

## F<sup>2</sup>MC-8FX Family MB95200 Series 8-Bit Microcontroller I/O API Usage

**Associated Part Family: MB95200 Series**

This document introduces API for I/O ports.

### Contents

|     |  |   |     |  |    |
|-----|--|---|-----|--|----|
| 1   | Introduction.....                                    | 1 | 4.2 | Digital Port Output .....  | 7  |
| 2   | I/O Ports Library Function List .....                | 1 | 4.3 | Pull Up Register Usage.....  | 7  |
| 3   | I/O Ports Function Detail .....                      | 2 | 4.4 | AD Input Allowed .....   | 7  |
| 3.1 | IO_Input Function .....                              | 2 | 4.5 | Especial_IO Usage .....  | 7  |
| 3.2 | IO_Output Function.....                              | 3 | 5   | Code of function .....   | 8  |
| 3.3 | IO_Pull_Up Function.....                             | 3 | 5.1 | IO_Input Code .....  | 8  |
| 3.4 | IO_Direction Function .....                          | 4 | 5.2 | IO_Output Code, IO_Pull_Up Code, IO_Direction Code, IO_AD_Select Code, Especial_IO Code..... | 9  |
| 3.5 | IO_AD_Select Function .....                          | 5 | 6   | Additional Information.....  | 10 |
| 3.6 | Especial_IO Function.....                            | 5 |     | Document History.....  | 11 |
| 4   | Usage Demo.....                                      | 7 |     |  |    |
| 4.1 | Digital Port Input or Peripheral Function Input .... | 7 |     |  |    |

## 1 Introduction

This document introduces API for I/O ports.

We should set up four functions to control I/O port: I/O input mode, I/O output mode, I/O pull up and I/O direction.

## 2 I/O Ports Library Function List

This section introduces the I/O port library all functions.

[Table 1](#) lists the I/O Ports library functions.

Table 1. I/O Ports Functions

| Function name  | Description   |
|--|---|
| void IO_Input(uint IO_Data_Port, uchar IO_Pin)                       | Setting the I/O ports as input and choose the port optionally.  |
| void IO_Output(uint IO_Data_Port, uchar IO_Pin, uchar Out_Value)     | Setting the I/O ports as output and choose the port optionally. |
| void IO_Pull_Up(uchar PU_Port, uchar IO_Pin, uchar IO_PU_Status)     | The P0 and PG ports, enable/disable the pull-up register.       |
| void IO_Direction(uchar PD_Port, uchar IO_Pin, uchar Direction)      | Setting the direction for each channel.                         |
| void IO_AD_Select(uint AD_IO,uchar IO_Pin,uchar AD_Status)           | Select Port0 as GPIO or AD function.                            |
| void Especial_IO(uint Spec_IO, uchar Function_IOPin, uchar Function) | Select Port F, Port G as GPIO or other function.                |

### 3 I/O Ports Function Detail

This section introduces the detail of I/O Ports function.

#### 3.1 IO\_Input Function

IO\_Input(Input parameter1, Input parameter2);

Table 2 describes IO\_Input function.

Table 2. IO\_Input Function

| Function name        | IO_Input                                       |
|----------------------|--|
| Function prototype   | void IO_Input(uint IO_Data_Port, uchar IO_Pin) |
| Behavior description | Set IO input pin                               |
| Input parameter1     | IO_Data_Port                                   |
| Input parameter2     | IO_Pin, choose operation pin                   |
| Return value         | None   |

Select the I/O port as input pins. In this parameter user can choose more than one of the I/O pins as I/O input pins, and get the external circuit value.

Table 3 describes the IO\_Data\_Port parameter values.

Table 3. IO\_Data\_Port Definition

| IO_Data_Port | Description |
|--------------|-------------|
| Data_Port0   | Set port 0  |
| Data_Port1   | Set port 1  |
| Data_Port6   | Set port 6  |
| Data_Port F  | Set port F  |
| Data_Port G  | Set port G  |

Table 4 describes the IO\_Pin parameter values.

Table 4. IO\_Pin Definition

| I/O_Pin | Description              |
|---------|--------------------------|
| P0      | First Pin in each Port   |
| P1      | Second pin in each port  |
| P2      | Third pin in each port   |
| P3      | Forth pin in each port   |
| P4      | Fifth pin in each port   |
| P5      | Sixth pin in each port   |
| P6      | Seventh pin in each port |
| P7      | Eighth pin in each port  |
| ALL     | All Pins in each Port    |

### 3.2 IO\_Output Function

IO\_Output(Input parameter1, Input parameter2, Input parameter3);

Table 5 describes IO\_Output function.

Table 5. IO\_Output Function

| Function name        | IO_Output  |
|----------------------|--|
| Function prototype   | void IO_Output(uint IO_Data_Port, uchar IO_Pin,uchar Out_Value)  |
| Behavior description | Control the IO output pin and values   |
| Input parameter1     | IO_Data_Port, Refer to table 3-2 for more details on the allowed values for this parameter.                    |
| Input parameter2     | IO_Pin, Choose operation pin.<br>Refer to table 3-3 for more details on the allowed values for this parameter. |
| Input parameter3     | Out_Value, Setting the output value for single pin   |
| Return value         | None   |

Select the I/O port as output pins. In this parameter user can choose more than one of the I/O pins as output pins and pass the control signal to the external circuit.

Table 6 describes the Out\_Value parameter values.

Table 6. Out\_Value Definition

| Out_Value | Description |
|-----------|-------------|
| Out_L     | out value 0 |
| Out_H     | out value 1 |

### 3.3 IO\_Pull\_Up Function

IO\_Pull\_up(Input parameter1, Input parameter2, Input parameter3);

Table 7 describes IO\_Output function.

Table 7. IO\_Output Function

| Function name        | IO_Pull_Up  |
|----------------------|---|
| Function prototype   | void IO_Pull_Up(uchar PU_Port,uchar IO_Pin,uchar IO_PU_Status)  |
| Behavior description | The P0 and PG ports, enable/disable the pull-up register.   |
| Input parameter1     | PU_Port, Setting the Pull-Up port   |
| Input parameter2     | IO_Pin, Choose operation pin<br>Refer to table 3-3 for more details on the allowed values for this parameter. |
| Input parameter3     | IO_PU_Status, only have two values: IO_PU_Enable and IO_PU_Disable, Setting the pull-up for single pin        |
| Return value         | None  |

Enable or disable the pull up register. This function can only used in port 0 and port G.

Table 8 describes the PU\_Port parameter values.

Table 8. PU\_Port Definition

| PU_Port       | Description |
|---------------|-------------|
| Pull_Up_Port0 | Port0       |
| Pull_Up_PortG | PortG       |

### 3.4 IO\_Direction Function

IO\_Direction(Input parameter1, Input parameter2, Input parameter3);

Table 9 describes IO\_Direction function.

Table 9. IO\_Direction Function

| Function name        | IO_Direction  |
|----------------------|---|
| Function prototype   | void IO_Direction(uchar PD_Port, uchar IO_Pin, uchar Direction)   |
| Behavior description | decide the pins as input pins or output pins  |
| Input parameter1     | PD_Port   |
| Input parameter2     | IO_Pin, Choose operation pin<br>Refer to table 3-3 for more details on the allowed values for this parameter. |
| Input parameter3     | Direction, only have two values: Direction_In and Direction_Out   |
| Return value         | None  |

Table 10 describes the PD\_Port parameter values.

Table 10. PD\_Port Definition

| PD_Port         | Description |
|-----------------|-------------|
| Direction_Port0 | Port0       |
| Direction_Port1 | Port1       |
| Direction_Port6 | Port6       |
| Direction_PortF | PortF       |
| Direction_PortG | PortG       |

### 3.5 IO\_AD\_Select Function

IO\_AD\_Select(Input parameter1, Input parameter2, Input parameter3);

Table 11 describes IO\_AD\_Select function.

Table 11. IO\_AD\_Select Function

| Function name        | IO_AD_Select   |
|----------------------|--|
| Function prototype   | void IO_AD_Select(uint AD_IO,uchar IO_Pin,uchar AD_Status)   |
| Behavior description | Set IO pin used as AD function pin   |
| Input parameter1     | AD_IO, only have one value: AD_IO_Port0  |
| Input parameter2     | IO_Pin, Choose operation pin.<br>Refer to table 3-3 for more details on the allowed values for this parameter. |
| Input parameter3     | AD_Status, only have two values: AD_Enable and AD_Disable  |
| Return value         | None   |

### 3.6 Especial\_IO Function

Especial\_IO(Input parameter1, Input parameter2, Input parameter3);

Table 12 describes Especial\_IO function.

Table 12. Especial\_IO Function

| Function name        | Especial_IO   |
|----------------------|---|
| Function prototype   | void Especial_IO(uint Espec_IO, uchar Function_IOPin, uchar Function)   |
| Behavior description | Set IO used as especial function IO   |
| Input parameter1     | Espec_IO, only have one values: Function_IO   |
| Input parameter2     | Function_IOPin  |
| Input parameter3     | Function, values be relate to Function_IOPin parameter.<br>For example: when Function_PF0, the Function only have two input values: IO_PF0 and MainC_PF0. |
| Return value         | None  |

Table 13 describes the Function\_IOPin parameter values.

Table 13. Function\_IOPin Definition

| Function_IOPin | Description |
|----------------|-------------|
| Function_PF0   | PF0         |
| Function_PF1   | PF1         |
| Function_PF2   | PF2         |
| Function_PG1   | PG1         |
| Function_PG2   | PG2         |

Table 14 describes the Function parameter values.

Table 14. Function Definition

| Function  | Description       |
|-----------|-------------------|
| IO_PF0    | I/O function      |
| MainC_PF0 | Especial Function |
| IO_PF1    | I/O function      |
| MainC_PF1 | Especial Function |
| IO_PF2    | I/O function      |
| RSTX_PF2  | Especial Function |
| IO_PG1    | I/O function      |
| SubC_PG1  | Especial Function |
| IO_PG2    | I/O function      |
| SubC_PG2  | Especial Function |

## 4 Usage Demo

This section is some demo which introduces how to use these I/O Ports.

### 4.1 Digital Port Input or Peripheral Function Input

This can be done by calling the following functions successively using appropriate parameters.

```
IO_Direction(Direction_Port0,P0, Direction_In); //port 0 as input pin
```

```
IO_Input(Data_Port0,P0); //port 0,bit0
```

### 4.2 Digital Port Output

```
IO_Direction(Direction_Port0,P0,Direction_Out); //port 0 as output pin
```

```
IO_Output(Data_Port0,P0,Out_H); //port 0,bit 0, output value 1
```

### 4.3 Pull Up Register Usage

This example shows that how to use the pull up register. note that there are only port 0 and port G have pull up register, and also only if the port as input pin could use pull up register.

```
IO_Direction(Direction_Port0,P0,Direction_In); //port 0 as input pin
```

```
IO_Pull_up(Pull_Up_Port0, P0, Pull_Up_Enable); //P00 use the pull up register
```

```
IO_Input(Data_Port0,P0); //port 0,bit0
```

### 4.4 AD Input Allowed

```
IO_AD_Select(AD_IO_Port0,P0, AD_Enable); //AD Input enable.
```

### 4.5 Especial\_IO Usage

```
Especial_IO(Function_IO, Function_PF0, IO_PF0); //PF0 as IO port
```

## 5 Code of function

This section list all function's code.

### 5.1 IO\_Input Code

```

IO_InPin:
    PUSHW  IX
    MOVW   A,SP
    MOVW   IX,A
    MOV    A,R0
    PUSHW  A
    MOV    A,R1
    PUSHW  A
    MOV    A,R2
    PUSHW  A

    MOV    A,@IX+07H
    CMP    A,#0x08
    BZ     In_ALL
    MOV    R0,A

    MOVW   A,@IX+04H
    MOV    A,@A

BIT_SHIFT:
    CMP    R0,#0
    BZ     KEEP_BIT0
    CLRC
    RORC  A
    DEC   R0
    JMP   BIT_SHIFT

KEEP_BIT0:
    AND   A,#01
    MOVW  EP,A
    JMP   RESUME

In_ALL:
    MOVW  A,@IX+04H
    MOV   A,@A
    MOVW  EP,A
    JMP   RESUME

```



## 5.2 IO\_Output Code, IO\_Pull\_Up Code, IO\_Direction Code, IO\_AD\_Select Code, Especial\_IO Code

```

_IO_Output:
_IO_Pull_Up:
_IO_Direction:
_IO_AD_Select:
_Especial_IO:
    PUSHW    IX
    MOVW     A,SP
    MOVW     IX,A
    MOV      A,R0
    PUSHW    A
    MOV      A,R1
    PUSHW    A
    MOV      A,R2
    PUSHW    A
;-----
IO_Pin:
    MOV      A,@IX+07H
    CMP      A,#0x08
    BZ       ALL_Init
    MOV      R0,A
    MOV      A,@IX+09H
    MOV      R1,A
    MOV      R2,#0xFE
BIT_shift:                                ;Shift bit operation
    CMP      R0,#0
    BZ       CHANGE_BIT
    MOV      A,R1
    CLRC
    ROLC    A
    MOV      R1,A
    MOV      A,R2
    SETC
    ROLC    A
    MOV      R2,A
    DEC      R0
    JMP      BIT_shift
CHANGE_BIT:                                ;Change corresponding bit
    MOVW     A,@IX+04H
    MOVW     EP,A
    MOV      A,@A
    AND      A,R2
    OR       A,R1
    MOV      @EP,A
    JMP      RESUME
ALL_Init:                                    ;If select all pins, operation here
    MOVW     A,@IX+04H
    MOVW     EP,A
    MOV      A,@A
    MOV      A,@IX+09H
    MOV      @EP,A
RESUME:
    POPW    A
    MOV     R2,A
    POPW    A
    MOV     R1,A
    POPW    A
    MOV     R0,A
    POPW    IX
    RET

```

## 6 Additional Information

For more information about how to use MB95200H EV-board, BGM Adaptor and SOFTUNE, please refer to SKT MB2146-410-01-E User Manual, or visit websites:

<http://www.cypress.com/documentation/software-and-drivers/f2mc-8fx-mb95200h210h-series-starter-kit-mb2146-410a-01-e-setup>

## Document History

Document Title: AN205443 – F<sup>2</sup>MC-8FX Family MB95200 Series 8-Bit Microcontroller I/O API Usage

Document Number: 002-05443

| Revision | ECN     | Orig. of Change | Submission Date | Description of Change   |
|----------|---------|-----------------|-----------------|---|
| **       | -       | HUAL            | 03/03/2009      | Initial release   |
|          |         |                 | 03/05/2009      | Add code to function  |
| *A       | 5278941 | HUAL            | 05/20/2016      | Migrated Spansion Application Note MCU-AN- 500024-E-11 to Cypress format. |

## 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 | <a href="http://cypress.com/arm">cypress.com/arm</a>               |
| Automotive                    | <a href="http://cypress.com/automotive">cypress.com/automotive</a> |
| Clocks & Buffers              | <a href="http://cypress.com/clocks">cypress.com/clocks</a>         |
| Interface                     | <a href="http://cypress.com/interface">cypress.com/interface</a>   |
| Lighting & Power Control      | <a href="http://cypress.com/powerpsoc">cypress.com/powerpsoc</a>   |
| Memory                        | <a href="http://cypress.com/memory">cypress.com/memory</a>         |
| PSoC                          | <a href="http://cypress.com/psoc">cypress.com/psoc</a>             |
| Touch Sensing                 | <a href="http://cypress.com/touch">cypress.com/touch</a>           |
| USB Controllers               | <a href="http://cypress.com/usb">cypress.com/usb</a>               |
| Wireless/RF                   | <a href="http://cypress.com/wireless">cypress.com/wireless</a>     |

### PSoC® Solutions

[PSoC 1](#) | [PSoC 3](#) | [PSoC 4](#) | [PSoC 5LP](#)

### Cypress Developer Community

[Forums](#) | [Projects](#) | [Videos](#) | [Blogs](#) | [Training](#) | [Components](#)

### Technical Support

[cypress.com/support](http://cypress.com/support)

PSoC is a registered trademark and PSoC Creator is a trademark of Cypress Semiconductor Corporation. All other trademarks or registered trademarks referenced herein are the property of their respective owners.



Cypress Semiconductor      Phone : 408-943-2600  
198 Champion Court      Fax : 408-943-4730  
San Jose, CA 95134-1709      Website : [www.cypress.com](http://www.cypress.com)

© Cypress Semiconductor Corporation, 2009-2016. 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, 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](http://cypress.com). Other names and brands may be claimed as property of their respective owners.