

3 SINGLE STEP

3.1 General

There is a simple single step mechanism in the ETRAX 100LX. The single step is controlled by 4 mode bits (19:16) in the R_TEST_MODE register.

Bit nr.	Name	Description
19	single_step	Turns single step on and off.
18	break_write	Enables the single step unit to break execution on memory write cycles.
17	break_read	Enables the single step unit to break execution on memory data read cycles (as opposed to instruction fetch).
16	break_fetch	Enables the single step unit to break execution on instruction fetch cycles. This will break both fetch from memory and fetch from the CPU internal prefetch register.

Table 3-1 Single Step Control Bits in the R_TEST_MODE Register

The single step unit is started by first setting the appropriate bits in the R_TEST_MODE register, and then using the RBF instruction to jump to the code through which the single step should be performed.

The single step unit will issue a bus fault on the first CPU cycle that matches the cycle types selected in the R_TEST_MODE register.

The single step bus fault uses the interrupt vector number 0x20, which is the same interrupt vector that is used by the HW break mechanism. As a result, it is not recommended that the single step and the HW break mechanisms be enabled at the same time.

After the bus fault has occurred, the single step unit is disabled until after the next RBF instruction has been executed and the restarted cycle has been completed.

If a single step bus fault occurs at the same time as an MMU bus fault, the single step bus fault will have priority. This means that the single step routine must be able to check for this case and invoke MMU handling as well, but the MMU handler doesn't need to be aware of the single step mechanism. To check for concurrent MMU and single step bus faults, there is a **both_faults** bit (R_BUS_STATUS bit 4) that is set when this happens. The **both_faults** bit is cleared when the CPU runs an RBF instruction.

3.2 Programming Considerations

A special case occurs with interrupts when the **break_read** bit is set. The single step unit will then break also on interrupt vector read cycles. This case can be identified by a bit in the CPU status record, but the interrupt sequence cannot be automatically restarted by the RBF instruction. The single step handler routine must check for this case and compensate for it.

When single stepping through code that uses the integral read-write mechanism, the 'old F flag' in the CPU status record must be checked, and the F flag in CCR must be restored if there wasn't a concurrent MMU bus fault. Otherwise, the conditional write will always fail in single step mode, and the program will hang.

Another problem that has to be taken care of occurs when single stepping through code that pops data from the supervisor stack. After a bus fault in the data read cycle of a pop, the stack pointer will already be incremented. If the single step handler uses the stack this will clobber the stack data that was going to be read when the bus fault occurred.

There are two possible solutions for this problem:

- 1 Reserve extra space on the stack.

Example:

```
SBFS [SP = SP - 80]                ; Reserve 64 extra bytes
PUSH DCCR
DI
:
:
:
MOVE [SP = SP + 84], DCCR          ; Dummy read to adjust SP, and
                                   ; lock out irq.
MOVE [SP - 84], DCCR              ; "POP" DCCR
RBF [SP - 80]
```

This will reserve stack space corresponding to the maximum SP increment that could occur with Autoincrement addressing mode. The maximum of 64 bytes occurs with the MOVEM [SP+], PC instruction.

This method will only work as long as the program you are single stepping through doesn't itself use the same method as described above to access values on the stack after the stack pointer has been incremented.

- 1 Store status at absolute address and have a separate stack for the single step handler.

Example:

```
SBFS [STATUS_RECORD]              ; Save CPU status to
                                   ; absolute address.
MOVE DCCR, [SAVED_DCCR]           ; Save DCCR to absolute address.
DI
MOVEM SP, [SAVED_REGS]            ; Save all registers to
                                   ; absolute address.
MOVE.D SINGLE_STEP_SP, SP        ; Load new sp value for
                                   ; single step handler.
:
:
:
MOVEM [SAVED_REGS], SP
MOVE [SAVED_DCCR], DCCR
RBF [STATUS_RECORD]
```