Technical Note

Using the GPIO Pins of the Z8 Encore! XP® MCU

TN002402-1107

Abstract

This technical note describes how to initialize and use the GPIO pins of the Z8 Encore! XP® MCU for input, output, and alternate function operations. The Technical Note contains examples on how to read/write to/from the Z8 Encore! XP® GPIO pins initialized for different operations.

Overview of General-Purpose I/O

The eZ8 CPU core of the Z8 Encore! XP® microcontrollers supports seven 8-bit Ports (Ports A-G) and one 4-bit Port (Port H) for general-purpose input or output operations. Specific Control and Data registers are associated with each port.The GPIO Control registers are used to determine data direction, open-drain, output drive current and alternate pin functions. Each port is individually programmable.

In Figure 1, a simplified block diagram of a GPIO Port pin is illustrated.

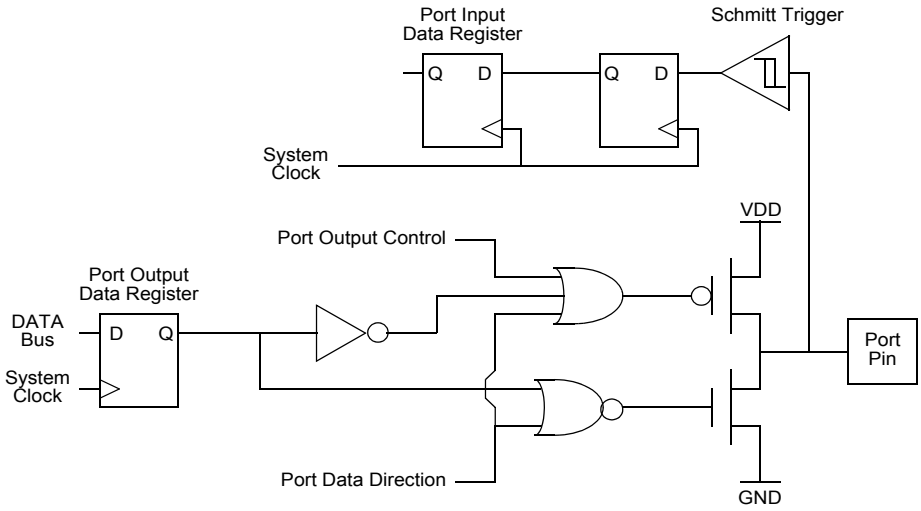

Figure 1. Z8 Encore! XP® GPIO Port Pin Block Diagram

When a GPIO pin is configured as input, the MCU reads the data on these pins into its registers and uses the data for further processing. The GPIO Port pins can be used as external interrupt sources when set for input operations. When a GPIO pin is configured as output, data from the MCU is sent on these pins to connected devices/circuits. When configured as an alternate function pin, a GPIO pin configuration depends on the peripheral function to which it provides access.

Not all devices of the Z8 Encore! XP® family of microcontrollers support all 8 ports. For a complete listing of the GPIO port availability according to the device type and packaging, and the address map for all the register files, refer to the *Z8 Encore! XP® 64K Series Flash Microcontrollers Product Specification* (PS0199) available on `www.zilog.com`.

The GPIO pin is configured for input, output or alternate function operations using the GPIO registers and sub-registers. Figure 2 illustrates the format to define the use of Port pins. The two main registers used in defining a port are:

- Port Address (PxADDR)
- Port Control (PxCTL)

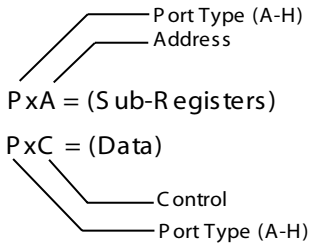

Figure 2. Format Defining Port Pins

The status of the GPIO pins at reset is as follows:

- The Port pins are in normal mode – no alternate function is selected
- All the Port pins are configured as inputs
- The content of the Input Data register (PxIN) is unknown
- The content of the Output Data register (PxOUT) is `0x00`
- The Port pin output driver is tri-stated.

The values of the Port registers and sub-registers at Reset are tabulated in Table 1 and Table 2.

Table 1. Z8 Encore! Port Register Values at Reset

Port Register Mnemonic	Value at Reset	Meaning
PxADDR	00H	Does not perform any function
PxCTL	00H	Does not perform any function
PxIN	xxH	Contents of this register are either 0 or 1
PxOUT	00H	Content of this register is 00

Table 2. Z8 Encore! Port Sub-Register Values at Reset

Port Sub-Register Mnemonic	Value at Reset	Meaning
DATA_DIR	FFH	Configures the Port pins as input pins. The output driver is tri-stated.
ALT_FUN	00H	Configures the Port pins to function in normal mode.
OUT_CTL	00H	Enables the drains of the Port pins.
HDR_EN	00H	Configures the Port pins for standard output current drive.
SMRS_EN	00H	Transitions on these pins during the STOP mode do not initiate STOP mode recovery.

All the GPIO pins must be initialized to the required or known state before use. The following sections first offer a brief overview of the Z8 Encore! XP® MCU register descriptions and then guide the user to set the Z8 Encore! XP® GPIO Port pins.

Z8 Encore! XP® Register Descriptions

There are four Z8 Encore! XP® MCU registers that provide access to GPIO control, input data, and output data. The Port A-H Address and Control registers together are used to provide access to the sub-registers for Port configuration and control.

The four registers of the Z8 Encore! XP® MCU are briefly described below:

- Port Address register (PXADDR) selects the sub-register suited for the GPIO Port functionality. It is used with the Port Control register to provide access to all GPIO Port control.
- Port Control register (PXCTL) sets the GPIO Port operation by providing access to sub-registers.
- Port Input Data register (PXIN) returns the values of the Port pins to be accessed.
- Port Output Data register (PXOUT) contains the data to be driven out from the Port pin and writes data to the Port pin.

Note: The four registers mentioned above are defined in the header file, `ez8.h`. To use the definitions, include the statement, `#include <eZ8.h>`, in the C-code file.

A brief description of the Port sub-registers used with the Port Address registers is given below:

- Data Direction sub-register (DATA_DIR) contains a defined data equal to `0x01`, and it is used to configure Port pins as either input or output. Assign 1 to set as input and 0 as output.

- Alternate Function sub-register (ALT_FUN) is used to activate on-chip peripheral functions and is defined as 0x02 data. Assign 1 to pins to use as alternate function.
- Output Control sub-register (OUT_CTL) sets the output pins as push-pull (0) and open-drain (1). This sub-register contains defined data equal to 0x03, and affects the pin directly.
- High Drive Enable sub-register (HDR_EN) is used to enable and disable pins operating in high current output drive. This sub-register contains a defined value equal to 0x04, and directly affects the pin.
- Stop Mode recovery Source Enable sub-register (SMRS_EN) configures the pins as Stop Mode recovery source and is defined as 0x05 data.

The sub-register values are defined as follows in all the examples cited in this document:

```
#define DATA_DIR     0x01        // Data Direction
#define ALT_FUN      0x02        // Alternate Function
#define OUT_CTL      0x03        // Output Control (Open-Drain)
#define HDR_EN       0x04        // High Drive Enable
#define SMRS_EN      0x05        // Stop Mode Recovery Source Enable
```

Note: To safeguard the sub-registers from being inadvertently over-written, please set the PxADDR register to 0x00 after setting the sub-registers for particular GPIO Port pins as required.

For further details on the sub-registers, refer to the *Z8 Encore! XP® 64K Series Flash Microcontrollers Product Specification* (PS0199) available on www.zilog.com.

Setting the GPIO Port as Input

To configure GPIO Port pins as input, use the appropriate Port Address register along with the Port Control register and the Data Direction sub-register. To prevent any effect on other functions of the Port, the Alternate Function sub-register is set to a default value—0x00. To access data from the Port pins, use the Port Input Data register.

For example, assume that you want to use Port F pin 6 (PF6) as an input pin. The following piece of code illustrates how you can set bit 6 of Port F as input.

```
/*******************************************************/
// Initialization for Port F bit 6 to be used as input  //
/*******************************************************/

PFADDR = 0x02;                // Port F Address register selects
                              // Alternate Function sub-reg
PFCTL &= 0x00;                // Port F Control register is set to
                              // default value because
                              // Port F has no Alternate Function

PFADDR = 0x01;                // Port F Addr reg selects Data Direction
                              // sub-register
PFCTL |= 0x40;                // Port F Control Register sets
                              // bit 6 as input
                              // No need to select Output Control sub-
                              // register since required configuration
                              // is input

 /*************************************/
//    Reading data from Port F bit 6     //
/*************************************/

while(1)
{
   Data = PFIN;               // Store Data from Port F to
                              // temporary register
                              // Data (Data is fetched from
                              // Port Input Data Register)

   if ((Data & 0x40) == 0x00) // Test if bit6 of Data is zero
   {
      user_routine();         // add user code here
   }
}
```

All GPIO pins of the Z8 Encore! XP® MCU can be configured to generate external interrupt requests on either the rising edge or falling edge of the input signal. Some Port pins can be set to generate interrupts on both the rising and falling edges of the input signal. External interrupts can only operate when the Port pins are configured as input, and proper initialization of interrupts must be adapted. For more information on initialization of external interrupts, refer to the *Z8 Encore! XP® 64K Series Flash Microcontrollers Product Specification* (PS0199) available on `www.zilog.com`.

Setting the GPIO Port as Output

To configure GPIO Port pins as output, the Port Address register and the Port Control register are used with the corresponding sub-registers (data direction and output control). To output data to the Port pins, the Port Output Data register is also used.

For example, assume that you want to use Port G and Port E as output Ports (where data is driven out through the pins); Port G is active high and Port E is active low.

The following piece of code illustrates how you can set the Port G and Port E as output and drive the data through them.

```
/************************************************/
// Initialization of Ports G and E as Output     //
/************************************************/

PGADDR = 0x02;                // Port G Address reg selects Alternate
                              // Function sub-register
PGCTL &= 0x00;                // Port G Control set to default value
                              // because Port G has no alternate
                              // function
PGADDR = 0x01;                // Port G Address reg accessed
                              // Data Direction sub-register
PGCTL &= 0x00;                // Port G Control reg sets all the bits
                              // as output
PGADDR = 0x03;                // Port G Address reg chooses
                              // Output control.
PGCTL &= 0x00;                // Port G Control reg is configured as
                              // push-pull (It gives off and sinks
                              // current)
PEADDR = 0x02;                // Port E Address reg selects Alternate
                              // Function sub-reg
PECTL &= 0x00;                // Port E Control reg set to no alternate
                              // function
PEADDR = 0x01;                // Port E Address reg selects
                              // Data Direction sub-reg
PECTL &= 0x00;                // Port E Control reg configures all bits
                              // as output
PEADDR = 0x03;                // Port E Address reg accessed
                              // Output Control sub-reg
PECTL &= 0x00;                // Port E Control reg sets Output Control
                              // to push-pull

/************************************/
// Routine to Drive Data High //
/************************************/

void Drive_High_to_Port_G(void)
{
     PGOUT = 0x7f;            // Set bit 0-6 to logic high
     PEOUT &= 0x00;           // Set all bits of Port E to low
}

/************************************/
// Routine to Drive Data Low          //
```

```
/************************************/

void Drive_Low_to_Port_G(void)
{
     PGOUT &= 0x00;           // Set all bits to logic low
     PEOUT &=0x00;            // Set all bits to low
}
```

Setting the GPIO Port for Alternate Function Operation

When a GPIO Port pin is configured for alternate function operation, it provides access to special on-chip peripheral functions such as timers and serial communication devices. To configure a GPIO Port pin for alternate function operation, the Port Address register must be defined as alternate function type and the Port Control register must assign values pertaining to the specific Port pins involved in using such functions. Table 3 lists the alternate functions associated with each Port pin for the Z8 Encore! XP® MCU.

Table 3. Port Alternate Function Mapping

Port	Pin	Mnemonic	Alternate Function Description
Port A	PA0	T0IN	Timer 0 Input
	PA1	T0OUT	Timer 0 Output
	PA2	N/A	No Alternate Function
	PA3	CTS0 (Active Low)	UART 0 Clear to Send
	PA4	RXD0 / IRRX0	UART 0 / IrDA 0 Receive Data
	PA5	TXD0 / IRTX0	UART 0 / IrDA 0 Transmit Data
	PA6	SCL	I^2C Clock (automatically open-drain)
	PA7	SDA	I^2C Data (automatically open-drain)
Port B	PB0	ANA0	ADC Analog Input 0
	PB1	ANA1	ADC Analog Input 1
	PB2	ANA2	ADC Analog Input 2
	PB3	ANA3	ADC Analog Input 3
	PB4	ANA4	ADC Analog Input 4
	PB5	ANA5	ADC Analog Input 5
	PB6	ANA6	ADC Analog Input 6
	PB7	ANA7	ADC Analog Input 7

Table 3. Port Alternate Function Mapping (Continued)

Port	Pin	Mnemonic	Alternate Function Description
Port C	PC0	T1IN	Timer1 Input
	PC1	T1OUT	Timer1 Output
	PC2	SS(Active Low)	SPI Slave Select
	PC3	SCK	SPI Serial Clock
	PC4	MOSI	SPI Master Out Slave In
	PC5	MISO	SPI Master In Slave Out
	PC6	T2IN	Timer2 In
	PC7	T2OUT	Timer2 Out (not available in 40-pin package)
Port D	PD0	T3IN	Timer3 In (not available in 40- and 44-pin package)
	PD1	T3OUT	Timer3 Out (not available in 40- and 44-pin package)
	PD2	N/A	No Alternate Functions
	PD3	N/A	No Alternate Functions
	PD4	RXD1 / IRRX1	UART 1 / IrDA 1 Receive Data
	PD5	TXD1 / IRTX1	UART 1 / IrDA 1 Transmit Data
	PD6	CTS1 (Active Low)	UART 1 Clear to Send
	PD7	RCOUT	Watch-Dog Timer RC Oscillator Output
Port E	PE [7:0]	N/A	No Alternate Functions
Port F	PF [7:0]	N/A	No Alternate Functions
Port G	PG [7:0]	N/A	No Alternate Functions
Port H	PH0	ANA8	ADC Analog Input 8
	PH1	ANA9	ADC Analog Input 9
	PH2	ANA10	ADC Analog Input 10
	PH3	ANA11	ADC Analog Input 11

For example, assume that you want to use the Z8 Encore! XP® Timer 0, by configuring a Port A pin for alternate function operation. Only the output features of Timer 0 are considered for this example.

The code in Example 1, illustrates how you can configure Port A pin 1 (PA1) for alternate function operation by setting the appropriate bits in the registers.

Example 1:

```
/*************************/
//    Initialization     //
/*************************/

PAADDR = 0x02;               // Port A Address reg selects alternate
                             // function sub-reg
PACTL |= 0x02;               // Port A Control reg sets Timer0 Output.

PAADDR = 0x01;               // Port A Address reg selects
                             // Data Direction sub-reg
PACTL &= 0xfd;               // Port A Control reg configures bit 1
                             // as output since Timer0 Out is selected
PAADDR = 0x03;               // Port A Address reg selects
                             // Output Control sub-reg
PACTL &= 0xfd;               // Port A Control reg configures bit 1
                             // as push-pull
PAADDR = 0x00;               // advised for proper settings to avoid
                             // inadvertent changes to port
                             // sub-registers
```

Example 2:

The following code illustrates how you can configure Port A pin 0 for alternate function operation, where the Timer0 input features are used.

```
/***************/
//Initialization  //
/***************/

PAADDR = 0x02;               // Port A Address reg selects alternate
                             // function sub-reg
PACTL |= 0x01;               // Port A Control reg sets Timer0 Input

PAADDR = 0x01;               // Port A Address reg selects
                             // Data Direction sub-reg
PACTL |= 0x01;               // Port A Control reg configures bit 0
                             // as Input since Timer0 Input is selected
PAADDR = 0x00;               // advised for proper settings to avoid
                             // inadvertent changes to port
                             // sub-registers
```

Note: The Port A-H Address registers select the GPIO Port functionality accessible through the Port A-H Control registers. Setting the Port pins to 1 in the Port A-H Address register through the Port Control register enables the corresponding Port for alternate function operation.

Setting the GPIO Port for Mixed Mode Operation

Now that we have seen how to configure Z8 Encore! XP® MCU Ports for input, output and alternate function operations, here is an example of how to simultaneously use different pins in the same Port as input, output, and for alternate function operation.

For example, assume that you want to use the Timer0 output pin by configuring Port A pin 1 (PA1) for alternate function operation. You also want to use Port A pin 4 and 5 as input, and Port A pins 6 and 7 as open-drain outputs. After configuration, you want to read bits [4, 5] and write [6,7] bits with [1,0] logic.

The code given below illustrates how to configure Port A for mixed mode operation.

```
/****************/
// Initialization //
/***************/

PAADDR = 0x02;                // Port A Address reg selects alternate
                              // function sub-reg
PACTL |= 0x02;                // Port A Control reg sets Timer0 Output.
                              // All other pins are Free for use as GPIO
PAADDR = 0x01;                // Port A Address reg selects
                              // Data Direction sub-reg
PACTL &= 0x3D;                // 0x3D = 0011-1101 configures bit 1,6,7
                              // as outputs and pins 4,5 (& others)
                              // as input
PAADDR = 0x03;                // Port A Address selects Output Control
PACTL &= 0xFD;                // Port A Control keeps bit 1 as default
                              // push-pull and configures bits 6,7
                              // as open drain
PAADDR = 0x00;                // advised for proper settings to avoid
                              // inadvertent changes to port
                              // sub-registers
data = PAIN;                  // Read the value of Port A Input reg into
                              // the variable ‘data’
data = data & 0x30;           // ‘AND’ this value with 00110000 to mask
                              // the values of bit 4 & 5
PAOUT = data | 0x40;          // ‘OR’ the resultant ‘data’ with 01000000
                              // and write to Port A Output reg to set
                              // bit [6,7] to [1,0]
```

Conclusion

- GPIO pins are easily configured using the Port Address register together with Port Control register and the sub-registers as explained in the document.
- A GPIO pin can be used as a source of external interrupt only when configured as input.

- Table 3 demonstrates that the on-chip peripheral function of the Z8 Encore! XP® device depends on the type and the bit position of its Ports.
- Access to the GPIO Port is made through Port Input Data register and Port Output Data register.

Warning: DO NOT USE IN LIFE SUPPORT

LIFE SUPPORT POLICY

ZILOG'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS PRIOR WRITTEN APPROVAL OF THE PRESIDENT AND GENERAL COUNSEL OF ZILOG CORPORATION.

As used herein

Life support devices or systems are devices which (a) are intended for surgical implant into the body, or (b) support or sustain life and whose failure to perform when properly used in accordance with instructions for use provided in the labeling can be reasonably expected to result in a significant injury to the user. A critical component is any component in a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system or to affect its safety or effectiveness.

Document Disclaimer