2BA4

Purpose of the Monitor Software

- To assist software development by…
  - Host computer communication
  - Loading and running **U**ser **S**oftware (**US**)
  - Providing US access to system resources
  - Aiding the debugging
  - Providing a useful interface

2BA4

Typical User Session

- Power-up machine
  - Set up transparent link to host
  - Edit & assemble on host
  - Download software to microprocessor
  - Set breakpoint
  - Start software running
  - At breakpoint:
    - Examine & set register
  - Continue program execution

2BA4

Components of Monitor Software

1. Monitor Initialisation
2. Exception Handling
3. System Services
   - Incl. I/O Device Control
4. Program state maintenance
5. Monitor Command Software
6. Main Command Loop

2BA4

1. Monitor Initialisation

- Main (obvious) purpose:
  - To initialise all monitor variables.
- Will occur at:
  - Power-up time
  - When RESET button is pressed
    - (Hard RESET)
- May Occur at other times
    - (Software RESET)
- May want partial initialisation at other times
    - (Soft RESET)

2BA4

2. Exception Handling (One)

- Exceptions must be handled properly
  - Most of little interest
  - or should not occur…
- Default Exception Handler

2BA4

2. Exception Handling (Two)

- Ideal Behaviour:
  - Write message saying unexpected exception occurred
    - GIVE DETAIL!
  - Tidy up
  - Return to Monitor Command Loop
- Needs I/O services
- Pay careful attention to system stack contents!

2BA4

3. System Services

- Provide controlled access to system resources
  - Both hardware and software
- Uniform system service calling system
- Designed for expandability
- Key requirement:
  - No need to re-assemble user software, if monitor software changes.

2BA4

4. Program State Maint. (One)

- Program State:
  - Contents of CPU Register & Memory
- Monitor keeps register copy as Saved State
- Set up Saved State when program downloaded.

2BA4

4. Program State Maint. (Two)

- (Re)Start Program:
  - Saved State -> CPU State
- Stop Program:
  - CPU State -> Saved State
- Allow user to Query and Modify Saved State.
- Special care needed with System Stack and Pointer

2BA4

Monitor Commands/ Main Loop

- Code to parse user commands
- Subroutines to support each user commands
- Good Help facilities

2BA4

Exception Processing (One)

- When 68008 starts to process an exception it:
  1. Saves current (SR, PC) using SSP
     - This is not done for a RESET.
  2. Sets new context in Status Register
     - (S=1 [Supervisor Mode], T=0 [Trace off])
- Additional information may be pushed onto SSP – depending on exception .

2BA4

Exception Processing (Two)

3. Determines:
   - Exception Vector Number (vno)
   - Exception Vector Address (va = 4vno)
4. Setup PC to execute “Handler”
   - For RESET: SSP <- (0) and PC <- (4)
   - For others: PC <- (va)

- Vector Table (1st 1kbyte of memory) is 256 long-words, each a pointer to an Exception Handler.

2BA4

Exception Groups

- The 68008 divides exceptions into 3 groups
  - Group 0: RESET, Bus Error & Address Error
    - Exception Processing starts immediately.
  - Group 1: Trace, Interrupt, Illegal Instr., Unimplemented Instr., Privilege Violation.
    - Exception Processing starts end end of current instruction.
  - Group 2:Trap Instruction
    - Exception Processing starts as consequence of current instruction

2BA4

Modifying Handlers

- User Programs often want own exception handlers.
  - But Vector Table is fixed in ROM.
- Idea:
  - Place "real" vector table in RAM (modifiable)
- Get ROM table to point "through" RAM table
- -> Need to initialise RAM table!

2BA4

Programming the ACIAs

- ACIAs take up four bytes
  - Base-Address + 0 … Base-Address +3
  - 0: Data Register (Transmit/Receive)
    - Write data for transmission to it.
    - Read received data from it.
  - 1: Status Register
    - Interrupt condition and other status
  - 3: Command Register
    - Controls communication functions
  - 4: Control Register
    - Controls communication speed.

2BA4

Data Register

- Two-in-one
- One address accesses both
  - Transmit Register (write-only)
  - Receive Register (read-only)
- Data is transmitted and received
  - Least-significant bit first.

Status Register

- Contents set by ACIA
- A READ transfers contents to CPU
- A Write causes a programmed RESET of ACIA
  - Bit 7: set if interrupt has occurred
  - Bit 4: set if Transfer Register empty
  - Bit 3: set if Receiver Register full
  - Bit 2: set if overrun has occurred
  - Bit 1: set if framing error has occurred
  - Bit 0: set if parity error occurred

Command Register

- Contents read/written by CPU
  - Bits 7-6:
    - Parity control
  - Bit 5:
    - Set if Parity mode enabled
  - Bit 3-2:
    - Transfer interrupt control
  - Bit 1:
    - Set if $\overline{\text{IRQ}}$ interrupt enabled

Command Register Example

- $0B or %00001011
  - Odd parity send and received
  - Parity disabled
  - Receiver Normal Mode
  - RTS low, interrupt disabled
  - IRQ (Receiver) disabled
  - DTR low

Control Register

- Contents read/written by CPU
  - Bit 7:
    - Control number of stop bits
  - Bit 6-5:
    - Word length (8…5 bits)
  - Bit 4:
    - Receiver baud rate source
  - Bit 3-0:
    - Transmitter baud rate

2BA4

Control Register Example

- $18 or % 0001 1000
  - 2 Stop Bits (depending on parity)
  - Word Length = 8
  - Rx baud = Tx baud
  - Tx baud = 1200