Technical Note No. 118

Jackrabbit (BL1800) Series External Interrupts

Introduction

The Rabbit 2000 microprocessor has four external interrupt inputs on Parallel Port E. Parallel Port E is accessed through pins PE0–PE7 on the Jackrabbit's header J4, and the pinout for header J4 is shown in Figure 1.

Table 1 lists the general-purpose Parallel Port E I/O pins that can be used for external interrupts. As shown in Table 1, the default use for PE0 and PE1 is with high-power outputs HV0 and HV1. This leaves PE4 and PE5 available for use as external interrupt inputs.

Table 1. Jackrabbit Use of Rabbit 2000 Parallel Port E

Pin	Jackrabbit Default Use	Jackrabbit Alternate Use
PE0	HV0 output	INT0A input
PE1	HV1 output	INT1A input
PE4	General-purpose I/O	INT0B input
PE5	General-purpose I/O	INT1B input

Figure 2 illustrates the use of these pins on the Jackrabbit.

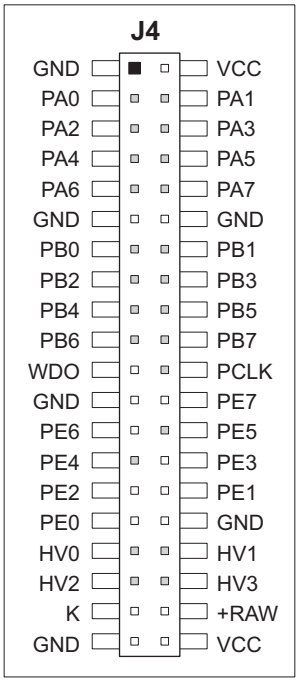

Figure 1. Jackrabbit Header J4 Pinout

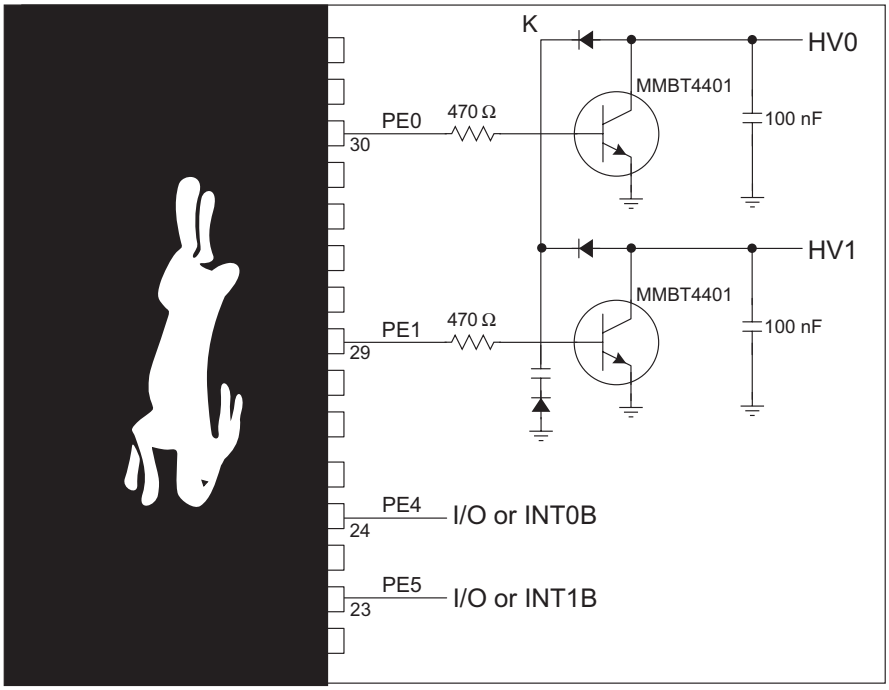

Figure 2. Jackrabbit Use of Rabbit 2000 Parallel Port E

Use of External Interrupts

Figure 3 shows a block diagram of how the Rabbit 2000 external interrupt logic is used in general.

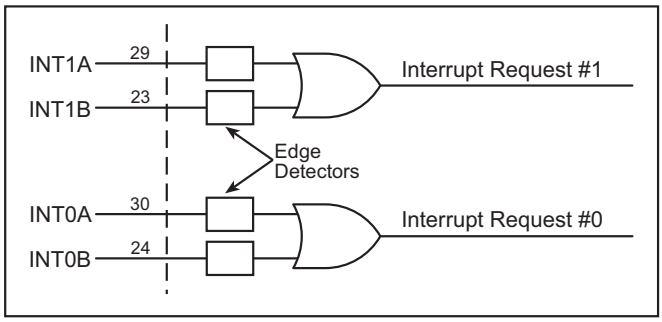

Figure 3. Rabbit 2000 External Interrupt Logic

Interrupts on the Rabbit 2000 can take place at three priority levels from low to high priority, and are numbered as 1, 2 and 3. Each on-chip device, including the two external interrupts, can be assigned a priority at which interrupts will take place. For interrupts that have been assigned the same programmed priority, there is an implicit priority with external interrupt #1 having the highest priority, external interrupt #0 the second highest, and the remaining on-chip devices having lower priorities in the order specified in Section 7.8, "Rabbit Interrupt Structure," in the ***Rabbit 2000 User's Manual***.

The two independent interrupts are generated by inputs to the four pins shown in Figure 3. Each pin is connected to an edge detector that can be configured under program control to detect rising edges, falling edges, or both. These same pins, a part of parallel port E, support alternate functionality as reflected in Table 1.

When the edge detector detects the rising or falling edge that it is programmed to detect, it sets a flip-flop that drives the output of the edge detector. The flip-flop should be cleared automatically when the interrupt takes place.

Instead, the flip-flop may be cleared spuriously because a different, *lower priority*, interrupt occurs nearly simultaneously (during an 8-clock window) with the occurrence of the edge that sets the flip-flop. This results in a lost interrupt.

Or the flip-flop might not be cleared when the interrupt takes place if a different, *higher priority*, interrupt is being requested nearly simultaneously (during an 8-clock window) with the occurrence of the external interrupt. This results in a spurious interrupt after the first interrupt because the interrupt request was not cleared.

In either case, precautions need to be taken if an interrupt request transitions during a short time period 8 clocks long. These sequences are shown schematically in Figure 4.

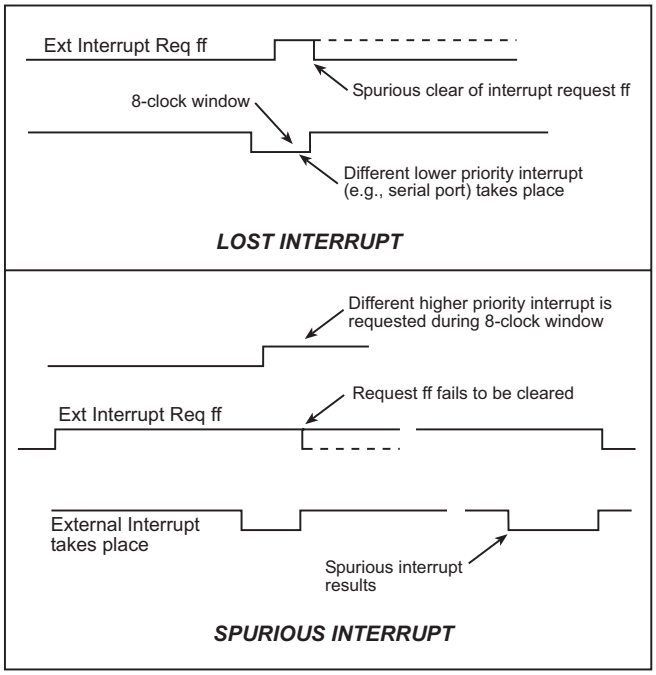

Figure 4. Interrupt Sequences with Lost or Spurious Interrupts

Single-Interrupt Request

Tie the inputs for external interrupt #1 and #0 together by adding a 1 kΩ resistor between pins 26 and 27 of Jackrabbit header J4 as shown in Figure 5.

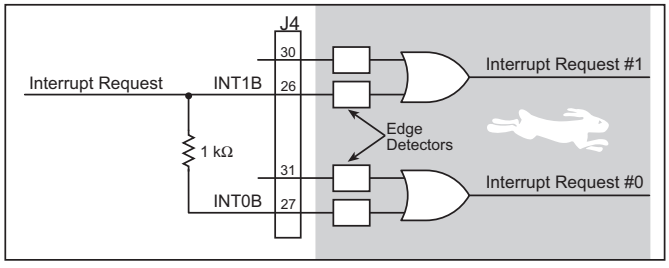

Figure 5. Jackrabbit Configuration for Single-Interrupt Request

Using this configuration, both interrupt #1 and #0 will be requested when an edge is detected. The #1 interrupt will take place first since it is of a higher priority.

The interrupt service routine for interrupt #1 should ignore the interrupt. The actual service routine will be the service routine for interrupt #0. If an interrupt is lost, it will always be #1 and never #0. The 1 kΩ resistor delays the edge slightly so that interrupt #1 is guaranteed to be latched earlier or simultaneously with interrupt #0. It is important that the programmed priority of interrupt #1 be higher than or equal to the programmed priority of interrupt #0. Normally they should be equal.

Spurious interrupts, which occur because of a failure to clear the request latch, are a possibility only if there are other interrupts of higher priority than external interrupt #1 and #0. These can only be the result of programming one of the on-chip peripheral interrupts to have a higher interrupt priority. This could be the case, for example, if the external interrupts are programmed to have priority 1, and one of the serial port interrupts is programmed to have priority 2. Spurious interrupts can always be eliminated by programming both external interrupts to have a priority equal to the highest priority used for another device. The priority can be reduced on entry to the service routine to avoid blocking the true high-priority interrupts. External interrupt #1 cannot cause interrupt #0 to have a spurious interrupt or vice versa. In some cases, spurious interrupts may not disturb function, but the fix is so simple that it is not usually worth the trouble to analyze this possibility.

Software

A new function, **SetVectExtern2000**, has been added to the Dynamic C **SYS.LIB** library, along with a sample program. This software is included with the Dynamic C 6.19 upgrade, which is available on the Z-World Web site at www.zworld.com.

The first parameter of the **SetVectExtern2000** function is now the priority level of the external interrupt, and the second is the ISR itself. This is different from the existing **SetVectExtern** function.

Sample Program

The sample program below demonstrates how to use the Jackrabbit external interrupt inputs to trigger an interrupt service routine. Two external interrupt inputs are used to ensure that no interrupts are missed. Before running the sample program, attach the input line to pin 26, header J4, as shown in Figure 5. The program will count pulses on that input line until 20 pulses have been counted. The keyword **interrupt** is used to create an interrupt service routine.

The sample program is available as **INT0DEMO.C** in the Dynamic C **SAMPLES/INTRUPTS** folder.

```
char count;
void my_isr();
main() {
    count = 0;

    WrPortI(PEDDR, &PEDDRShadow, 0x00);  // set port E as all inputs

    SetVectExtern2000(1, my_isr);
    WrPortI(I0CR, NULL, 0x2B);           // enable external INT0 on
                                         // PE4, rising edge
    WrPortI(I1CR, NULL, 0x2B);           // enable external INT1 on
                                         // PE5, rising edge

    while (count < 20) {

        // outout the interrupt count every second
        costate {
            printf("count = %3d\n", count);
            waitfor(DelaySec(1));
        }
    }

    WrPortI(I0CR, NULL, 0x00);           // disable external interrupt 0
}

/****** interrupt routine for external interrupt 0 ******/

nodebug root interrupt void my_isr() {
    count++;
}
```

OR'ed Interrupt Request

When two separate interrupt requests need to be used, you will lose high-power outputs HV0 and HV1 on the Jackrabbit. To ensure that sufficient current is available to drive the source of the interrupt request and to ensure that interrupt #1 is received before interrupt #0, either R49 and R53 ***or*** Q26 and Q27 need to be removed from the Jackrabbit board. The locations of these components are shown in Figure 6.

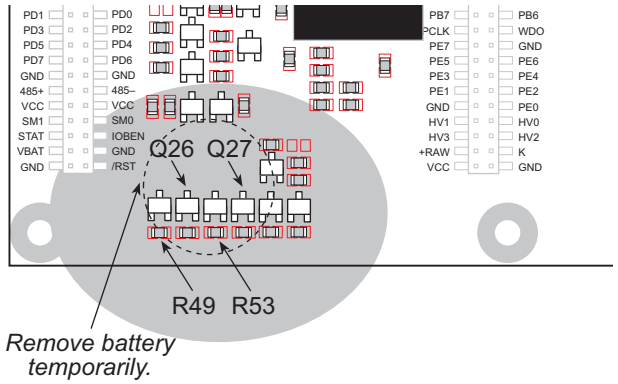

Figure 6. Location of Jackrabbit Components to Remove for Second Interrupt Request

Note that the lithium battery needs to be removed temporarily to access these components. Replace the lithium battery after removing these components.

Tie the inputs for external interrupt #1 and #0 together by adding a 1 kΩ resistor between pins 26 and 27 and between pins 30 and 31 of Jackrabbit header J4 as shown in Figure 7.

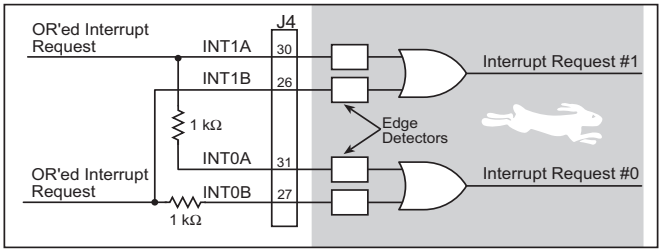

Figure 7. Jackrabbit Configuration for OR'ed Interrupt Request

Z-World Corporate Headquarters
2900 Spafford Street
Davis, California 95616 U.S.A.
530.757.3737 • Fax: 530.753.5141

http://www.zworld.com