PUERTOS

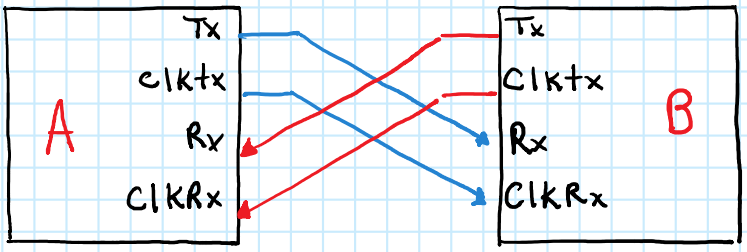
Thursday, October 3, 2019 3:11 PM

// Tipos de comunicación

Emisor Receptor

- Simplex "A → B" (Comunicación en un sólo sentido)
- Duplex "A ↔ B" (Comunicación en ambos sentidos, un solo canal)
  - ↳ WalkingTalkyng*
- Full duplex "A ⇄ B" Se da mediante 2 canales
  - ↳ Teléfono

Puerto Serial Síncrono

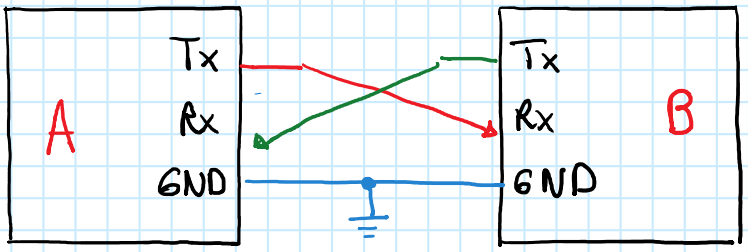


Tx ≙ Transmisión
Rx ≙ Recepción
clktx ≙ Reloj de transmisión (out)
clkRx ≙ Reloj de recepción (in)

Full Duplex

- A y B pueden manejar frecuencias de transmisión diferentes.
- Transmiten señal (Tx) junto con la señal de reloj con la que están sincronizados para que el receptor decodifique el mensaje.

Puerto Serial Asincrono



Full Duplex

Deben tener la misma frecuencia para poder establecer comunicación

- Baud Rate → Velocidad de transmisión de palabra (De 8 o (9) bits)

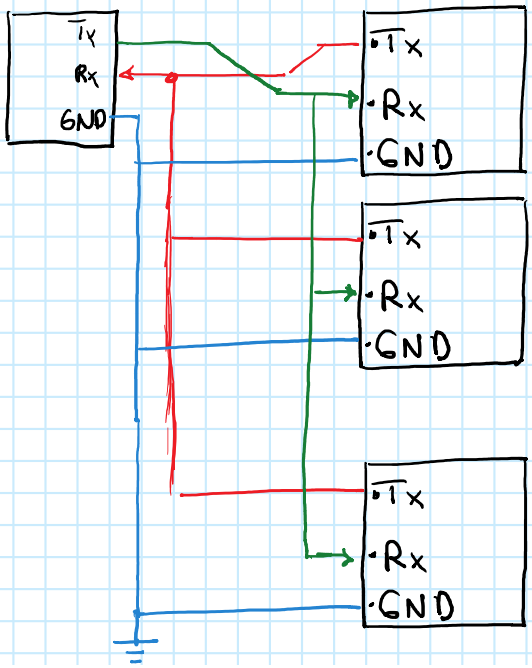
$$\frac{\text{Palabras}}{\text{Segundo}}$$

// Si se puede configurar como Bit Rate → $\frac{\text{Bit}}{\text{Segundo}}$

(9) es para el "Bit de paridad"

- Baud Rate { 9600, 19200 } [Baudios]

Red - Maestro → Esclavo



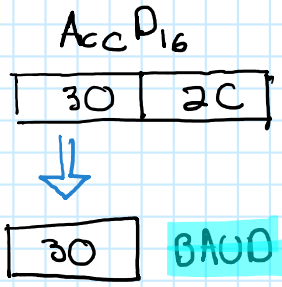
- DB9

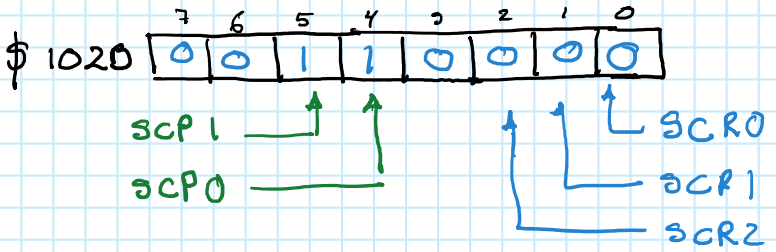
// Puerto Serial

- View → Memory → Dumb

THRsim11

↳ ¿Cómo introducir un número hexadecimal?





SCCR2

SCCR2 — SCI Control Register 2 S102D

BIT 7	6	5	4	3	2	1	BIT 0
TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
RESET: 0	0	0	0	0	0	0	0

TIE — Transmit Interrupt Enable
0 = TDRE interrupts disabled (software polling mode).
1 = An SCI interrupt is requested when TDRE is set to one.

TCIE — Transmit Complete Interrupt Enable
0 = TC interrupts disabled (software polling mode).
1 = An SCI interrupt is requested when TC is set to one.

RIE — Receive Interrupt Enable
0 = RDRF and OR interrupts disabled (software polling mode).
1 = An SCI interrupt is requested when either RDRF or OR is set to one.

ILIE — Idle-Line Interrupt Enable
0 = IDLE interrupts disabled (software polling mode).
1 = An SCI interrupt is requested when IDLE is set to one.
The idle-line function is inhibited while the receiver wake-up function is enabled.

TE — Transmit Enable
0 = SCI transmitter disabled
1 = SCI transmitter enabled

RE — Receive Enable
0 = SCI receiver disabled
1 = SCI receiver enabled

RWU — Receiver Wake Up
0 = Normal SCI receiver operation (wake-up feature not enabled).
1 = Places the SCI receiver in a standby mode where receiver-related interrupts are inhibited until some hardware condition is met to wake up the sleeping receiver.

SBK — Send Break
0 = Normal transmitter operation.
1 = Enable transmitter to send synchronous break characters.

- Si se transmite a 9600 [Baudios] :

$$T_{Tx} = \frac{1}{9600} = 1{,}0417 \times 10^{-4} \text{ [segundos]}$$

- Cristal 8 [MHz] : E ≙ Frecuencia de reloj interno

$$E = \frac{8 \text{ MHz}}{4} = 2 \text{ MHz}$$

$$T_{\text{ciclos reloj}} = \frac{1}{2\,000\,000 \text{ Hz}} = 5 \times 10^{-7} \text{ [seg]}$$

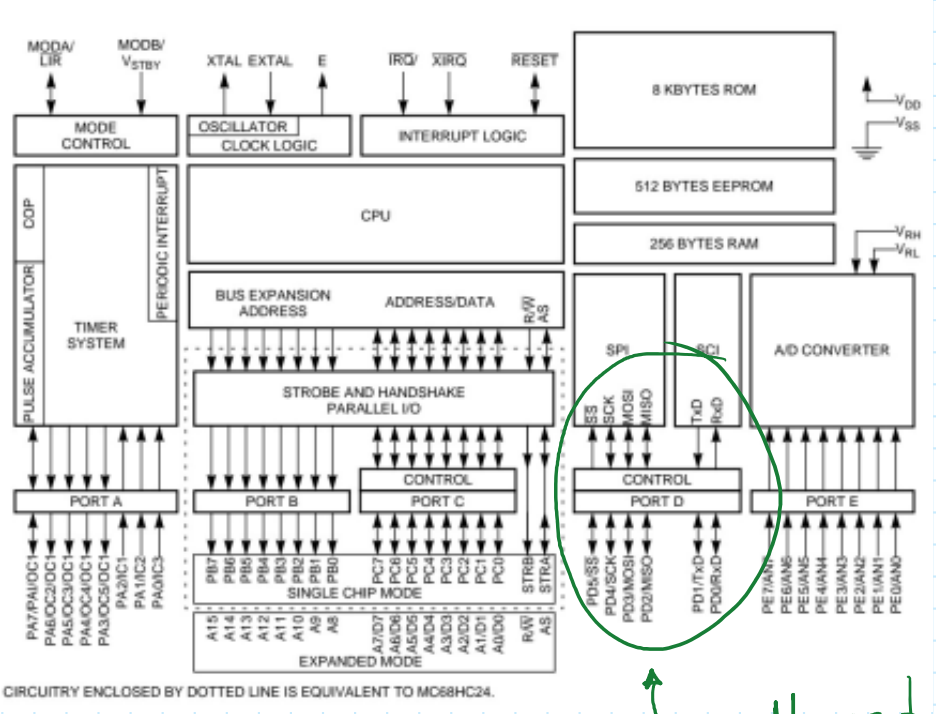
$$T_{ciclos\ reloj} = \frac{1}{2\,000\,000\ Hz} = 5 \times 10^{-7}\ [seg]$$

// ¿Cuántos ciclos de reloj demorará la transmisión de 1 byte?

$$\text{Núm. Ciclos Empleadas} = \frac{T_{Tx}}{T_{ciclo\,reloj}} = \frac{1.0417 \times 10^{-4}}{5 \times 10^{-7}} = 208.34$$

∴ 209 [ciclos] //

Análisis Code (EXEMPLO)



Necesitamos configurar

- LDAA #$FE

  F E

  1 1 1 1 1 1 1 0

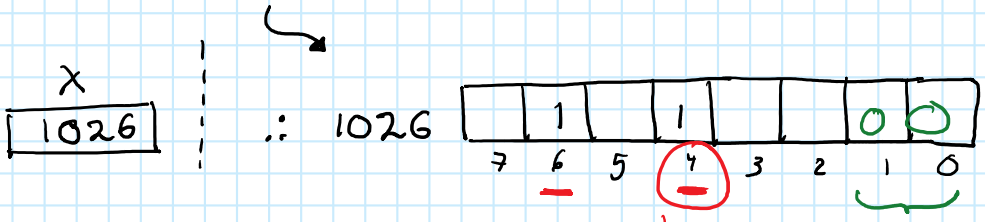
  S S S S S S S E

∴ Ya "sacrificamos" una salida como entrada en el puerto D

- LDAA #$F4
  STAA TEMPO

TEMPO = $F4 ; TEMPO = 244

Bset $00, X, #$50



//Pulsos

- Niveles
- Flancos

} Formas de detectar el pulso en el micro

//Real Time → Contador de 8 bits que cuenta tiempo real

RTR1
RTR0

∴ Nos interrumpe cada 4.1 [ms]
//Checa haga con tabla PACTL

$$\text{Núm Interrupciones} = \frac{1000\ [ms]}{4.1\ [ms]} = 243.9 \cong 244$$

//Forma de garantizar que ya pasó un segundo

Exemplo

Friday, October 18, 2019 8:07 AM

→ Scg = 0 → ORDEN = 3F (Ascii "?")

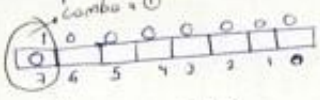
→ Tempo = F4 (244 en decimal)

- Puerto serial Asincrono 9600 Baudios, 8 bits, Tx, Rx, $\overline{Irx}$
- Real Time (Interrumpe cada 4.1 milisegundos)
- Acumulador pulsos

; JSR → Salto a "SubRutine
; RTS → return ...

//Frecuencimetro ; contador de 8 bits hasta 256 Hertz
// Podemos partir el segundo para obtener frecuencias más altas

//SCSR | Reg de control del estado del puerto ~~serial~~ serial asincrono
// Tenemos que leerlo para que funcione
∴ LDAA SCSR



$SCSR $102E

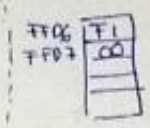
BRCLR $00,X, #$80
que valga esto — 8.t más significativo

//SCDR | Es el Buffer del puerto asincrono

// Cuando cambia de 0 a 1 entonces ya no salta
↳ Ejecutaremos la misma instrucción hasta que no sea 0
// Echo → Para verificar que salé bien el micro o bien configurado los puertos

ZONA NOTAS

ORG = $FFD6
PCB $F1, $00
//Vector interrupción



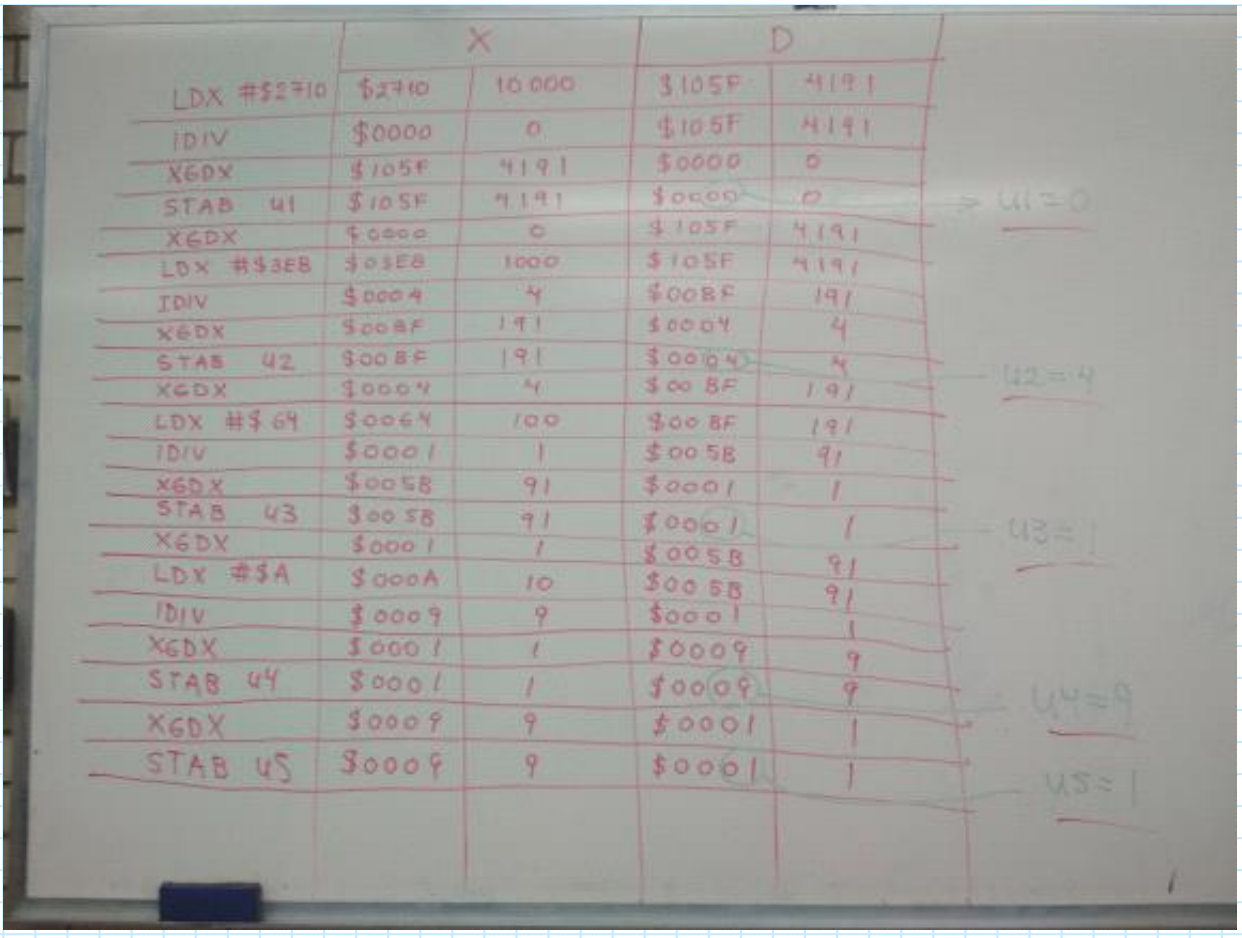
//Para un "Hola Mundo"

```
    LDX  #SCDR
    LDAA SCSR
    LDAA #1A
    STAA SCDR
ETO
    BRCLR $00,X, #$80 ETO
    LDAA SCSR
    LDAA #'O
    STAA SCDR
ETI
    BRCLR $00, X, #$80 ETI
```

//Función/Etiqueta BINBCD

	X	D	
LDX #$2710	$2710	10000	$105F	4191
IDIV	$0000	0	$105F	4191
XGDX ...				
STAB ...				
XGDX ...				
LDX ...				

//Foto



	X		D		
LDX #$2710	$2710	10000	$105F	4191	
IDIV	$0000	0	$105F	4191	
XGDX	$105F	4191	$0000	0	
STAB 41	$105F	4191	$0000	0	→ 41=0
XGDX	$0000	0	$105F	4191	
LDX #$3E8	$03E8	1000	$105F	4191	
IDIV	$0004	4	$008F	191	
XGDX	$008F	191	$0004	4	
STAB 42	$008F	191	$0004	4	→ 42=4
XGDX	$0004	4	$008F	191	
LDX #$64	$0064	100	$008F	191	
IDIV	$0001	1	$005B	91	
XGDX	$005B	91	$0001	1	
STAB 43	$005B	91	$0001	1	→ 43=1
XGDX	$0001	1	$005B	91	
LDX #$A	$000A	10	$005B	91	
IDIV	$0009	9	$0001	1	
XGDX	$0001	1	$0009	9	
STAB 44	$0001	1	$0009	9	→ 44=9
XGDX	$0009	9	$0001	1	
STAB 45	$0009	9	$0001	1	→ 45=1

Downloader

Tuesday, October 29, 2019 2:12 PM

- Diagrama de Flujo: La primera parte válida la cadena "START"

// Insert Image

- Parte del Downloader

S - T - A - R - T - XX_0 - XX_1 ... - XX_n - END

  - Bytes que conforman el código objeto de un programa previamente compilado

- Diagrama de Flujo
  - VAR será la variable que utilizaremos para ver que caracter llegó

- VAR = 0
- DIRBASE = 0030

Simulador Ausm11.exe

- Para cargar un programa: L - P Enter filename "D:68HC11/"
- Resetear programa: R^reset - C^PU
- Para usar ventana: D - 1 - A Expresión in memory Address space

Bubble sort

martes, 5 de noviembre de 2019 14:35

Algoritmo:

```
void bubble(int size){
int i, j, aux;
i=0;
j= size-1;
 for( ; j>0 ;j-){
    for( i=0; i<j;i++){
      if( arr[i]>arr[i+1])
         aux = arr[i];
         arr[i]= arr[i+1];
         arr[i+1]= aux;
      }
    }
}
```

// Code

j = n-1

n = x - Dir_Base - 5

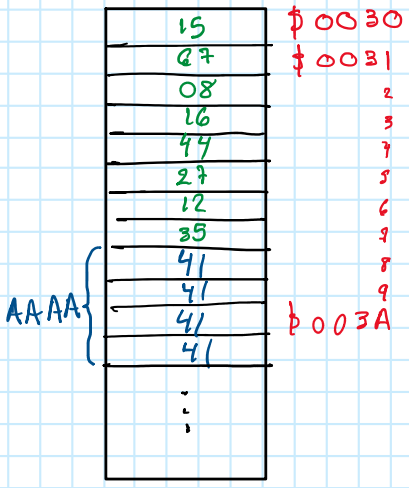
j = x - 6

Y_{16} = x - 6

// Juega el papel de j

Y_{16} ≙ Juega el papel de i

Y_{16} = Dir_Base



//Código en ensamblador

```
ALTO    EQU   $0007   } Variables agregadas
BAJO    EQU   $0008   }
```

Burbuja

```
LDY   # DirBase   * I=0
XGDX              * j=n-1
SUBD  #6
XGDX
```

CicloFor1

```
CPX   # DirBase   * if  j>0   ¡ Condición del first for
BHI   CicloFor2   * Caso true
JMP   INICIO      * Caso Falso
```

CicloFor2

```
STY   ALTO        * Guarda Y en alto y bajo // Lo guarda indirectamente en bajo
CPX   ALTO        * X - Y
BHI   COMPARA
DEX
LDY   #DirBase
JMP   CicloFor1
```

COMPARA

```
LDAA  $00,Y
LDAB  $01,Y       * Cargamos a Y
CBA
BLO   INTERCAMBIA * Vamos al if(arr[i]>arr[i+1])
INY               * Incrementamos y
JMP   CicloFor2
```

BLO: Acomoda al revés

INTERCAMBIA

```
STAB  $00,Y   } Parte del intercambio las 3 lineas
STAA  $01,Y
INY
JMP   CicloFor2
```