AN205493

F²MC-16FX, All Series, I/O-Port

The I/O-port functionality is the simplest peripheral function of the 16FX microcontroller. Nevertheless, some details should be considered while programming. This application note reflects the functionality and describes the different modes.

Contents

1 Introduction 1
1.1 Key features 1
2 The I/O-port 2
2.1 Block Diagram 2
2.2 Registers 3
2.3 Input Mode 5
2.4 Pull-up control register 6
2.5 Output-mode 7
3 Port Input / Unused Pins 8
3.1 Port Input / Unused Pins 8
4 Technical information 9
4.1 Hysteresis Inputs 9
5 Using the same I/O port simultaneously as in- and output 10
6 Tips and Tricks 12
6.1 Initial Value 12
6.2 Bit Instructions 12
6.3 RMW Instructions 12
7 Additional Information 13
Document History 14

1 Introduction

The I/O-port functionality is the simplest peripheral function of the 16FX microcontroller.

Nevertheless, some details should be considered while programming.

This application note reflects the functionality and describes the different modes.

Please note that in this document each port number is given with the 2-digit placeholder "*xy*". "*z*" always means the bit position 0 – 7.

Example: "PDR02_P3" means Port 02 Bit 3.

1.1 Key features

- Port direction settable
- Usage of I/O Port or Resource Pin, both states readable
- Input can be disabled, if corresponding pin is unused
- Internal pull-up resistor can be enabled
- Input level can be set to CMOS (0307), CMOS (0208), Automotive Hysteresis, and TTL
- Output drive strength can be set

2 The I/O-port

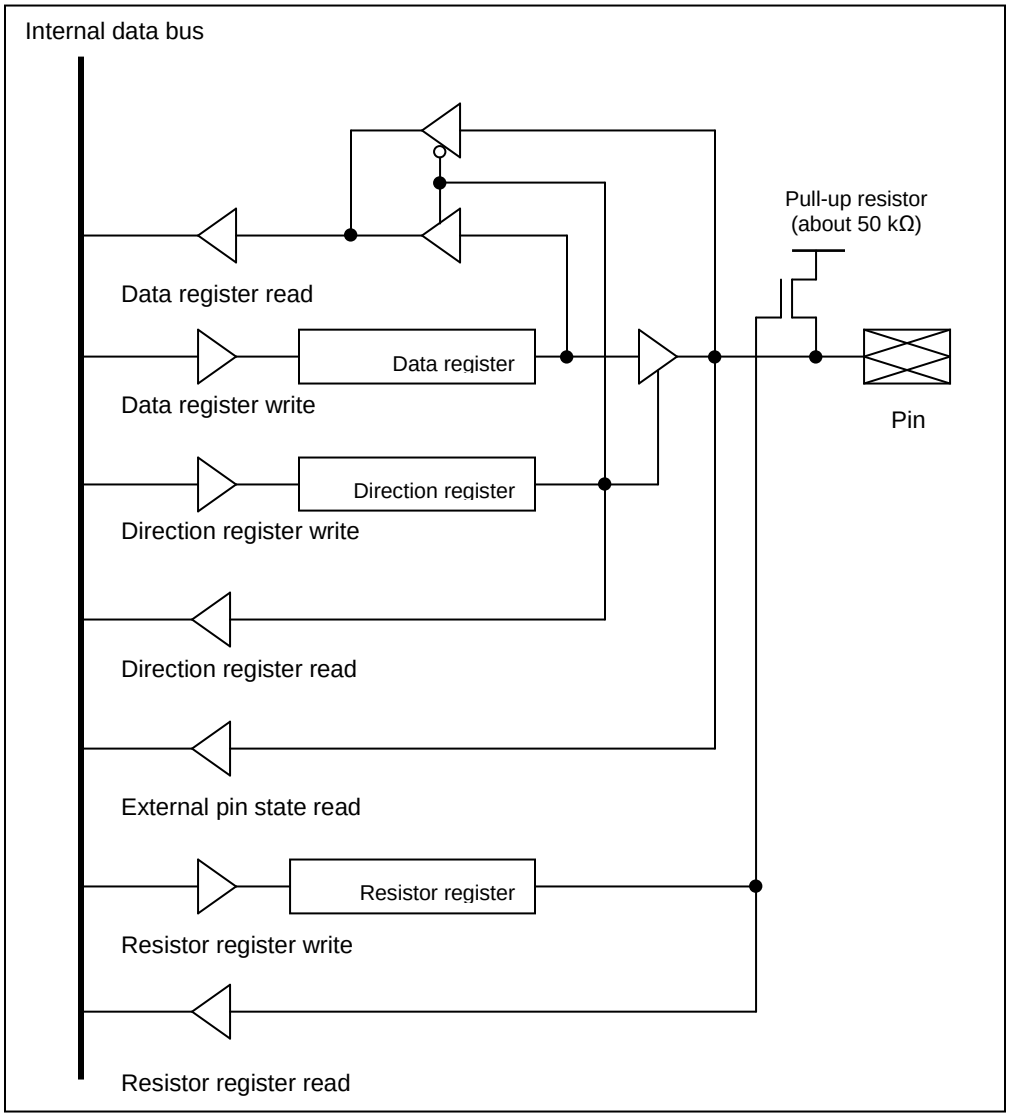
Basic functionality of the I/O ports

2.1 Block Diagram

Figure 1 shows the internal block diagram of an external I/O-pin.

Up to 8 I/O-pins may be encapsulated into one port and one register set. The registers are described below.

Figure 1. I/O-port block diagram



2.2 Registers

2.2.1 Port Data Register (PDR)

This register contains the data bits, if the corresponding port acts as a simple digital output. The contents are output, if the Port Direction Register is set to output mode.

Please note that a resource output control bit overwrites the `PDR` bit value.

Table 1. PDR

PDRxy_Pz	Pin Function
0	Pin State Low (V_{SS})
1	Pin State High (V_{DD})

The read value of PDR register depends on the following:

- The corresponding bit in DDR register
- The Current status of the resource that is connected to the same pin
- The type of instruction used (Read/Read-modify-write)

The following table describes the above discussed behavior:

Table 2. Reading PDR

DDR Value	Resource Output	Pin Value	Read Value of 'Read Instructions'	Read Value of 'Read-modify-write Instructions'
0 (Input)	Enabled	Value of Resource Input	Input Pin State	Value of PDR
	Disabled	Port Input	Input Pin State	Value of PDR
1 (Output)	Enabled	Value of Resource Output	Value of PDR	Value of PDR
	Disabled	Value of PDR	Value of PDR	Value of PDR

2.2.2 Data Direction Register (DDR)

This register contains the bit information of the corresponding pins if they should act as input or output.

Table 3. DDR

DDRxy_Dz	Resource Output	Pin Function
0	Disabled	Port Input[1]
0	Enabled	Resource Output
1	Disabled	Port Output
1	Enabled	Resource Output logically OR with Port Output

[1] Please note that an input ***always must*** be enabled by the `PIER`, regardless whether digital input or resource input is used.

2.2.3 Port Input Enable Register (PIER)

This registers enables the input functionality of a MCU pin. If a corresponding bit is set to “0” (default value) the pin can be left unconnected regardless of the settings of all port and resource registers.

Table 4. PIER

PIERxy_IEz	Function
0	Digital Input disabled
1	Digital Input enabled

2.2.4 External Pin State Register (EPSR)

This register always contains the state of the corresponding external pin.

Please note that the corresponding bit in the Port Input Enable Register must also be set to “1”, otherwise the read-back value is undefined.

2.2.5 (Extended) Port Input Level Register (PILR, EPILR)

With these registers one of the following input levels can be chosen. The input level applies to both pin input and resource input (e.g. TINn input of Reload Timer, INn input of Input Capture).

Table 5. PILR, EPILR

PILRxy_ILz	EPILRxy_EILz	Input Level	V_{IL}	V_{IH}
0	0	CMOS (0307)	0.3 V_{DD}	0.7 V_{DD}
1	0	Automotive Hysteresis	0.5 V_{DD}	0.8 V_{DD}
0	1	TTL	0.8 V	2.1 V
1	1	CMOS (0208)	0.2 V_{DD}	0.8 V_{DD}

2.2.6 Port Output Drive Register (PODR) and Port High Drive Register (PHDR)

With these registers the strength of the output current of a pin can be adjusted:

Table 6. PODR, PHDR

PODRxy_ODz	PHDRxy_HDz	Output Current
0	0	Normal Current
1	0	Reduced Current
X	1	High Current

High current outputs are only available with ports those provide the stepper motor functionality i.e. Port 8, 9 and 10. It is driven by DV_{CC}. It should be noted that the Output Current also influences the slew rate and therefore EMI emission. The higher the output current the more the slew rate and hence higher the EMI noise.

Please see datasheet for current values. The current values depend on V_{CC} and DV_{CC} respectively.

2.2.7 Pull-up Control Register (PUCR)

These registers connect an internal pull-up resistor to a port pin.

PIERxy_IEz	Function
0	Digital Input disabled
1	Digital Input enabled

Table 7. PUCR

PUCRxy_PUz	Pull-Up Resistor
0	Disabled
1	Enabled

The nominal value for this pull-up resistor is 50 kΩ. It may vary between 25 kΩ to 100 kΩ depending upon the doping process and temperature.

2.3 Input Mode

In general, if a pin should acts as a digital input, the corresponding bit in the Port Input Enable Register (0) ***must*** be set to “1”.

2.3.1 Digital Port Input

The following example shows the register configuration that needs to be done on MB96340 Series, if a pin should act as a digital input:

```
PIER06_IE0 = 1;      // port input enable
DDR06_D0 = 0; // data direction - input
ADER0_ADE0 = 0;    // disable AN0 input that is shared with port 06
                     // pin 0
```

As shown above, the resource input AN0 (Analog Input 0 of ADC) which shares port 06 pin 0 input has to be disabled for digital port input functionality.

After configuring a pin as digital port input, the level of the input pin can be determined as follows:

```
if (1 == PDR06_P0)// if pin high?
{
                     // do something
}
else                 // pin low
{
                   // do something
}
```

Additionally the level of the input pin can also be read out via the 2.2.4 External Pin State Register (EPSR) as follows:

```
if ( 1 == EPSR06_PS0 )     // if pin high?
{
                          // do something
}
else                       // pin low
{
                         // do something
}
```

Optionally the input level can be set via the 2.2.5 (Extended) Port Input Level Register (PILR, EPILR) as follows:

```
PILR06_IL0 = 1;      // set input detection level as CMOS (0208)
EPILR06_EIL0 = 1;    //
```

If the connected external source may change to high-Z state, please use an external pull-up or –down resistor or set the corresponding bit in the 2.2.7 Pull-up Control Register (PUCR).

2.3.2 Resource Input

The following example shows the register configuration that needs to be done on MB96340 Series, if a pin should act as resource input (ADC input AN0 in this case):

```
PIER06_IE0 = 1;      // port input enable
ADER0_ADE0 = 1;    // AN0 pin as analog input
```

Optionally the input level can be set via the 2.2.5 (Extended) Port Input Level Register (PILR, EPILR).

If the connected external source may change to high-Z state, please use an external pull-up or –down resistor or set the corresponding bit in the 2.2.7 Pull-up Control Register (PUCR).

2.4 Pull-up control register

All ports, while in input-mode, have the possibility to enable an internal pull-up resistor (about 50 kΩ) by programming the 2.2.7 Pull-up Control Register (PUCR)

The initial value of “0” disconnects the internal pull-up resistor, writing “1” to the corresponding bit-position in the `PUCRxy` enables the resistor.

If the port-pin is used as an output the value of the register-bit has no meaning and the pull-up resistor is disabled (Exception: For I^2C pins SDA and SCL, the setting remains. Also for UART output SOT the internal pull-up can be used if not provided by line driver).

Enabled pull-up resistors will be disabled while the microcontroller is in stop mode or timer mode, if the SPL bit of SMCR register is configured as 1 before entering these modes. The resistor is also disabled if the pin is used as ADC input.

If the external pin is used by the external bus-interface, the internal pull-up resistor cannot be used too.

2.5 Output-mode

2.5.1 Digital Port Output

The following example shows the register configuration that needs to be done on MB96340 Series, if a pin should act as a digital output:

```
PIER08_IE1 = 1;   // this configuration is required only if the external
                  // pin status needs to be read via EPSR
PDR08_P1 = 0; // clear output before setting the data direction,
                  // this is required to guarantee the initial value
                  // when the data direction is set as output
DDR08_D1 = 1; // data direction - output
TMCSR0_OUTE = 0;// disable TOT0 output that is shared with port 08
                   // pin 1
```

As shown above, the resource output TOT0 (Output of RLT0) which shares port 08 pin 1 output has to be disabled for digital port output functionality.

Optionally the output current strength can be set by the 2.2.6 Port Output Drive Register (PODR) and Port High Drive Register (PHDR)as follows:

```
PODR08_OD1 = 1;    // reduced current output
PHDR08_HD1 = 0;    //
```

2.5.2 Resource Output

The following register settings have to be done, if a pin should act as a resource output:

```
PIER08_IE1 = 1;  // this configuration is required only if the external
                  // pin status needs to be read via EPSR
TMCSR0_OUTE = 1;// enable TOT0 output
```

Optionally the output current strength can be set by the 2.2.6 Port Output Drive Register (PODR) and Port High Drive Register (PHDR).

3 Port Input / Unused Pins

How to connect Input Port Pins and how to proceed with unused Pins

3.1 Port Input / Unused Pins

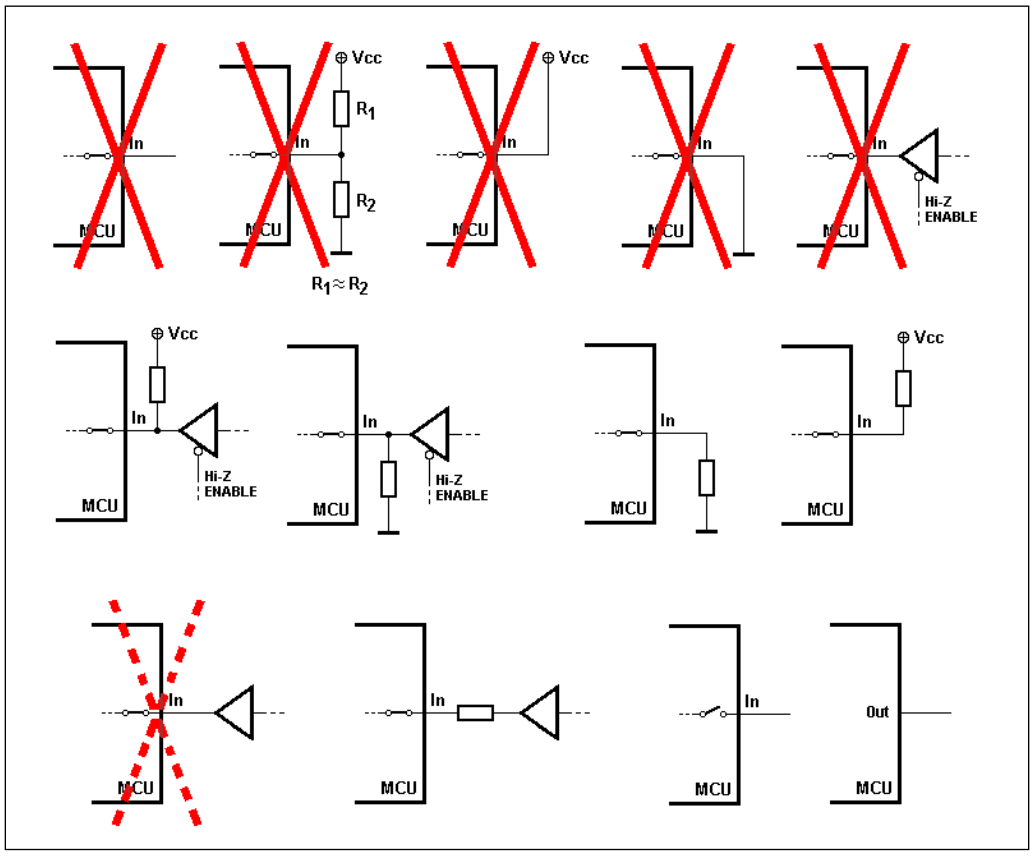
It is strongly recommended not to leave the pins unconnected, while they are switched to input and are enabled. In this case those pins can enter a so-called *floating state*. This can cause a high I_{CC} current, which is adverse to low power modes. Also damage of the MCU can happen.

Use the internal pull-up resistors in this case. If not, use external pull-up or pull-down resistors to define the input-level.

The recommended way is to set the port input enable to "0", if a port pin is unconnected.

Never connect a potential divider with almost same resistor values.

Figure 2. Recommended Connections for Port Input and Unused Pins



Be careful with connection of input pins to other devices, which can go into High-Z states. Always use internal pull-up or external pull-up or pull-down resistors in this case.

Outputs from external circuits should always be connected via a serial resistor to a MCU input pin to prevent latch-up effects caused by under- or overshoots.

Debouncing and decoupling capacitors should always be chosen as smallest as possible. For detailed information please refer the chapter 3.2 of Hardware Setup application note AN204772.

All pins are set to input disabled (i.e. corresponding DDR and PIER bits are 0) after any reset.

Do not connect any input ports directly to V_{CC} or V_{SS} (GND)! Always use pull up or down resistors

4 Technical information

Electrical characteristics of the input hysteresis

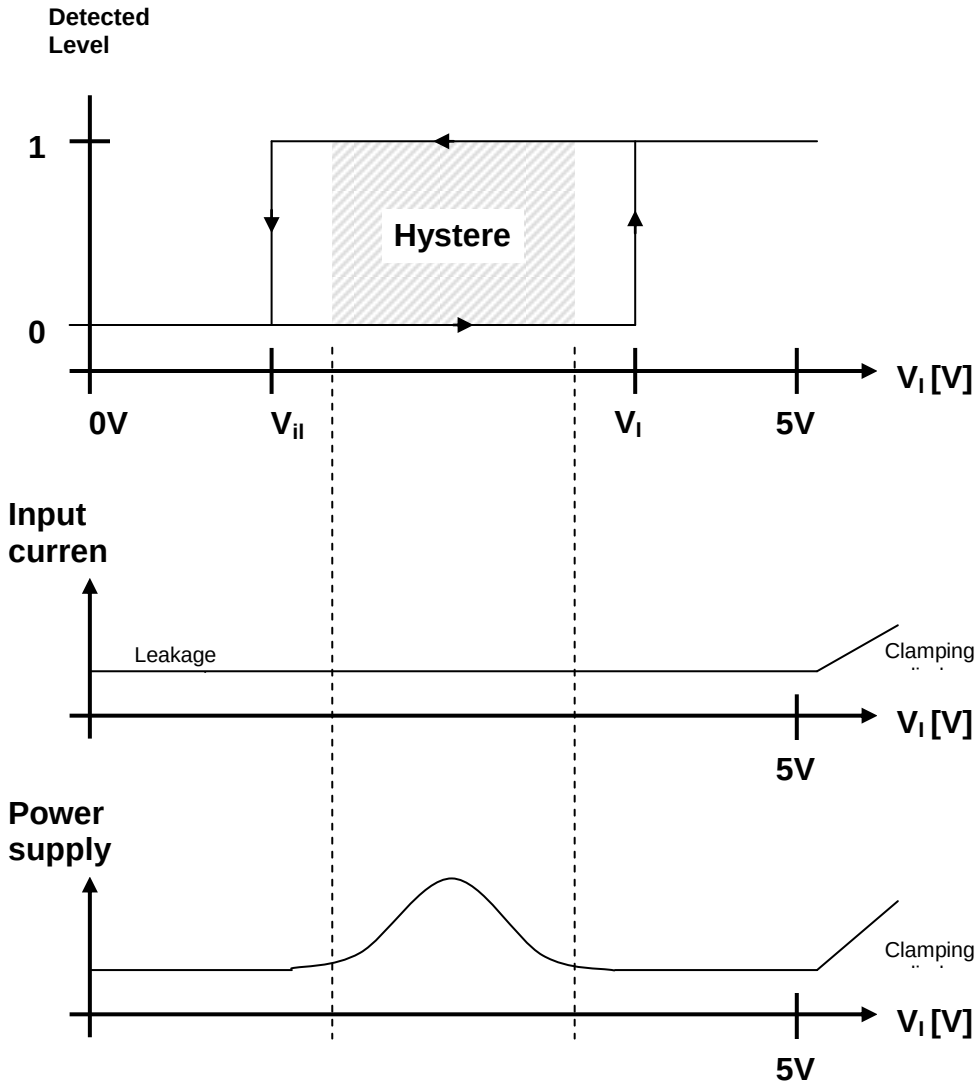
4.1 Hysteresis Inputs

A hysteresis describes the behavior of an input pin where the input level at which '1' is detected, and the level at which '0' is detected are different.

The levels are described in chapter 2.2.5.

Kindly note that the power supply current i.e. the power consumption of the device may increase, while the input voltage is within the hysteresis area, however the input current of the I/O pin remains constant.

Figure 3. Hysteresis



V_{IL} Port hysteresis input level (low-level) specified in the datasheet

V_{IH} Port hysteresis input level (high-level) specified in the datasheet

Real hysteresis area

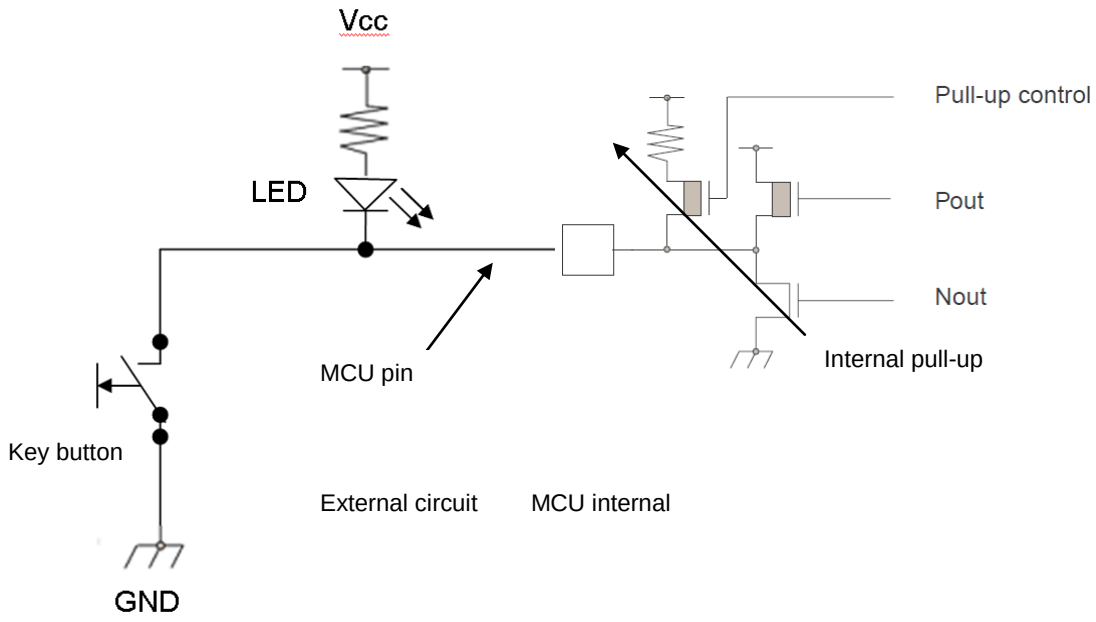
5 Using the same I/O port simultaneously as in- and Output

This Chapter Explains using one Pin Simultaneously as in-and Output

With the circuit shown in Figure 4 enabled internal pull-up resistor, pin state set low and some small considerations in the software it's possible to use the same port simultaneously as input and output – for polling a key button and driving a LED at the same pin.

If the port is used as output the LED is on. If the port is used as input the LED is off and the key button's status can be polled.

Figure 4. I/O-port example circuit



When the port direction is changed from output to input, the pin's level becomes high. The high-level is not reached immediately after the port's DDR register is set to input.

Some minor parasitic capacities (~ 30 pF) caused by chip-internal capacities and by the PCB are connected to the pin. These capacities are loaded via the internal pull-up (50 kΩ) as soon as the port is set to input. The pin reaches high-level after a typical charging time of 1 to 2 µs.

Charging time: $\tau = R * C = 50\ k\Omega * 30\ pF = 1, 5\ us$

Polling the port within this time may return a false value. As workaround we recommend to implement a short delay loop after the port's direction is switched to input and before the port polling is started. Figure 5 shows a code example for using port 09 pin 0 as in- and output.

Figure 5. Example code

```
void KeyLED_Init(void)
{
  PDR09_P0   = 0;   // Preset Port register (never output high, this might
cause
                    // shortcircuit if key button is pressed)
  PUCR09_PU0 = 1;   // Enable Pull-Up resistor to be used while in input
state
  PIER09_IE0 = 1;   // Enable Port Input
}


void LED_on(void)
{
   DDR09_D0 = 1;    // Switch port to output
}


void LED_off(void)
{
   DDR09_D0 = 0;    // Switch port to input (never output high, this might
cause
                    // shortcircuit if key button is pressed)
}


unsigned char KeyPressed(void)
{
   if (PDR09_P0 == 0)
     return (1);     // return '1' in case that the key button is pressed
   else
     return (0);
}


void main(void)
{
  ...

   KeyLED_Init();

   LED_on();                              /* switch the LED on */
   for (delay=0; delay<500000; delay++) /* keep the LED on for some time
delay */
    __asm("\tNOP");
   LED_off();                             /* switch the LED off */

   for (delay=0; delay<10; delay++)     /* short delay to get pull-up
resistor */
    __asm("\tNOP");                      /* active */

   while (!KeyPressed())                /* wait until switch is pressed */
    __asm("\tNOP");

  ...
```

6 Tips and Tricks

This chapter gives some hints on using I/O ports

6.1 Initial Value

Ensure that the port-data is defined before the pin-direction is changed to output. Otherwise undefined data might be output to the I/O-pin, until PDR00 is written.

```
PDR00 = 0x00;  // define initial value before port 0 is set to output
DDR00 = 0xFF;  // set port 0 to output, after initial value is defined
```

6.2 Bit Instructions

Use byte-instructions which will be executed faster instead of using bit instructions since all bit instructions are essentially read-modify-write instructions.

6.3 RMW Instructions

Accessing to the 2.2.1 Port Data Register (PDR) via a read-modify-write instruction always returns the contents of the register itself during read cycle (of the same read-modify-write instruction) regardless whether the resource output of the corresponding pin is enabled or not.

7 Additional Information

Information about Cypress Microcontrollers can be found on the following Internet page:

http://www.cypress.com/cypress-mcu-product-softwareexamples

The software example related to this application note is:

96340_io

Document History

Document Title: AN205493 - F²MC-16FX, All Series, I/O-port

Document Number: 002-05493

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	-	MKEA	04/20/2006	V1.0, First release, MWi
			12/05/2006	V1.1, Reviewed the document and updated with review findings, MPi
			02/21/2007	V1.2, some clarifications added, MPi
			08/02/2007	V1.3, typos corrected; small clarifications, MPi
			09/27/2009	V1.4, Added chapter “Using the same IO-port…” MHz
*A	5077748	MKEA	04/06/2016	Converted Spansion Application Note “MCU-AN-300200-E-V14” to Cypress format
*B	5869316	AESATMP9	08/31/2017	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, 2006-2017. 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.