Technical Note

Setting Interrupts in ADL Mode on eZ80F91 MCU

TN002004-0707

General Overview

This Technical Note describes how to set maskable interrupts for the eZ80F91 MCU 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. For more information, refer to *Setting Interrupts with the eZ80 CPU Application Note (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. Other eZ80 devices have a 16-bit address. Table 1 lists the interrupt vector addresses in ADL mode for all the 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

The interrupt vector table is constrained to start at the boundary of 512 bytes. Using the 15 bits of the I Register [15:1], the interrupt vector table is mapped to any of the 32,768 pages (16 MB divided into pages of 512 bytes).

Figure 1 illustrates the boundaries where the interrupt vector table is located. While relocating the interrupt vector table, the vector starting address (for example, `000000h` and `000200h`) must be placed at the beginning of the page address and not in the middle of the page.

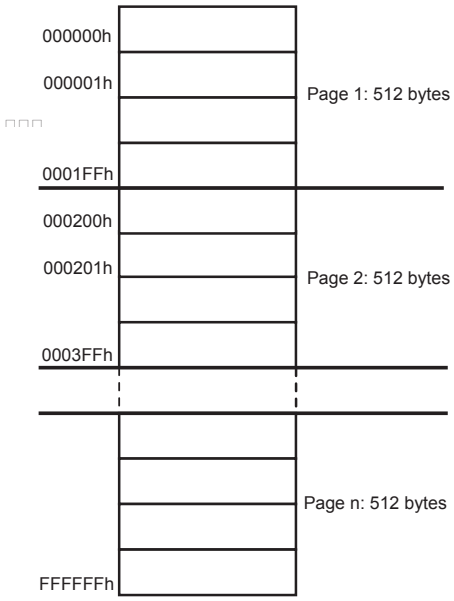

Figure 1. Interrupt Vector Table Boundaries for Relocation on the eZ80F91 MCU

The following start-up code is an illustration of code that must be added to the existing `startup.asm` file when relocating the interrupt vector table for the eZ80F91 MCU.

In this example, the interrupt vector table is relocated to address location `FFE000h` within on-chip SRAM.

```
;*****************************************************************************
The start-up code presented below defines an interrupt vector table for
the eZ80F91 MCU. It must be aligned to the 512-byte boundary of RAM.
;*****************************************************************************
.assume ADL =1                  ;This is an assembler directive
  NUM_VECTORS EQU 64            ;Initialize all the interrupt vector
                                ;location
.def _vector table              ;
define _vectab, space=RAM, align= 512
.sect "_vectab"
ORG %FFE000                     ;Base address of the Interrupt vector.
                                ;The interrupt vector table is to be
                                ;relocated to ADDRESS 204800. This
                                ;address must be a multiple of 512.
                                ;The ORG directive is used to locate
                                ;the base address
_vector_table:
ds NUM_VECTORS*4                ;Each vector is a 4-byte address
                                ;pointing to the _vectptr segment.
                                ;The number of interrupts is 64 and
                                ;hence 256 bytes of memory is defined
```

```
                                ;Each vector takes up four bytes of
                                ;memory location.
;*****************************************************************
; The following start-up code sets the eZ80 CPU in interrupt mode 2 and
; loads the base address (FFE0 in this example) to the 16-bit I Register.
; Relocates the vector table.
;*****************************************************************
im 2                             ; Interrupt mode 2
ld hl, _vector_table >> 8 & 0ffffh  ;
ld i,hl
;*****************************************************************
;*****************************************************************
```

In the following sample code, the interrupt vector table is relocated to address location 003E00h on the on-chip Flash.

```
;*****************************************************************
The start-up code presented below defines an interrupt vector table for
the eZ80F91 MCU. It must be aligned to the 512-byte boundary of ROM.
;*****************************************************************
. assume ADL =1                 ;This is an assembler directive
  NUM_VECTORS EQU 64            ;Initialize all interrupt vector
                                ; location
. def _vector table             ;
define _vectab, space=ROM, align= 512
.sect "_vectab"
ORG %003E00                     ;Base address of the Interrupt vector.
                                ;The interrupt vector table is to be
                                ;relocated to ADDRESS 003E00h This
                                ;address must be a multiple of 512.
                                ;The ORG directive is used to locate
                                ;the base address

_vector_table:                 ;Base address + the timer 0 interrupt
ORG %003E00+%54                ;vector address
dl_ISR_timer0                  ;The address of the timer0_isr is
XREF _ISR_timer0               ;mapped at the relocated vector table

;*****************************************************************
; The set_vector function cannot be used when mapping the interrupt
; service routines to the corresponding vector address.
;*****************************************************************
; The following start-up code sets the eZ80 CPU in interrupt mode 2 and
; loads the base address (003E in this example) to the 16-bit I Register.
; Relocates the vector table.
;*****************************************************************
im 2                             ; Interrupt mode 2
ld hl, _vector_table >> 8 & 0ffffh ;
ld i,hl
;*****************************************************************
;*****************************************************************
```

Mapping the ISR Location in the Interrupt Vector Table

The maximum addressable capacity of eZ80 devices is 16 MB.

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.*

All the interrupt vector addresses are located anywhere in the 16 MB address space, barring the locations where the nonmaskable interrupt (NMI) is located (`066h`) and the locations between RST0 to RST8 and RST38, which are all absolute addresses. The program must store the ISR's starting address in a 4-byte interrupt vector location.

For example, consider a TIMER0 interrupt service routine that starts at 3-byte address location `123456h`. The default TIMER0 interrupt vector location for the eZ80F91 MCU is at `054h` and the TIMER0 interrupt service routine's address is therefore mapped as follows:

```
{I Register [15:1], 054h}   ------------>   56h
{I Register [15:1], 055h}   ------------>   34h
{I Register [15:1], 056h}   ------------>   12h
{I Register [15:1], 057h}   ------------>   NOT USED
```

Note: *The address location* `{I Register [15:1], 057h}` *is not used. The least-significant byte is stored at the lower address per the Little Endian format.*

The full 24-bit interrupt vector is located at starting address `{I [15:1], IVECT [8:0]}` the lower bit of the I Register (bit 0) is overwritten with the most-significant bit (msb) of the IVECT Register. Setting bit 0 of the I Register has no effect on the interrupt vector locations.

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` is used.

For example, to write an interrupt service routine for TIMER0, use either of the two code segments described 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 code illustrates how the _set_vector() routine is called.

```
# define VECTOR_TIMER0 0x54    // Vector offset value as mentioned in
                               // interrupt vector table for F91
# 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);

main( )
{
  _di(  );                     // disable interrupts
   Init_TIMER0;                // Initialize timer0
  _ei( );1                     // enable interrupts

}

Init_TIMER0 ( )
{
  _set_vector (VECTOR_TIMER0, ISR_Timer0);
  Initialize timer0
  ...;
  ...;
}
```

The Init_TIMER0() function calls the _set_vector() function. The TIMER0 interrupt vector address is located at 054h. This address is passed to the _set_vector() function along with the address of the ISR, ISR_TIMER0. The _set_vector() function is written in assembly and is defined in the startup.asm file. The code snippet provided below is for the _set_vector() function.

The following start up code must be added in the existing startup.asm file.

```
;*****************************************************************
; The address of the interrupt handler (interrupt service routine) is
; copied into corresponding 3-byte vector address location.
;
; void _set_vector (unsigned short vector, void (*hndlr)(void));
```

```
;*******************************************************************
. def    _set_vector
_set_vector:
  ld      iy,0
  add     iy, sp
  ld  hl,(iy+3)
  ld bc,_vector_table
  add     hl,bc
  ld      bc, (iy+6)
  ld      (hl), bc
  ret
;*******************************************************************
;*******************************************************************
```

Warning: DO NOT USE IN LIFE SUPPORT

LIFE SUPPORT POLICY

ZiLOG'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS PRIOR WRITTEN APPROVAL OF THE PRESIDENT AND GENERAL COUNSEL OF ZiLOG CORPORATION.

As used herein

Life support devices or systems are devices which (a) are intended for surgical implant into the body, or (b) support or sustain life and whose failure to perform when properly used in accordance with instructions for use provided in the labeling can be reasonably expected to result in a significant injury to the user. A critical component is any component in a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system or to affect its safety or effectiveness.

Document Disclaimer