The following document contains information on Cypress products. The document has the series name, product name, and ordering part numbering with the prefix “MB”. However, Cypress will offer these products to new and existing customers with the series name, product name, and ordering part number with the prefix “CY”.

How to Check the Ordering Part Number

1. Go to www.cypress.com/pcn.
2. Enter the keyword (for example, ordering part number) in the **SEARCH PCNS** field and click **Apply**.
3. Click the corresponding title from the search results.
4. Download the Affected Parts List file, which has details of all changes

For More Information

Please contact your local sales office for additional information about Cypress products and solutions.

About Cypress

Cypress is the leader in advanced embedded system solutions for the world's most innovative automotive, industrial, smart home appliances, consumer electronics and medical products. Cypress' microcontrollers, analog ICs, wireless and USB-based connectivity solutions and reliable, high-performance memories help engineers design differentiated products and get them to market first. Cypress is committed to providing customers with the best support and development resources on the planet enabling them to disrupt markets by creating new product categories in record time. To learn more, go to www.cypress.com.

AN205013

F²MC-8FX Family 8-Bit Microcontroller MB95F430 Series Operational Amplifier

Associated Part Family: CYxxx

This application note describes how to use the amplifier function on the MB95F430 series.

Contents

1 Introduction....1
2 Amplifier Overview....1
2.1 Block Diagram of Operational Amplifier....3
2.2 Pins of Operational Amplifier....4
2.3 OPAMP Control Register....4
3 Operations of Operational Amplifier....7
4 Amplifier Setting Procedure....8
5 Amplifier Driver....8
5.1 Peripheral Usage....8
5.2 Driver Code....8
6 Typical Application....10
6.1 HW Design....10
6.2 Sample Code....10
7 Additional Information....11
A Appendix....12
A.1 Sample Code....12

1 Introduction

In this document, we will introduce how to use the amplifier function on the MB95F430 series.

Section 2 gives an overview on operational amplifier.

Section 3 introduces the operations of operational amplifier.

Section 4 introduces Operational Amplifier setting procedure.

Section 5 introduces amplifier drivers.

Section 6 introduces amplifier application demo.

2 Amplifier Overview

The operational amplifier can be used to sense the ground current, and support front-end analog signal conditioning prior to A/D conversion. It can operate in either closed loop mode or standalone open loop mode.

- Closed Loop Mode

  The operational amplifier can be configured as a non-inverting closed loop operational amplifier.

  It has six software-selectable closed loop gain options for ground current sensing according to different sense voltage values.

No.	Gain
1	10 V/V
2	20 V/V
3	30 V/V
4	40 V/V
5	50 V/V
6	60 V/V

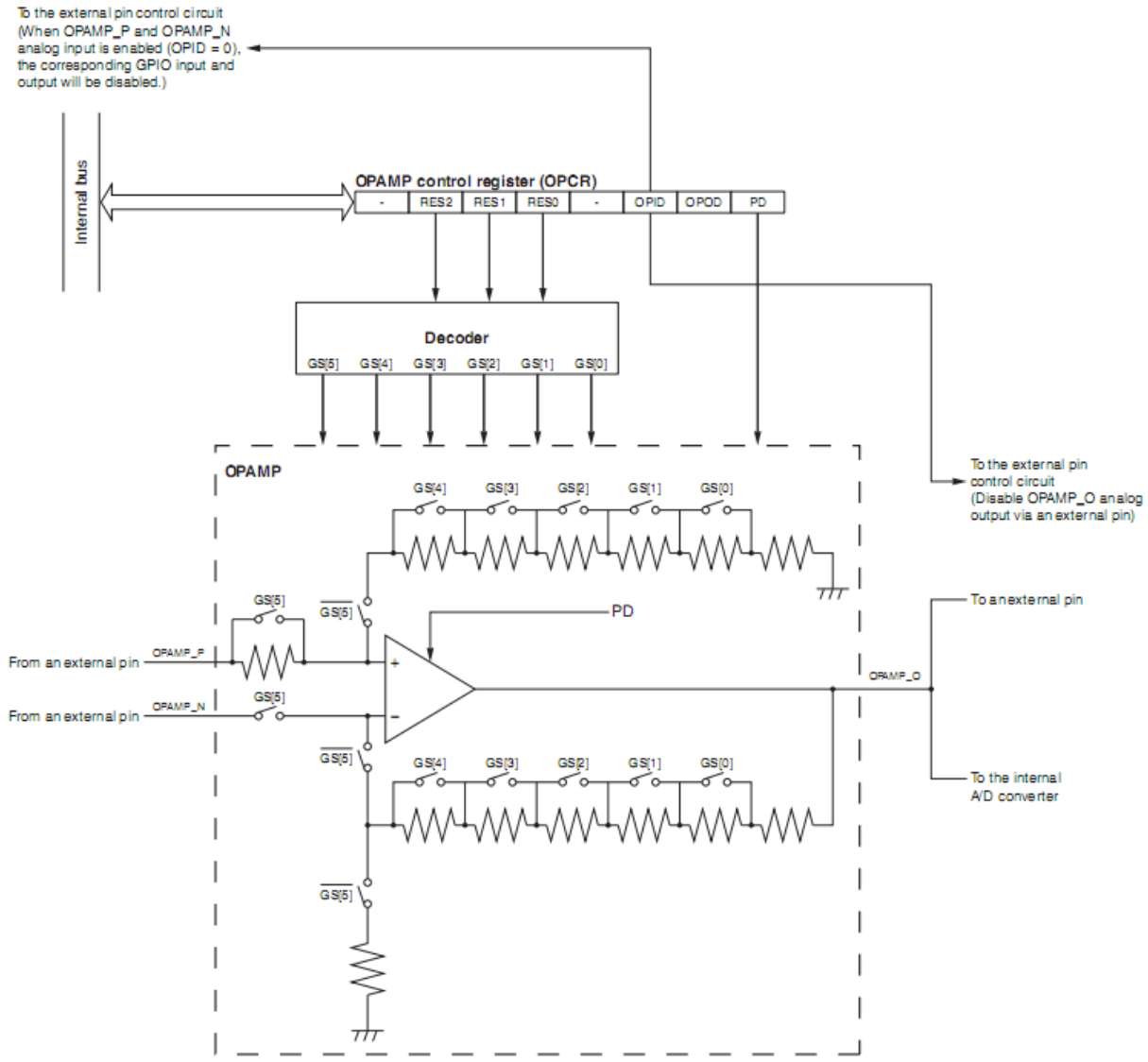
- Standalone Open Loop Mode

  In this mode, the operational amplifier input pins are connected to external signals without any output feedback.

  The standalone open loop mode is designed for users that can choose more flexible gain using external resistors.

2.1 Block Diagram of Operational Amplifier

Figure 2-1: Block Diagram of Operational Amplifier



2.2 Pins of Operational Amplifier

The OPAMP uses the OPAMP_P pin and the OPAMP_N pin as the analog input pins of the operational amplifier, and uses the OPAMP_O pin as the analog output pin of the operational amplifier.

When GS [5] is set to "1B" and GS [4:0] is set to "00000B", the OPAMP will work as a standalone open loop operational amplifier.

When GS [5] is set to "0B", the OPAMP will work as a non-inverting closed loop operational amplifier. It provides six different closed loop gain settings through the software.

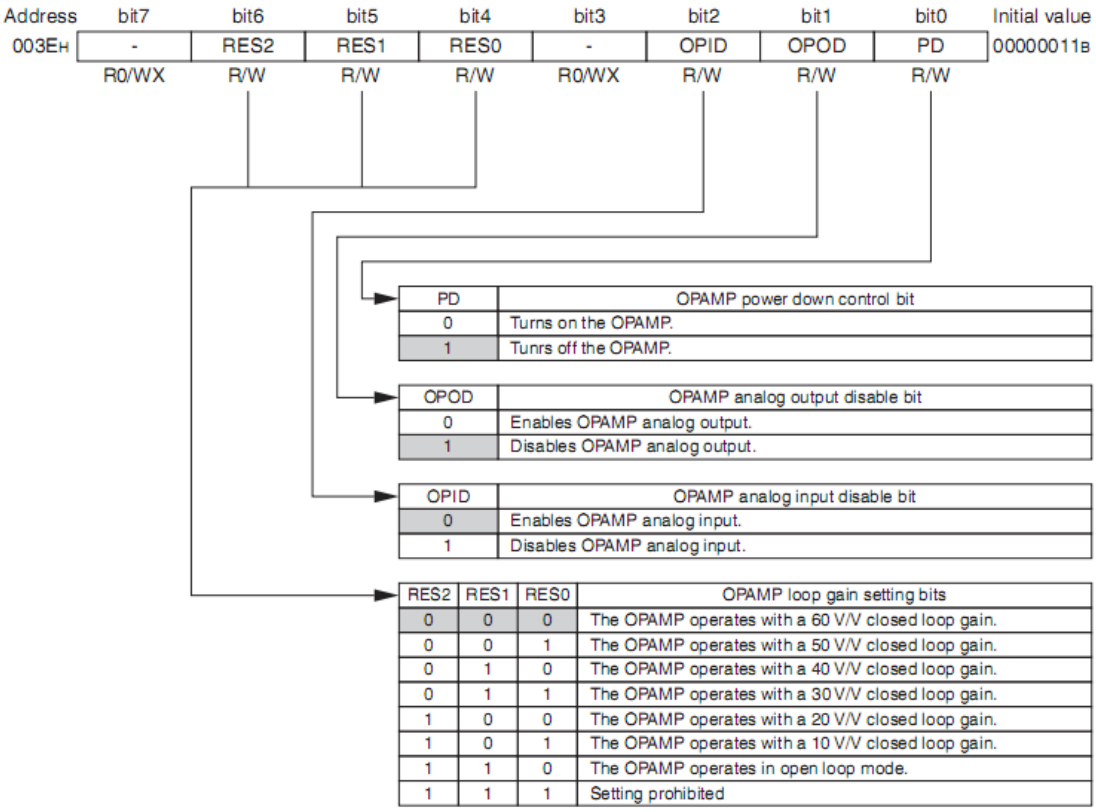
Pin Name	Pin Function	I/O Type	Pull-up Option	Standby Control	Settings Required for Using The Pin	Default Status
P60/OPAMP_P	GPIO/ OPAMP positive analog input	CMOS input/ CMOS output/ Analog input	Unavailable	Available	OPCR:OPID = 0 (Enables analog input)	GPIO input disabled; GPIO output disabled; analog input enabled
P61/OPAMP_N	GPIO/ OPAMP negative analog input	CMOS input/ CMOS output/ Analog input			OPCR:OPID = 0 (Enables analog input)	GPIO input disabled; GPIO output disabled; analog input enabled
P62/OPAMP_O	GPIO/ OPAMP analog output	CMOS input/ CMOS output/ Analog output			OPCR:OPOD = 0 (Enables analog output)	GPIO input enabled; GPIO output disabled; analog output disabled

2.3 OPAMP Control Register

The OPAMP control register (OPCR) is used to turn on and off the OPAMP, to enable and disable OPAMP analog output, and to enable and disable OPAMP analog input.

The register can also be used to set the OPAMP to operate as a standalone open loop operational amplifier, or a non-inverting closed loop operational amplifier with six different closed loop gain settings that can be selected by the software.

Figure 2-2: OPAMP Control Register



Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
003EH	-	RES2	RES1	RES0	-	OPID	OPOD	PD	00000011B
	R0/WX	R/W	R/W	R/W	R0/WX	R/W	R/W	R/W	

PD	OPAMP power down control bit
0	Turns on the OPAMP.
1	Tunrs off the OPAMP.

OPOD	OPAMP analog output disable bit
0	Enables OPAMP analog output.
1	Disables OPAMP analog output.

OPID	OPAMP analog input disable bit
0	Enables OPAMP analog input.
1	Disables OPAMP analog input.

RES2	RES1	RES0	OPAMP loop gain setting bits
0	0	0	The OPAMP operates with a 60 V/V closed loop gain.
0	0	1	The OPAMP operates with a 50 V/V closed loop gain.
0	1	0	The OPAMP operates with a 40 V/V closed loop gain.
0	1	1	The OPAMP operates with a 30 V/V closed loop gain.
1	0	0	The OPAMP operates with a 20 V/V closed loop gain.
1	0	1	The OPAMP operates with a 10 V/V closed loop gain.
1	1	0	The OPAMP operates in open loop mode.
1	1	1	Setting prohibited

R/W : Readable/writable (The read value is the same as the write value.)
R0/WX : The read value is "0". Writing a value to it has no effect on operation.
- : Undefined bit
: Initial value

- Functions of Bits in OPAMP Control Register (OPCR)

Bit name		Function
bit7	Undefined bit	The read value is always "0". Writing a value to it has no effect on operation.
bit6 to bit4	RES2, RES1, RES0: OPAMP loop gain setting bits	These bits select an OPAMP loop gain in closed loop mode from six options and can set the OPAMP to operate in open loop mode.
bit3	Undefined bit	The read value is always "0". Writing a value to it has no effect on operation.
bit2	OPID: OPAMP analog input disable bit	This bit enables and disables OPAMP analog input. **Writing "0":** enables OPAMP analog input. **Writing "1":** disables OPAMP analog input.
bit1	OPOD: OPAMP analog output disable bit	This bit enables and disables OPAMP analog output. **Writing "0":** enables OPAMP analog output. **Writing "1":** disables OPAMP analog output.
bit0	PD: OPAMP power down control bit	This bit turns on and off the OPAMP. **Writing "0":** turns on the OPAMP. **Writing "1":** turns off the OPAMP.

- OPAMP Operating Mode Settings

RES2	RES1	RES0	OPAMP Loop Gain Settings
0	0	0	The OPAMP operates with a 60 V/V closed loop gain.
0	0	1	The OPAMP operates with a 50 V/V closed loop gain.
0	1	0	The OPAMP operates with a 40 V/V closed loop gain.
0	1	1	The OPAMP operates with a 30 V/V closed loop gain.
1	0	0	The OPAMP operates with a 20 V/V closed loop gain.
1	0	1	The OPAMP operates with a 10 V/V closed loop gain.
1	1	0	The OPAMP operates in open loop mode.
1	1	1	Setting prohibited

Note:

- While the OPAMP is operating, modifying the settings of RES2, RES1 and RES0 is allowed, however, do not use the output signal of the OPAMP or execute A/D conversion until OPAMP output becomes stable.
- It is recommended to turn off the operational amplifier before modifying the settings of RES2, RES1 and RES0.

3 Operations of Operational Amplifier

The operational amplifier can be activated by setting the PD bit in the OPCR register using the software. It can operate in closed loop mode or open loop mode, depending on the settings of the RES2, RES1 and RES0 bits in the OPCR register.

- Activating Operational Amplifier by Software

The settings shown in Figure 3-1 are required for activating the operational amplifier using the software.

Figure 3-1: Settings for Activating Operational Amplifier

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
OPCR	-	RES2	RES1	RES0	-	OPID	OPOD	PD
	×	❍	❍	❍	×	0	0	0

❍ : Bit to be used
× : Unused bit
0 : Set to "0"

After the bits in the OPCR register are set as shown above, the operational amplifier will not start operating until it stabilizes.

- Operations of OPAMP in Closed Loop Mode

Before being activated, the operational amplifier can be set to operate in closed loop mode in advance by setting RES[2:0] in the OPCR register to "000B", "001B", "010B", "011B", "100B" or "101B".

Six different closed loop gains are available to be used in closed loop mode. Select a desired closed loop gain by setting RES[2:0] in OPCR to the value corresponding to that gain.

Notes:

- In closed loop mode, connecting the P61/OPAMP_N pin to the ground is recommended.
- While the OPAMP is operating, modifying the settings of RES2, RES1 and RES0 is allowed, however, do not use the output signal of the OPAMP or execute A/D conversion until OPAMP output becomes stable.
- It is recommended to turn off the operational amplifier before modifying the settings of RES2, RES1 and RES0.

- Operations of OPAMP in Open Loop Mode

Before being activated, the operational amplifier can be set to operate in open loop mode in advance by setting RES [2:0] in the OPCR register to "110B".

Note:

- While the OPAMP is operating, switching it from closed loop mode to open loop mode, and vice versa, is allowed, however, do not use the output signal of the OPAMP or execute A/D conversion until OPAMP output becomes stable.

---

4 Amplifier Setting Procedure

Below is an example of procedure for setting the operational amplifier.

Initial settings

1. Set both OPCR: OPID and OPCR: OPOD to "0" to enable both OPAMP analog input and OPAMP analog output.
2. Set the feedback resistor and RES [2:0] in OPCR.
3. Set OPCR: PD to "0" to turn on the operational amplifier.
4. Wait until the operation amplifier stabilizes.
5. Start A/D conversion if necessary.

5 Amplifier Driver

This is OPAMP driver description.

5.1 Peripheral Usage

The MCU pins used as below:

OPAMP_N,used as amplifier negative input;

OPAMP_P,used as amplifier positive input;

OPAMP_O,used as amplifier output;

5.2 Driver Code

5.2.1 General Definition

```
typedef unsigned char  BOOLEAN;
typedef unsigned char  INT8U;            /* Unsigned  8 bit quantity */
typedef signed   char  INT8S;            /* Signed    8 bit quantity */
typedef unsigned int    INT16U;           /* Unsigned 16 bit quantity */
typedef signed   int    INT16S;           /* Signed   16 bit quantity */
typedef unsigned long    INT32U;          /* Unsigned 32 bit quantity */
typedef signed   long  INT32S;           /* Signed   32 bit quantity */

#define BOOL                  BOOLEAN
#define BYTE                  INT8U
#define UBYTE                 INT8U
#define WORD                  INT16U
#define UWORD                 INT16U
#define LONG                  INT32S
#define ULONG       INT32U
#define UCHAR       INT8U
#define UINT                        INT16U
#define DWORD                 INT32U

#define TRUE           1
#define FALSE          0

#define BYTE_LO(w)     ((UBYTE)(w))
#define BYTE_HI(w)     ((UBYTE)(((UWORD)(w)>>8)&0xFF))
```

5.2.2 Amplifier Routine

void AmpOpenLoop()

Return : none.
Parameters : none.
Description : open-loop setting.
Example : AmpOpenLoop();

```
void AmpOpenLoop()
{
       DDR6_P60=0;
       DDR6_P61=0;
       DDR6_P62=1;
       OPCR=0x60;//Amplifier gain is R3/R1
}
```

void AmpCloseLoop()

Return : none.
Parameters : none.
Description : close-loop setting.
Example : AmpCloseLoop();

```
void AmpCloseLoop()
{
       DDR6_P60=0;
       DDR6_P61=0;
       DDR6_P62=1;
       OPCR=0x40;//Amplifier gain is 20V/V
}
```

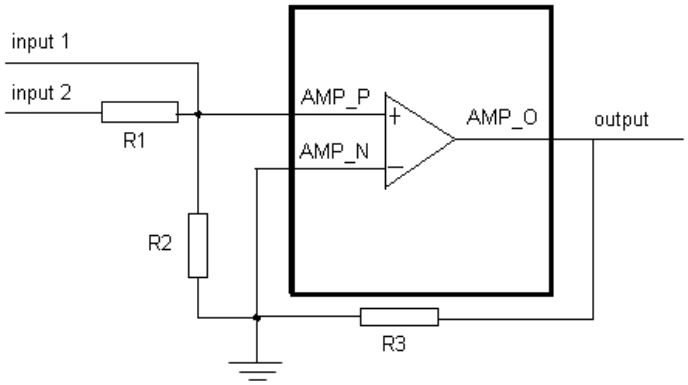
6 Typical Application

This is the typical application introduction.

6.1 HW Design

In this application, we will test the operational amplifier in the MB95F430K. The HW is designed as below. The R1, R2, R3 is used in open-loop amplifier.

Figure 6-1: Hardware Design



6.2 Sample Code

void main(void)

Return : none.

Parameters : none;

Description : system main programm.

Example : main();

```
void main(void)
{
      __DI();
      __set_il(3);
      InitIrqLevels();

      WDTH =0xA5;//Disable WTG
      WDTL =0x96;

      WATR =0xEE;
      SYCC =0xF0;//Main Clock
      SYCC2=0xF4;//Main Clock
      SYSC  =0xBC;//BUZZ(P01)
      SYSC2 =0x02;//PPG(P73),Disable I2C
      while(!STBC_MRDY);
      __EI();

      AmpOpenLoop();
      AmpCloseLoop();
}
```

7 Additional Information

Please contact your local support team for any technical question.

A Appendix

A.1 Sample Code

main.c

```
#include "mb95430.h"

#include "TypeDef.h"


/*-----------------------------------------------------------------------------*/

/* Amplifier Setting

/*-----------------------------------------------------------------------------*/

void AmpOpenLoop()

{

      DDR6_P60=0;

      DDR6_P61=0;

      DDR6_P62=1;

      OPCR=0x60;//Amplifier gain is R3/R1

}


void AmpCloseLoop()

{

      DDR6_P60=0;

      DDR6_P61=0;

      DDR6_P62=1;

      OPCR=0x40;//Amplifier gain is 20V/V

}
```

```
void main(void)
{
	__DI();
	__set_il(3);
	InitIrqLevels();

	WDTH =0xA5;
	WDTL =0x96;

	WATR =0xEE;
	SYCC =0xF0;//Main Clock
	SYCC2=0xF4;//Main Clock
	SYSC  =0xBC;//BUZZ(P01)
	SYSC2 =0x02;//PPG(P73),Disable I2C
	while(!STBC_MRDY);

	__EI();

	AmpOpenLoop();
	AmpCloseLoop();
}
```

VECTORS.C

```
 #include "mb95430.h"


void InitIrqLevels(void)

{

/*  ILRx                    IRQs defined by ILRx */


    ILR0 = 0xFF;       //  IRQ0:  external interrupt ch0 | ch4

                       //  IRQ1:  external interrupt ch1 | ch5

                       //  IRQ2:  external interrupt ch2 | ch6

                       //  IRQ3:  external interrupt ch3 | ch7


    ILR1 = 0xFF;       //  IRQ4:  UART/SIO ch0

                       //  IRQ5:  8/16-bit timer ch0 (lower)

                       //  IRQ6:  8/16-bit timer ch0 (upper)

                       //  IRQ7:  Output Compare ch0


    ILR2 = 0xFF;       //  IRQ8:  Output Compare ch1

                       //  IRQ9:  none

                       //  IRQ10: Voltage Compare ch0

                       //  IRQ11: Voltage Compare ch1


    ILR3 = 0xFF;       //  IRQ12: Voltage Compare ch2

                       //  IRQ13: Voltage Compare ch3

                       //  IRQ14: 16-bit free run timer

                       //  IRQ15: 16-bit PPG0
```

```
    ILR4 = 0xFF;        //  IRQ16: I2C ch0
                        //  IRQ17: none
                        //  IRQ18: 10-bit A/D-converter
                        //  IRQ19: Timebase timer

    ILR5 = 0xFF;        //  IRQ20: Watch timer
                        //  IRQ21: none
                        //  IRQ22: none
                        //  IRQ23: Flash Memory
}

/*-----------------------------------------------------------------------
   Prototypes

   Add your own prototypes here. Each vector definition needs is proto-
   type. Either do it here or include a header file containing them.
-----------------------------------------------------------------------*/
__interrupt void DefaultIRQHandler(void);

/*-----------------------------------------------------------------------
   Vector definiton

   Use following statements to define vectors.
   All resource related vectors are predefined.
```

```
   Remaining software interrupts can be added hereas well.

------------------------------------------------------------------------*/

#pragma intvect DefaultIRQHandler 0   //  IRQ0:  external interrupt ch0 | ch4

#pragma intvect DefaultIRQHandler 1   //  IRQ1:  external interrupt ch1 | ch5

#pragma intvect DefaultIRQHandler 2   //  IRQ2:  external interrupt ch2 | ch6

#pragma intvect DefaultIRQHandler 3   //  IRQ3:  external interrupt ch3 | ch7


#pragma intvect DefaultIRQHandler 4   //  IRQ4:  UART/SIO ch0

#pragma intvect DefaultIRQHandler 5   //  IRQ5:  8/16-bit timer ch0 (lower)

#pragma intvect DefaultIRQHandler 6   //  IRQ6:  8/16-bit timer ch0 (upper)

#pragma intvect DefaultIRQHandler 7   //  IRQ7:  Output Compare ch0


#pragma intvect DefaultIRQHandler 8   //  IRQ8:  Output Compare ch1

#pragma intvect DefaultIRQHandler 9   //  IRQ9:  none

#pragma intvect DefaultIRQHandler 10  //  IRQ10: Voltage Compare ch0

#pragma intvect DefaultIRQHandler 11  //  IRQ11: Voltage Compare ch1


#pragma intvect DefaultIRQHandler 12  //  IRQ12: Voltage Compare ch2

#pragma intvect DefaultIRQHandler 13  //  IRQ13: Voltage Compare ch3

#pragma intvect DefaultIRQHandler 14  //  IRQ14: 16-bit free run timer

#pragma intvect DefaultIRQHandler 15  //  IRQ15: 16-bit PPG0


#pragma intvect DefaultIRQHandler 16  //  IRQ16: I2C ch0

#pragma intvect DefaultIRQHandler 17  //  IRQ17: none

#pragma intvect DefaultIRQHandler 18  //  IRQ18: 10-bit A/D-converter
```

```
#pragma intvect DefaultIRQHandler 19  //  IRQ19: Timebase timer


#pragma intvect DefaultIRQHandler 20  //  IRQ20: Watch timer
#pragma intvect DefaultIRQHandler 21  //  IRQ21: none
#pragma intvect DefaultIRQHandler 22  //  IRQ22: none
#pragma intvect DefaultIRQHandler 23  //  IRQ23: Flash Memory


/*------------------------------------------------------------------------
   DefaultIRQHandler()

   This function is a placeholder for all vector definitions.
   Either use your own placeholder or add necessary code here
   (the real used resource interrupt handlers should be defined in the main.c).
------------------------------------------------------------------------*/
__interrupt void DefaultIRQHandler(void)
{
      __DI();                                   // disable interrupts
      while(1)
            __wait_nop();                 // halt system
}
```

Document History

Document Title: AN205013 - F²MC-8FX Family 8-bit Microcontroller MB95F430 Series Operational Amplifier

Document Number: 002-05013

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	—	HUAL	03/22/2010	Initial release
*A	5267510	HUAL	05/11/2016	Migrated Spansion application note “MCU-AN-500080-E-10” to Cypress format..
*B	5848974	MALI	08/09/2017	Updated logo and copyright

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

ARM® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6

Cypress Developer Community

Forums | WICED IOT Forums | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

All other trademarks or registered trademarks referenced herein are the property of their respective owners.

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

© Cypress Semiconductor Corporation, 2010-2017. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC ("Cypress"). This document, including any software or firmware included or referenced in this document ("Software"), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress's patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage ("Unintended Uses"). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.