Interrupt Handler — PLS179

As an example of designing a microprocessor family part, consider Figure 1, which depicts an interrupt handler. In particular, note that interrupt inputs will be latched into an 8-bit register. This in turn will be encoded to a 3-bit vector which may be appropriately enabled and applied to the microbus. Figure 1 shows the eight flip-flops as having J-K and /D inputs which will be generated with a PLS179 by switching the flip-flop control. Appropriate control signals for the various transactions might be as follows:

1. CLOCK – the system synchronous time base.
2. Interrupt Enable – when asserted high from the microprocessor, allows interrupts to be generated to the microprocessor.
3. Interrupt – a strobe or level defined to indicate a pending interrupt and a valid encoded vector.
4. Interrupt Acknowledge – a response signal from the microprocessor which may be used to enable the 3-bit vector onto the bus. As well, it may initiate clearing the currently asserted interrupt latch.
5. /INT0–/INT7 – eight possible interrupt request signals which must be asserted low and held there until service for that device has occurred.
6. Reset – this is a system override signal which will clear all flip-flops during initial operation.

Basic Operation

Initially, the part should be reset by asserting the RESET pin high, asynchronously. Then, when interrupts are enabled, the /D-inputs to the 8 flip-flops will be synchronously scanning for interrupt inputs (asserted low). This will put a nonzero value into the eight bit register which will generate an interrupt output, combinationally through the Complement array. In parallel, a 3-bit encoded vector will be applied on the VEC0, VEC1, VEC2 lines. Asserted high logic will be assumed for the vector. Presumably, a microprocessor will interrupt this, transfer control to a service routine and clear the interrupt. The clear will be accomplished by disabling interrupts and strobing the vector value back into the PLS179, using the IACK signal. Disabling the interrupts will put the registers into J-K mode. J is tied to zero and K is decoded from the specifically strobed vector. Therefore, synchronous clear of the high priority bit is done. Interrupts are then re-enabled and the process continues.

The PLS179 solution offers room for user alteration. For example, the IACK condition could be redefined as a combination of the Z80 IOREQ and M1 signals, or any specific splitting of internal signals could be easily done. The design could fit into a PLS159A, but there would be less room for variation for specific users exact needs. Figure 3 shows the pinlist for the handler. Figure 4 gives the corresponding design file.

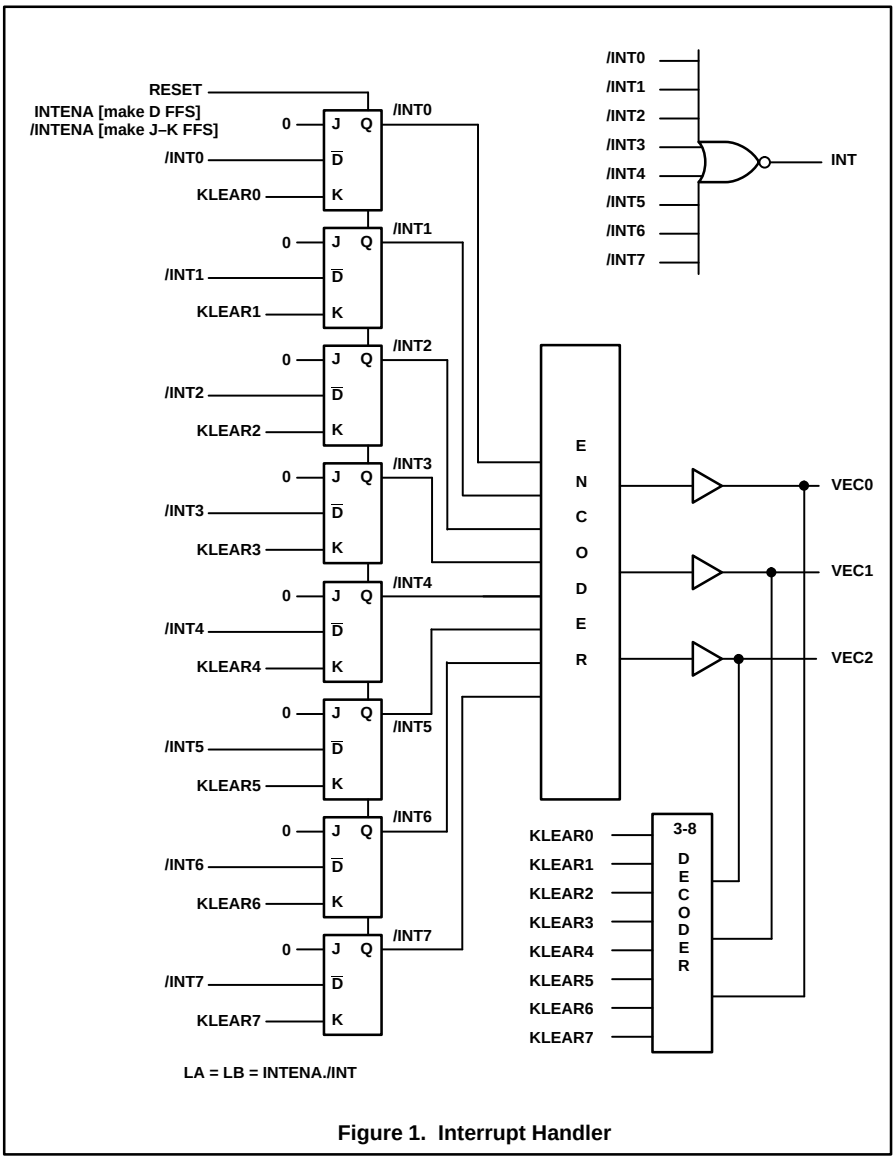

Figure 1. Interrupt Handler

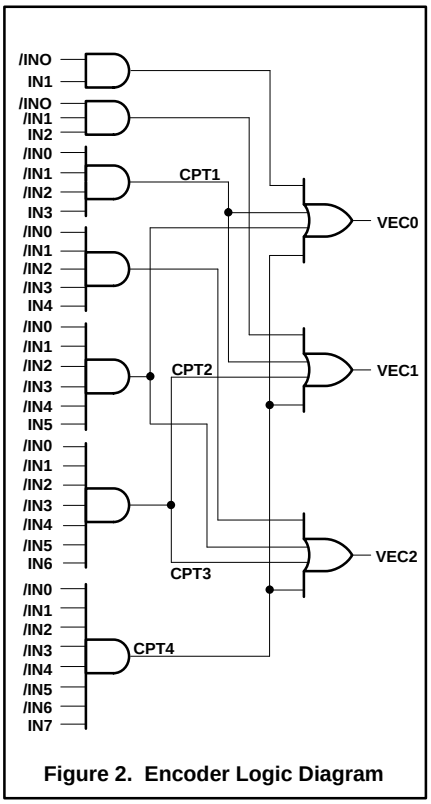

Figure 2. Encoder Logic Diagram

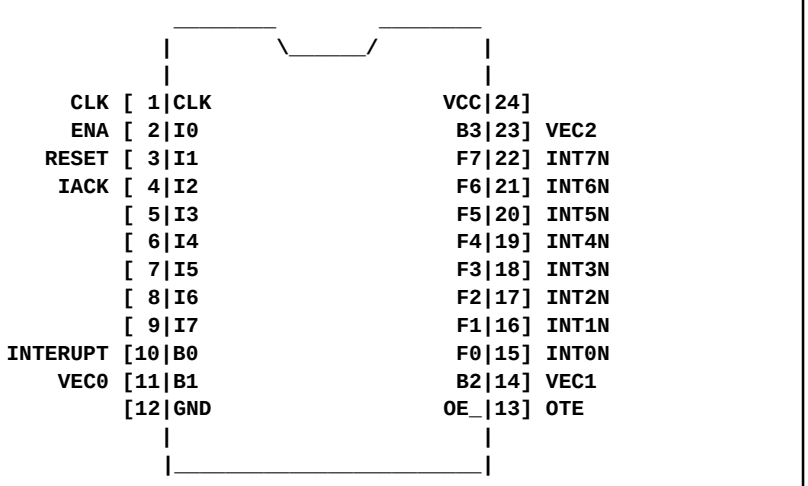

Figure 3. Interrupt Handler Pin List

```
@PINLIST
OTE          I;          INT4N        B;
IACK         I;          INT3N        B;
RESET        I;          INT2N        B;
ENA          I;          INT1N        B;
CLK          I;          INT0N        B;
VEC2         B;          VEC1         B;
INT7N        B;          VEC0         B;
INT6N        B;          INTERUPT     O;
INT5N        B;

@LOGIC EQUATIONS

”Encoder Equations”
CPT1 = INT0*INT1*INT2*/INT3;
CPT2 = INT0*INT1*INT2*INT3*INT4*/INT5;
CPT3 = INT0*INT1*INT2*INT3*INT4*INT5*/INT6;
CPT4 = INT0*INT1*INT2*INT3*INT4*INT5*INT6*/INT7;
VEC0 = (INT0*/INT1+CPT1+CPT2+CPT4);
VEC1 = (INT0*INT1*/INT2+CPT1+CPT3+CPT4);
VEC2 = (INT0*INT1*INT2*INT3*/INT4+CPT2+CPT3+CPT4);
VEC0.oe = ENA;
VEC1.oe = ENA;
VEC2.oe = ENA;

C = /(/INT0+/INT1+/INT2+/INT3+/INT4+/INT5+/INT6+/INT7);
INTERUPT = C;
INTERUPT.oe = ENA;

”Decoder Equations”
KLEAR0 = /VEC2*/VEC1*/VEC0*IACK; ”DECODE VECTOR 0”
KLEAR1 = /VEC2*/VEC1*VEC0*IACK;  ”DECODE VECTOR 1”
KLEAR2 = /VEC2*VEC1*/VEC0*IACK ; ”DECODE VECTOR 2”
KLEAR3 = /VEC2*VEC1*VEC0*IACK ;  ”DECODE VECTOR 3”
KLEAR4 =  VEC2*/VEC1*/VEC0*IACK; ”DECODE VECTOR 4”
KLEAR5 =  VEC2*/VEC1*VEC0*IACK ; ”DECODE VECTOR 5”
KLEAR6 =  VEC2*VEC1*/VEC0*IACK ; ”DECODE VECTOR 6”
KLEAR7 =  VEC2*VEC1*VEC0*IACK  ; ”DECODE VECTOR 7”

”Register equations”
INT0.J=in0j;                  INT4.J=in4j;
INT0.K=KLEAR0+in0k;           INT4.K=KLEAR4+in4k;
INT1.J=in1j;                  INT5.J=in5j;
INT1.K=KLEAR1+in1k;           INT5.K=KLEAR5+in5k;
INT2.J=in2j;                  INT6.J=in6j;
INT2.K=KLEAR2+in2k;           INT6.K=KLEAR6+in6k;
INT3.J=in3j;                  INT7.J=in7j;
INT3.K=KLEAR3+in3k;           INT7.K=KLEAR7+in7k;

”Register RESET equations”
INT0.rst=RESET;               INT4.rst=RESET;
INT1.rst=RESET;               INT5.rst=RESET;
INT2.rst=RESET;               INT6.rst=RESET;
INT3.rst=RESET;               INT7.rst=RESET;
```

Figure 4. Interrupt Handler Design File