

# General purpose Input/Output

---

---

---

This chapter starts with a description of one of the simplest integrated peripherals of the MSP430 – the General Purpose 8-bit Input Output (GPIO). The Input/Output (I/O) ports can be configured as interruptible or non-interruptible. Additionally, the port pins can be individually configured for general-purpose use, or as special function I/Os, such as USARTs, comparator signals and ADCs.

Following the technical details, there is the first in a series of hands-on laboratories. The laboratory uses the input lines to read the state of push button switches and uses the output lines to feed light emitting diodes (LEDs).

Topic	Page
<b>6.1 Introduction</b> .....	<b>6-3</b>
<b>6.2 Registers</b> .....	<b>6-4</b>
6.2.1 Direction Registers (PxDIR) .....	6-4
6.2.2 Input Registers (PxIN).....	6-4
6.2.3 Output Registers (PxOUT) .....	6-4
6.2.4 Pull-up / Pull-down Resistor Enable Registers (PxREN) .....	6-5
6.2.5 Function Select Registers: (PxSEL) and (PxSEL2) .....	6-5
6.2.6 Interruptible ports (P1 and P2).....	6-5
Interrupt Enable (PxIE) .....	6-6
Interrupt Edge Select Registers (PxIES).....	6-6
Interrupt Flag Registers (PxIFG) .....	6-6
<b>6.3 Laboratory 2: Flashing LED</b> .....	<b>6-6</b>
6.3.1 Lab2A: Blinking the LED .....	6-7
6.3.2 Lab2B: Blinking the LED half the speed .....	6-8
6.3.3 Lab2C: Toggle the LED state by pressing the push button .....	6-9
6.3.4 Lab2D: Enable / disable LED blinking by push button .....	6-9

<b>press</b>	.....	<b>6-11</b>
<b>6.4 Quiz</b>	.....	<b>6-12</b>
<b>6.5 FAQs</b>	.....	<b>6-13</b>

## 6.1 Introduction

Depending on the MSP430 device, there can be up to ten 8-bit digital Input/Output (I/O) ports, named P1 to P10.

Typically, the MSP430 I/O ports P1 and P2 have interrupt capability. Each interrupt on these I/O lines can be individually configured to provide an interrupt on a rising edge or falling edge of an input signal. All I/O lines with interrupt capacity use a single interrupt vector.

Depending on the hardware development tool used, the available digital I/O pins are:

- ❑ eZ430-F2013 MSP430 USB Stick: 10 pins - Port P1 (8 bits) and Port P2 (2 bits);
- ❑ eZ430-RF2500 MSP430 USB Stick: 32 pins - Ports P1 to P4 (8 bits);
- ❑ MSP430FG4618/F2013 Experimenter board: 80 pins – Ports P1 to P10 (8 bits). Ports P7/P8 and P9/P10 can be accessed as 16-bit values (words) as ports PA and PB respectively.

Each of these I/O ports has the following capacity:

- ❑ Independently programmable;
- ❑ Combined input, output, and interrupt conditions;
- ❑ Edge-selectable interrupt inputs for all the 8 bits of ports P1 and P2;
- ❑ Read/write access to port-control registers supported by all two- or one-address instructions;
- ❑ Each I/O has an individually programmable pullup/pulldown resistor (2xx family only).

Additionally, and depending on the device (see specific datasheet for details), the port pins can be individually configured as special functions I/O, for example:

- ❑ USART – Universal Synchronous/Asynchronous Receive/Transmit;
- ❑ Comparative signals;
- ❑ Analogue-to-Digital converter;
- ❑ Amongst others...

## 6.2 Registers

Independent of the I/O port type (non-interruptible: P3 and others) or interruptible (P1 and P2), the configuration of the port operation is defined in software using the following registers:

### 6.2.1 Direction Registers (PxDIR)

- ❑ Read/write 8-bit registers;
- ❑ Selects the direction of the corresponding I/O pin, regardless of the selected function of the pin (general purpose I/O or as a special function I/O);
- ❑ Other module functions must be set as required by the other modules;
- ❑ PxDIR configuration:
  - Bit = 1: The port pin is set up as an output;
  - Bit = 0: the port pin is set up as an input.

### 6.2.2 Input Registers (PxIN)

- ❑ Each bit of these read-only registers reflects the input signal at the corresponding I/O pin (pin configured as general purpose I/O);
- ❑ PxIN configuration:
  - Bit = 1: The input is high;
  - Bit = 0: The input is low;
- ❑ Tip: Avoid writing to these read-only registers because it will result in increased current consumption.

### 6.2.3 Output Registers (PxOUT)

- ❑ The output registers are read-write. Each bit of these registers reflects the value written to the corresponding output pin.
- ❑ PxOUT configuration:
  - Bit = 1: The output is high;
  - Bit = 0: The output is low.
  
- ❑ The 2xx family provides the additional feature that each I/O has a pullup/pulldown resistor that can be individually programmed. If the pin's pullup/pulldown resistor is enabled, the corresponding bit in the PxOUT register selects the pull-up or pull-down:
  - Bit = 1: The pin is pulled up;
  - Bit = 0: The pin is pulled down.

### 6.2.4 Pull-up/Pull-down Resistor Enable Registers (PxREN)

- ❑ Applies to the 2xx family only.
- ❑ Each bit of this register enables or disables the pullup/pulldown resistor of the corresponding I/O pin.
- ❑ PxREN configuration:
  - Bit = 1: Pullup/pulldown resistor enabled;
  - Bit = 0: Pullup/pulldown resistor disabled.

### 6.2.5 Function Select Registers: (PxSEL) and (PxSEL2)

- ❑ Some port pins are multiplexed with other peripheral module functions (see the device-specific datasheet);
- ❑ The bits: (PxSEL) and (PxSEL2 – 2xx family and some devices of the 47x(x) family), are used to select the pin function: I/O general-purpose port or peripheral module function.
- ❑ PxSEL configuration:
  - Bit = 0: I/O function is selected for the pin;
  - Bit = 1: Peripheral module function is selected for the pin.
- ❑ The 2xx family devices provide the PxSEL2 bit to configure additional features of the device. The PxSEL and PxSEL2 combination provides the following configuration of the 2xx devices:
  - Bit = 0: I/O function is selected for the pin;
  - Bit = 1: Peripheral module function is selected for the pin.

Table 6-1. Pin ports 2xx family devices configuration by PxSEL and PxSEL2 combination.

PxSEL	PxSEL2	Pin Function
0	0	Selects general-purpose I/O function
0	1	Selects the primary peripheral module function
1	0	Reserved (See device-specific data sheet)
1	1	Selects the secondary peripheral module function

- ❑ Note: When the P1 and P2 ports are configured for peripheral module function (PxSEL = 1 and/or PxSEL2), the interrupts are disabled.

### 6.2.6 Interruptible ports (P1 and P2)

Each pin of ports P1 and P2 is able to generate an interrupt request (pin is interruptible) and is configured using the PxIFG, PxIE, and PxIES registers. The port makes use of all the same configuration registers as non-interruptible ports (as described above), but with three additional registers:

### Interrupt Enable (PxIE)

- ❑ Read-write register to enable interrupts on individual pins;
- ❑ PxIE configuration:
  - Bit = 1: The interrupt is enabled;
  - Bit = 0: The interrupt is disabled.
- ❑ Each PxIE bit enables the interrupt request associated with the corresponding PxIFG interrupt flag;
- ❑ Writing to PxOUT and/or PxDIR can result in setting PxIFG.

### Interrupt Edge Select Registers (PxIES)

- ❑ This read-write register selects the transition on which an interrupt occurs for the corresponding I/O pin (if PxIE and GIE are set);
- ❑ PxIES configuration:
  - Bit = 1: Interrupt flag is set on a high-to-low transition;
  - Bit = 0: Interrupt flag is set on a low-to-high transition.

### Interrupt Flag Registers (PxIFG)

- ❑ The bit of this read-write register is set automatically when the programmed signal transition (edge) occurs on the corresponding I/O pin, provided that the corresponding PxIE bit and the GIE bit are set;
- ❑ Each PxIFG flag can be set by software, enabling an interrupt generated by software;
- ❑ Each PxIFG flag must be reset with software;
- ❑ PxIFG configuration:
  - Bit = 0: No interrupt is pending;
  - Bit = 1: An interrupt is pending.

## 6.3 Laboratory 2: Flashing LED

The first hands-on laboratory consists of configuring the I/O ports, setting up the input lines to read push buttons and the output lines to feed LEDs. The following exercises have been developed for the three hardware development tools.

The first to be discussed is the MSP-EXP430FG4618 Experimenter's board. Modifications are later made to suit the other development boards. The main differences between the boards are related to the specific ports in which the buttons and LED are (or can be) connected. For the development of this laboratory, *Code Composer Essentials v3* has been used.

This laboratory will be expanded into Labs 2A, 2B, 2C and 2D. It will start without using any input or interrupt. Then, the initial lab (2A) will be improved until it is using inputs, interrupts and low power modes. Also, the watchdog timer is used in these exercises.

Because the eZ430-F2013 does not include a push button, Lab2C and Lab2D will not support this hardware development tool.

### 6.3.1 Lab2A: Blinking the LED

#### *Project files*

- C source files: **Chapter 6 > Lab2\_X > Lab2A\_student.c**
- Solution file: **Chapter 6 > Lab2\_X > Lab2A\_solution.c**

Where **X** corresponds to the folder of the selected hardware development tool:

- eZ430: eZ430-F2013 MSP430 USB Stick
- RF2500: eZ430-RF2500 MSP430 USB Stick
- Exp\_board: MSP430FG4618/F2013 Experimenter's board

#### *Procedure*

- By analysis of the schematics, determine which I/O port pin is connected to the LED on the board:
  - Consult the MSP430FG4618/F2013 Experimenter's Board User's Guide <slau213a.pdf>;
  - Consult the eZ430-F2013 Development Tool User's Guide <slau176b.pdf>;
  - Consult the eZ430-RF2500 Development Tool User's Guide <slau227a.pdf>;
  - Or the schematics shown in Chapter 3, depending on the device.
- Port control registers:
  - Set the LED port pin as an output;
  - Switch the port state between low and high during program execution.
- Configure the watchdog timer to prevent a PUC during the program execution;
- Use an infinite loop to modify the state of the port;
- Use a software delay loop to generate the pause interval. (a long software delay loop is used here for simplicity - in real applications, a timer would be used);
- Try different values for the counter, until you can observe a visible blink.

**MSP-EXP430FG4618****SOLUTION**

Using the MSP-EXP430FG4618 Development Tool and the MSP430FG4618 device, blink LED1.

- ❑ LED1 is connected to Port 2.2
- ❑ Port control registers:
  - P2DIR: Port 2.2 is set as an output:  $P2DIR \mid = 0x04$  {to force the pin setting. It uses an OR operation (  $\mid$  ) with P2DIR and 0x04}
  - P2OUT: To switch the port state between low and high state. It uses an XOR operation (  $\wedge$  ) between P2OUT and 0x04
- ❑ The watchdog timer must be prevented from generating a PUC. Write 0x5A to the eight MSBs of the Watchdog timer control register, WDTCTL (Additional details are given in Chapter 5);
  - $WDTCTL = WDTHOLD \mid WDTPW$ ;
- ❑ Because no clock is defined, the device will use the 32.768 kHz watch crystal. In order for a rate of one blinking LED state transition each second, the software delay loop should count to approximately 30000 {  $30000/32768 \cong 1 \text{ sec}$ };

**eZ430-RF2500****SOLUTION**

Using the eZ430-RF2500 Development Tool blink the red LED.

- ❑ LED1 is connected to Port 1.1
- ❑ The programming code follows the same sequence as given above, requiring only configuration of this port.

**eZ430-F2013****SOLUTION**

Using the Ez430-F2013 Development Tool blink the LED.

- ❑ LED is connected to Port 1.0
- ❑ The programming code follows the same sequence as given above, requiring only configuration of this port.

### 6.3.2 Lab2B: Blinking the LED half the speed

**Project files**

- C source files: **Chapter 6 > Lab2\_X > Lab2B\_student.c**
- Solution file: **Chapter 6 > Lab2\_X > Lab2B\_solution.c**



Where **X** corresponds to the folder of the selected hardware development tool:

- eZ430: eZ430-F2013 MSP430 USB Stick
- RF2500: eZ430-RF2500 MSP430 USB Stick
- Exp\_board: MSP430FG4618/F2013 Experimenter's board

### **Procedure**

- Using the previous example, independently of the hardware development tool, reduce the value of the software delay to half its previous value.

## **6.3.3 Lab2C: Toggle the LED state by pressing the push button**

### **Project files**

- C source files: **Chapter 6 > Lab2\_X > Lab2C\_student.c**
- Solution file: **Chapter 6 > Lab2\_X > Lab2C\_solution.c**

Where **X** corresponds to the folder of the selected hardware development tool:

- RF2500: eZ430-RF2500 MSP430 USB Stick
- Exp\_board: MSP430FG4618/F2013 Experimenter's board

### **Procedure**

- By analysis of the schematics, determine to which port pin the push button is connected:
  - Consult the MSP430FG4618/F2013 Experimenter's Board User's Guide <slau213a.pdf>;
  - Consult the eZ430-RF2500 Development Tool User's Guide <slau227a.pdf>;
  - Schematics are given in Chapter 3.
- Ports control registers:
  - Set push button pin port as an input;
  - Enable interrupts to this pin port;
- Configure the watchdog timer to prevent a PUC during the program execution;
- Configure a low power mode.

**MSP-EXP430FG4618****SOLUTION**

Use the S1 button of the development tool to toggle the state of LED1. You must use interrupts to detect the button press and you should ensure that the MSP430 is in Low Power Mode 3 when it is not in use.

- The LED1 configuration is given in Lab 2A;
- Button S1 is connected to Port 1.0
- Port control registers:
  - P1DIR: Port 1.0 is set as an input: `P1DIR &= ~0x01` {to force the pin setting to 0. It uses an AND operation ( & ) between P1DIR and 0xFE}
  - P1IE: Enable interrupt to port 1.0;
  - P1IES: Call the port interrupt on a high-to-low transition;
- Configure the Watchdog timer;
- Enable Global Interrupts;
- Configure low power mode 3;
- Create a interrupt service routine, that includes:
  - Toggle LED1 pin port;
  - Delay for button debounce;
  - Clear interrupt flag.

**eZ430-RF2500****SOLUTION**

Use the button of the development tool to toggle the state of the red LED. You must use interrupts to detect the button press and you should ensure that the MSP430 is in Low Power Mode 3 when it is not in use.

- Follow the same sequence as given above, taking into consideration that:
- Button S1 is connected to Port 1.2;
- Port control registers:
  - The eZ430-RF2500 uses a device in 2xx family, so you need to additionally configure the button as pull-up or pull-down, in the P1REN register.

### 6.3.4 Lab2D: Enable/disable LED blinking by push button press

#### *Project files*

- C source files: **Chapter 6 > Lab2\_X > Lab2D\_student.c**
- Solution file: **Chapter 6 > Lab2\_X > Lab2D\_solution.c**

Where **X** corresponds to the folder of the selected hardware development tool:

- RF2500: eZ430-RF2500 MSP430 USB Stick
- Exp\_board: MSP430FG4618/F2013 Experimenter's board

#### *Procedure*

- Improve the last exercise (Lab 2C) by enabling the LED blinking by button press;
- Use the previous configuration;
- Include a control flow program variable that detects if the LED is blinking or not, when the button is pressed.

#### **MSP-EXP430FG4618**

#### **SOLUTION**

Use the S1 button of the development tool to start and stop the LED blinking. It is not necessary to use Lower Power Modes in this exercise.

- Follow the same sequence as presented in Lab 2C;
- Detect of the button is pressed: `if (!(P1IN & 0x01))`
- Define a variable that indicates whether the LED is blinking;
- Set the program flow depending on the state of the variable.

#### **eZ430-RF2500**

#### **SOLUTION**

Use the button of the development tool to start and stop the LED blinking. It is not necessary to use Lower Power Modes in this exercise.

- Follow the same sequence as given in Lab 2C for this hardware development tool and the additional steps provided above.

## 6.4 Quiz

1. In the MSP430 system, all I/O ports:
  - (a) Are initially set as outputs when the system powers up;
  - (b) Remain constant as the applications program proceeds;
  - (c) Vary with each step of the program;
  - (d) Are initially set as inputs when the system powers up.
  
2. To set an I/O port pin as output, the:
  - (a) PxDIR direction register bit is set;
  - (b) PxSEL function-select register bit is set;
  - (c) PxIN register bit is set;
  - (d) PxOUT register bit is set.
  
3. An input pin on the MSP430 I/O port configuration requires:
  - (a) Reset the direction register bit;
  - (b) Set the PxIN register bit to whatever the input data dictates;
  - (c) PxOUT register bit inactive;
  - (d) All of above.
  
4. For the 2xx family hardware development tools (eZ430-F2013 and eZ430-RF2500), an output pin is pulled down when:
  - (a) PxDIR and PxREN are set;
  - (b) PxDIR is set, PxREN is reset;
  - (c) PxDIR, PxREN and PxOUT are set;
  - (d) PxDIR and PxOUT are reset.
  
5. The eZ430-F2013 Port 1 I/O pin as Sigma/Delta ADC (SD16\_A) positive analogue input A0 requires:
  - (a) Set P1SEL.0 (P1SEL = 0x01h) and P1DIR.0 (P1DIR = 0x01h);
  - (b) Set P1OUT.1 (P1OUT = 0x02h) and reset P1DIR.1 (P1DIR = 0x00h);
  - (c) Set P1IN.2 (P1IN = 0x04h) and reset P1DIR.2 (P1DIR = 0x00h);
  - (d) Set P1SEL.1 (P1SEL = 0x02h) and reset P1DIR.0 (P1DIR = 0x00h).

**6.** To configure I/O Port 2 to generate an interrupt request during a low-to-high transition, you must configure:

- (a) Set the P2DIR, P2IE and P2IES associated bits;
- (b) Set the P2IE and reset the P2DIR and P2IES associated bits;
- (c) Set the P2IE and P2DIR and reset the P2IES associated bits;
- (d) None of the above.

Solution: 1. (d); 2. (a); 3. (a); 4. (a); 5. (a); 6. (b)

## 6.5 FAQs

**1.** What happens when the read-only Registers PxIN are written?

Writing to these read-only registers results in increased current consumption while the write attempt is active.

**2.** Is it appropriate to set PxREN = 1 when PxSEL = 1?

This is not recommended. It may result in unwanted current flow through the internal resistor.

**3.** What will be the status of P1 and P2 interrupts when peripheral module function is enable (PxSEL = 1)?

Regardless of the state of the corresponding P1IE or P2IE bit, signals on the associated pins will not generate interrupts, because the pin's corresponding interrupt function is disabled.

**4.** What will be the state of the PxIFG flags when PxOUT or PxDIR are changed (only for the interruptible ports, Port 1 and Port 2)?

Writing to P1OUT, P1DIR, P2OUT, or P2DIR can result in setting the corresponding P1IFG or P2IFG flags.

**5.** What should be the minimum length of time of an external I/O pin interrupt event?

It should be active for a minimum time period equal to  $1.5 \times \text{MCLK}$ , to ensure that it is recognized and that the corresponding interrupt flag is set.

**6. What happens to the PxIFGx status when writing to PxIESx?**

It depends on the PxIN status. If PxIN is reset and there is a low-to-high transition on PxIESx, then the flag may be set. If PxIN is set and a high-to-low transition occurs on the PxIESx, the flag also may be set.

**7. How should the unused port pins be configured?**

They should be configured as I/O function, output direction, and left unconnected on the PC board. This prevents floating inputs and reduces power consumption. For the 2xx family of devices, an integrated pull-up/pull-down resistor can be enabled by setting the PxREN bit of the unused pin, which prevents the input from floating.