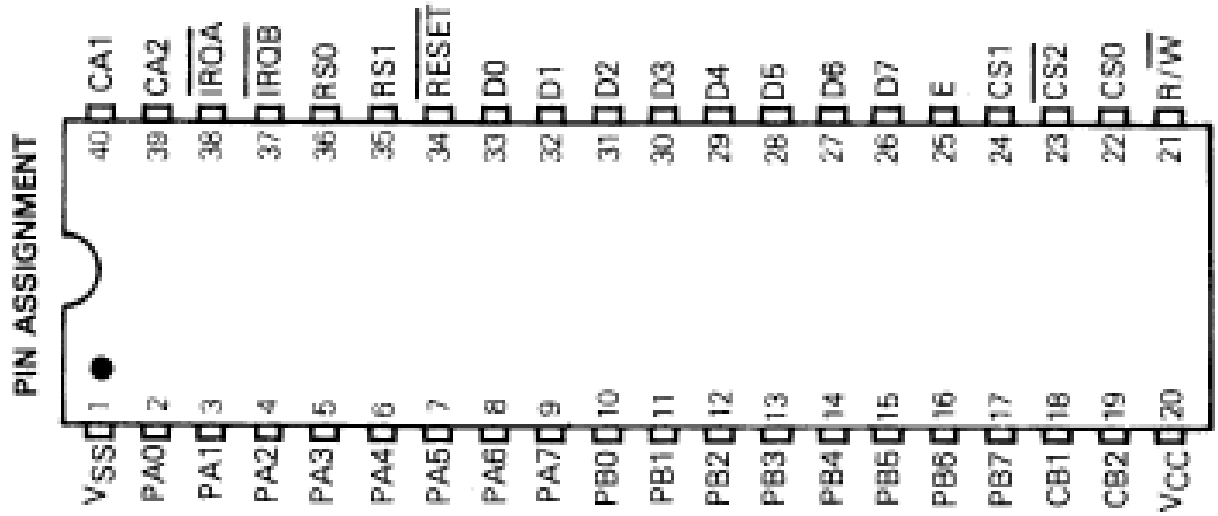
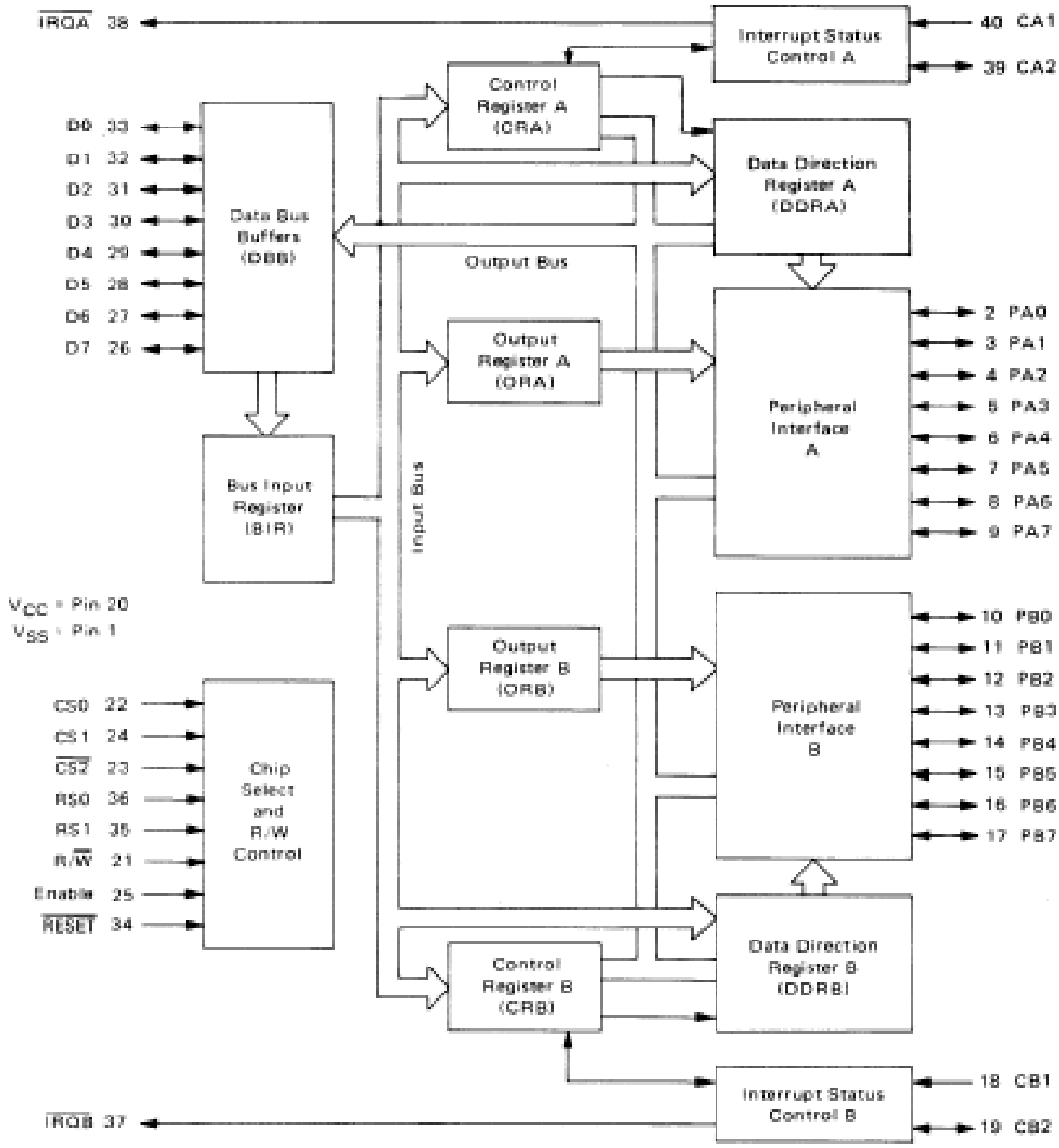
The 6821 Peripheral Interface Adaptor (PIA)

The PIA interfaces to the 68xx microprocessor with an 8-bit bidirectional data bus, three chip select lines, two register select lines, two interrupt request lines, a read/write line, an enable line and a reset line.

- PIA has two ports; each port may drive two TTL loads
- Each individual signal line (PA0 through PA7 and PB0 through PB7) can be programmed as an input or an output



PIA Block Diagram



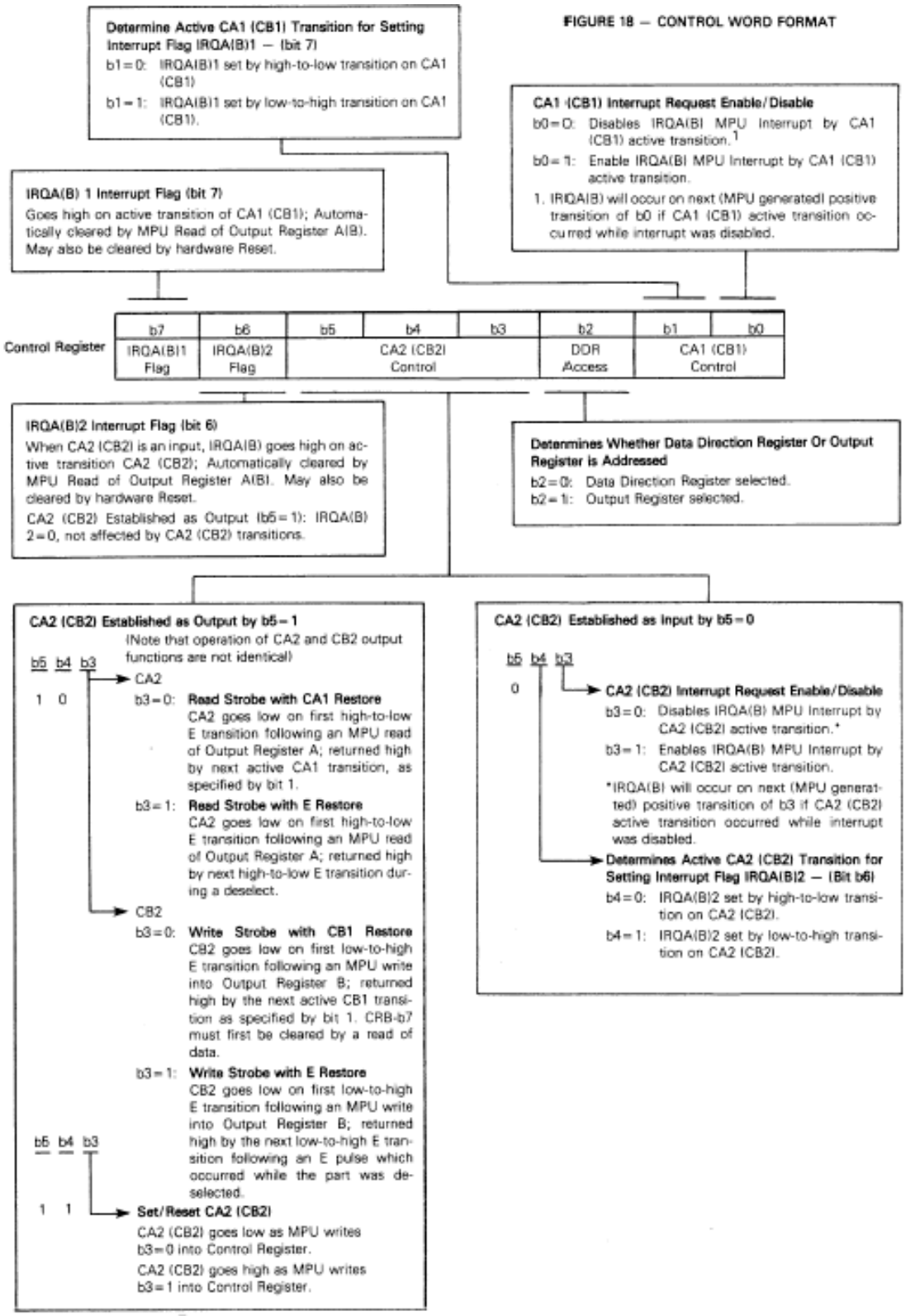
PIA Registers

TABLE 1 – INTERNAL ADDRESSING

RS1	RS0	Control Register Bit CRA-2	Control Register Bit CRB-2	Location Selected
0	0	1	X	Peripheral Register A
0	0	0	X	Data Direction Register A
0	1	X	X	Control Register A
1	0	X	1	Peripheral Register B
1	0	X	0	Data Direction Register B
1	1	X	X	Control Register B

X = Don't Care

FIGURE 18 — CONTROL WORD FORMAT



Initializing PIA

Procedure to initialize PIA port A for simple I/O

- **Fill bit 2 of CRA with 0 to access DDRA**
- **Fill DDRA with proper value to determine the role of each signal line of port A (1 for output and 0 for input)**
- **Fill bit 2 of CRA with 1 to access DRA**
- **Write or read data using DRA**

6802 prog to set up A side as i/ps, B side as o/ps

```
CLR     CRA     ;Access DDRA
CLR     DRA     ;all i/ps
LDAA    #4      ;Set CRA2  = 1
STAA    CRA      ;Access DRA

CLR     CRB     ;Access DDRB
LDAA    #$FF    ;all o/ps
STAA    DRB
LDAA    #4      ;Set CRA2  = 1
STAA    CRA      ;Access DRA
```

Interface switch to PIA

Connect on/off switch between 0V and PA7 on PIA. Set VAL to 1 if switch OPEN, to 0 if CLOSED.

```
        CLR     CRA     ;Access DDRA
        CLR     DRA     ;all i/ps
        LDAA    #4      ;Set CRA2  = 1
        STAA    CRA      ;Access DRA
        CLR     VAL     ;VAL = 0
        LDAA    DRA     ;Read PA7-0
        ANDA    $80     ;keep PA7
        BEQ     DONE    ;switch closed
        INC     VAL     ;switch open
DONE    …
```

LED interface

Connect 8 LEDs between 0V and PB 0-7 on PIA. Send contents of LITES to LEDs

```
CLR     CRB     ;Access DDRB
LDAA    #$FF    ;all o/ps
STAA    DRB
LDAA    #4      ;Set CRB2 = 1
STAA    CRB     ;Access DRB
LDAA    LITES   ;Get LED data
STAA    DRB     ;Sent to LEDs
```

Serial Input/Output Interface

- Serial data transmission
  - One bit at a time
  - Asynchronous or synchronous
  - Simplex, half-duplex or full-duplex

- Advantages
  - Cheap
  - Simple
  - Easy to use

- Disadvantages
  - Slow
  - Needs conversion to and from parallel format

- Serial interface
  - Parallel-to-series and series-to-parallel adaptors
  - Connects CPU to remote serial peripherals
  - E.g. ACIA 6850

6850 ACIA
Asynchronous Communications Interface Adaptor

- 6850 ACIA
  - series-parallel data conversion
  - asynchronous data formatting

- CPU Side Interface
  - 1 clock input (E enable)
  - synchronous Data bus (8 bits)
  - 3 chip select inputs (CS0, CS1, !CS2)
  - 1 register select input (RS)
    - in general connected to A0
  - Read/write control line (R/!W)

ACIA Peripheral Interface

- Receiver
  - serial data input line
    - receiver data input line (RxD)
  - data carrier detect signal (!DCD)
    - indicates that incoming data is valid

- Transmitter
  - serial data output line
    - transmitter data output line (TxD)
  - request to send output line (!RTS)
    - Set when ACIA is ready to transmit data
    - Set or cleared by software control
  - clear to send input (!CTS)
    - Active when the peripheral device is ready to transmit data
    - Setting !CTS inhibits the transmission

Asynchronous Serial I/O

- Asynchronous
  - transmitter and receiver do not synchronize timing
  - clocks at transmitter and receiver are not synchronized
  - data at each end is synchronized to local clock

- Data exchange - terminology
  - Mark level => logic 1
  - Space level => logic 0

- Data format
  - Idle – mark level
  - Start bit – space level
  - Data bits (7 or 8 bits)
    - usually ASCII characters
  - Even or odd parity bit (optional)
  - Stop bit – mark level
    - 1 or 2 bit times in length
  - 12 combinations in total

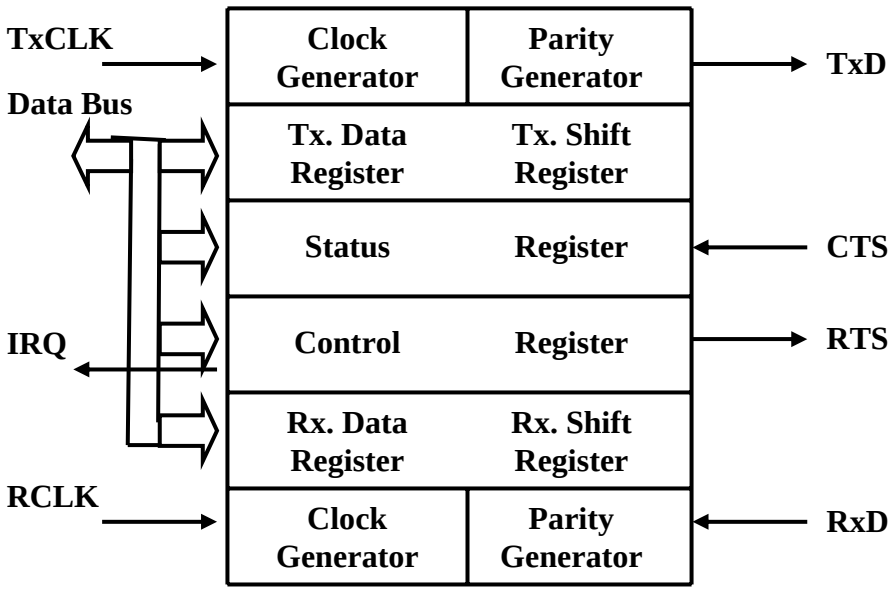
Data Exchange Issues

- Efficiency
  - 7-bit data, start, 1-bit stop, parity => 70% overall efficiency

- Data Timing
  - Bit time = T
  - Receiver waits for falling edge of the Start bit
    - triggers local clock
  - Samples next N bits at their centers
    - using local clock, compute T/2
  - Clock precision
    - < T/2 error in 9-11 bits between transmitter and receiver clocks
    - < 5% error - trivial with crystal oscillators

- Baud vs. Bit-rate
  - Bit rate = no of bits of DATA sent per second
  - Baud = no of bits of DATA+ CONTROL sent per second
  - E.g. start bit + 7 data bits + parity + stop bit
    - Bit-rate = 7 bits/s
    - Baud = 10 bits/s

ACIA Internal Structure



- Registers – 4 registers accessed by RS and R/!W pins
  - **Transmission Data/Shift Register** – accepts parallel data from Data Bus, inserts necessary parity bits and shifts data, one-bit at a time, to TxD serial line
  - **Reception Data/Shift Register** – receives serial data from RxD line, shifts the data, removing also the parity bits and delivers parallel data to the Data Bus
  - **Control Register** – determines the format of the serial data (transmission frequency, number of data bits, parity), the clock to be used (external or built-in) and enables interrupts during the transmission and reception of data respectively
  - **Status Register** – gives information about the status of the conversion process transmit/receive (e.g. errors, parity, interrupts, handshaking signals, etc.)
  - **Clock Generator** – generates clock signal for data Tx. and Rx.
  - **Parity Generator** – generates parity bits for data Tx. and Rx.

ACIA Registers

- Transmit data register (TDR)
  - contains data to transmit
  - it is write only
- Receive data register (RDR)
  - contains received data
  - it is read only
- Control register (CR)
  - defines the operating mode: relationship between transmitter and receiver clocks, number of data bits, number of start and stop bits, parity type, etc.
  - it is write only
- Status register (SR)
  - indicates the status of both transmission and reception
  - it is read only

ACIA Software Control

- Register Selection

RS	R/!W	Operation	Register
0	0	Write	Control
0	1	Read	Status
1	0	Write	Tx. Data
1	1	Read	Rx. Data

- Control Register Format

7	6	5	4	3	2	1	0
RIE	TC1	TC0	WS2	WS1	WS0	CD1	CD0

- RIE – Receiver interrupt enable
- TC – Transmitter control
- WS – Word select
- CD – Clock division

ACIA Control Register Format (1)

- Clock Division (CD) Bits

CD1	CD0	Division Ratio
0	0	: 1
0	1	: 16
1	0	: 64
1	1	Master reset

- Word Select (WS) Bits

WS2	WS1	WS0	Word Length	Parity	Stop bits	Total
0	0	0	7	even	2	11
0	0	1	7	odd	2	11
0	1	0	7	even	1	10
0	1	1	7	odd	1	10
1	0	0	8	none	2	11
1	0	1	8	none	1	10
1	1	0	8	even	1	11
1	1	1	8	odd	1	11

ACIA Control Register Format (2)

- Transmitter Control (TC) Bits

TC1	TC0	!RTS	Tx Interrupt
0	0	low	disabled
0	1	low	enabled
1	0	high	disabled
1	1	low	Disabled*

*Note: a break level is put on the transmitter output

ACIA Status Register Format (1)

- Status Register Format

7	6	5	4	3	2	1	0
IRQ	PE	OVRN	FE	!CTS	!DCD	TDRE	RDRF

- IRQ – Interrupt request
- set whenever the ACIA wishes to interrupt CPU:
  - Received data register full (SR bit 0 set)
  - Transmitter data register empty (SR bit 1 set)
  - !DCD bit set (SR bit 2)

- PE – Parity error
- set when the parity bit received does not match the parity bit generated locally for the received data

- OVRN – Receiver Overrun
- set when data is received by the ACIA and not read by the CPU when new data is received over-writing old data
- it indicates that data has been lost

- FE – Framing error
- set when received data is incorrectly framed by the start and stop bits

ACIA Status Register Format (2)

- Status Register Format

7	6	5	4	3	2	1	0
IRQ	PE	OVRN	FE	!CTS	!DCD	TDRE	RDRF

- !CTS – Clear to send
- directly indicates the status of the ACIA's !CTS input

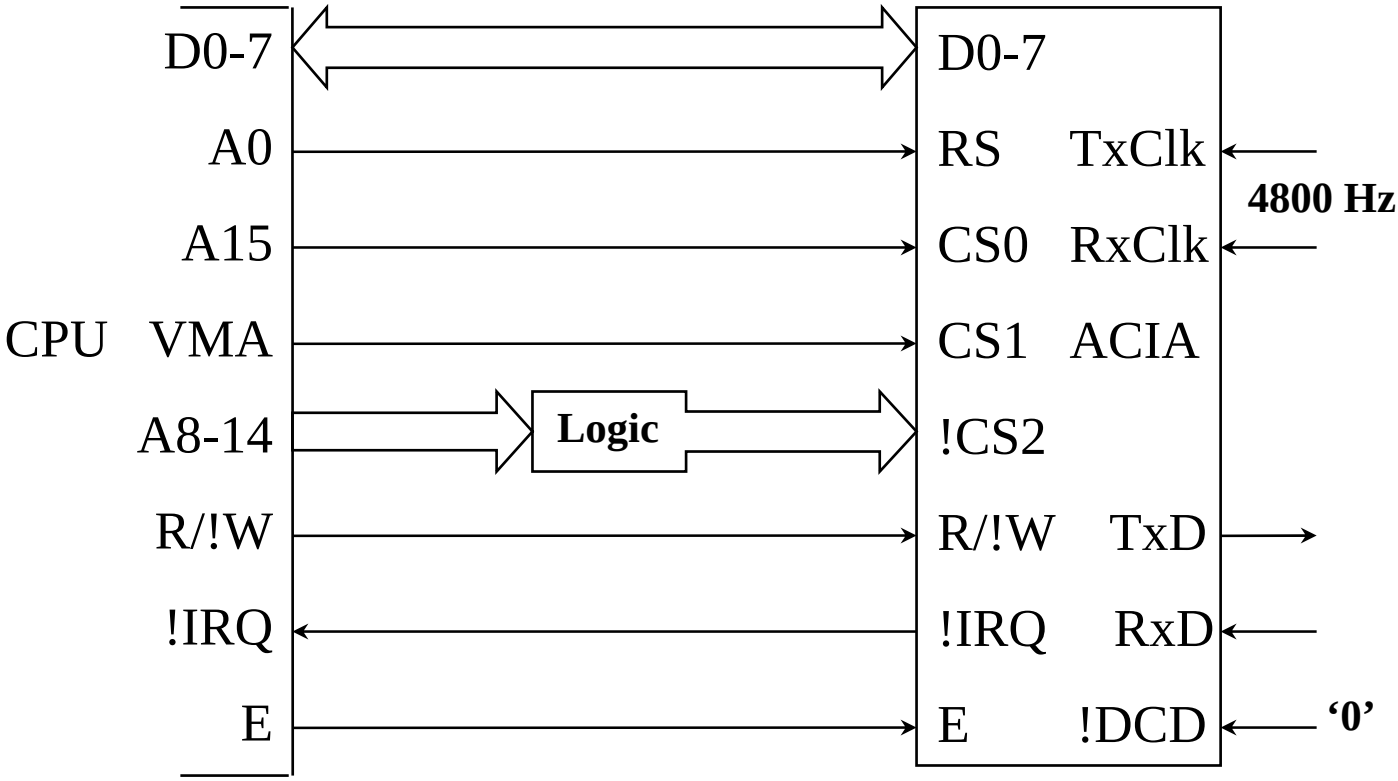
- !DCD – Data Carrier Detect
- set when the ACIA's !DCD input is high
- reset when the CPU reads both the status register and the data register or when ACIA is master reset

- TDRE - Transmitter data register empty
- set when the transmitter data register is empty, indicating that data has been sent
- reset when transmitter data register is full or when !CTS is high, indicating that the peripheral is not ready

- RDRF – Receiver data register full
- set when the receiver data register is full, indicating that data has been received
- reset when the data has been read from the data register

Interfacing ACIA

- Connect an ACIA at $E0XX to interface a 300 baud serial line (4800 Hz clock) with data format as follows:
- 7 data bits, even parity, 2 stop bits
- and having !RTS=0 and the interrupts enabled



- Write an interrupt service routine to determine if an error occurred when transmitting or receiving data
- The data from location $D002 is to be sent
- Received data is stored at location $D000

Interfacing ACIA

```
START   LDAA #3  reset ACIA
        STAA $E000      store in control register
        LDAA #$A1       sets IRQ, !RTS=0, 7 data bits
                        even parity, 2 stop bits, clk/16
        STAA $E000      store in control register
          …             set interrupt vector
DMY     BRA DMY         main program

ISR     LDAA $E000      read status register
        RORA            rotate right (RDRF -> Carry)
        BCS RECV        if carry is set -> jump RECV
        BITA #$38       test error bits (PE, OVRN, FE)
                        note: right shift was performed
        BNE ERR         if not zero -> jump ERR
                        perform transmission:
TX      LDAA $D002      take data from address $D002
        STAA $E001      send data to ACIA Tx Register
        RTI

RECV    LDAA $E001      get data from Rx Register
        STAA $D000      store to address $D000
        RTI

ERR       …             do something
        LDAA #3         reset flags
        STAA $E000      store in control register
        RTI
```

Timers

- Programmable Timing
  - Allows for controlled clock signal generation
  - Permits synchronisation of processes
  - Permits controlled generation of CPU interrupt requests

- Advantages
  - High flexibility
  - Control in relation to timing

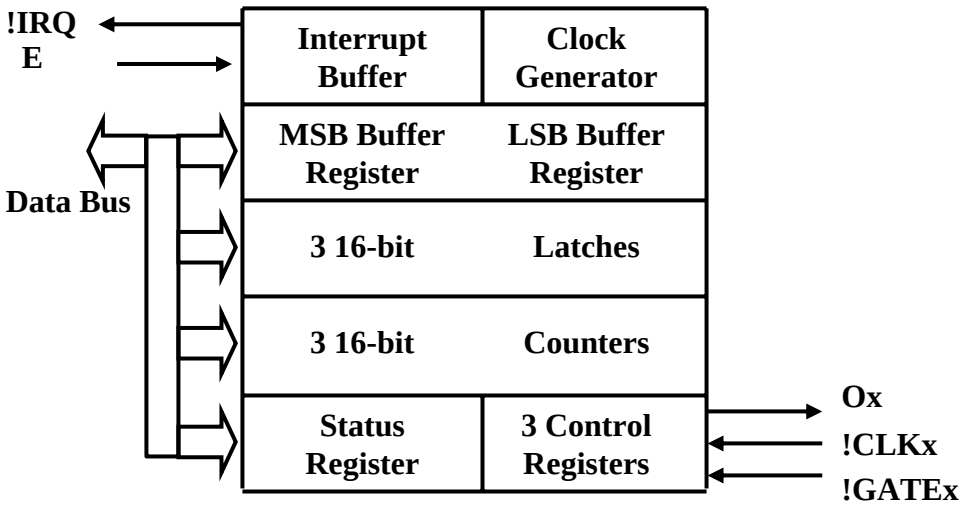
- Disadvantages
  - Relatively complex
  - Needs good machine language programming skills

- Programmable Timer Module
  - E.g. PTM 6840

6840 PTM
Programmable Timer Module

- 6840 PTM
  - Programmable timer
- CPU Side Interface
  - 1 clock input (E enable)
  - synchronous Data bus (8 bits)
  - 2 chip select inputs (!CS0, CS1)
  - 3 register select inputs (RS0, RS1, RS2)
    - in general connected to A0-A2
  - Read/write control line (R/!W)
  - Interrupt output line (!IRQ)
- Peripheral Side Interface
  - 3 groups of 3 lines:
    - 1 output (Ox, x=1,2,3)
    - 1 input clock (!CLKx, x=1,2,3)
    - 1 input control line (!GATEx, x=1,2,3)
  - each group is associated with a Timer

PTM Internal Structure



- Registers – 18 registers – selected by 3 RS lines only
  - **16-bit Counters (3)** – store the current values for the Timers
  - **16-bit Latches (3)** – hold the start values for the Counters
  - **16-bit Buffer Register (1)** – temporarily stores data in its MSB and LSB prior to its exchange with the CPU via the 8-bit Data Bus that is performed in two clock intervals
  - **Control Registers (3)** – determine the operation of the PTM and the meaning of some of its lines
  - **Status Register** – gives information about whether one of the Timers has generated an interrupt or not
  - **Clock Generator** – generates the internal clock for PTM based on the CPU clock

PTM Software Control

- Register Selection

RS2	RS1	RS0	Read (R/!W = 1)	Write (R/!W = 0)
0	0	0	No operation	Control reg. 1 or 3
0	0	1	Status register	Control register 2
0	1	0	MSB Counter 1 (C1)	MSB L1 - Buff. Reg.
0	1	1	LSB C1 - Buff. Reg.	LSB Latch 1 (L1)
1	0	0	MSB Counter 2 (C2)	MSB L2 - Buff. Reg.
1	0	1	LSB C2 - Buff. Reg.	LSB Latch 2 (L2)
1	1	0	MSB Counter 3 (C3)	MSB L3 - Buff. Reg.
1	1	1	LSB C3 - Buff. Reg.	LSB Latch 3 (L3)

- Using Timers' Latches and Counters
- Lets assume that PTM's base address is $9000
- To set value $1234 into Timer 1's latch the following code has to be executed:

```
LDX     #$1234
STX     $9002
```

- The STX has the following effect:

  MSB register X -> MSB buffer register ($9002)
  LSB register X -> LSB timer 1's latch ($9003)
  MSB buffer register -> MSB timer's 1 latch

PTM Status Register

- Status Register Format

7	6	5	4	3	2	1	0
CI	NU3	NU2	NU1	NU0	I3	I2	I1

- CI – Composite Interrupt Flag (I3 + I2 + I1)
- NU3 – Not Used
- NU2 – Not Used
- NU1 – Not Used
- NU0 – Not Used
- I3 – Timer 3 Interrupt Flag
- I2 – Timer 2 Interrupt Flag
- I1 – Timer 1 Interrupt Flag

PTM Control Registers (1)

- Control Registers' Format

7	6	5	4	3	2	1	0
OE	IE	OM2	OM1	OM0	CMC	CS	TDB

- OE – Output Enable
  - 0 – output disabled
  - 1 – output enabled
- IE – Interrupt Enable
  - 0 – interrupt disabled
  - 1 – interrupt enabled
- OM2-0 – Operating Mode Select bits
- CMC - Counting Mode Control
  - 0 – normal (16-bit) counting mode
  - 1 – dual 8-bit counting mode
- CS - Clock Select
  - 0 – external clock
  - 1 – internal clock
- TDB – Timer Dependent Bit
  - Timer 1
    - 0 – normal operation
    - 1 – reset
  - Timer 2
    - 0 – select control register 3
    - 1 – select control register 1
  - Timer 3
    - 0 – Timer 3 CLK / 1
    - 1 – Timer 3 CLK / 8

PTM Control Registers (2)

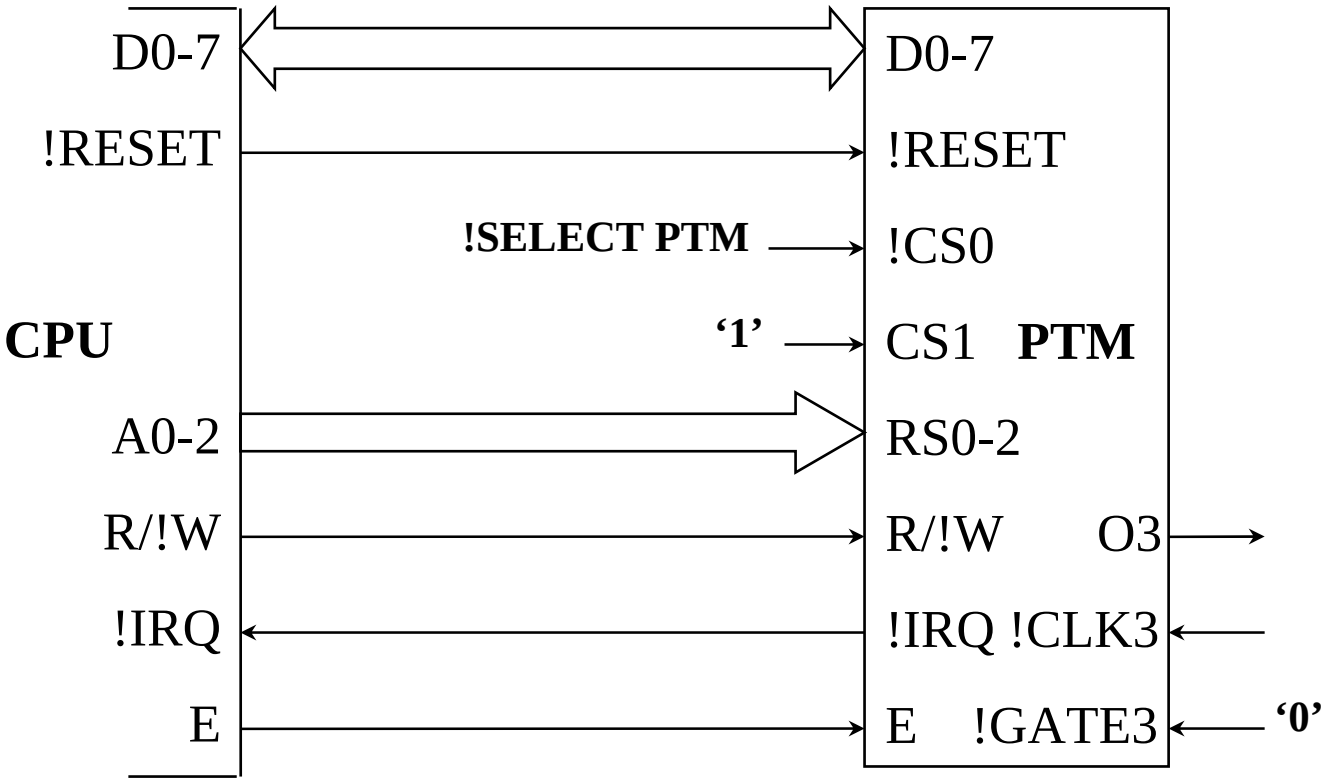
- Operating Mode Select (OM2-0) Bits

OM2	OM1	OM0	Operation Mode	Description
0	0	0	Continuous	Gate negative edge, write to latches or reset causes counter initialisation
0	0	1	Frequency comparison	Interrupt if gate period is shorter than counter timeout
0	1	0	Continuous	Gate negative edge or reset causes counter initialisation
0	1	1	Pulse width comparison	Interrupt if gate pulse width is shorter than counter timeout
1	0	0	Single shot	Gate negative edge, write to latches or reset causes counter initialisation
1	0	1	Frequency comparison	Interrupt if gate period is longer than counter timeout
1	1	0	Single shot	Gate negative edge or reset causes counter initialisation
1	1	1	Pulse width comparison	Interrupt if gate pulse width is longer than counter timeout

- Timeout = (Count+1)/Clock_Freq

Interfacing PTM (1)

- Connect an PTM at $9000 and configure its Timer 3 to generate square waves with frequency 62.5 Hz. Assume 1 MHz [internal] clock.



- Timeout = (Count+1)/Clock_Freq
- 16-bit counting mode:
  - Output waveform freq = Clock_Freq/(Count+1)
- 8-bit counting mode:
  - Output waveform freq = Clock_Freq/[2*(Count+1)]

Interfacing PTM (2)

- Output waveform freq = Clock_Freq/[2*(Count+1)]

- For divide-by-1 solution:
- Count = Clock_Freq / [2* Output_Freq] – 1
- Count = 1 000 000 / [2 * 62.5] - 1 = 7999
- Count = $1F3F

- For divide-by-8 solution:
- Count = Divided_Clock_Freq / [2* Output_Freq]
- Count = 250 000 / [2 * 62.5] – 1= 999
- Count = $3E7

Solution divide-by-1:

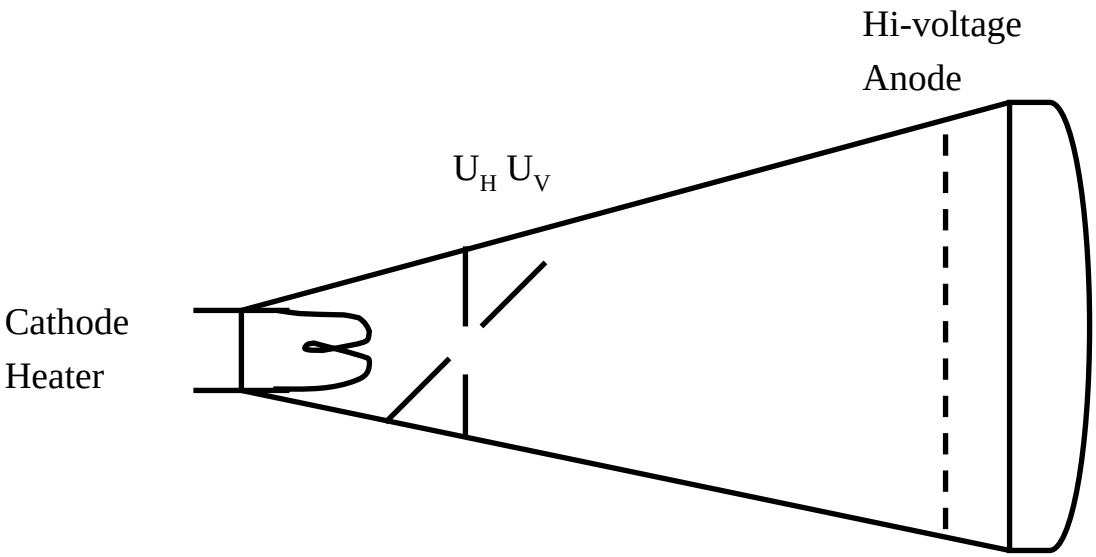
- **Clear CR2 (control register 2) – bit 2 to allow access to CR1**
- **Software reset PTM by storing ‘1’ in CR1 bit 0**
- **Store starting count value of $1F3F to latch 3**
- **Configure CR3**
- **Start counting by clearing reset bit from CR1**

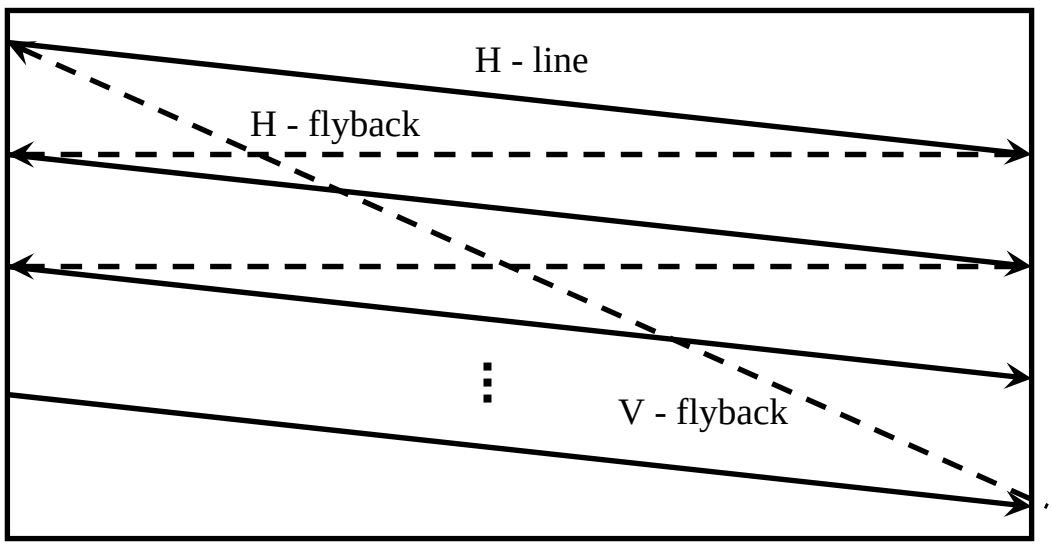
- **CR3 => %1000 0010 = $82**

Interfacing PTM (3)

```
START   CR1     EQU     $9000
        CR2     EQU     $9001
        CR3     EQU     $9002
        L3      EQU     $9006

        LDAA #$01       set bit 0 in register A
        STAA CR2        select CR1 - set bit 0 in CR2
        STAA CR1        reset PTM – set bit 0 in CR1
        CLR CR2         access CR3

        LDX #$1F3F      store $1F3F in register X
        STX L3          init counter 3:  X -> latch 3

        LDAA #$82       #%1000 0010 in CR3
        STAA CR3

        LDAA #$01       set bit 0 in register A
        STAA CR2        select CR1 - set bit 0 in CR2

        CLR CR1         start counting
        WAI
```

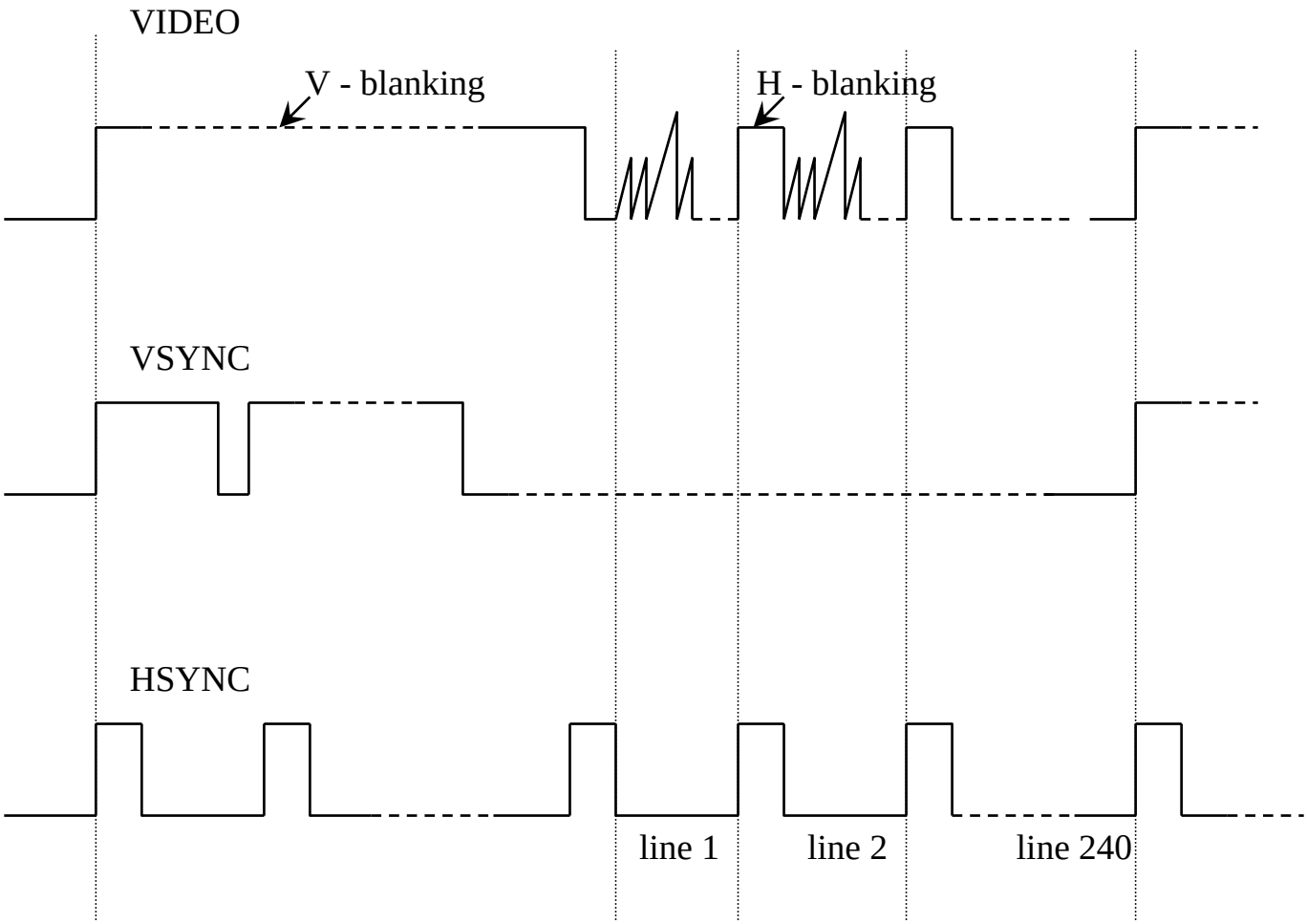
Raster-scanned Video Display





- For low resolution displays:
  - $freq_H$ = 60 Hz, $freq_V$ = 15,600 Hz
- No. of lines = 15,600 / 60 = 260
- Displays have only 240 display lines
- 20 potential lines lost for V-flyback

Raster-scanned Video Display



- <u>Video display:</u>
- 240 lines display 24 character rows, 10 V-dots each
- Each line display 80 characters, 7 H-dots each
- On the 7 x 10 matrix a 5 x 9 dot pattern is displayed

6845 CRTC
Cathode Ray Tube Controller

- 6845 CRTC
  - CRT Controller for interfacing 6800 processor with CRT and TV raster scan displays
- CPU Side Interface
  - 1 clock input (E enable)
  - synchronous Data bus (8 bits)
  - 1 chip select input (!CS)
  - 1 register select input (RS)
  - Read/write control line (R/!W)
- Peripheral Side Interface
  - Video interface:
    - HSYNC – H-line sychronisation
    - VSYNC – V-line synchronisation
    - DE – display enable
    - !RES – reset
  - Memory addressing:
    - MA0-MA13 – Refresh RAM address lines
    - RA0-RA4 – Raster address lines

CRTC Internal Structure

- Registers
  - 18 registers
  - selected using Address Register
  - status given by Status Register

R0 (8 bits) – H. Total (total no. chars per line – 1)

R1 (8 bits) – H. Displayed (no. chars. H. displayed)

R2 (8 bits) – H. Sync Pos. (position of HSYNC on line)

R3 (8 bits) – H. & V. Sync Widths (HSYNC & VSYNC widths)

R4 (7 bits) - Vertical Total (no. char. rows – 1)

R5 (5 bits) - Vertical Total Adjust (no. of additional scan lines needed to complete an entire frame scan)

R6 (7 bits) - Vertical Displayed (no. of rows displayed)

R7 (7 bits) - Vertical Sync Pos. (position of VSYNC on frame)

R8 (8 bits) – Mode Control (set addressing, interlace and skew)

R9 - Maximum Raster Address (no. of scan lines per char. row)

R10 (5 bits) - Cursor Start (start scan line for cursor)

R11 (5 bits) - Cursor End (end scan line for cursor)

R12/13 (14 bits) - Display Start Address (memory address of the first character to be displayed – top-left on the frame)

R14/15 (14 bits) - Cursor Address (current cursor position)

R16/17 (14 bits) - Light Pen Address (light pen strobe position)

CRTC Software Control

- Register Selection

!CS	RS	AR	Read (R/!W = 1)	Write (R/!W = 0)
0	0	X	Status register	Address register
0	1	n	Register Rn (n>13)	Register Rn (n<16)
1	X	X	-	-

- Status Register Format

7	6	5	4	3	2	1	0
NU5	LRF	VB	NU4	NU3	NU2	NU1	NU0

- NU5 – Not Used
- LRF – LPEN register full
  - 0 – when either R16 or R17 is read
  - 1 – when LPEN strobe occurs
- VB – vertical blanking
  - 0 – when scan is not in vertical blanking portion
  - 1 – when scan is in vertical blanking portion
- NU4 – Not Used
- NU3 – Not Used
- NU2 – Not Used
- NU1 – Not Used
- NU0 – Not Used

CRTC Mode Control Register

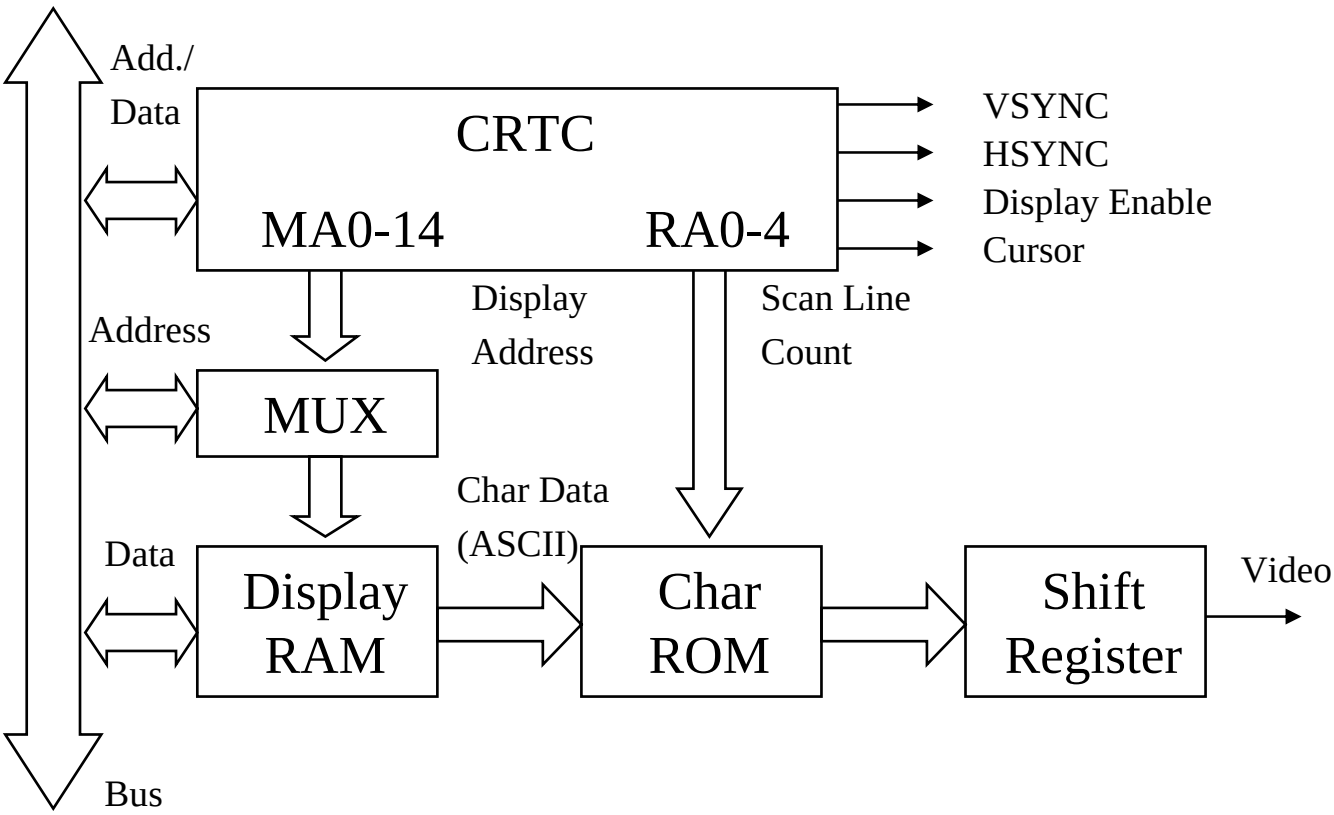
- Mode Control Register Format

7	6	5	4	3	2	1	0
NU1	NU0	CSk	DES	MP	VDA	IC1	IC0

- NU1 – Not Used
- NU0 – Not Used
- CSk – Cursor Skew
  - 0 – new delay
  - 1 – delay cursor one character time
- DES – Display Enable skew
  - 0 – no delay
  - 1 – delay display enable one character time
- MP – Must Program
  - 0 - compulsory
- VDA – Video Display RAM Addressing
  - 0 – straight binary
  - 1 – row/column
- IC1-0 – Interlaced Mode Control
  - 10 – interlace
  - X0 – non interlace
  - X1 – invalid

Interfacing CRTC (1)

- Connect an CRTC at $9000 and configure it in order to interface a CRT-display TV



Interfacing CRTC (2)

```
CRTCAR EQU    $9000
CRTCRG EQU    $9001

       CLRB           initialise register counter
       LDX #CRTTAB    initialise X with address of data
CRT    STAB CRTCAR    store in Add. Reg. reg. no. to init
       LDAA 0, X      load in A info indicated by X + 0
       STAA CRTCRG    store data in selected register
       INCX           increment value from X
       INCB           increment value from B
       CMPB #16       compare value from B with 16
       BNE CRT        jump to CRT if not equal (loop)
       SWI

CRTTAB FCB    92, 80   meant for R0 and R1
       FCB    6, $21   meant for R2 and R3
       FCB    26, 0    meant for R4 and R5
       FCB    24, 1    meant for R6 and R7
       FCB    0, 10    meant for R8 and R9
       …
```