

F²MC-16LX FAMILY
16-BIT MICROCONTROLLER
MB90XXX

**RELOCATED INTERRUPT
VECTOR TABLE**

APPLICATION NOTE

Revision History

Date	Issue
2002-10-21	M. Willam; 1 st version

This document contains 10 pages.

Warranty and Disclaimer

To the maximum extent permitted by applicable law, Fujitsu Microelectronics Europe GmbH restricts its warranties and its liability for **all products delivered free of charge** (eg. software include or header files, application examples, target boards, evaluation boards, engineering samples of IC's etc.), its performance and any consequential damages, on the use of the Product in accordance with (i) the terms of the License Agreement and the Sale and Purchase Agreement under which agreements the Product has been delivered, (ii) the technical descriptions and (iii) all accompanying written materials. In addition, to the maximum extent permitted by applicable law, Fujitsu Microelectronics Europe GmbH disclaims all warranties and liabilities for the performance of the Product and any consequential damages in cases of unauthorised decompiling and/or reverse engineering and/or disassembling. **Note, all these products are intended and must only be used in an evaluation laboratory environment.**

1. Fujitsu Microelectronics Europe GmbH warrants that the Product will perform substantially in accordance with the accompanying written materials for a period of 90 days from the date of receipt by the customer. Concerning the hardware components of the Product, Fujitsu Microelectronics Europe GmbH warrants that the Product will be free from defects in material and workmanship under use and service as specified in the accompanying written materials for a duration of 1 year from the date of receipt by the customer.
2. Should a Product turn out to be defect, Fujitsu Microelectronics Europe GmbH's entire liability and the customer's exclusive remedy shall be, at Fujitsu Microelectronics Europe GmbH's sole discretion, either return of the purchase price and the license fee, or replacement of the Product or parts thereof, if the Product is returned to Fujitsu Microelectronics Europe GmbH in original packing and without further defects resulting from the customer's use or the transport. However, this warranty is excluded if the defect has resulted from an accident not attributable to Fujitsu Microelectronics Europe GmbH, or abuse or misapplication attributable to the customer or any other third party not relating to Fujitsu Microelectronics Europe GmbH.
3. To the maximum extent permitted by applicable law Fujitsu Microelectronics Europe GmbH disclaims all other warranties, whether expressed or implied, in particular, but not limited to, warranties of merchantability and fitness for a particular purpose for which the Product is not designated.
4. To the maximum extent permitted by applicable law, Fujitsu Microelectronics Europe GmbH's and its suppliers' liability is restricted to intention and gross negligence.

NO LIABILITY FOR CONSEQUENTIAL DAMAGES

To the maximum extent permitted by applicable law, in no event shall Fujitsu Microelectronics Europe GmbH and its suppliers be liable for any damages whatsoever (including but without limitation, consequential and/or indirect damages for personal injury, assets of substantial value, loss of profits, interruption of business operation, loss of information, or any other monetary or pecuniary loss) arising from the use of the Product.

Should one of the above stipulations be or become invalid and/or unenforceable, the remaining stipulations shall stay in full effect

Contents

REVISION HISTORY	2
WARRANTY AND DISCLAIMER	3
CONTENTS	4
0 INTRODUCTION	5
1 THE INTERRUPT TABLE	6
1.1 Location	6
1.2 Normal Reference	7
2 SECOND TABLE - WORKAROUND	8
2.1 Relocated Reference.....	8
2.2 Interrupt Flow	9
2.3 Code Flow	10

0 Introduction

This document describes how to use a customized Interrupt Vector Jump Table.

1 The Interrupt Table

DESCRIPTION OF THE INTERRUPT VECTOR TABLE

1.1 Location

The interrupt vector table is located fix in the Bank FF_H. There is no way for the user to move it to another memory location.

Below is a list of the interrupt vector address locations:

Interrupt Vector	Hardware Interrupt	Interrupt Control Reg.	Vector		
			Address L	Address M	Address H
INT0	None	-	FFFFFC _H	FFFFFD _H	FFFFFE _H
.
.
.
INT7	None	-	FFFFE0 _H	FFFFE1 _H	FFFFE2 _H
INT8	Reset	-	FFFFDC _H	FFFFDD _H	FFFFDE _H
INT9	INT9 Instruction	-	FFFFD8 _H	FFFFD9 _H	FFFFDA _H
INT10	Exception	-	FFFFD4 _H	FFFFD5 _H	FFFFD6 _H
INT11	(device specific)	ICR00	FFFFD0 _H	FFFFD2 _H	FFFFD3 _H
INT12	(device specific)		FFFFCC _H	FFFC _D	FFFFCE _H
...
INT254	(device specific)	ICR122	FFFC04 _H	FFFC05 _H	FFFC06 _H
INT255	(device specific)		FFFC00 _H	FFFC01 _H	FFFC02 _H

Note, that a device can have theoretical 255 interrupts, but in the most devices the number of interrupts is less than 45.

1.2 Normal Reference

An example of the usual interrupt vector reference without relocation is shown below. This has to be done in the file "vectors.c".

```

. . .
/* Prototypes */
__interrupt void IRQ_Handler1(void);
__interrupt void IRQ_Handler2(void);
__interrupt void IRQ_Handler3(void);
. . .
/* Vector definition */
#pragma intvect IRQ_Handler1 9
#pragma intvect IRQ_Handler2 10
#pragma intvect IRQ_Handler3 11
. . .

```

usual "vectors.c"

The following illustration shows the corresponding Interrupt handler routines. In this example they can be found in the main program "main.c".

```

. . . /* End of void main(void) */
}

__interrupt void IRQ_Handler1(void)
{
. . . /* Code here */
}

__interrupt void IRQ_Handler2(void)
{
. . . /* Code here */
}

__interrupt void IRQ_Handler3(void)
{
. . . /* Code here */
}

```

"main.c"

2 Second Table - Workaround

USING A SECOND TABLE

2.1 Relocated Reference

It is possible to use a second so-called “Jump-Table”, which is then independent from the hard-wired table.

This table then has to be like in the following example in a separate assembly file:

```
.PROGRAM NEWVECT
.TITLE "ALTERNATIVE IRQ VECTOR TABLE"
.SECTION NEWINTVECT, CONST, LOCATE=H'FF8000 ; depends on MCU type

.IMPORT _IRQ_Handler1
.IMPORT _IRQ_Handler2
.IMPORT _IRQ_Handler3
. . .

.EXPORT _Vector1
.EXPORT _Vector2
.EXPORT _Vector3
. . .
_Vector1: JMPP _IRQ_Handler1
_Vector2: JMPP _IRQ_Handler2
_Vector3: JMPP _IRQ_Handler3
. . .

.END

"newvect.asm"
```

“_IRQ_Handler n ” is the basic interrupt handler function (ISR), which is normally referenced in “vectors.c” without relocation.

Depending on the Code segment in which this table and the interrupt service routine are located, you have to be careful using the “JMP” (16-Bit-Addressing) or the “JMPP” (24-Bit-Addressing) instruction.

Note, that “JMPP” will always work.

The next step is to reference to these Jump-Locations in the “vectors.c” file:

```

. . .
/* Prototypes */
__interrupt void Vector1(void);
__interrupt void Vector2(void);
__interrupt void Vector3(void);
. . .
/* Vector definition */
#pragma intvect Vector1 9
#pragma intvect Vector2 10
#pragma intvect Vector3 11
. . .

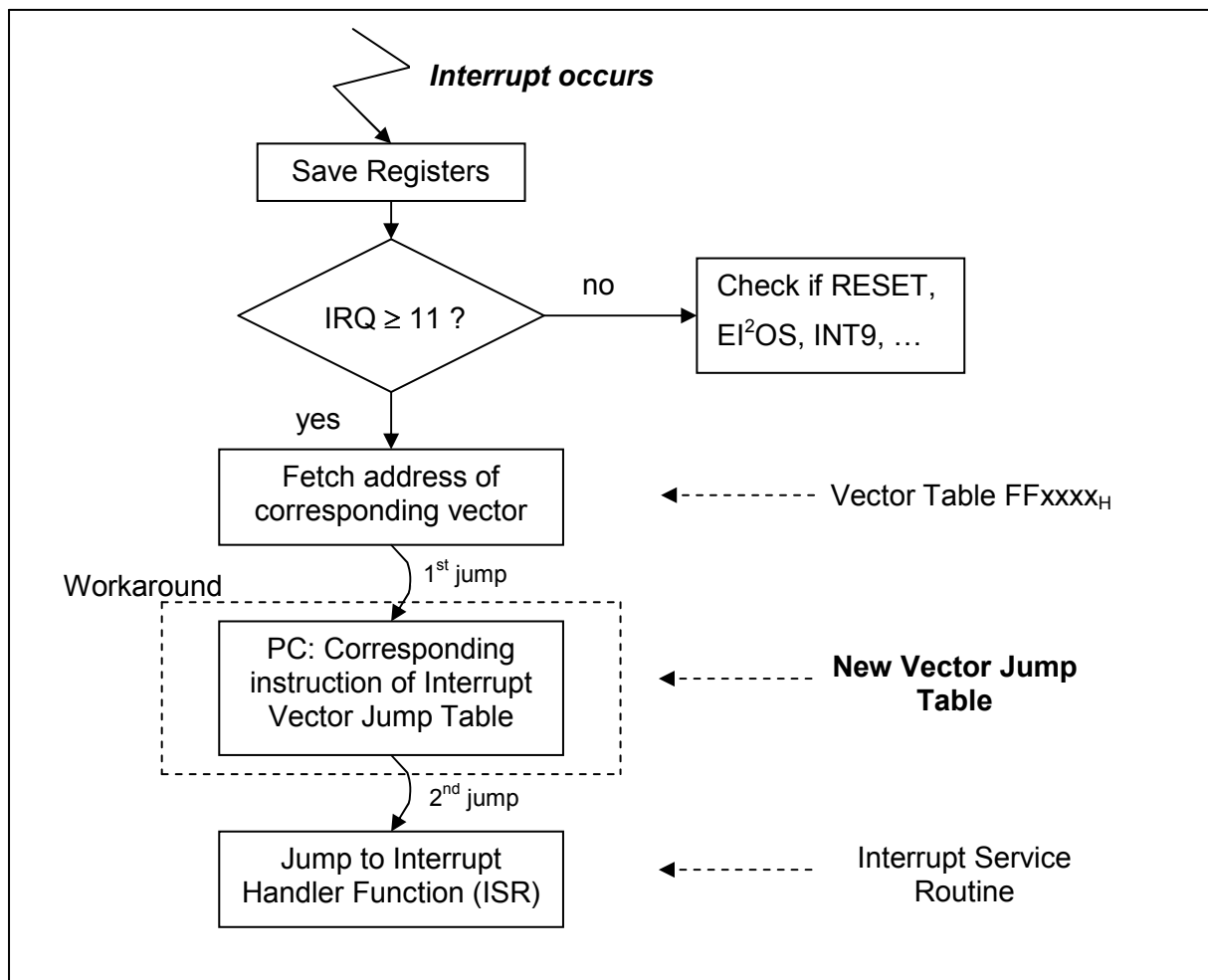
```

"vectors.c"

The file in which the Interrupt routines are coded ("Main.c") remains unaffected.

2.2 Interrupt Flow

Now the instruction flow is like in the following illustration.



2.3 Code Flow

The following graphic illustrates the program code flow for the Jump Table Workaround:

