2BA4

Final Coursework Requirements

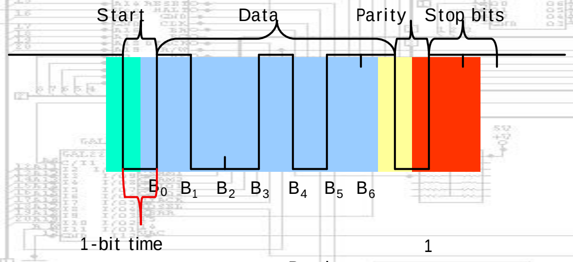
- Deadline:
  - 4:30pm, 20th February 2004
- What:
  - Project Kit, Tools, and individual Notebooks.
- To: Technicians, in Tech. Lab ORI

Note: Lab. Notebooks must be handed up at the same time as the project kits. As they are a diary of your work in the project there will be nothing to add or change once the project work has stopped.

- Project Demo:
  - 16th February 2004

2BA4

Standard serial data waveform



$$\text{Baud rate} = \frac{1}{\text{bit time}}$$

2BA4

I/O Memory Locations

- Some useful assembler symbols:
  - Base Address of $IODev_1$
    - `IODEV1      EQU     $80000`
  - Base Address of $IODev_2$
    - `IODEV2      EQU     $C0000`
  - Offset to ACIA Data Register
    - `IODATA      EQU     0`
  - Offset to ACIA Status Register
    - `IOSTAT      EQU     1`
  - Offset to ACIA Command Register
    - `IOCOMM      EQU     2`
  - Offset to ACIA Control Register
    - `IOCNTL      EQU     3`

2BA4

Initialising $IODev_1$

- Write $0B to command register
  - No parity and no interrupts
- Write $18 to control register
  - 1200 baut

2BA4

Initialising IODev$_1$ (Code)

```
* Software delay of 1/10 second
      move.w       #40000,d1         *
WAIT  DJUMP        d1,WAIT           * Assembler Macro
* Put IODev1's address into a0
      move.b       #IODEV1,a0        *
* Software reset of ACIA
      move.b       d0,IOSTAT(a0)     *
* Clear received character bit
      tst.b        IODATA(a0)        *
* Initialise command register
      move.b       #$0B,IOCOMM(a0)   *
* Initialise control register
      move.b       #$18,IOCNTL(a0)   *
```

2BA4

The Assembler Macro

- Macros allow a block of statements to be treated as a single unit.

```
addem  MACRO                 * Header
       move\0      \1,do     * Body
       add\0       \2,do
       ENDM                  * Terminator

       addem.w               num1,num2 * call
```

2BA4

Clearing Status Bits

- Bit3
  - Set when receiver data is FULL
- Bit 4
  - Set when Transmit Data is EMPTY
- How are they cleared?
  - Bit 3 cleared when processor reads receiver data
  - Bit 2,1,0 also cleared when receiver data is read
    - (overrun, framing and parity errors)

2BA4

Writing to IODev$_2$

- Character to be written in d0
- Using Polled IO

```
* Put IODev2's status register address into a0
      move.l       #IODEV2+IOSTAT,a0

* Wait until transmitter register is empty
WAIT  btst         #4,(a0)
      beq.s        WAIT

* Write character
      move.b       d0,-(ao)
```

- Note: Use of status register as address base

2BA4

Reading from IODev$_1$

- Character read into d0
- Using Polled IO

```
* Put IODev1's status register address into a0
          move.l          #IODEV1+IOSTAT,ao
* Wait until receive register is full
WAIT      btst            #3,(ao)
          beq.s           WAIT
* Read received character
          move.b          -(ao),do
* Clear top 3 bytes
          andi.l          #$FF,do
* Or clear top 3 bytes + bit 7 of byte
          andi.l          #$7F,do        * clears parity bit
```

2BA4

Transparent Linking

- All characters received on one ACIA are transmitted out the other.
- Escape mechanism usually provided:
  - Control Character from keyboard
    - (Not used by host computer OS)
- Polled IO version:
  - Polling loop must avoid **deadlock**.
    - Check all events in rotation

2BA4

When Overrun Occurs

- Receiver Overrun occurs if:
  - Receiver Data Register has filled
  - Receiver Data has not been read
  - Another complete character has been received
- ACIA contains;
  - Receiver Data Register
  - Receiver Shift Register

2BA4

T-Linking Forever (One)

```
* Runs forever!
* Character received by IODEV1?
TPCOM0      btst        #3, IODEV1+IOSTAT
            beq.s       TPCOM2
* Can character be accepted by IODEV2?
            btst        #4, IODEV2+IOSTAT
            beq.s       TPCOM2
* Read received character
            move.b       IODEV1,d0
* Clear top 3 bytes + parity
            andi.l      #$7F,d0
* Write character to IODEV2
            move.b      d0,IODEV2
* See next page:
```

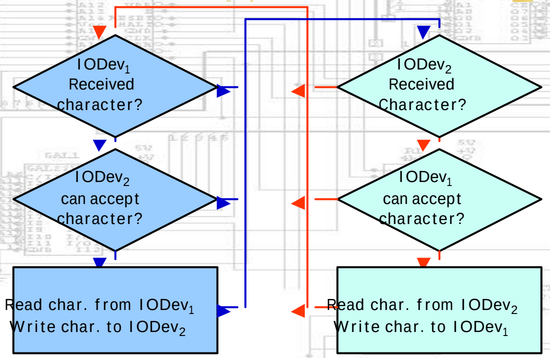
2BA4

T-Linking Forever (Two)

```
* Continued from previous page.
* Character received by IODEV2?
TPCOM2          btst            #3,IODEV2+IOSTAT
                beq.s           TPCOM0
* Can character be accepted by IODEV1?
                btst            #4,IODEV1+IOSTAT
                beq.s           TPCOM0
* Read received character
                move.b          IODEV2,d0
* Clear top 3 bytes + parity
                andi.l          #$7F,d0
* Write character to IODEV1
                move.b          d0,IODEV1
                bra             TPCOM0
```

2BA4

T-Linking Forever (Flowchart)


IODev$_1$ Received character?
IODev$_2$ can accept character?
Read char. from IODev$_1$ Write char. to IODev$_2$
IODev$_2$ Received Character?
IODev$_1$ can accept character?
Read char. from IODev$_2$ Write char. to IODev$_1$