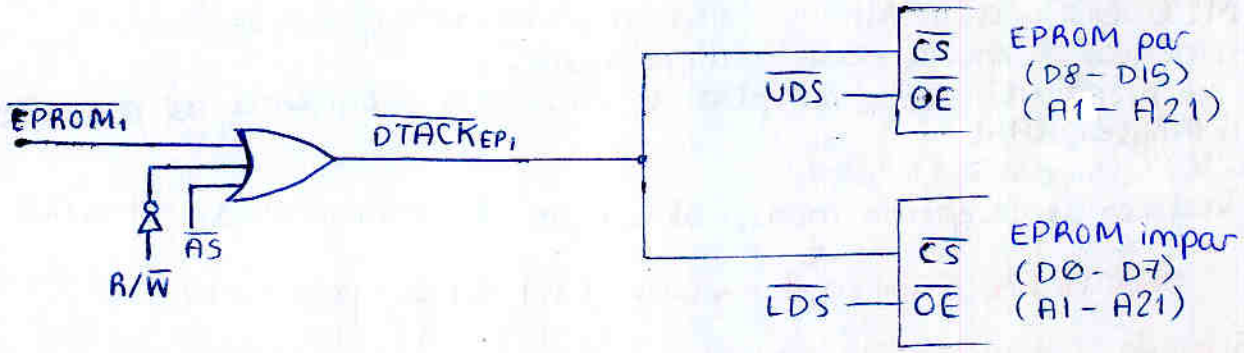




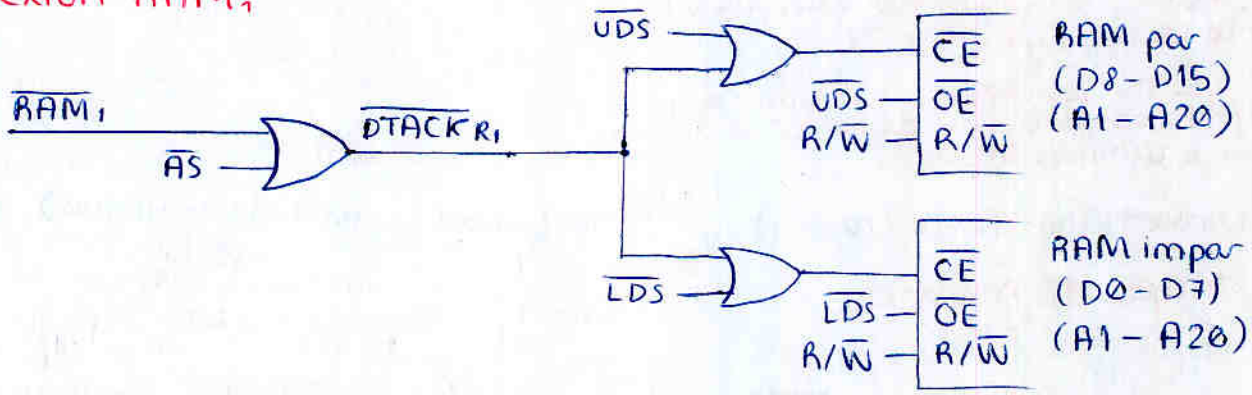
si hubiera problemas temporales $\overline{AS}$ convendría poner lo después del 3 a 8.

• Conexión EPROM₁

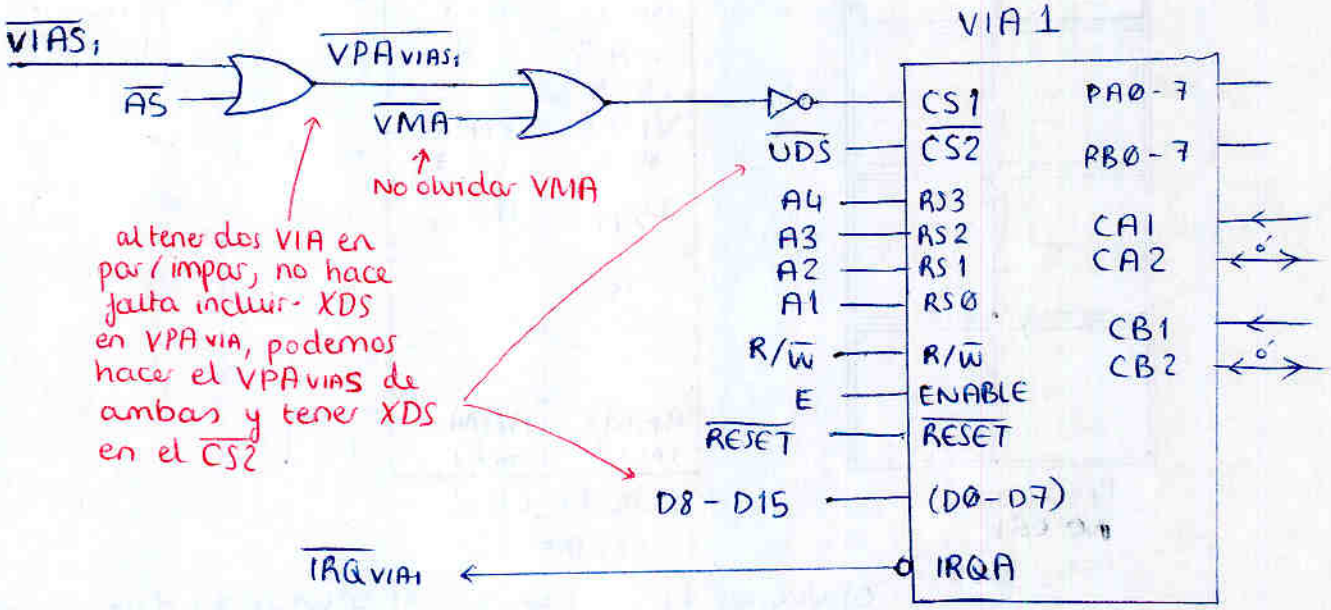
vamos a colocar $\overline{AS}$ detras del decodificador. si no hiciera falta no habria mas que quitarla



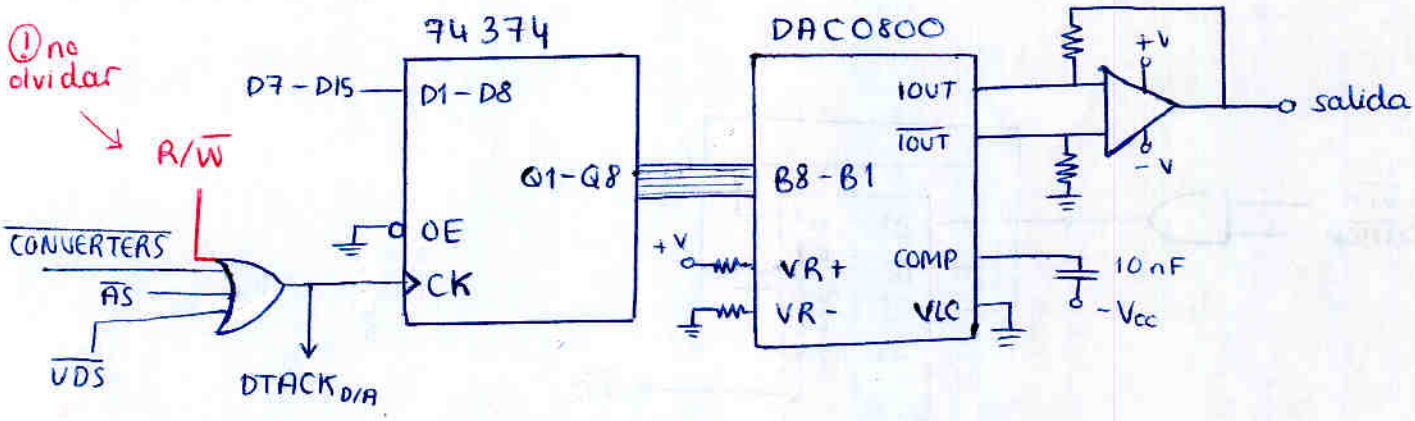
• Conexión RAM₁



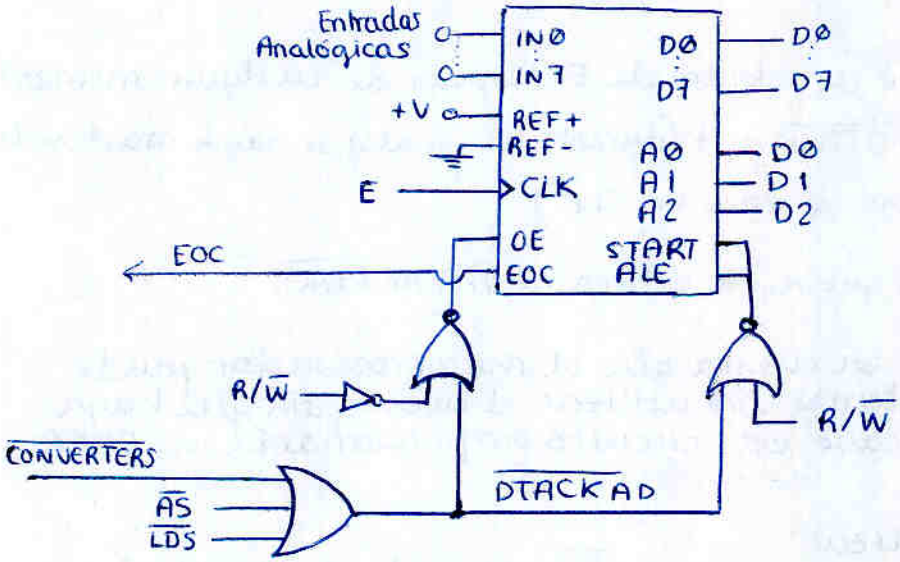
• Conexión VIA₁



· Conexión D/A

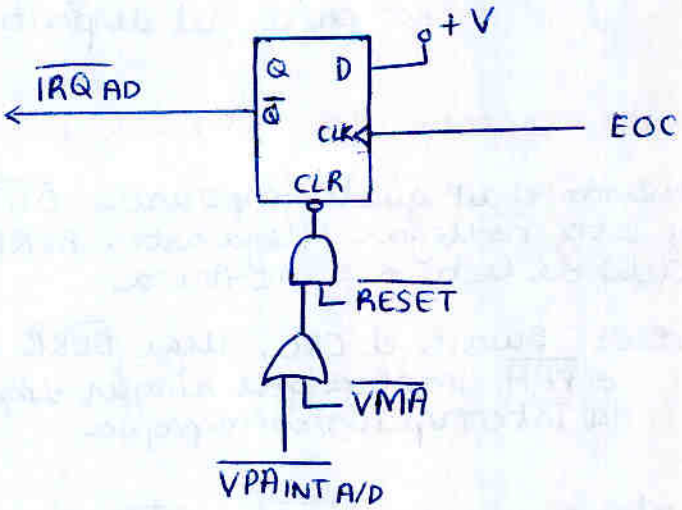


· Conexión A/D

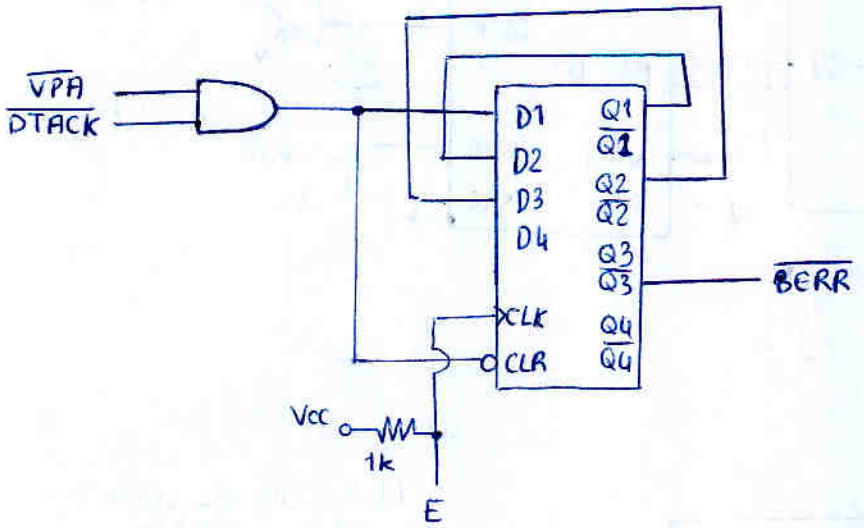


El hecho de haber decidido responer con DTACK significa que habrá que comprobar los tiempos en el ciclo de lectura (muy posiblemente haya un ciclo de espera ya que DTACK depende de $\overline{LDS}$)

Para la interrupción; EOC (flanco) debe generar IRQ (nivel)



Circuito propuesto para la generación de la señal BERR

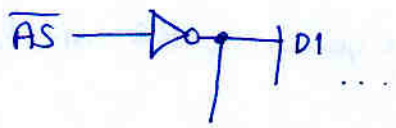


Se activará $\overline{BERR}$ 3 flancos de subida de E despues de cualquier momento en que se cumpla $\overline{VPA} \cdot \overline{DTACK} = 1$ (durante los cuales se haya mantenido $\overline{VPA} \cdot \overline{DTACK}$ a 1, ya que si no se hace un CLR)

$\overline{VPA} \cdot \overline{DTACK} = 1$ siempre que no se activen ni $\overline{VPA}$ ni $\overline{DTACK}$

El circuito no tiene en cuenta que el microprocesador puede estar haciendo ciclos internos sin utilizar el bus y sin que haya error alguno, en cuyo caso este circuito proporcionaría un BERR indebido.

La solución sería utilizar



ya que $\overline{AS}$ sólo vale 1 durante 3 flancos de E en un ciclo en el que se utiliza el BUS y no hay respuesta por parte del dispositivo.

- Listar procesos de excepción que activa la señal BERR

· Error de Bus: cuando el µP queda esperando $\overline{DTACK}$ o $\overline{VPA}$ y éstos no llegan (llega antes BERR) en un ciclo de lectura o escritura

· Interrupción espúrea: Durante el CRC, llega $\overline{BERR}$ antes que $\overline{DTACK}$ o $\overline{VPA}$, implica que ningun dispositivo reconoce la interrupción como propia.

Nota: el ciclo de relanzamiento ($\overline{BERR}$ y $\overline{HALT}$) no se considera proceso de excepción (ya que no sigue los pasos de un proceso de excepción, que incluye obtener número de vector)

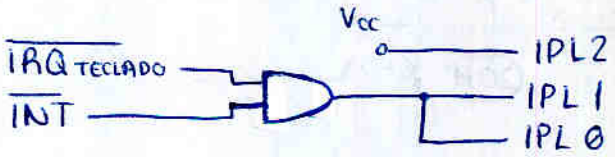
Problema 4

1. Diseñar circuito de reconocimiento de interrupciones

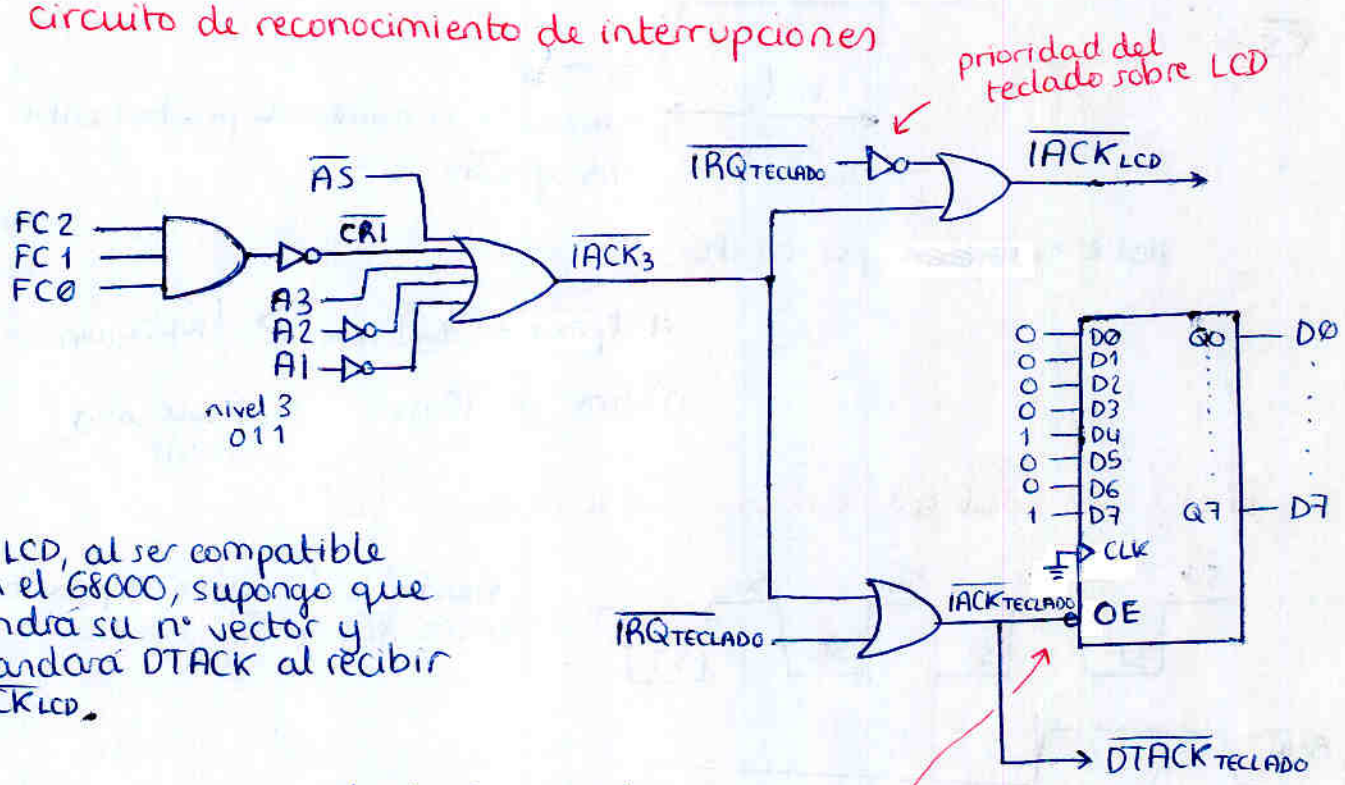
Sabiendo que los unicos dispositivos que interrumpen son:

- Teclado con salida $\overline{IRQ}$. nivel 3. Deve devolver nº vector \$90
- PCF 8584 (LCD). nivel 3. Interrupción vectorizada.
  - ↳ compatible con 68000 ($\overline{INT}$, $\overline{IACK}$ y $\overline{DTACK}$)
  - ↳ unica diferencia con DUART – $\overline{IACK}$ provoca la desactivación de $\overline{INT}$ pasado un tiempo t_{ALIE}

nivel 3: $\overline{IPL2}$: 1
$\overline{IPL1}$: 0
$\overline{IPL0}$: 0



2. Circuito de reconocimiento de interrupciones

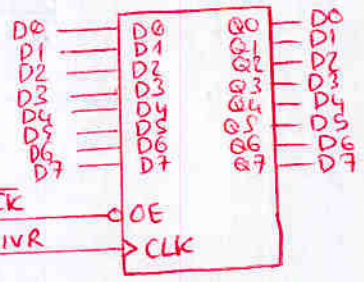


El LCD, al ser compatible con el 68000, supongo que pondrá su nº vector y mandará DTACK al recibir $\overline{IACK}$ LCD.

He puesto prioridad al teclado sobre el LCD para evitar conflictos (que no puedan ambos emitir su número de vector por D0 – D7)

No podriamos haberlo hecho al revés (utilizando $\overline{INT}$ LCD ya que el enunciado dice que éste se desactiva)

CLK a masa porque no queremos cambiar el nº de vector; si quisieramos, es típico hacer

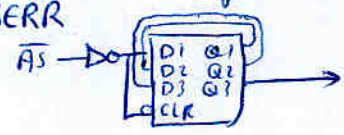


mapeado en una dirección de memoria

Registro vector interrupción

ya que el teclado puede tener rebotes

Interrupciones espúreas: es especialmente importante hacer circuito generación de BERR

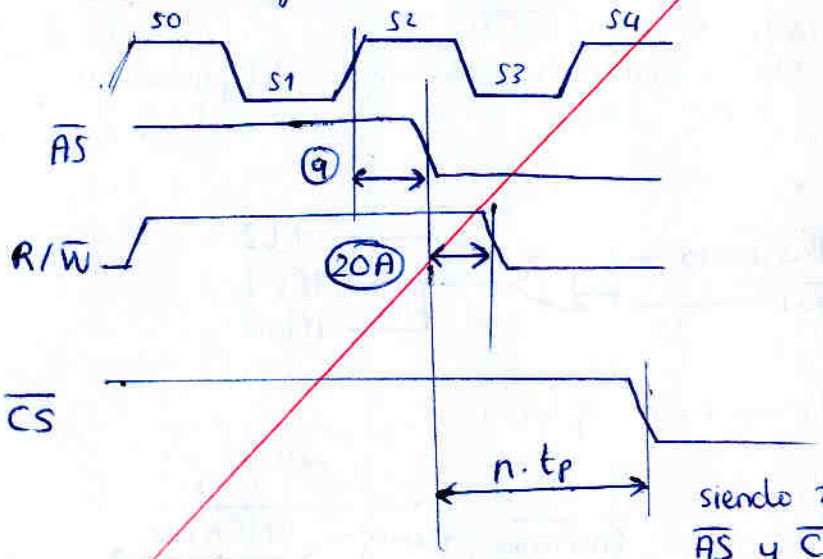


4. Comprobar inecuación que garantice parámetro t_{WLCL} del PCF8584 en el peor caso posible

$t_{pmin} = 5ns$
$t_{p\,max} = 20\,ns$

t_{WLCL} es el tiempo desde que se pone $R/\overline{W}$ a 0 hasta que se pone $\overline{CS}$ a 0 en el ciclo de escritura.

Del catálogo del 68000



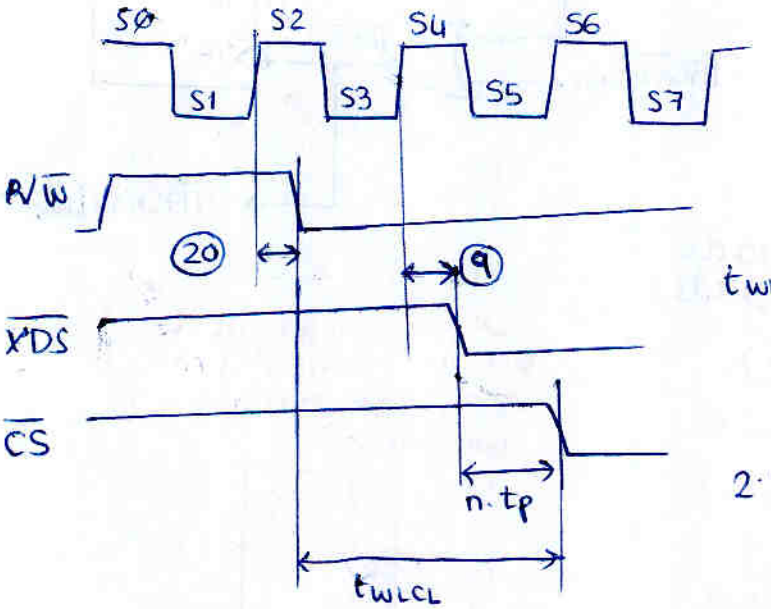
siendo n el número de puertas entre $\overline{AS}$ y $\overline{CS}$

Debe cumplirse por tanto

$$n \cdot t_{pmin} - (20A)_{max} > t_{WLCL\,min}$$

$$n \cdot 5ns - 10ns > t_{WLCL\,min}$$
$$10ns$$

Error. La señal que activa $\overline{CS}$ última es $\overline{XDS}$



siendo n el número de puertas entre $\overline{XDS}$ y $\overline{CS}$ $n = 1$

$$t_{WLCL} = 2 \cdot S - (20) + (9) + n \cdot t_p > t_{WLCL\,min}$$

$$2 \cdot 50ns - (20)_{max} + (9)_{min} + n \cdot t_{pmin} > t_{WLCL\,min}$$

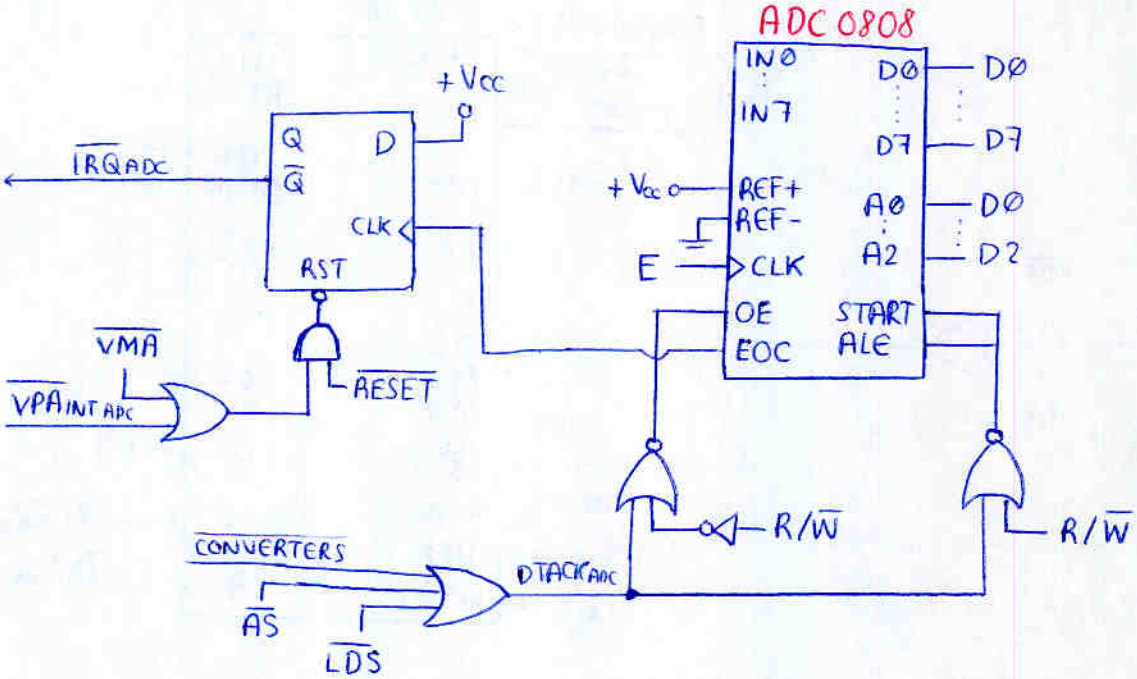
$$2 \cdot 50ns - 45ns + 3ns + 1 \cdot 5ns > 10\,ns$$

$$63\,ns > 10\,ns$$

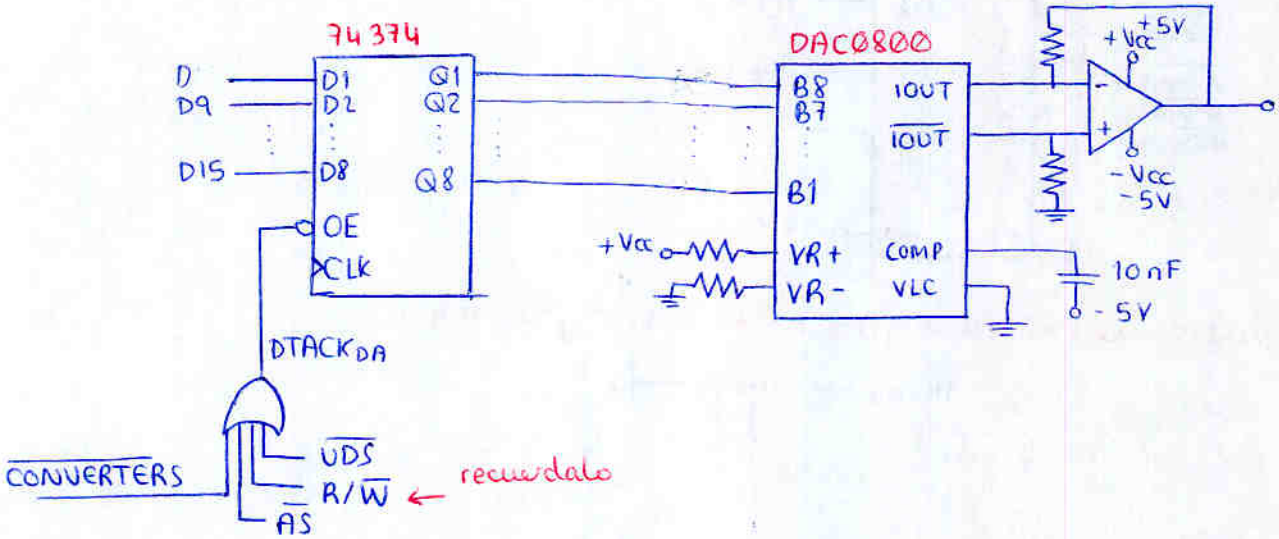
se cumple la ecuación

• Conexión A/D y D/A

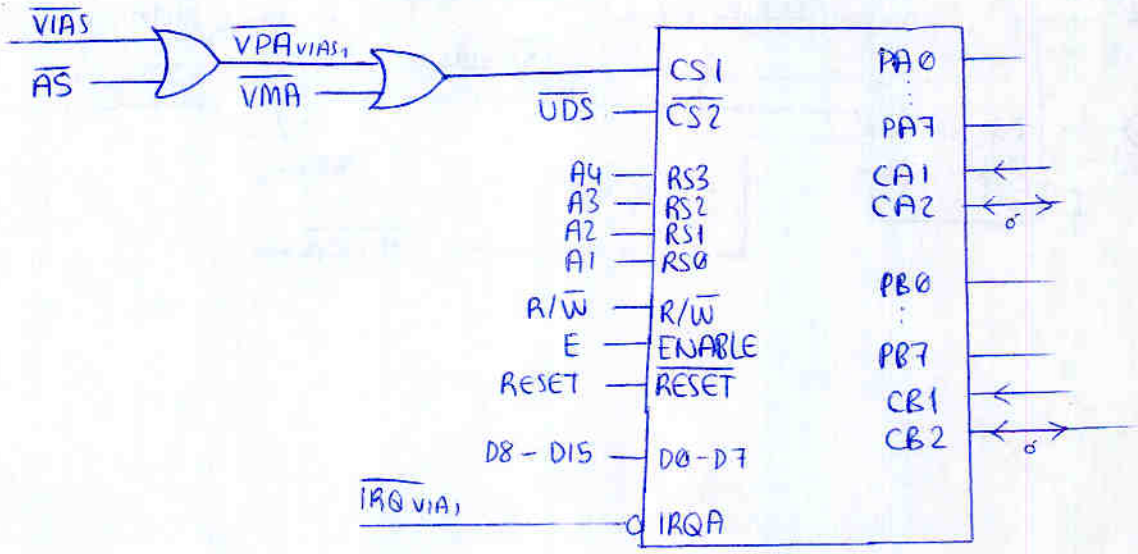
- A/D



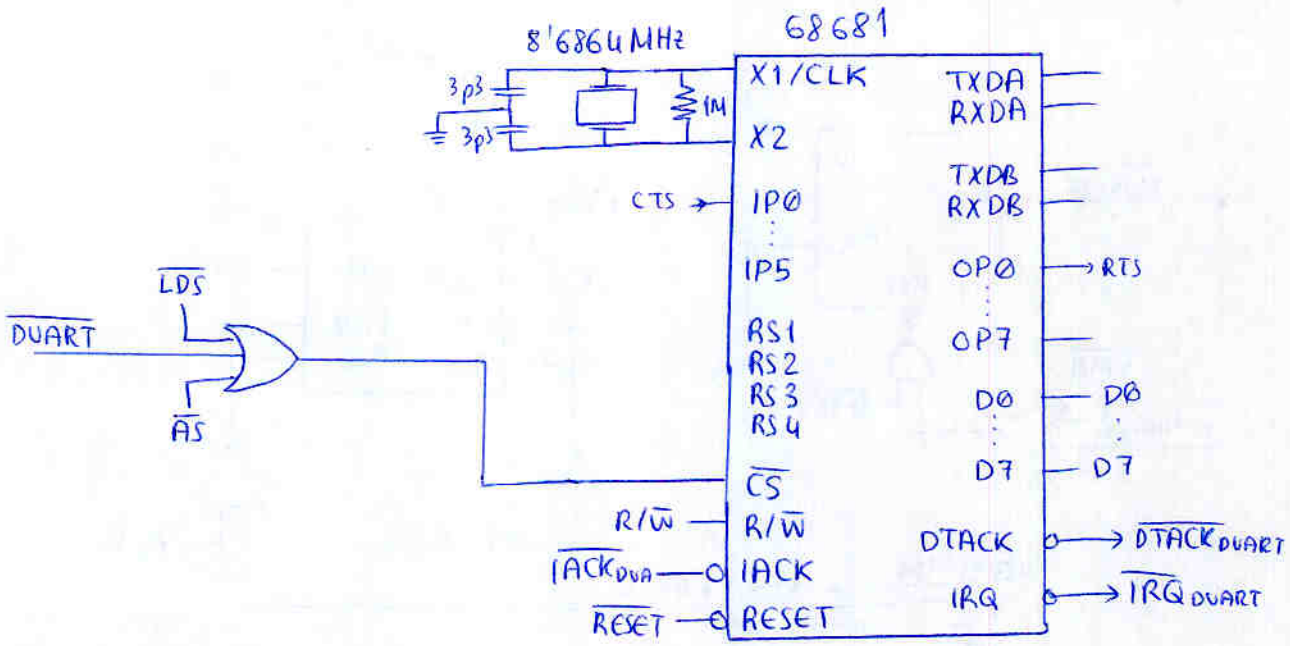
- D/A



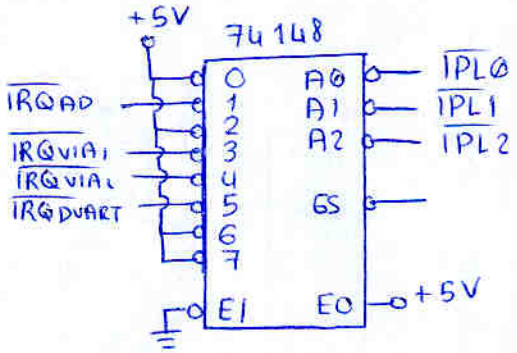
- VIA



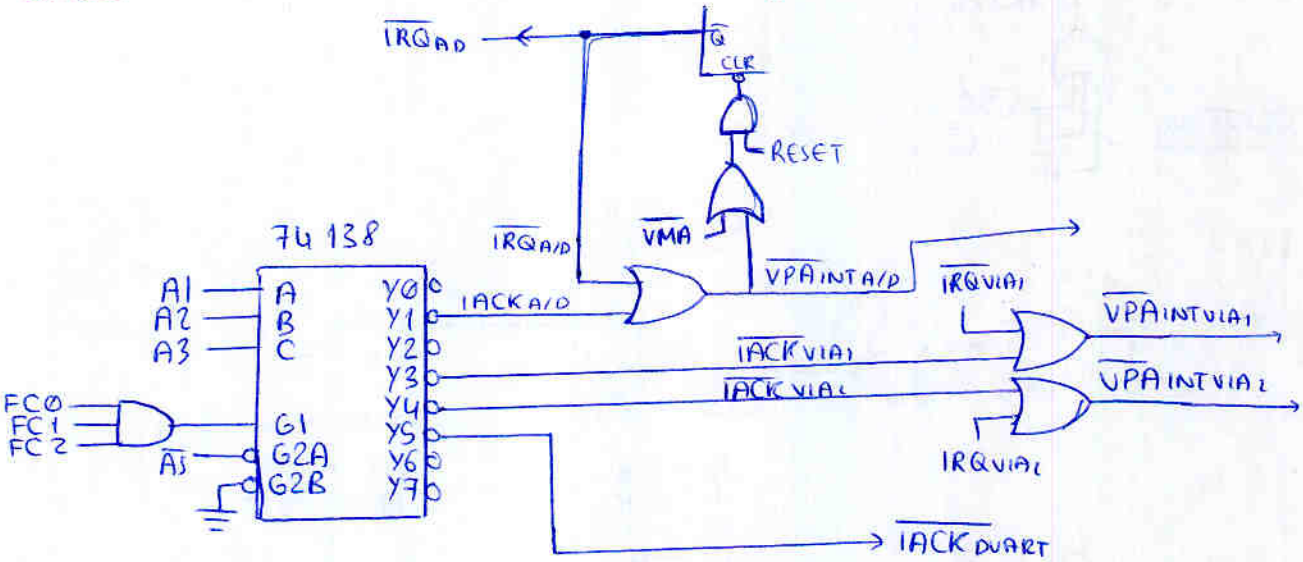
- DUART



- circuito de petición de interrupciones



- circuito de reconocimiento de interrupciones



Problemas SED

Problema 1 del libro

MC68000 a 8MHz

- 128 kbytes EPROM a partir de dir 0
- 128 kbytes RAM ~~a partir de dir~~ al final del mapa
- 2 VIAs (niv. 3 y 4)
- 1 DUART (niv. 5)
- Convertidor A/D (niv 1) y D/A
- Pulsador que genera interrupción nivel 6 ABORT

$t_p = 20ns$
$T_{acc} = 100ns$
$t_{hold} = 0ns$

RAM 32k x 8
EPROM 32k x 8

128 kbytes – 17 líneas 32kbytes – 15 líneas

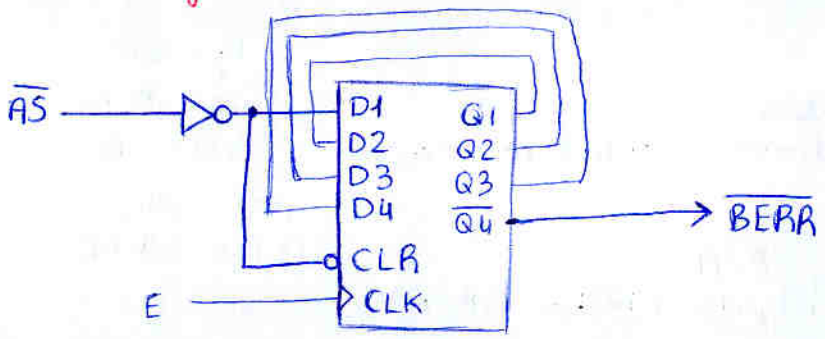
A16
0000 0000 0000 0000 0000 0000
0000 0001 1111 1111 1111 1111

Como sobra bastante espacio haré bloques de 64 k x 8 (tamaño de un pastilla)

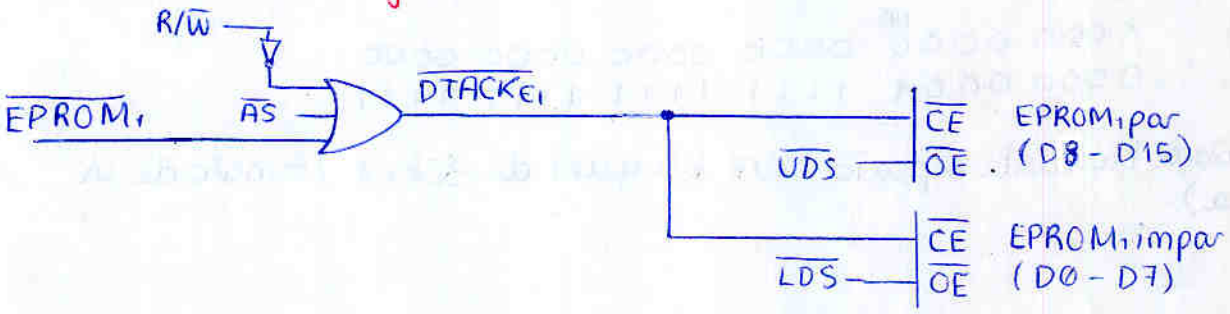
			A17 A16 A15	
EPROM1 32k x 8	EPROM1 32k x 8	xxxx xx	00 0	000 0000 0000 0000 111 1111 1111 1111
EPROM2 32k x 8	EPROM2 32k x 8	xxxx xx	0 0 1	000 0000 0000 0000 111 1111 1111 1111
VIA 1 int: 3	VIA 2 int: 4	xxxx xx	01 0	xxx xxxx xxx 0 0000 1 1111
	DUART int: 5		0 1 1	
D/A	A/D int: 1		1 0 0	
			1 0 1	
RAM1 32k x 8	RAM1 32k x 8		1 1 0	
RAM2 32k x 8	RAM2 32k x 8		1 1 1	
replicado 64 veces				

- La EPROM y la RAM responden con DTACK ya que es el método más rápido y son dispositivos de mucho acceso.
- La DUART responde con DTACK puesto que es de la familia del 68000
- La VIA responde con VPA puesto que es de la familia del 6800
- El convertidor D/A responde con DTACK ya que se accede a el a través de un latch que no suele presentar problemas temporales.
- El convertidor A/D en principio podría contestar con DTACK, pero si vemos que hay problemas temporales que requieren introducir estados de espera sería mejor utilizar VPA para ahorrar hardware a costa del tiempo de acceso a un dispositivo de raro acceso.

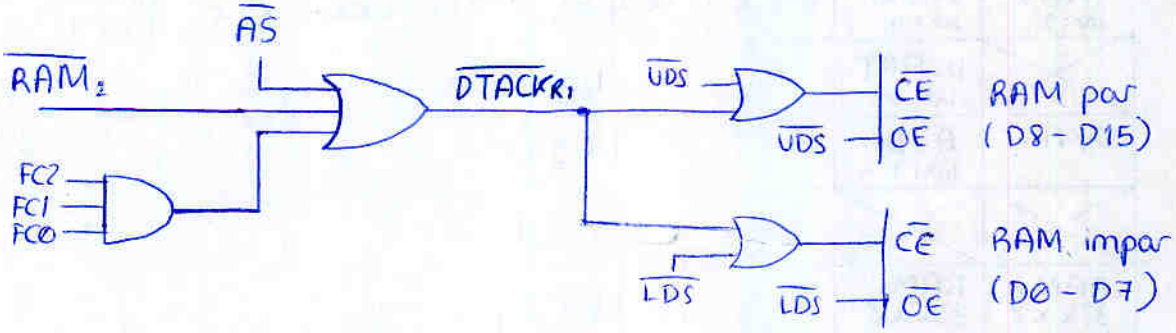
2. Circuito de generación de error de bus



3. Conexión EPROM y RAM

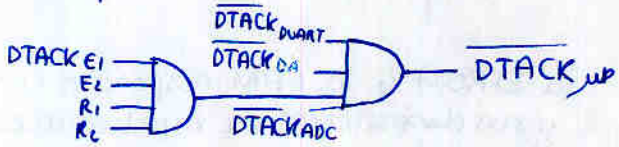


Por estar la RAM en las ultimas posiciones habrá que tener cuidado con el CR/



4. Verificación temporal

En este caso las direcciones estan sólo un tp más alejadas de DTACK y sin embargo tienen todo un semiciclo más de tiempo.
Por tanto la más limitante es $\overline{AS}$.



Ciclo de lectura

- Aparición de estados de espera

$$3S_{S1/S2 \to S4/S5} > t_{S1/S2 \to \overline{AS}} + t_{\overline{AS} \to \overline{DTACK}} + t_{setup\,DTACK}$$

$$3 \cdot \frac{1}{2}\left(\frac{1}{8MHz}\right) > 9 + 3\,t_p + 47$$

$$3 \cdot 62'5ns > 60ns + 3 \cdot 20ns + 10ns$$

$$187'5 > 130ns$$

- Necesidad de estados de espera

$$5S > t_{S1/S2 - \overline{AS}} + t_{\overline{AS} - \overline{CS}} + T_{acc} + t_{setup\,DATA}$$

$$5 \cdot 62'5 > 9 + 2 \cdot t_p + 100 + 27$$

$$312'5 > 60 + 2 \cdot 20 + 100 + 10$$

$$312'5 > 210$$

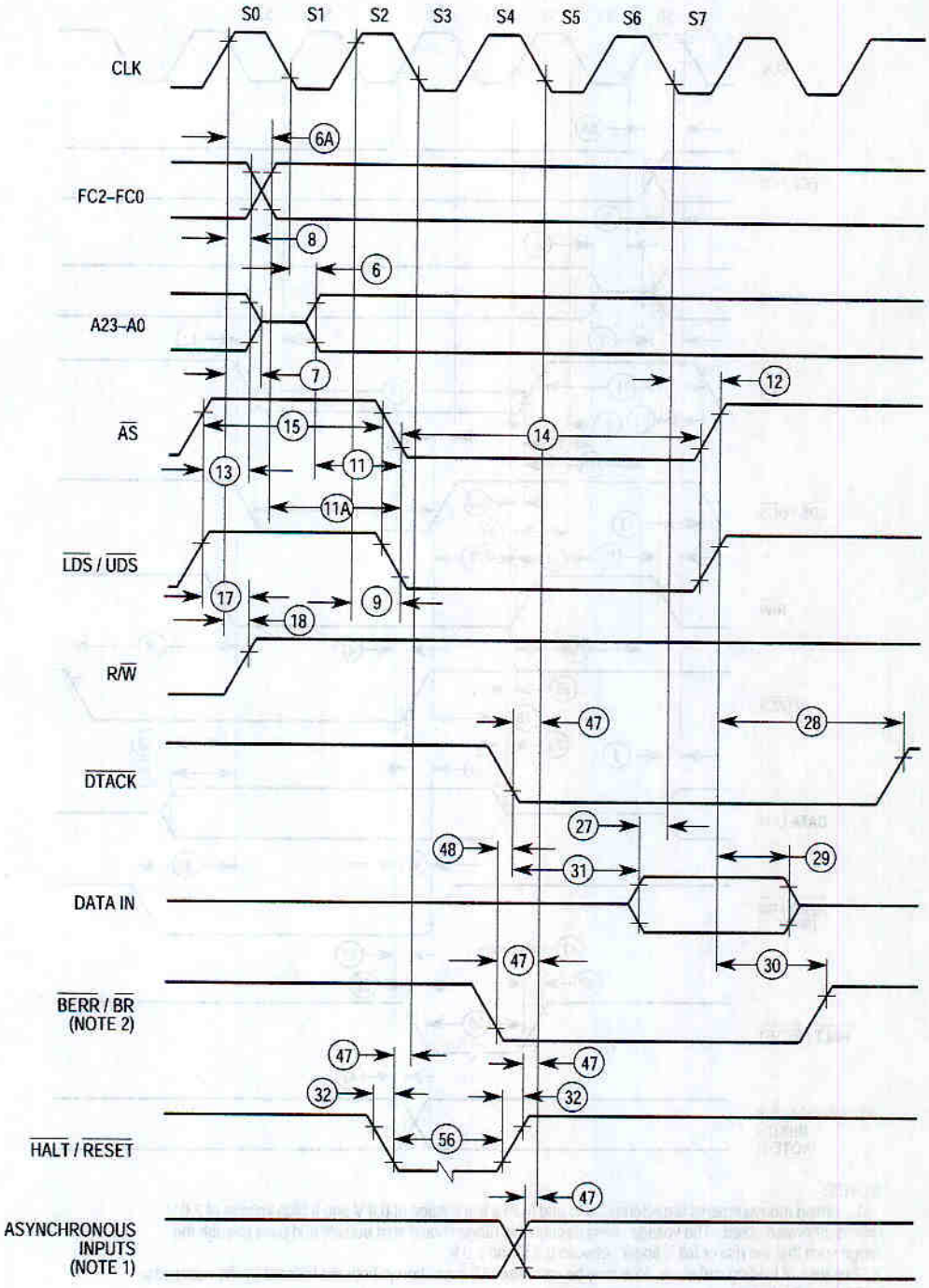
↑ en el caso de EPROM sólo uno

Ciclo escritura

$$25 > t_{\overline{XDS} - \overline{CS}} + t_{hold\,mem}$$

$$40ns > 20 + 0$$

LECTURA

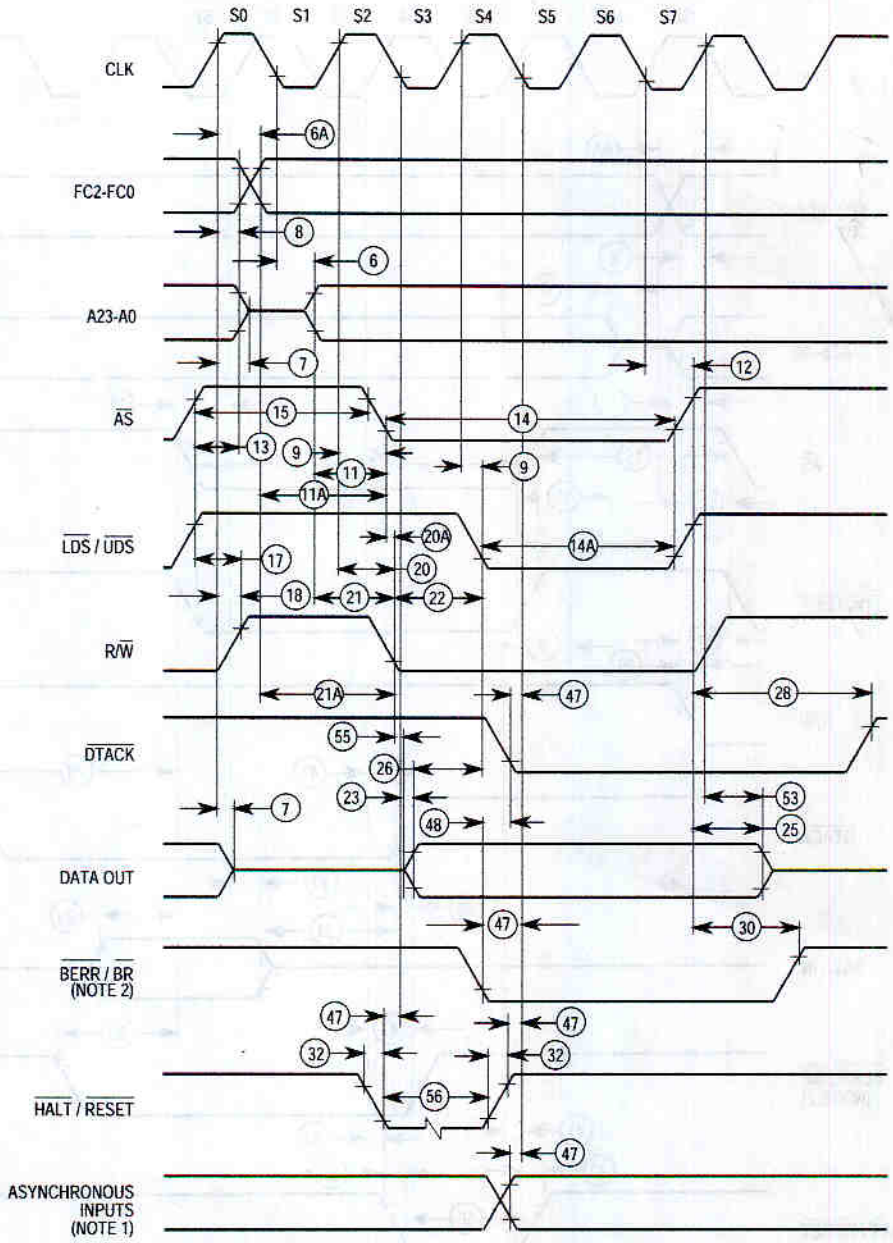


NOTES:

1. Setup time for the asynchronous inputs $\overline{IPL2}$–$\overline{IPL0}$ and $\overline{AVEC}$ (#47) guarantees their recognition at the next falling edge of the clock.
2. $\overline{BR}$ need fall at this time only to insure being recognized at the end of the bus cycle.
3. Timing measurements are referenced to and from a low voltage of 0.8 V and a high voltage of 2.0 V, unless otherwise noted. The voltage swing through this range should start outside and pass through the range such that the rise or fall is linear between 0.8 V and 2.0 V.

Figure 10-12. MC68EC000 Read Cycle Timing Diagram

ESCRITURA



NOTES:

1. Timing measurements are referenced to and from a low voltage of 0.8 V and a high voltage of 2.0 V, unless otherwise noted. The voltage swing through this range should start outside and pass through the range such that the rise or fall is linear between 0.8 V and 2.0 V.
2. Because of loading variations, R/W may be valid after AS even though both are initiated by the rising edge of S2 (specification #20A).

Figure 10-13. MC68EC000 Write Cycle Timing Diagram

Num	Characteristic	8 MHz*		10 MHz*		12.5 MHz*		16.67 MHz 12F		16 MHz		20 MHz*		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	
6	Clock Low to Address Valid	—	62	—	50	—	50	—	50		30	—	25	ns
6A	Clock High to FC Valid	—	62	—	50	—	45	—	45	0	30	0	25	ns
7	Clock High to Address, Data Bus High Impedance (Maximum)	—	80	—	70	—	60	—	50		50	—	42	ns
8	Clock High to Address, FC Invalid (Minimum)	0	—	0	—	0	—	0	—	0	—	0	—	ns
9[1]	Clock High to $\overline{AS}$, $\overline{DS}$ Asserted	3	60	3	50	3	40	3	40	3	30	3	25	ns
11[2]	Address Valid to $\overline{AS}$, $\overline{DS}$ Asserted (Read)/AS Asserted (Write)	30	—	20	—	15	—	15	—	15	—	10	—	ns
11A[2]	FC Valid to $\overline{AS}$), $\overline{DS}$ Asserted (Read)/$\overline{AS}$) Asserted (Write)	90	—	70	—	60	—	30	—	45	—	40	—	ns
12[1]	Clock Low to $\overline{AS}$, $\overline{DS}$ Negated	—	62	—	50	—	40	—	40	3	30	3	25	ns
13[2]	$\overline{AS}$, $\overline{DS}$ Negated to Address, FC Invalid	40	—	30	—	20	—	10	—	15	—	10	—	ns
14[2]	$\overline{AS}$and $\overline{DS}$ Read) Width Asserted	270	—	195	—	160	—	120	—	120	—	100	—	ns
14A	$\overline{DS}$ Width Asserted (Write)	140		95		80		60		60	—	50	—	ns
15[2]	$\overline{AS}$, $\overline{DS}$ Width Negated	150	—	105	—	65	—	60	—	60	—	50	—	ns
16	Clock High to Control Bus High Impedance	—	80	—	70	—	60	—	50	—	50	—	42	ns
17[2]	$\overline{AS}$, $\overline{DS}$ Negated to R/$\overline{W}$ Invalid	40	—	30	—	20	—	10	—	15	—	10	—	ns
18[1]	Clock High to R/$\overline{W}$ High (Read)	0	55	0	45	0	40	0	40	0	30	0	25	ns
20[1]	Clock High to R/$\overline{W}$ Low (Write)	0	55	0	45	0	40	0	40	0	30	0	25	ns
20A[2,6]	$\overline{AS}$ Asserted to R/$\overline{W}$ Valid (Write)	—	10	—	10	—	10	—	10	—	10	—	10	ns
21[2]	Address Valid to R/$\overline{W}$ Low (Write)	20	—	0	—	0	—	0	—	0	—	0	—	ns
21A[2]	FC Valid to R/$\overline{W}$ Low (Write)	60	—	50	—	30	—	20	—	30	—	25	—	ns
22[2]	R/$\overline{W}$ Low to $\overline{DS}$ Asserted (Write)	80	—	50	—	30	—	20	—	30	—	25	—	ns
23	Clock Low to Data-Out Valid (Write)	—	62	—	50	—	50	—	550	—	30	—	25	ns
25[2]	$\overline{AS}$, $\overline{DS}$) Negated to Data-Out Invalid (Write)	40[1] 0	—	30	—	20	—	15	—	15	—	10	—	ns

Num	Characteristic	8 MHz*		10 MHz*		12.5 MHz*		16.67 MHz 12F		16 MHz		20 MHz*		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	
26[2]	Data-Out Valid to $\overline{DS}$ Asserted (Write)	40	—	30	—	20	—	15	—	15	—	10	—	ns
27[5]	Data-In Valid to Clock Low (Setup Time on Read)	10	—	10	—	10	—	7	—	5	—	5	—	ns
27A[5]	Late $\overline{BERR}$ Asserted to Clock Low (setup Time)	45	—	45	—	45	—	—	—	—	—	—	—	ns
28[2]	$\overline{AS}$, $\overline{DS}$ Negated to $\overline{DTACK}$ Negated (Asynchronous Hold)	0	240[1] 1	0	190	0	150	0	110	0	110	0	95	ns
28A	$\overline{AS}$, $\overline{DS}$ Negated to Data-In High Impedance	—	187	—	150	—	120	—	110	—	110	—	95	ns
29	$\overline{AS}$, $\overline{DS}$ Negated to Data-In Invalid (Hold Time on Read)	0	—	0	—	0	—	0	—	0	—	0	—	ns
29A	$\overline{AS}$, $\overline{DS}$ Negated to Data-In High Impedance	—	187	—	150	—	120	—	90	—	90	—	75	ns
30	$\overline{AS}$, $\overline{DS}$) Negated to $\overline{BERR}$ Negated	0	—	0	—	0	—	0	—	0	—	0	—	ns
31[2,5]	$\overline{DTACK}$ Asserted to Data-In Valid (Setup Time)	—	90	—	65	—	50	—	40	—	50	—	42	ns
32	$\overline{HALT}$) and $\overline{RESET}$ Input Transition Time	0	200	0	200	0	200	0	150	—	150	0	150	ns
33	Clock High to $\overline{BG}$ Asserted	—	62	—	50	—	40	—	40	0	30	0	25	ns
34	Clock High to $\overline{BG}$ Negated	—	62	—	50	—	40	—	40	0	30	0	25	ns
35	$\overline{BR}$ Asserted to $\overline{BG}$ Asserted	1.5	3.5	1.5	3.5	1.5	3.5	1.5	3.5	1.5	3.5	1.5	3.5	Clks
36[7]	$\overline{BR}$ Negated to$\overline{BG}$ Negated	1.5	3.5	1.5	3.5	1.5	3.5	1.5	3.5	1.5	3.5	1.5	3.5	Clks
37	$\overline{BGACK}$ Asserted to $\overline{BG}$ Negated	1.5	3.5	1.5	3.5	1.5	3.5	1.5	3.5	1.5	3.5	1.5	3.5	Clks
37A[8]	$\overline{BGACK}$ Asserted to $\overline{BR}$ Negated	20	1.5 Clks	20	1.5 Clks	20	1.5 Clks	10	1.5 Clks	10	1.5 Clks	10	1.5 Clks	ns
38	$\overline{BG}$ Asserted to Control, Address, Data Bus High Impedance ($\overline{AS}$ Negated)	—	80	—	70	—	60	—	50	—	50	—	42	ns
39	$\overline{BG}$ Width Negated	1.5	—	1.5	—	1.5	—	1.5	—	1.5	—	1.5	—	clks
40	Clock Low to $\overline{VMA}$ Asserted	—	70	—	70	—	70	—	50	—	50	—	40	ns
41	Clock Low to E Transition	—	55[12]	—	45	—	35	—	35	—	35	—	30	ns
42	E Output Rise and Fall Time	—	15	—	15	—	15	—	15	—	15	—	12	ns
43	$\overline{VMA}$ Asserted to E High	200	—	150	—	90	—	80	—	80	—	60	—	ns
44	$\overline{AS}$, $\overline{DS}$ Negated to $\overline{VPA}$ Negated	0	120	0	90	0	70	0	50	0	50	0	42	ns
45	E Low to Control, Address Bus Invalid (Address Hold Time)	30	—	10	—	10	—	10	—	10	—	10	—	ns
46	$\overline{BGACK}$ Width Low	1.5	—	1.5	—	1.5	—	1.5	—	1.5	—	1.5	—	ns

Num	Characteristic	8 MHz*		10 MHz*		12.5 MHz*		16.67 MHz 12F		16 MHz		20 MHz**		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	
47[5]	Asynchronous Input Setup Time	10	—	10	—	10	—	10	—	5	—	5	—	ns
48[2,3]	$\overline{BERR}$ Asserted to $\overline{DTACK}$ Asserted	20	—	20	—	20	—	10	—	10	—	10	—	ns
48[2,3,5]	$\overline{DTACK}$ Asserted to $\overline{BERR}$ Asserted (MC68010 Only)	—	80	—	55	—	35	—	—	—	—	—	—	ns
49[9]	$\overline{AS}$, $\overline{DS}$, Negated to E Low	-70	70	-55	55	-45	45	-35	35	-35	35	–30	30	ns
50	E Width High	450	—	350	—	280	—	220	—	220	—	190	—	ns
51	E Width Low	700	—	550	—	440	—	340	—	340	—	290	—	ns
53	Data-Out Hold from Clock High	0	—	0	—	0	—	0	—	0	—	0	—	ns
54	E Low to Data-Out Invalid	30	—	20	—	15	—	10	—	10	—	5	—	ns
55	R/$\overline{W}$ Asserted to Data Bus Impedance Change	30	—	20	—	10	—	0	—	0	—	0	—	ns
56[4]	$\overline{HALT}$ ($\overline{RESET}$ Pulse Width	10	—	10	—	10	—	10	—	10	—	10	—	clks
57	$\overline{BGACK}$ Negated to $\overline{AS}$, $\overline{DS}$, R/$\overline{W}$ Driven	1.5	—	1.5	—	1.5	—	1.5	—	1.5	—	1.5	—	clks
57A	$\overline{BGACK}$ Negated to FC, $\overline{VMA}$ Driven	1	—	1	—	1	—	1	—	1	—	1	—	clks
58[7]	$\overline{BR}$ Negated to $\overline{AS}$, $\overline{DS}$, R/$\overline{W}$ Driven	1.5	—	1.5	—	1.5	—	1.5	—	1.5	—	1.5	—	clks
58A[7]	$\overline{BR}$ Negated to FC, $\overline{AS}$ Driven	1	—	1	—	1	—	1	—	1	—	1	—	clks

*These specifications represent improvement over previously published specifications for the 8-, 10-, and 12.5-MHz MC68000 and are valid only for product bearing date codes of 8827 and later.

** This frequency applies only to MC68HC000 and MC68HC001.

NOTES:

1. For a loading capacitance of less than or equal to 50 pF, subtract 5 ns from the value given in the maximum columns.
2. Actual value depends on clock period.
3. If #47 is satisfied for both $\overline{DTACK}$ and $\overline{BERR}$, #48 may be ignored. In the absence of $\overline{DTACK}$, $\overline{BERR}$ is an asynchronous input using the asynchronous input setup time (#47).
4. For power-up, the MC68000 must be held in the reset state for 100 ms to allow stabilization of on-chip circuitry. After the system is powered up, #56 refers to the minimum pulse width required to reset the processor.
5. If the asynchronous input setup time (#47) requirement is satisfied for $\overline{DTACK}$, the $\overline{DTACK}$ asserted to data setup time (#31) requirement can be ignored. The data must only satisfy the data-in to clock low setup time (#27) for the following clock cycle.
6. When $\overline{AS}$ and R/$\overline{W}$ are equally loaded (±20;pc), subtract 5 ns from the values given in these columns.
7. The processor will negate $\overline{BG}$ and begin driving the bus again if external arbitration logic negates $\overline{BR}$ before asserting $\overline{BGACK}$.
8. The minimum value must be met to guarantee proper operation. If the maximum value is exceeded, $\overline{BG}$ may be reasserted.
9. The falling edge of S6 triggers both the negation of the strobes ($\overline{AS}$ and $\overline{DS}$) and the falling edge of E. Either of these events can occur first, depending upon the loading on each signal. Specification #49 indicates the absolute maximum skew that will occur between the rising edge of the strobes and the falling edge of E.
10. 245 ns for the MC68008.
11. 50 ns for the MC68008
12. 50 ns for the MC68008.

Item	Characteristic	[illegible]		[illegible]		[illegible]		[illegible]		[illegible]		[illegible]		[illegible]		[illegible]	
		Min	Max	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible] Time	[illegible]	—	[illegible]	—	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	—	[illegible]	—	[illegible]	—
[illegible]	[illegible] Address [illegible]	20	—	20	—	[illegible]	—	15	[illegible]	[illegible]	[illegible]	[illegible]	—	[illegible]	—	[illegible]	—
[illegible]	[illegible] Address [illegible]	[illegible]	[illegible]	—	78	[illegible]	[illegible]	[illegible]	[illegible]	—	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible] Address [illegible]	[illegible]	70	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	20	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	—	[illegible]	—	[illegible]	—	[illegible]	[illegible]	[illegible]	—	[illegible]	[illegible]	[illegible]	—	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	—	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	—	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	—	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	—	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	—	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	—	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	—	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	—	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	—	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	—	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	—	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

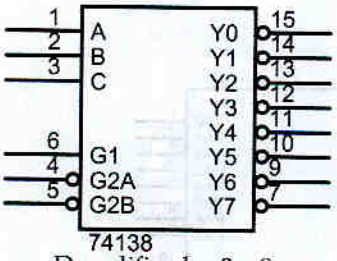
[illegible]

[illegible]

[illegible]

NOTES:

1. [illegible]
2. [illegible]
3. [illegible]
4. [illegible]
5. [illegible]
6. [illegible]
7. [illegible]
8. [illegible]
9. [illegible]
10. [illegible]
11. [illegible]
12. [illegible]

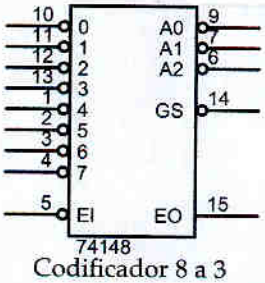


Decodificador 3 a 8

Inputs						Outputs							
Enable			Select										
G1	$\overline{G2A}$	$\overline{G2B}$	C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
X	H	X	X	X	X	H	H	H	H	H	H	H	H
X	X	H	X	X	X	H	H	H	H	H	H	H	H
L	X	X	X	X	X	H	H	H	H	H	H	H	H
H	L	L	L	L	L	L	H	H	H	H	H	H	H
H	L	L	L	L	H	H	L	H	H	H	H	H	H
H	L	L	L	H	L	H	H	L	H	H	H	H	H
H	L	L	L	H	H	H	H	H	L	H	H	H	H
H	L	L	H	L	L	H	H	H	H	L	H	H	H
H	L	L	H	L	H	H	H	H	H	H	L	H	H
H	L	L	H	H	L	H	H	H	H	H	H	L	H
H	L	L	H	H	H	H	H	H	H	H	H	H	L

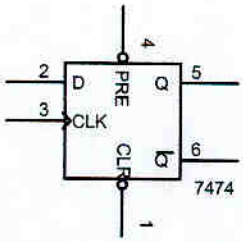
74139

Decodificador 2 a 4



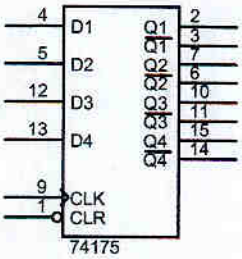
Codificador 8 a 3

Inputs									Outputs				
EI	0	1	2	3	4	5	6	7	A2	A1	A0	GS	EO
H	X	X	X	X	X	X	X	X	H	H	H	H	H
L	H	H	H	H	H	H	H	H	H	H	H	H	L
L	X	X	X	X	X	X	X	L	L	L	L	L	H
L	X	X	X	X	X	X	L	H	L	L	H	L	H
L	X	X	X	X	X	L	H	H	L	H	L	L	H
L	X	X	X	X	L	H	H	H	L	H	H	L	H
L	X	X	X	L	H	H	H	H	H	L	L	L	H
L	X	X	L	H	H	H	H	H	H	L	H	L	H
L	X	L	H	H	H	H	H	H	H	H	L	L	H
L	L	H	H	H	H	H	H	H	H	H	H	L	H



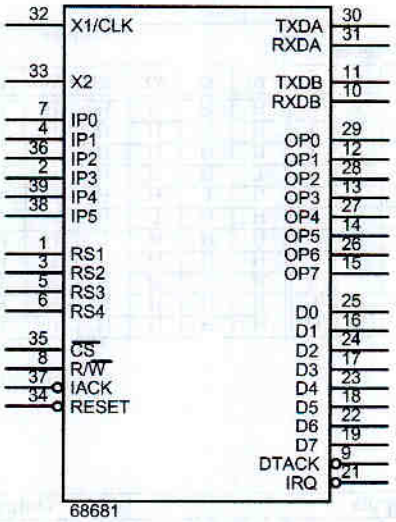
FUNCTION TABLE

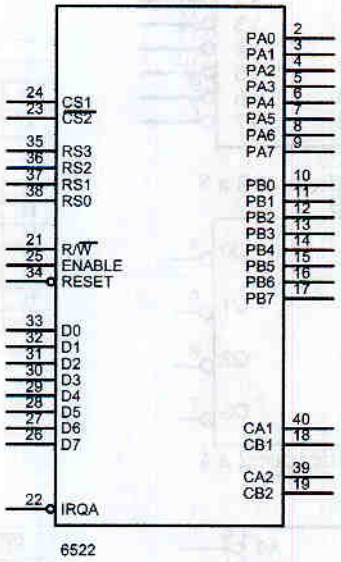
Inputs				Outputs	
$\overline{PRE}$	$\overline{CLR}$	CLK	D	Q	$\overline{Q}$
L	H	X	X	H	L
H	L	X	X	L	H
L	L	X	X	H↑	H↑
H	H	↑	H	H	L
H	H	↑	L	L	H
H	H	L	X	Q_0	$\overline{Q_0}$



FUNCTION TABLE
(each flip-flop)

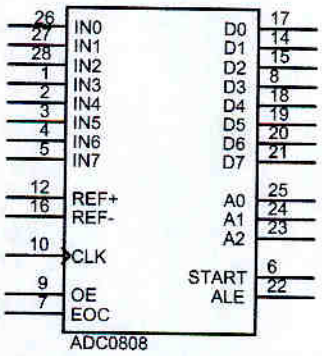
Inputs			Outputs	
$\overline{CLR}$	CLK	D	Q	$\overline{Q}$
L	X	X	L	H
H	↑	H	H	L
H	↑	L	L	H
H	L	X	Q_0	$\overline{Q_0}$



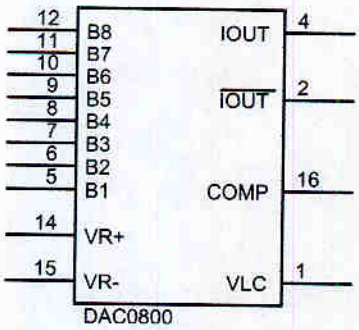


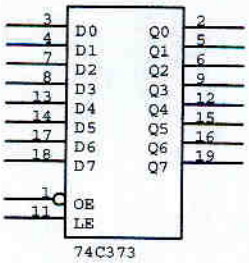
RS3	RS2	RS1	RS0	Lectura	Escritura
0	0	0	0	IRB	ORB
0	0	0	1	IRA	ORA
0	0	1	0	DDRB	
0	0	1	1	DDRA	
0	1	0	0	T1C_L	T1L_L
0	1	0	1	T1C_H	
0	1	1	0	T1L_L	
0	1	1	1	T1L_H	
1	0	0	0	T2C_L	T2L_L
1	0	0	1	T2C_H	
1	0	1	0	SR	
1	0	1	1	ACR	
1	1	0	0	PCR	
1	1	0	1	IFR	
1	1	1	0	IER	
1	1	1	1	IRA	ORA

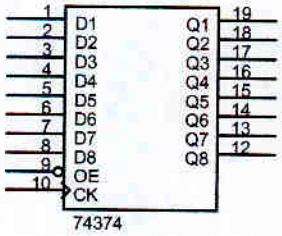
Lógica de selección de los registros internos de la VIA.



El tiempo de acceso en lectura del convertidor A/D (desde la activación de OE) es de 250 ns como máximo.

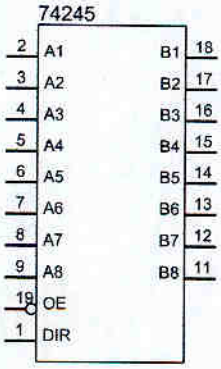






FUNCTION TABLE
(each flip-flop)

Inputs			Output
$\overline{OE}$	CLK	D	Q
L	↑	H	H
L	↑	L	L
L	H or L	X	Q_0
H	X	X	Z

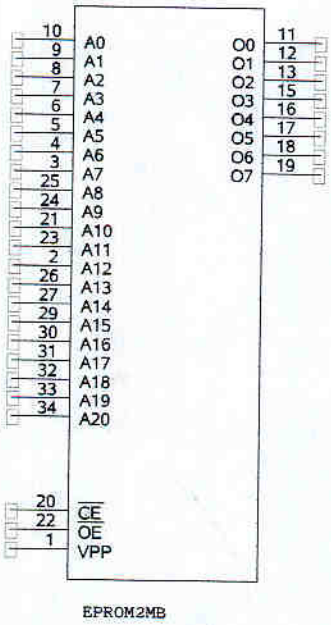
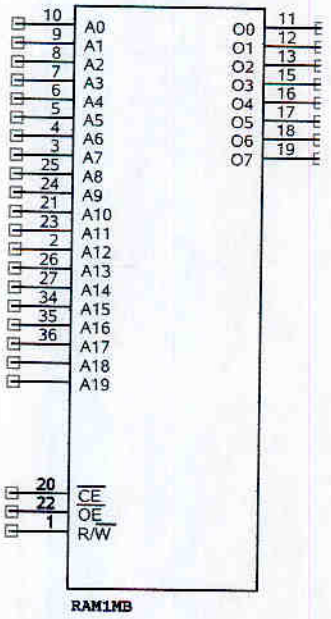
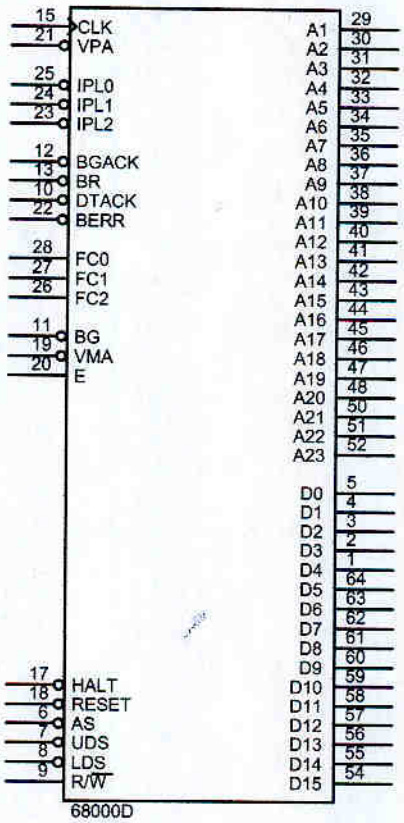


FUNCTION TABLE

Inputs		Inputs/Outputs	
$\overline{OE}$	DIR	A_n	B_n
L	L	A=B	inputs
L	H	inputs	B=A
H	X	Z	Z

Patillajes de dispositivos

EPROM 2Mx8



RAM 1MX8