Z i L O G

Technical Note

Setting Interrupts in ADL Mode on eZ80190, eZ80L92, eZ80F92, and eZ80F93 Devices

TN002102-1203

General Overview

This Technical Note describes how to set maskable interrupts for the eZ80190, eZ80L92, eZ80F92, and eZ80F93 devices in ADL mode. The document discusses how to relocate the interrupt vector table and map interrupt service routines in the interrupt vector table.

A broader discussion about this topic covers the entire family of eZ80® devices in both ADL and Z80 modes. Please refer to the ZiLOG Application Note titled *Setting Interrupts with the eZ80® CPU* (AN0170).

Discussion

All maskable interrupts for the eZ80® family of devices use the eZ80® CPU's vectored interrupt function. The eZ80F91-based interrupt vector locations have a 24-bit address. The remainder of the eZ80® devices, which are discussed in this Technical Note, have a 16-bit address. See Table 1, which lists the interrupt vector addresses in ADL mode for each of these eZ80® devices.

Table 1. Interrupt Vector Address for eZ80® Devices in ADL Mode

eZ80® Device	Size of I Register	Size of IVECT Register	ISR Address (ADL Mode)
eZ80F91	16 bits	9 bits	{I[15:1], IVECT[8:0]}*
eZ80F92	8 bits	8 bits	{I[7:0], IVECT[7:0]}
eZ80F93	8 bits	8 bits	{I[7:0], IVECT[7:0]}
eZ80L92	8 bits	8 bits	{I[7:0], IVECT[7:0]}
eZ80190	8 bits	8 bits	{I[7:0], IVECT[7:0]}

Note: Only 15 bits of the I Register are used. The 16th bit is overwritten by the msb of the IVECT Register.

Relocating the Interrupt Vector Table

TIMER0 (PRT 0) ISR starts at the three-byte address location `204800h`. The default TIMER0 interrupt vector location resides at `0Ah` for the eZ80F92 MCU. Assuming that the interrupt vector table is relocated to start at address `E000h`, this start location points to location `E114h`, which contains a jump instruction to the TIMER0 ISR address. Figure 1 illustrates the default and relocated jump tables for the TIMER0 ISR location.

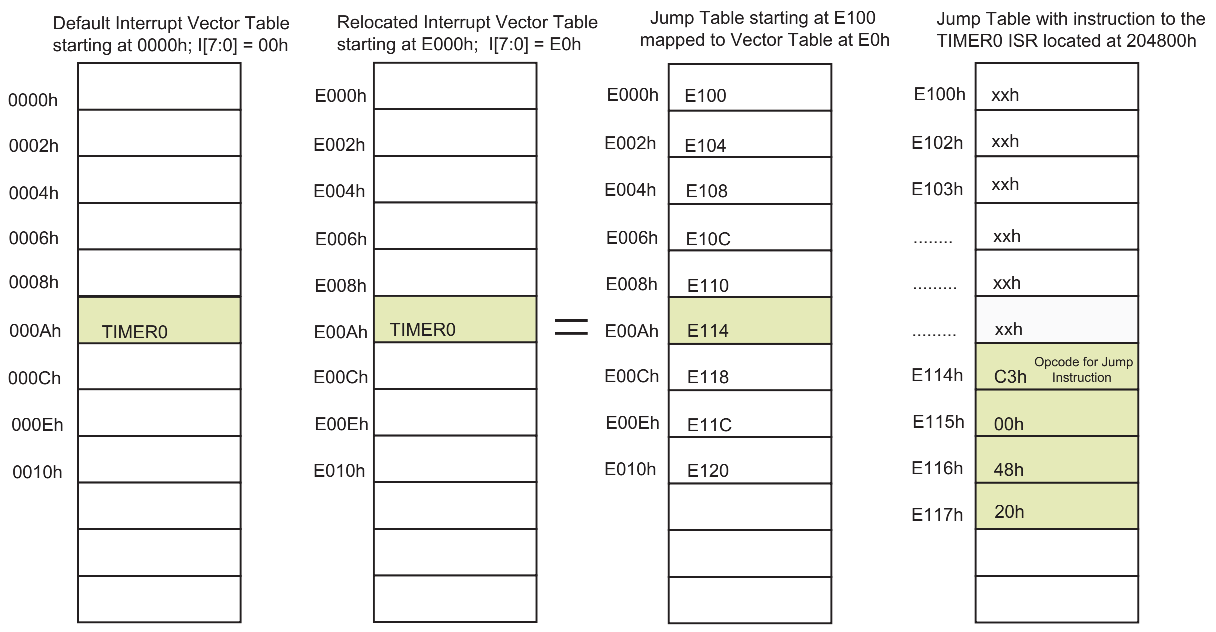

Figure 1. Memory Map Relocating Interrupt Vector Address with Jump Table

The following start-up code illustrates the type of code that must be added to the user's existing `startup.asm` file, when relocating the interrupt vector table using eZ80® devices other than the eZ80F91 MCU, with the 8-bit I Register and the 8-bit IVECT Register.

In this example, the interrupt vector table is relocated to address location `E000h` within on-chip SRAM.

```
;***********************************************************************
;Each interrupt vector is a 16-bit address pointing into the __vecptr
;segment. This segment must be aligned on a 256 byte boundary of RAM ;
;and must reside in the lower 64KB of memory.
;***********************************************************************
.assume ADL=1                       ;This is an assembler
                                    ;directive

RELOCATED_VECTOR_TABLE  EQU  E000h
NUM_VECTORS             EQU  128
. def __vector_table
define __vectab, space=RAM, align=256
. sect "__vectab"
ORG RELOCATED_VECTOR_TABLE
__vector_table:
ds NUM_VECTORS*2                    ;Each vector is a 2-byte
                                    ;address pointing into the
                                    ;__vectptr segment
```

```
;*************************************************************************
; This start-up code relocates the vector table from absolute 0000h
; location to E000h location. It loads I Registers with the value E0h.
;*************************************************************************
im 2                            ; Interrupt mode 2
ld, __vector_table >> 8 & 0ffh
ld i, a                         ; Load interrupt vector base
;************************************************************************
*;************************************************************************
```

The following code is an illustration of the kind of code that can be added to the user's existing `startup.asm` file to locate the jump table within the 64 KB memory space, when using eZ80® devices other than the eZ80F91 MCU.

In the following sample code, the jump table is relocated to address location `E100h` within on-chip SRAM.

```
;*************************************************************************
;Define jump table. Each entry is a JP.LIL to an interrupt handler.
;This segment must reside in the lower 64KB of RAM.
;*************************************************************************
RELOCATED_JUMP_TABLE  EQU E100h
define __jumptab,space=RAM
.sect "__jumptab"             ; __vectors is predefined
ORG RELOCATED_JUMP_TABLE
__jump_table:
ds NUM_VECTORS*4              ; Each entry is a JP.LIL to a handler
;************************************************************************
*;************************************************************************
```

The following lines of code illustrate how to map the jump table to the relocated vector table.This start-up code must be added to the user's existing `startup.asm` file.

```
;************************************************************************
ld hl,__vector_table
ld b,NUM_VECTORS
ld iy,__jump_table
$1:
ld.sis (hl),iy                ; store vector
inc hl
inc hl                        ; next vector address
lea iy,iy+4                   ; next jp.lil address
dec b
jr nz,$1
;************************************************************************
*;************************************************************************
```

The following start-up code illustrates the type of code that must be added to the user's existing `startup.asm` file when relocating the interrupt vector table and using the 8-bit `I` Register and the 8-bit IVECT Register on eZ80® devices other than eZ80F91.

In the following sample code, the interrupt vector table is relocated to address location `3E00h` in on-chip Flash memory.

```
;*********************************************************************
; Each interrupt vector is a 16-bit address pointing into the __vecptr
; segment. This segment must be aligned on a 256 byte boundary of ROM ;
;and must reside in the lower 64KB of memory.
;*********************************************************************
.assume ADL=1                         ; This is an assembler directive

RELOCATED_VECTOR_TABLE  EQU  3E00h
NUM_VECTORS             EQU  128
. def __vector_table
define __vectab, space=ROM, align=256
. sect "__vectab"
ORG RELOCATED_VECTOR_TABLE
__vector_table:

dw __jump_table, __jump_table+2,------------- __jump_table+254

;*********************************************************************
; This start-up code relocates the vector table from absolute 0000h
; location to E000h location. It loads I Registers with the value E0h.
;*********************************************************************
im 2                            ; Interrupt mode 2
ld, __vector_table >> 8 & 0ffh
ld i, a                         ; Load interrupt vector base
;********************************************************************
*;*******************************************************************
```

The following code is an illustration of the kind of code that can be added to the user's existing `startup.asm` file to locate the jump table within the 64KB memory space, when using the eZ80® devices other than the eZ80F91 MCU.

In the following sample code, the jump table is relocated to address location `3F00h` in on-chip Flash memory.

```
;********************************************************************
;Define jump table. Each entry is a JP.LIL to an interrupt handler.
;This segment must reside in the lower 64KB of ROM.
;*********************************************************************
RELOCATED_JUMP_TABLE EQU E100h
define __jumptab,space=ROM
.sect "__jumptab"                          ; __vectors is predefined
ORG RELOCATED_JUMP_TABLE
__jump_table:
```

```
ORG RELOACATED_JUMP_TABLE+%0A            ;%0A is the address of the
                                         ;timer0 interrupt vector
Db      %C3;Op Code for jump
.trio _ISR_timer0                        ;maps the address of the
                                         ;timer0_isr
XREF _ISR_timer0                         ;to relocated jump table
;**********************************************************************
;
*;*********************************************************************
 ;
```

Mapping the ISR Location in the Interrupt Vector Table

The maximum addressable capacity of eZ80® devices is 16MB.

Note: ZiLOG recommends that the I Register value for eZ80® devices be changed from its default value of 00h to avoid conflict between the NMI, RST instruction addresses, and the maskable interrupt vectors.

The interrupt vectors must be located within the 64KB address space for eZ80® devices other than the eZ80F91 MCU. Because the interrupt vector location can take only two-byte addresses (see Table 1), the ISR must also be located within the same 64KB address space. However, by using a jump table, the ISR can be located anywhere in the 16MB address space. The jump table, however, must be located within the 64KB memory space where the interrupt vectors are also located.

For example, the default TIMER0 (PRT 0) interrupt vector for the eZ80190 MPU is located at 06h. Therefore, the TIMER0 interrupt service routine's address—123456h, is mapped as follows:

```
{I Register [7:0], 06h}  ------------>   56h
{I Register [7:0], 07h}  ------------>   34h
{I Register [7:0], 08h}  ------------>   12h
{I Register [7:0], 09h}  ------------>   NOT USED
```

Writing the Interrupt Service Routine

To create an interrupt service routine in ADL mode, the interrupt keyword is used. This keyword is a storage class that is applicable only to functions. Alternatively, a keyword combination of #pragma interrupt can be used.

For example, to write an interrupt service routine for TIMER0, use either of the two code segments presented below.

```
interrupt void ISR_Timer0 (void)
{
  unsigned char temp= 10;
  ....;
  ....;
}
```

or

```
#pragma interrupt
void ISR_Timer0 (void)
{
 unsigned char temp = 10;;
 ....;
 ....;
}
```

The `_set_vector` function is used to attach an interrupt service routine (which is a C function) to an interrupt vector. The `_set_vector` routine takes two arguments—the first is an integer defining the interrupt number, and the second is the name of the associated interrupt service routine.

The following sample code illustrates how the `_set_vector` function is called for the remainder of the eZ80® devices.

```
# define TIMER0 0x14          // Vector offset value for TIMER0
      //(PRT 0)as mentioned in vector table
      // for eZ80F92 * 2 (0Ah * 2)
      // Vector offset for TIMER0 (PRT 0)
      // for eZ80190 is 06h (06h * 2)

# include <ez80.h>

void ISR_TIMER0();            // Function prototype declaration
void  Init_TIMER0 (void);     // Function prototype declaration

void set_vector(unsigned short int, void(*handlr)(void);

Init_TIMER0 ( )
{
  _set_vector(TIMER0, ISR_Timer0);
  Initialize TIMER0;

  ....;
  ....;
}
```

The following start-up code must be added to the user's existing `startup.asm` file.

```
;*********************************************************************
;Define __set_vector to install a user interrupt handler
;
;void _set_vector(unsigned short vector, void (*hndlr)(void));
;
;
;Argument1 - address of user interrupt handler
```

```
;Argument2 - define  TIMER0  0x14 (for eZ80F92)
;
;*********************************************************************
  .def __set_vector
  __set_vector:
  ld ix, 0
   add      ix, sp
   ld       hl,0                       ; Clear UHL
   ld.sis   hl, (ix+3)                 ; Vector offset
   ld bc, RELOCATED_JUMP_TABLE

   add       hl, bc                    ; hl is address of jp
   ld        (hl), %C3                 ; Op Code for jump
   inc       hl                        ; hl is address of handler
   ld        bc, (ix+6) handler
   ld        (hl), bc                  ; store new vector address
   ret

;********************************************************************
*;*******************************************************************
```

This publication is subject to replacement by a later edition. To determine whether a later edition exists, or to request copies of publications, contact:

ZiLOG Worldwide Headquarters
532 Race Street
San Jose, CA 95126
Telephone: 408.558.8500
Fax: 408.558.8300
www.zilog.com

Information Integrity

Document Disclaimer