Application Note

Initializing Peripherals in Z8 Encore! XP® 8-Pin Devices

AN027801-0808

Z8 Encore! XP® Flash Microcontrollers

Abstract

This application note describes the initialization of the peripherals of the Z8 Encore! XP® 8-Pin Devices (Z8F082A and Z8F0823 family of devices). The initialization covered in this Application Note include the following components:

- Stack Pointer
- System Clock and IPO
- General Purpose Input/Output (GPIO) Port A
- Universal Asynchronous Receiver Transmitter (UART)
- Timer1
- Analog-to-Digital Converter (ADC)
- Temperature Sensor
- Watchdog Timer
- Comparator

Initialization

To ensure that the Z8 Encore! XP® is powered up to a known and desirable state, the device's peripheral must be properly initialized. Initialization is achieved by running the initialization codes. Either the Assembly version or the C version of the code can be run to initialize the peripheral.

In this Application Note, for each peripheral, one example each of Assembly code and C code is given to illustrate the initialization process. Reference to the control register section of the peripherals in the Product Specifications is also given.

Stack Pointer

The stack pointer holds the current stack address and is used for all stack operations of the CPU. It must be set to point to a section of the Register File that does not cause user data program to be overwritten.

Stack Pointer Assembly Code Initialization Example

```
;Stack starting address
NEAR_STACK EQU %00FF ; Define Stack at RAM Address 00FF hex
; Startup and Stack Pointer Initialization code
; Startup code
startup:
    DI                                  ; Prevent interruption
    SRP #%00                            ; Working registers 00-0f
    LDX SPL, #LOW(NEAR_STACK)           ; Initialize stack pointer
    LDX SPH, #HIGH(NEAR_STACK)
```

Stack Pointer C Code Initialization Example

```
// Initialize Stack Pointer
DI();
SPL = NEAR_STACK & 0x00FF; // Initialize stack pointer
SPH = NEAR_STACK >> 8;
```

System Clock

The system clock drives the CPU and all the peripherals. The source of the system clock is user-selectable and can be one of the following:

- On-chip Internal Precision Oscillator (IPO)
- On-chip low power Watchdog Timer (WDT) Oscillator
- External clock drive

Selection of the clock source for system clock is made during initialization of the system clock. For 8-pin packages, most applications configures the Internal Precision Oscillator (IPO) as the system clock source because the IPO requires no external components.

System Clock Assembly Code Initialization Example with IPO selected as system clock

```
; Initialize system clock with Internal Precision Oscillator (IPO)
; In this example IPO, WDT, SOFEN and WDFEN are enabled
    LDX OSCCTL, #%E7 ;Unlock sequence
    LDX OSCCTL, #%18 ;Unlock sequence
    LDX OSCCTL, #%B8 ;Selecting IPO as system clock at 5.53 MHz with
                     ;WDT, SOFEN and WDFEN enabled.
```

System Clock C Code Initialization Example

```
// Initialize system clock with Internal Precision Oscillator (IPO)
// In this example IPO, WDT, SOFEN and WDFEN are enabled
OSCCTL = 0xE7; //Unlock Sequence;
OSCCTL = 0x18; //Unlock Sequence
OSCCTL = 0xB8; //Selecting IPO as system clock at 5.53 MHz with
               //WDT, SOFEN and WDFEN enabled
```

General Purpose Input/Output (GPIO)

The Z8 Encore! XP® products support a maximum of 25 port pins in four ports (Port A, B, C and D) for general-purpose input/output (GPIO) operations. For 8-pin packages, only Port A is available.

To initialize the GPIO, run the example Assembly code or the C code below.

GPIO Assembly Code Initialization Example

```
; Initializes General Purpose IO
;In the example below, Port A pins are configured as follows:

init_gpio:
   ; Port A initialization
    LDX PAOUT, #%02  ; Load Port A Output Data Register bit PA1 with
                     ; value 1.
    LDX PAADDR, #%07 ; Set Address Pointer to AFS1 Register
    LDX PACTL, #%2C  ;Set Port A pin configuration as follows:
       ;PA0 as Input
       ;PA1 as Output
       ;PA2 as Timer1 output
       ;PA3 as Analog Input
       ;PA4 as UART 0 Receive
       ;PA5 as Comparator Input (CINP)
    LDX PAADDR, #%08 ; Set Address Pointer to AFS2 register
    LDX PACTL, #%28  ; Set Port A pin Configuration as follows:
       ;PA0 as Input
       ;PA1 as Output
       ;PA2 as Timer1 output
       ;PA3 as Analog Input
       ;PA4 as UART 0 Receive
       ;PA5 as Comparator Input (CINP)
    LDX PAADDR,#%02 ;Set Address Pointer to AF register
    LDX PACTL, #%3C; Set Port A Pin Configuration
       ;PA0 as Input
       ;PA1 as Output
       ;PA2 as Timer1 output
       ;PA3 as Analog Input
       ;PA4 as UART 0 Receive
```

```
    ;PA5 as Comparator Input (CINP)
  LDX PAADDR #%01 ;Set Address Pointer to Data Direction (DD)Register
  LDX PACTL, #%FD ;Set Port A PA1 to output mode and rest of pins to
                  ;Input Mode
  LDX PAADDR,#%00 ;Lock Port A Control Register
```

GPIO C Code Initialization Example

```
//Initialize GPIO
   PAOUT = 0x02;  //Load Port A Output Data Register bit PA1 with value
                    1.
   PAADDR = 0x07;//Set Address Pointer to AFS1 Register
   PACTL = 0x2C; /* Set Port A pin configuration as follows:
                    PA0 as Input
                    PA1 as Output
                    PA2 as Timer1 output
                    PA3 as Analog Input
                    PA4 as UART 0 Receive
                    PA5 as Comparator Input (CINP)*/
   PAADDR =0x08; //Set Address Pointer to AFS2 register
   PACTL = 0x28; /* Set Port A pin Configuration as follows:
                    PA0 as Input
                    PA1 as Output
                    PA2 as Timer1 output
                    PA3 as Analog Input
                    PA4 as UART 0 Receive
                    PA5 as Comparator Input (CINP) */
   PAADDR = 0x02;   //Set Address Pointer to AF register
   PACTL = 0x3C;    /* Set Port A Pin Configuration as follows:
                       PA0 as Input
                       PA1 as Output
                       PA2 as Timer1 output
                       PA3 as Analog Input
                       PA4 as UART 0 Receive
```

```
                    PA5 as Comparator Input (CINP) */
   PAADDR = 0x01; //Set Address Pointer to Data Direction (DD)Register
   PACTL = 0xFD;    /*Set Port A PA1 to output mode and rest of pins to
                      Input Mode */
   PAADDR = 0x00;   //Lock Port A Control Register
```

Further References

For detailed definition of the GPIO Control Registers, please refer to GPIO Control Register Definitions Section of either the Z8 Encore! XP® F082A Series Product Specifications (PS0228) or Z8 Encore! XP® F0823 Series Product Specifications (PS0243) as applicable.

Universal Asynchronous Receiver/Transmitter (UART)

The universal asynchronous receiver/transmitter (UART) is a full-duplex communication channel capable of handling asynchronous data transfers. The UART supports 8- and 9-bit data modes and selectable parity. The UART also supports multi-drop address processing in hardware. The UART baud rate generator (BRG) can be configured and used as a basic 16-bit timer.

UART0 Assembly Code Initialization Example

```
; Initialize UART0
init_uart:
    LDX U0BRH, #HIGH(UART_BRG);Load High Byte value of UART_BRG
    LDX U0BRL, #LOW(UART_BRG); Load Low Byte value of UART_BRG
    LDX U0CTL0, #%40 ; Transmit disable, Receive Enable,No Parity,1 stop
    LDX U0CTL1, #%00 ; clear for normal non-Multiprocessor operation,
                     ; DE Signal is active low if enabled

RET
; End of Code
```

Note: *To configure PA4 as UART0, please refer to* GPIO Assembly Code Initialization Example.

UART0 C Code Initialization Example

```
//Initialize UART0
   U0BRH = (UART_BRG >> 8);//Load High Byte value of UART_BRG
   U0BRL = (UART_BRG & 0x00FF); //Load Low Byte value of UART_BRG
   U0CTL0 = 0x40;  //Transmit disable, Receive Enable,No Parity,1 stop
   U0CTL1 = 0x00; //clear for normal non-Multiprocessor operation
```

```
                    //DE Signal is active low if enabled
//End of code
```

Note: *To configure PA4 as UART0, please refer to* GPIO C Code Initialization Example.

Further References

For detailed definitions of the UART Control Registers, please refer to the UART Control Register Section of either the Z8 Encore! XP® F082A Series Product Specifications (PS0228) or Z8 Encore! XP® F0823 Series Product Specifications (PS0243) as applicable.

Timers

The Z8 Encore! XP® 8K and 4K Series products contain two 16-bit reloadable timers (Timer 0 and Timer 1) that can be used for timing, event counting, motor control operations or generation of pulse-width modulated (PWM) signals. These timers provide a 16-bit programmable reload counter and operate in ONE-SHOT, CONTINUOUS, GATED, CAPTURE, CAPTURE RESTART, COMPARE, CAPTURE and COMPARE, PWM SINGLE OUTPUT AND PWM DUAL OUTPUT modes.

To initialize Timer1, run the example Assembly code or C code below.

Timer1 Assembly Code Initialization Example

```
; Initialize Timer1
init_timer1:
    LDX T1CTL0, #0 ; Timer1 interrupts on all defined Reloads, compares
                   ; and Input Events and PWM has no delays
    LDX T1H, #HIGH(TIMER_RR);Load High Byte value of TIMER_RR to High
                             Byte of Timer1;
    LDX T1L, #LOW(TIMER_RR); Load Low Byte value of TIMER_RR to Low Byte
                             of Timer1;
    LDX T1RH, #HIGH(TIMER_RR);Load High Byte value of TIMER_RR to High
                              Byte of Timer1 Reload Register
    LDX T1RL, #LOW(TIMER_RR); Load Low Byte value of TIMER_RR to Low
                              Byte of Timer1 Reload Register
    LDX T1CTL1, #(%C1 | (TIMER_PRES << 3)); Enable Timer1, set Timer to
                                          ; continuous counting mode, set
                                        ; TPOL = 1 for initial T1Out = 1,
                                         ; and T1Out is complemented upon
                                         ; timer reload; Load value of
                                         ; TIMER_PRES after shifting left
```

```
                                        ; 3 spaces.
    ; End of Code
```

Note: *To configure PA2 as Timer1 Output (T1OUT), please refer to* GPIO Assembly Code Initialization Example.

Timer1 C Code Initialization Example

```
//Initialize Timer1
 T0CTL1 = 0;                    /* Timer1 interrupts on all defined
                                   Reloads, compares and Input Events
                                   and PWM has no delays */
 T1H = HIGH(TIMER_RR >> 8);     /*Load High Byte value of TIMER_RR to
                                  High Byte of Timer1 */
 T1L = LOW(TIMER_RR & 0x00FF); /*Load Low Byte value of TIMER_RR to Low
                                  Byte of Timer1 */
 T1RH = HIGH(TIMER_RR >> 8);    /*Load High Byte value of TIMER_RR to
                                  High Byte of Timer1 Reload Register */
 T1RL = LOW(TIMER_RR & 0x00FF);/*Load Low Byte value of TIMER_RR to
                                  Low Byte of Timer1 Reload Register*/
 T1CTL1 = (0xC1 | (TIMER_PRES << 3)); /* Enable Timer1, set Timer to
                                         continuous counting mode, set
                                        TPOL = 1 for initial T1Out = 1,
                                         and T1Out is complemented upon
                                         timer reload; Load value of
                                         TIMER_PRES after shifting left
                                         3 spaces.*/
//End of Code
```

Note: *To configure PA2 as Timer1 Output (T1OUT), please refer to* GPIO C Code Initialization Example

Further References

For detailed definition of the Timer Control Register, see Timer Control Register Definitions Section of either the Z8 Encore! XP® F082A Series Product Specifications (PS0228) or Z8 Encore! XP® F0823 Series Product Specifications (PS0243) as applicable.

Analog-to-Digital Converter (ADC)

The Analog-to-Digital converter (ADC) converts an analog input signal to a 10-bit binary number. The ADC Control Register 0 (ADCCTL0) selects the analog input channel and initiates the analog-to-digital conversion. It also selects the voltage reference configuration. The ADC Control/Status Register 1 (ADCCTL1)configures the input buffer stage, enables the threshold interrupts and contains the status of both threshold triggers. It is also used to select the voltage reference configurations.

To initialize the ADC, run the example Assembly code or C code below.

ADC Assembly Code Initialization Example

```
;initialize ADC
   ANDX PWRCTL0, #%FB  ;Power on the ADC Peripheral
   LDX ADCCTL1, #%80   ;Internal Reference set to 2.0v, single-ended,
                       ;unbuffered input
   LDX ADCCTL0, #%92   ; Enable conversion, Internal Reference set to
                       ;2.0v, Reference Buffer disabled, ANA2 is input,
                       ; and continuous conversion mode.
; End of Code
```

Note: *To configure PA3 as ANA2 Input, please refer to* GPIO Assembly Code Initialization Example.

ADC C Code Initialization Example

```
//Initialize ADC
   PWRCTL0 &= 0xFB; //Power on the ADC Peripheral
   ADCCTL1 = 0x80;  /*Internal Reference set to 2.0v, single-ended,
                      unbuffered input */
   ADCCTL0 = 0x92;  /* Enable conversion, Internal Reference set to
                       2.0v, Reference Buffer disabled, ANA2 is input,
                       and continuous conversion mode.*/
//End of Code
```

Note: *To configure PA3 as ANA2 Input, please refer to* GPIO C Code Initialization Example.

Further References

For detailed definition of the ADC Control Registers, see ADC Control Register Definition section of either the Z8 Encore! XP® F082A Series Product Specifications (PS0228) or Z8 Encore! XP® F0823 Series Product Specifications (PS0243) as applicable.

Temperature Sensor

The on-chip Temperature Sensor measures temperature on the die by producing an analog output signal proportional to the device temperature. This signal can be sent to either the ADC or the analog comparator. The Temperature Sensor is factory calibrated for in-circuit software correction. Uncalibrated accuracy is significantly worse, therefore the Temperature Sensor is not recommended for uncalibrated use.

Note: *Please note that the Temperature Sensor is only applicable to the Z8 Encore! XP® F082A series of device only.*

Temperature Sensor Assembly Code Initialization Example

;Temperature Sensor Initialization

```
ANDX PWRCTL0, #%F3  ;Power On Temperature Sensor, Peripheral and ADC
LDX ADCCTL1, #%81   ;Internal Reference set to 2.0v, Single-ended,
                    ; buffered input with unity gain
LDX ADCCTL0, #%9E   ; Enable Conversion, Internal Reference set to
                    ; 2.0v, Reference Buffer disabled, temperature
                    ; sensor is input
```

;End of Code

Temperature Sensor C Code Initialization Example

// Temperature Sensor Initialization

```
PWRCTL0 &= 0xF3; /*Power On Temperature Sensor, Peripheral and ADC*/
ADCCTL1 = 0x81   /*Internal Reference set to 2.0v, Single-ended,
                   buffered input with unity gain */
ADCCTL0 = 0x9E   /*Enable Conversion, Internal Reference set to
                   2.0v, Reference Buffer disabled, temperature
                   sensor is input */
```

//End of Code

Further Reference

See the Temperature Sensor section of the Z8 Encore! XP® F082A Series Product Specifications (PS0228) for detailed operation of the Temperature Sensor.

Comparator

The Comparator is a general-purpose comparator that compares two analog input signals. These analog signals may be external stimulus from a pin (CINP and/or CINN) or internally generated signals. Both a programmable voltage reference and the temperature sensor output voltage are available internally for comparison. The output

of the Comparator is available as an interrupt source (to Interrupt Controller) or can be routed to an external pin (COUT).

Comparator Assembly Code Initialization Example

; Comparator Initialization

```
ANDX PWRCT0, #%FD     ;Power up Comparator
LDX CMP0, #%52        ;CINP used as positive comparator input,
                      ;internal reference enabled as negative
                        ;comparator Input, internal reference voltage
                        ;level set to 0.90v.
```

;End of Code

Note: *To configure PA5 as Comparator Input (CINP), please refer to* GPIO Assembly Code Initialization Example.

Comparator C Code Initialization Example

//Comparator Initialization

```
PWRCT0 &= 0xFD; /* Power up Comparator*/
CMP0 = 0x52;    /* CINP used as positive comparator input,internal
                   reference enabled as negative comparator Input,
                   internal reference voltage level set to 0.90v.*/
```

//End of Code

Note: *To configure PA5 as Comparator Input (CINP), please refer to* GPIO C Code Initialization Example

Further Reference

For detailed operation of the Comparator and the definitions of the Comparator Control Register, please refer to the Comparator section of either the Z8 Encore! XP® F082A Series Product Specifications (PS0228) or Z8 Encore! XP® F0823 Series Product Specifications (PS0243) as applicable.

Watchdog Timer

The Watchdog Timer (WDT) protects the device against corrupt or unreliable software, power faults, and other system-level problems which may place the Z8 Encore! XP® 8K and 4K Series device into unsuitable operating states. The WDT uses a dedicated on-chip RC oscillator as its clock source and generates an interrupt or system reset signal when the countdown reaches its terminal value of 000000 Hex.

WDT Assembly Code Initialization Example

;WDT Initialization for 7456.5 msec WDT Timeout

```
LDX WDTCTL, #%55  ;write 55 Hex for unlock sequence
LDX WDTCTL, #%AA  ;write AA Hex for unlock sequence
LDX WDTU, #%01    ;write WDT Reload Upper Byte Register
LDX WDTH, #%23    ;write WDT Reload High Byte Register
LDX WDTL, #%45    ;write WDT Reload Low Byte Register
```

;End of Code

WDT C Code Initialization Example

// WDT Initialization for 7456.5 msec WDT Timeout

```
WDTCTL = 0x55; //write 55 Hex for unlock sequence
WDTCTL = 0xAA; //write AA Hex for unlock sequence
WDTU = 0x01;   //write WDT Reload Upper Byte Register
WDTH = 0x23;  //write WDT Reload High Byte Register
WDTL = 0x45;  //write WDT Reload Low Byte Register
```

//End of Code

Further Reference

For detailed operation of the WDT and definitions of the WDT Control Registers, please refer to the Watchdog Timer section of either the Z8 Encore! XP® F082A Series Product Specifications (PS0228) or Z8 Encore! XP® F0823 Series Product Specifications (PS0243) as applicable.

Summary

This application note provides a clean example of how programmers can set up and initialize the Z8 Encore! XP® 8-pin device in both the C and Assembly language environments. These are only simple examples and can be embellished upon depending on the specific application needs. This application note is meant as a basis to understand how the initialization process works for the stack, system clock, GPIOs, and some of the on-chip peripherals like the UART, Timer1, ADC, Temperature Sensor (applicable to Z8 Encore! XP® F082A Series only), WDT, and the Comparator.

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