Application Note

ZMOTION™ Detection Module Application Walkthrough

AN030704-1110

Abstract

This application note describes in detail the application code associated with the ZMOTION Detection Module and how it can be used for detection and control applications. The ZMOTION Detection Module can be used either in Hardware Mode or Serial Mode. In the Hardware Mode, you use the hardware circuit selection for sensitivity adjustments, delay time settings, ambient light settings, and Sleep Mode. In the Serial Mode, you can use the UART interface to fine-tune the ZMOTION Detection Module with advanced configuration settings.

Note: The application code (AN0307-SC01) associated with this application note has been tested with ZDS II—Z8 Encore! version 4.11.0.

ZMOTION Overview

Zilog's ZMOTION Detection Module is a complete, compact, and easy to interface solution for motion detection and direction. It is designed using advanced passive infrared technology and Zilog's Z8FS04 Motion Detection MCU with a powerful embedded software engine that delivers high-performance motion detection. Figure 1 on page 2 shows the location of the pins on the ZMOTION Detection Module. Table 1 on page 2 describes the pin functions.

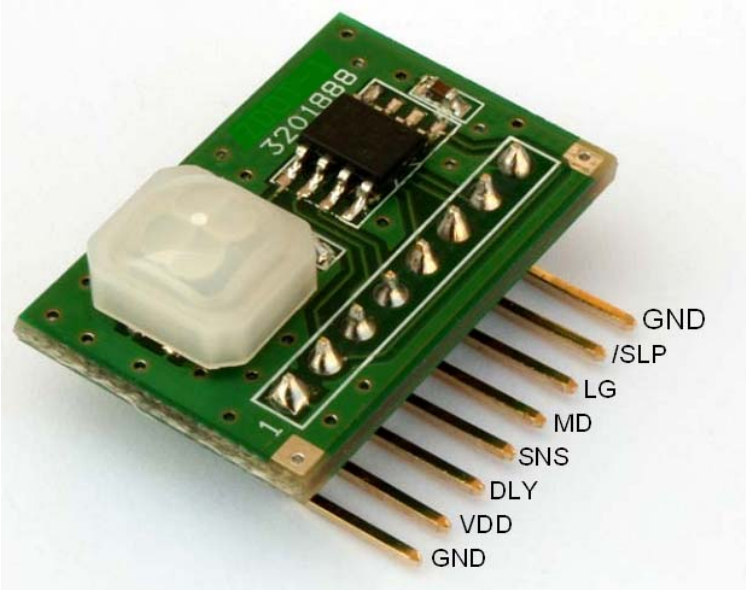

Figure 1. ZMOTION Detection Module Pin Diagram

Operation Modes

There are two operation modes: Hardware Interface Mode and Serial Interface Mode.

In the Hardware Interface Mode, you can make the following adjustments:
- Use the hardware interface pins for basic configuration.
- Adjust the motion sensitivity with the voltage on the SNS pin.
- Adjust the time delay (Output Active Time) with the voltage on the DLY pin
- Set the optional ambient light input.
- Use the Sleep Mode to reduce power consumption.

In the Serial Interface Mode, you can use the serial interface (Rxd and Txd) for advanced configuration:
- 9600 baud rate
- No parity
- 8 data bits
- 1 stop bit
- No flow control

The /MD, LG, and SLP pins remain functional.

Table 1. ZMOTION Detection Module Pin Description

Pin No.	Signal Name	Hardware Interface Mode	Serial Interface Mode	Description
1	GND	Ground	Ground	—
2	VDD	Supply Voltage	Supply Voltage	—
3	RXD/DLY	DLY-Delay (analog input)	RXD Receive Data (digital input)	—
4	TXD/SNS	SNS Sensitivity (analog input)	TXD Transmit Data (digital input)	Mode select during Reset

Pin No.	Signal Name	Hardware Interface Mode	Serial Interface Mode	Description
5	/MD/RST	Motion Detect (digital output)	Configurable: /RST Reset (digital input) /MD Motion Detect (digital input)	Default is /RST (Reset) in Serial Interface Mode
6	LG	Light Gate (analog input)	Light Gate (analog input)	If unused, connect to V_{dd}.
7	/SLP/DBG	/SLP Sleep (digital input)	/SLP Sleep (digital input)	DBG is used for programming and debug.
8	GND	Ground	Ground	—

Hardware Interface Mode Description

Figure 2 shows a typical example of how to connect the ZMOTION Detection Module using Hardware Interface Mode. This mode of operation is selected when a voltage between 0 V and 1.8 V is presented to the SNS pin during power ON (or after a reset caused by Vbo). When the Hardware Interface Mode has been established, this pin becomes the Sensitivity input and accepts a voltage between 0 V and 1.8 V as reference for motion detection sensitivity level.

- 0 V = Highest Sensitivity
- 1.8 V = Lowest Sensitivity

In Figure 2, R2 and R4 form a simple potentiometer resistor divider to ensure that the Hardware Mode is entered upon reset and to control sensitivity levels.

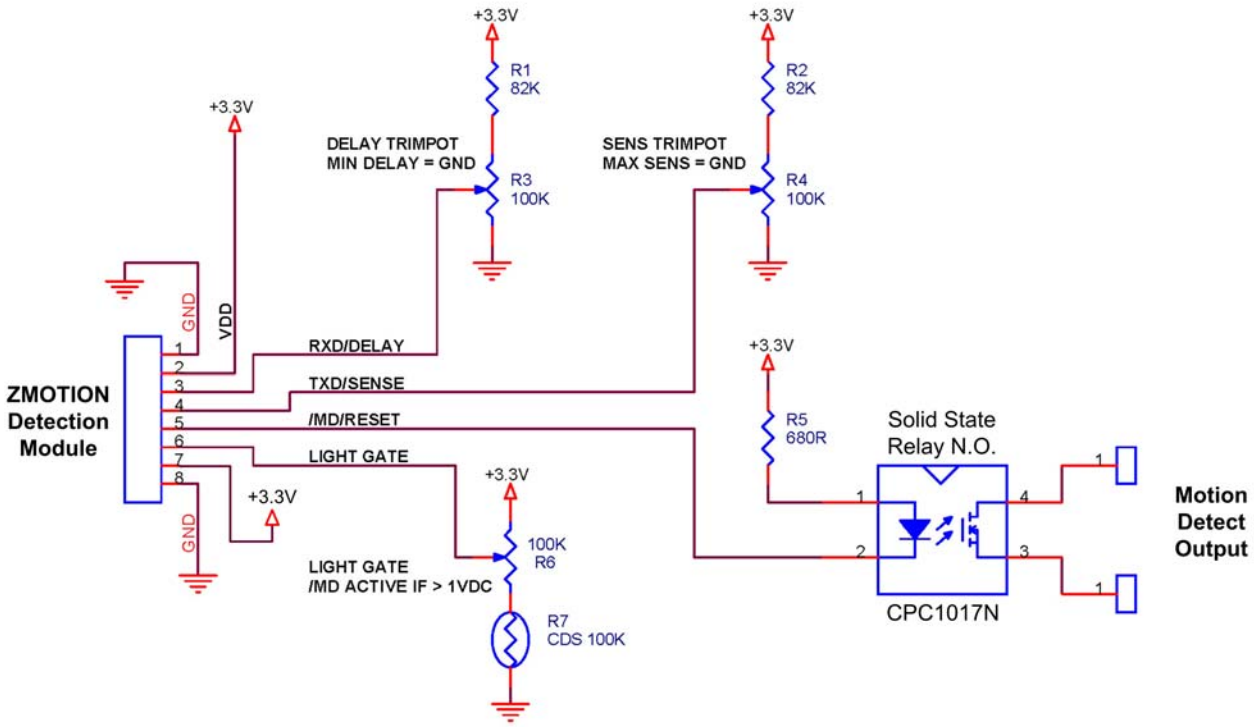

Figure 2. Application Example of Hardware Interface Mode

After the application of power, the passive infrared sensor needs some time to stabilize. This typically takes about 20 seconds but varies depending on environmental conditions. The software dynamically monitors the pyroelectric sensor during power-up and begins detecting motion as soon as the sensor is stable.

The /MD (Motion Detect) pin is driven active (Low) when motion is detected. The voltage on the delay pin (DLY) is used to determine the active duration of the /MD signal. This can be easily set using a resistor divider circuit shown in Figure 2 on page 4. With an 82-kΩ (R1) resistor tied to Vcc and values ranging from 0 Ω to 100 kΩ (R3) tied to ground, you can select delay times from 2 seconds to 15 minutes. See Figure 2 on page 4 and Table 15 on page 21.

Table 2. Delay Times Using R1 = 82 KΩ

Delay Time	R_DLY Voltage	R_DLY Standard Resistor Value
2 sec	0 V	0 Ω
5 sec	0.2 V	5.1 kΩ
10 sec	0.4 V	11 kΩ
30 sec	0.6 V	18 kΩ
1 min	0.8 V	24 kΩ
2 min	1.0 V	33.2 kΩ
3 min	1.2 V	43 kΩ
5 min	1.4 V	56 kΩ
10 min	1.6 V	68 kΩ
15 min	1.8 V	91 kΩ

The Light Gate (LG) signal acts as a disable (gate) for the /MD output signal. In a typical application, this signal is a representation of the ambient light in the environment. If there is light detected, the /MD signal does not activate even in the presence of motion.

Serial Interface Mode Commands

The Serial Interface Mode operates as a host-client relationship where the ZMOTION Detection Module is the client. Commands are sent from the host, and the module responds with the requested information or confirmation. The only exception is when the module is configured for "/MD Unsolicited" operation. Under this condition, it will send motion-detected information without first receiving a command from the host.

All the serial commands sent to the ZMOTION Detection Module are in ASCII characters format, but the data sent to and from the module can be ASCII or decimal.

There are three types of commands accepted by the module:

- Read commands
- Write commands
- Confirmation commands

Read Command Structure

Read commands are used to request information from the module. Read commands are sent from the host, and the module responds with the requested data. See Figure 3 on page 6.

- All read commands are initiated by single lower-case letters.
- When received, the device will return the applicable value as described in

Table 16 on page 22.

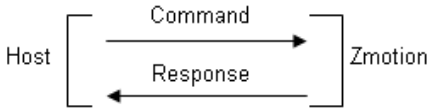

Figure 3. Read Command Structure

Write Command Structure

Write commands are used to update configuration of the module. The command is sent from the host, and the module responds with the current value as an acknowledgment. Then the host sends the new data, and the module responds with an 'ACK'. See Figure 4.

- All write commands are initiated by single upper-case letters.
- When a write command is received, the device returns the current value and expects an appropriate single-byte data value.
- When the data value is received, the device returns an 'ACK'. If no data is received after the inactivity timeout of 2.5 seconds, the device returns a 'NACK'.

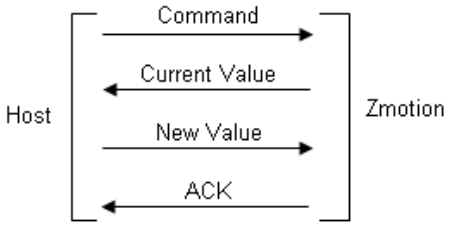

Figure 4. Write Command Structure

Confirmed Command Structure

Certain commands require a specific sequence of characters to be sent in order to help prevent accidental initiation. These commands require a 4-character confirmation sequence. When a command requiring confirmation is received, the device returns an 'ACK' . See Figure 5.

- If the sequence is correct, the device returns an 'ACK' and executes the command.
- If the sequence is incorrect, or there is an inactivity delay of more than 2.5 seconds between any characters of the sequence, the device immediately sends a 'NACK' and does not execute the command.

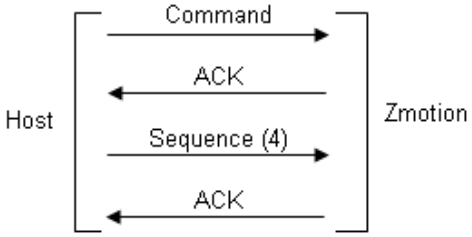

Figure 5. Confirmed Command Structure

Notes: 1. ACK = 0x06 (ASCII ACK character).
2. NACK = 0x15 (ASCII NACK character). The ZMOTION will respond with a 'NACK' on all

unrecognized commands or when a command requires data (that is, Write, Clear, and Confirmation types) but does not receive the required data within the inactivity timeout period.

Software Overview

The application code in the ZMOTION Detection Module first executes an initialization procedure, which is discussed in the "Setting the Operational Mode" section on page 8. When the module is enabled, the ADC interrupt runs in the background (see Figure 6). Every ADC conversion generates an interrupt and the ZMOTION engine performs its functions during this time (see Figure 6). The user application code runs in the foreground and monitors the status through the API and performs any other functions required for the application, which is discussed in the "Main Application Loop" section on page 9.

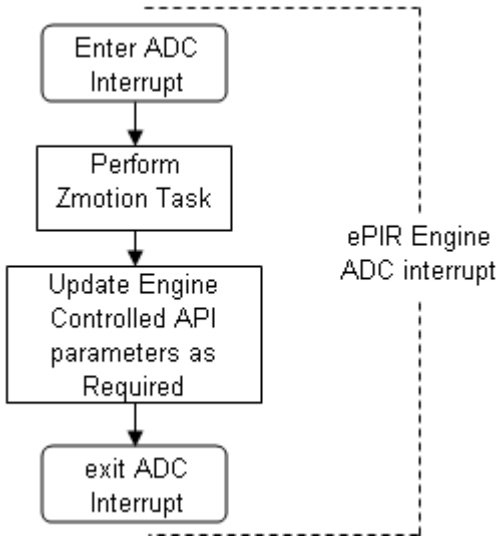

Figure 6. ADC Interrrupt for ZMOTION Detection

One-Second Timer Tick

The ZMOTION engine requires a 1-second time base to perform various housekeeping operations (see Figure 7 on page 8). This is handled in the application by the timer interrupt. The Timer0 Interrupt is configured for 100 ms. The application code counts 10 of these interrupts and sets the Engine Timer Tick bit in ePIR_SC1.

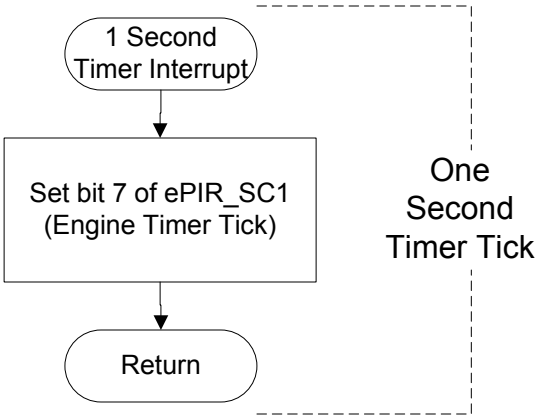

Figure 7. One-Second Engine Timer Tick

Setting the Operational Mode

The mode of operation (HW or Serial mode) can be selected using the TXD/SNS pin during the power ON or when exiting the Sleep Mode. The following explains the program in the operational mode (see Figure 8 on page 9):

1. The program starts by initializing all of the needed peripherals.
2. Check the Stop Mode Recovery SMR reset. If there is NO SMR reset, then the sensitivity level is set to default and the ePIR status register is initialized.
3. The ePIR_ENABLE register is set to EPIR_ENABLE_PATTERN, so the engine can be enabled.
4. The EPIR_INIT macro is executed to initialize the ePIR engine.
5. Initialize Timer0 and set to 100-ms interrupt.
6. Check analog channel 0 (ANA0), which is connected to PIN4 or TXD/SNS if the analog level exceeds the threshold or not. This is repeated 10 times to ensure a stable reading. If the TXD/SNS pin is above the threshold, then the cCmdState is set to 1; else, it is set to 0. This is done to stabilize the ePIR engine.
7. Next, the mode of operation is determined with the value of the cCmdState. If the cCmdState is 1, then the Mode of Operation is Serial Interface, and if the cCmdState is 0, then the Mode of Operation is Hardware Interface.
8. Finally, the mode of operation is set, and cCmdState is set to IDLE state; it then proceeds to the main application loop.

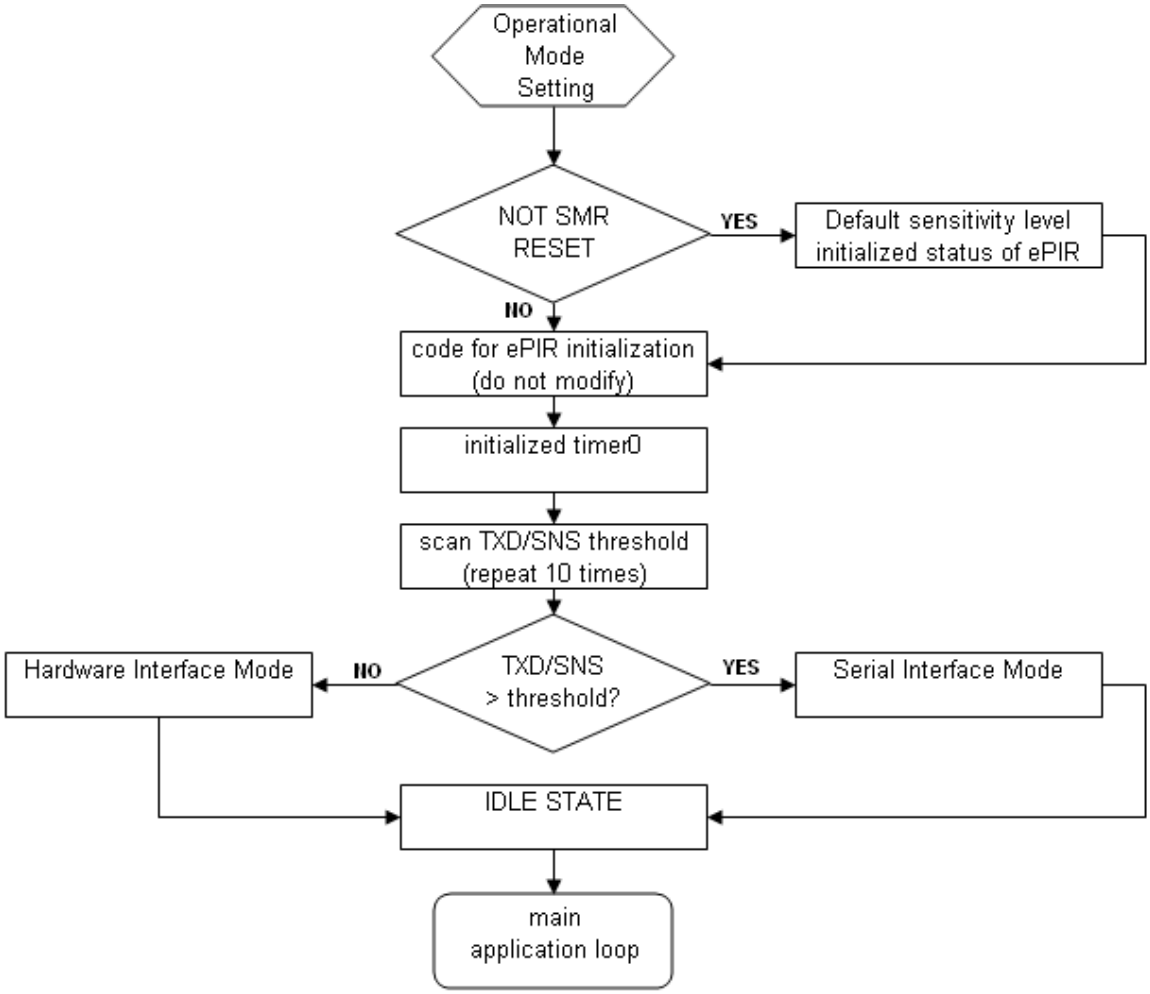

Figure 8. Flow Diagram of Operational Mode

Main Application Loop

Figure 9 on page 10 describes the main application loop of the ZMOTION Detection Module.

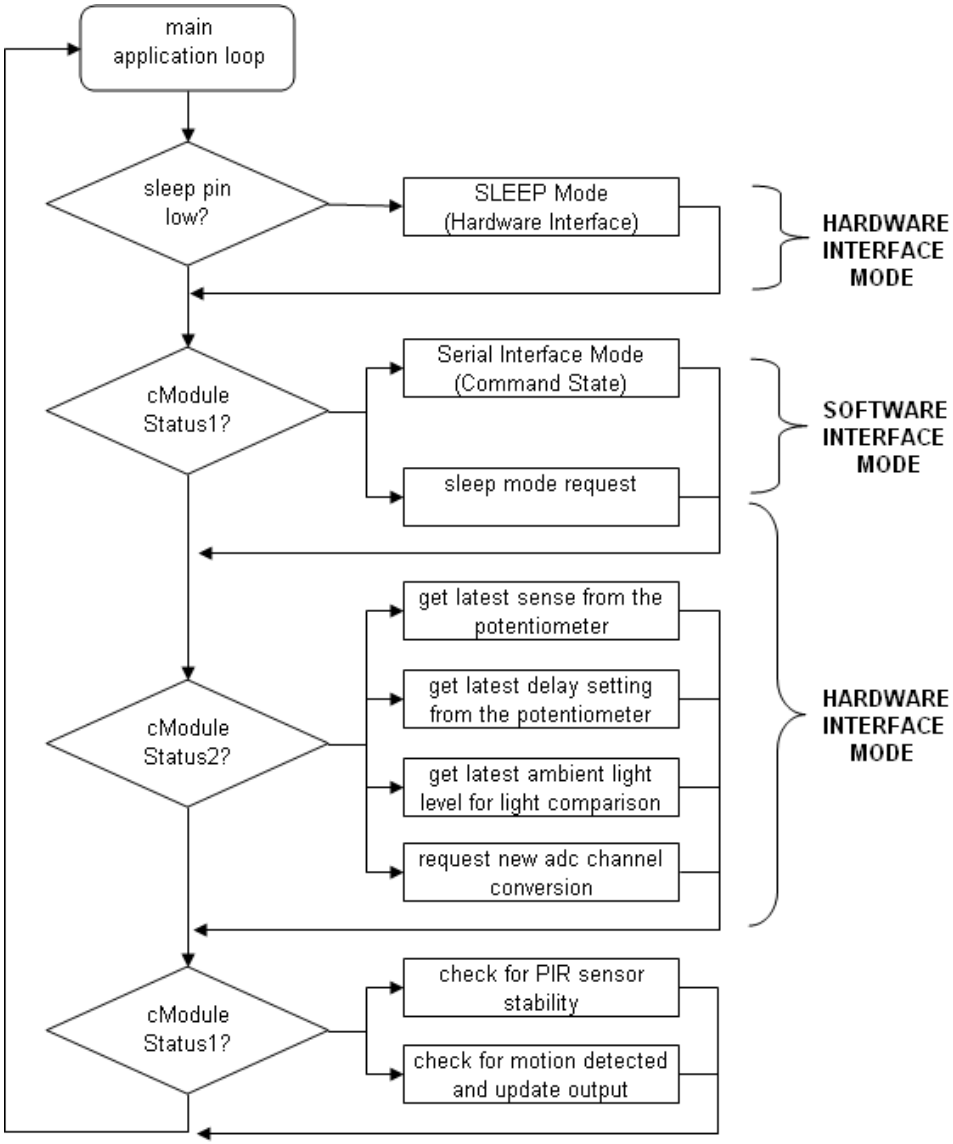

Figure 9. Flow Diagram of Main Application Loop of Application Code

The following describes the operation of the main application loop:

1. The program checks the Sleep pin if it is low. If the Sleep pin is low, then the program enters to the Sleep Mode.
2. Next the program checks the content of the cModule Status1 (cModuleStatus1) flag. See Table 20 on page 24.
   - If the Serial Mode Register (SER_MODE_ENABLED) is HIGH, several select cases are available depending on the inputs of the select case. See

Table 17 on page 23. The contents of the cModuleStatus1 flag are evaluated and updated.
- If the Sleep Mode Request (MOD_SLEEP_REQ) is HIGH, then the MCU Sleep Flag will be on, and the MCU will go to sleep. This is to reduce the power consumption of the MCU especially for battery-powered devices.

3. The program checks the state of the cModule Status2 (cModuleStatus2) flag. See Table 21 on page 24.
   - If the ADC current channel scan (SCAN_SENS_POT) is HIGH and the conversion for Sense potentiometer is complete, then the program gets the latest Sense value from the potentiometer.
   - If the Scan delay pot (SCAN_DELAY_POT) is HIGH and the conversion for Delay potentiometer is complete, then the program gets the latest Delay setting from the potentiometer
   - If the Scan Light Gate (SCAN_LIGHT_GATE) is HIGH and the conversion for the Light Gate is complete, then the program gets the latest ambient light level for light comparison.
   - If the Scan Request (SCAN_REQUEST_NEW) new is high, then the program will request new ADC channel conversion.
4. Finally, the program checks again the status of the cModule Status1 (cModuleStatus1) flag. See Table 20 on page 24.
   - If the Mode Pir Stable (MOD_PIR_STABLE) is LOW, then the program checks the PIR sensor stability after power up.
   - If the Mode Pir Stable (MOD_PIR_STABLE) is HIGH, the engine will indicate an alarm and checks for motion detected and updates the output.

Sleep Mode in Hardware Interface Mode

Sleep signal can be used to put the device into a low-power mode. Using this feature will allow a shorter PIR stabilization time.

If the Sleep (/SLP) input signal is driven LOW, the device enters into a low-power Sleep Mode. It can be awakened by either deactivating the /SLP signal (driving the signal HIGH) or sending a character over the serial interface; the received character is received and processed.

Serial Interface Mode (Software Interface Mode)

The Serial Interface Mode is responsible for handling serial command input and processing. See Table 3.

Table 3. Command State (cCmdState)

Command State (cCmdState)	Description
Idle State	This state puts the ZMOTION into standby mode
Real Time State	This state puts the program in real-time mode. This state is enabled if the received command is “Y” and disabled if the received command is “N”.
Suspend Status State	This state temporarily enables or disables the motion detection of the program. This state is enabled if the received command is “Y” and disabled when the receive command is “N”.
Serial Interface Mode State	This state let the user change the mode of sending the data format of the ZMOTION to either ASCII or Decimal. The ASCII mode is enabled if the command received is “A” and Decimal Mode if the command received is “D”.
Sensitivity Change State	This state changes the sensitivity of the ePIR engine to the target motion. Sensitivity is higher with a lower number value. 0x00 being the most sensitive and 0xFF being the least sensitive.

Command State (cCmdState)	Description
Sleep Duration State	This state changes the time duration of Sleep Mode.
Delay Change State	This state changes the time delay of changing the state from one state to another.
Light Threshold Change State	This state is used to control and monitor the signal associated with the Light Gate pin.
MD Output Change State	This state indicates how the ePIR engine detected the last motion-detected event. When the ZMOTION sets the Motion Detected bit in ePIRStatus0, it also sets this bit according to which detection engine registered the event.
Configuration Change	This state is used to configure the motion detection. If the received command is 'R', then it is requesting for a Reset. If the receive command is 'M', then it is requesting for make motion detection.
Direction Change State	This state is used to configure the direction of the motion to positive, negative, or disabled. Positive movement is requested when a "+" is received, negative movement is requested when a "-" is received, and disabled motion detection is requested when "A" is received.
Dual Direction Change State	This state determines if the engine should accept signals from one or two ePIR sensor. If configured as single operation then only one sensor is used which is connected to ANA2. If it is configured as dual operation then ZMOTION will scan two sensors simultaneously with the second sensor connected to ANA3.
Hyper-sense Change State	This state changes the sensitivity of the ZMOTION. If this state is enabled, the engine considers smaller signal changes as valid motion events. This significantly increases sensitivity at the cost of more potential false motion detections.
Ping Write Request State	This state is used to clear and temporarily save the received command.
Frequency Response State	This state determines the frequency response of the motion detection system. Higher values allow lower frequencies to be accepted by the ePIR engine. Lower values cause the engine to ignore targets that generate lower frequencies.
Range Control State	This state determines the relative range of motion detection. Larger values decrease the range of detection.
Reset Mode State	This state uses the watch-dog timer to have system reset.
Sleep Mode State	This state puts the program in Sleep Mode. This state uses the watch-dog timer to have system reset that wakes up MCU.

Sleep Mode Request in Serial and Hardware Interface Mode

The program enters Sleep Mode request if MOD_SLEEP_REQ in cModuleStatus1 register is enabled and sleeps for the given time duration depending on the value of the set sleep duration.

Sensitivity Setting from Potentiometer (Hardware Interface Mode)

Get the latest Sense setting from the Pot. The program gets the latest sense from the potentiometer if the SCAN_SENS_POT is enabled in cModuleStatus2 register and the SC3_ANA0_SCAN_REQUEST is disabled. Then the ePIR_sensitivity is updated.

Delay Setting from Potentiometer (Hardware Interface Mode)

Get the latest Delay setting from the Pot. The program checks if the SCAN_DELAY_POT is enabled in cModuleStatus2 register and then updates the cDelayTime.

Ambient Light Level for Gate comparison (Hardware Interface Mode)

Get the latest Ambient light level for Light Gate comparison. The program checks if the SCAN_LIGHT_GATE is enabled in cModuleStatus2 register and the updates the value of cLGAmbient.

ADC Conversion and Request Next Channel Conversion (Hardware Interface Mode)

Request new ADC channel conversion. The program checks if the SCAN_REQUEST_NEW is enabled in cModuleStatus2 register and then the cModuleStatus2 register is updated.

Sensor Stability

Check for PIR sensor stability. The program checks the stability of the system by verifying if the MOD_PIR_STABLE is enabled in cModuleStatus1 register after power up.

Motion Detection

Check for motion detected and update the output on time as needed. The program checks if motion is detected by checking if the SCO_MOTION_DETECTED is enabled in ePIR_SCO register then send it once it is needed.

References

The following documents are associated with ZMOTION™ Detection Module and are available on www.zilog.com:

- ePIR Motion Detection Zdots Single Board Computer Product Brief (PB0223)
- ePIR Motion Detection Zdots SBC Product Specification (PS0284)
- ZMOTION Detection Module Evaluation Kit User Manual (UM0223)
- ZMOTION Detection Module Development Kit Quick Start Guide (QS0073)
- Motion Detection and Control with ePIR Zdots Single Board Computer Cut Sheet (CS0005)
- ZMOTION—A New PIR Motion Detection Architecture White Paper (WP0017)
- Power Management and Customer Sensing with Zilog's ZMOTION Detection Module Application Note (AN0301)
- ZMOTION Detection and Control Family Featuring ePIR Technology Product Specification (PS0285)
- ZMOTION Lens and Pyroelectric Sensor Product Specification (PS0286)
- Z8 Encore! XP F0822 Series Product Specification (PS0225)
- Z8 Encore! XP F082A Series Product Specification (PS0228)

Appendix A—Pre-Build Setup

This appendix lists the settings and resources you need for your build.

Use the following Compiler and Linker settings to prepare for your build:

ZDS Version	ZDSII Encore! 4.11.0
CPU Family	Z8Encore_XP_F082A_8Pin_Series
CPU	Z8F042AXB
Limit Optimizations for Easier Debugging	unchecked
Memory Model	Small
Frames	Static
Parameter Passing	Memory
Use Register Variables	Aggressive
Generate printf Inline	unchecked
Bit-Field Packing	Backward Compatible
Place CONST Variables in ROM	unchecked
Disable ANSI Promotions	unchecked
Address Space	Default settings + Use PRAM checked

Table 4 lists the resources used by the build.

Table 4. Resources Used

Clock Source	Internal clock source, 5.52960 MHz	
Peripherals	TMR0	One-second timer used in low-power mode
	UART0	RS232 interface
	WDT	Software reset and watch-dog function
	PA0 or Debug, PA1, PA2 or RESET and PA4	Used for input and output pins.
	ADC – ANA2 to ANA3	These are used for the ePIR output for motion detection.

Appendix B—Library and Software Tool Files

This appendix describes the standard project files and external dependencies.

Figure 10 shows the block diagram of the ZMOTION application program.

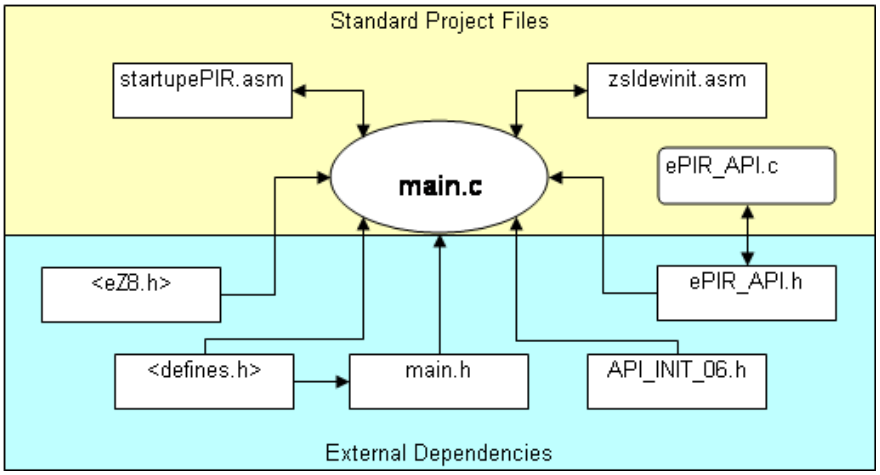

Figure 10. Block Diagram of the ZMOTION Application Project

Table 5 describes the standard project files.

Table 5. Standard Project files

File	Description
`main.c`	• Handles the main program. • Includes several libraries such as `eZ8.h`, `defines.h`, `main.h`, `ePIR_API.h`, and `API_INIT_06.h` to provide support to the functions and subprograms.
`startupePIR.asm`	Used to set the Linker address spaces in project settings.
`Zsldevinit.asm`	Used to set the ZSL in the project settings to include the Zilog Standard Library (Peripheral Support).
`ePIR_API.c`	Contains the ePIR standard and advanced API registers WARNING: Do not modify or remove the given values in this file.

Table 6 lists the external dependencies.

Table 6. External Dependencies

File	Description
eZ8.h	Contains the standard Z8 library.
defines.h	Define the names for control register access.
main.h	Contains project header that includes: • Predefined application compile switches • Peripheral configuration macro definitions • Reserve interrupt vector • Application definitions • Serial interface Macro definitions • Application configuration Macro definitions • Flags for cModuleStatus1, cModuleStatus2, and cModuleStatus3 • Delay pot voltage thresholds (Hardware Interface Mode) • MD Delay values (Hardware Interface Mode) • Serial Command List • Serial Command States • Motion Alarm output macro • Inline assembly macro
API_INIT_06.h	• Handles the functions and subprograms declared in ePIR_API.c. • Contains the API configuration for Normal Scan and Low Scan Rate Mode. WARNING: Do not modify or remove the given values in this file.
ePIR_API.h	Defines the standard and advanced API interface. WARNING: Do not modify or remove the given values in this file.

Appendix C—main.c

This appendix describes the function prototypes in `main.c`.

The following is a list of the function prototypes in `main.c`:

Function Name	makeMDReset
Returns	-none-
Parameters	-none-
Description	Routine for making the /MD pin a reset input

Function Name	makeMDMotion
Returns	-none-
Parameters	-none-
Description	Configure /MD pin as Motion Detected.

Function Name	checkCommand
Returns	-none-
Parameters	-none-
Description	Check the newly received command.

Function Name	SerialReceive
Returns	unsigned char intData
Parameters	-none-
Description	Character received through serial communication

Function Name	cSerialRX
Returns	unsigned char rxData
Parameters	-none-
Description	Copy the character in the serial buffer

Function Name	cSerialTX
Returns	-none-
Parameters	unsigned char rxData
Description	Transmit the character in the serial buffer

Function Name	TxDirect
Returns	-none-
Parameters	unsigned char rxData
Description	Transmit byte via UART0

The following is a list of the interrupt function prototypes in `main.c`:

Function Name	interrupt isrADC_EOC
Returns	-none-
Parameters	-none-
Description	ADC for End of Character (EOC) interrupt

Function Name	interrupt isrTimer0
Returns	-none-
Parameters	-none-
Description	TIMER0 timeout interrupt

Function Name	interrupt isrRX0
Returns	-none-
Parameters	-none-
Description	UART0 RX interrupt

Appendix D—ePIR_API.c

This appendix describes the contents of `ePIR_API.c`.

Table 7. Register Address of Standard API Interface (x100 to 0x10F)

Address	Size	Name	Description
100h	1 byte	ePIR_Enable	ePIR API Enable Register
101h	1 byte	ePIR_Sensitivity	ePIR API Sensitivity Register
102h	1 byte	ePIR_SC0	ePIR API Status and Control Reg. 0
103h	1 byte	ePIR_SC1	ePIR API Status and Control Reg. 1
104h	1 byte	ePIR_SC2	ePIR API Status and Control Reg. 2
105h	1 byte	ePIR_SC1	ePIR API Status and Control Reg. 3
106h	1 byte	ePIR_Reserved_106	Reserve Standard API Register
107h	1 byte	ePIR_Reserved_107	Reserve Standard API Register
108h	1 byte	ePIR_Reserved_108	Reserve Standard API Register
109h	1 byte	ePIR_Reserved_109	Reserve Standard API Register
10Ah	2bytes	ePIR_ADC_Result	ePIR ADC result Register
10Ch	1 byte	ePIR_Version	ePIR Engine S/W Version Register
10Dh	1 byte	ePIR_Reserved_10D	Reserve Standard API Register
10Eh	1 byte	ePIR_Reserved_10E	Reserve Standard API Register
10Fh	1 byte	ePIR_Reserved_10F	Reserve Standard API Register

Table 8. Register Address of Advanced API Interface (0xF0 to 0xFF)

Address	Size	Name	Description
F0h	1 byte	ePIR_ASC0	Advanced API Status\Control Reg.0
F1h	1 byte	ePIR_Reserved_F1	Reserved Advanced API Register
F2h	1 byte	ePIR_ASC2	Advanced API Status\Control Reg.2
F3h	2bytes	ePIR_Process_Rate	ePIR Process Rate Register
F5h	1 byte	ePIR_Sample_Size	ePIR Sample Size Register
F6h	1 byte	ePIR_Debounce	ePIR Debounce Time Register
F7h	1 byte	ePIR_Debounce_Batch	ePIR Debounce Batch Size Register
F8h	1 byte	ePIR_Transient_Sense	ePIR Transient Sensitivity Reg.
F9h	1 byte	ePIR_Noise_Sense	ePIR Noise Sensitivity Register
FAh	2bytes	ePIR_Signal	ePIR PIR Signal Register
FCh	2bytes	ePIR_Signal_DC	ePIR PIR Signal DC Level Register
FEh	1 byte	ePIR_Reserved_FE	Reserved Advanced API Register
FFh	1 byte	ePIR_Reserved_FF	Reserved Advanced API Register

Appendix E—main.h

This appendix describes the contents of `main.h`.

Table 9. Predefined Application Compile Switches

Define	Value	Description
ZDOT_APP_ID	BEh	Identification for user to uniquely identify each released version of the application

Table 10. Predefined Peripheral Configuration to Macro Definitions

Define	Value	Description
U0_BAUD_HIGH	00h	UART 0 BRG high setting for 9600 Baud with IPO
U0_BAUD_LOW	24h	UART 0 BRG low setting for 9600 Baud with IPO
U0CTL0_VAL	C0h	UART 0 Control Register 0 value
U0CTL1_VAL	00h	UART 0 Control Register 1 value
T0RH_VAL	87h	Timer 0 reload value high when set for 5.5MHz standard clock
T0RL_VAL	02h	Timer 0 reload value low when set for 5.5MHz standard clock
T0CTL0_VAL	00h	Timer 0 Control 0 Register value
T0CTL1_VAL	E1h	Timer 0 Control 1 Register value
WDTU_VAL	00h	WDT Upper byte, ~1000 ms
WDTH_VAL	27h	WDT High byte, ~1000 ms
WDTL_VAL	10h	WDT Lower byte, ~1000 ms
OSCCTL_VAL	A0h	Oscillator control
PWRCTL0_VAL	8Ah	Power control 0 register value

Table 11. Predefined Application Definitions with Initial Default Values

Define	Value	Description
SENSE_ADJUST_HW_MIN	10h	Sensitivity adjustment, lower is more sensitive
SENSE_ADJUST_HW_MAX	FFh	
LG_THRESHOLD	64h	Light Gate threshold POR Default value
LOW_FREQUENCY_RESPONSE	28h	Default Frequency Response value
HIGH_FREQUENCY_RESPONSE	00h	High Frequency Response value
SER_IDLE_TIMEOUT	19h	Serial Interface Mode Inactivity timeout (x 100ms)

Table 12. Serial Interface Macro Definitions

Define	Function	Description
SER_ACK	TxDirect(0x06)	Serial Acknowledge
SER_NACK	TxDirect(0x15)	Serial Not Acknowledge
SER_AWAKE	TxDirect(0x16)	Serial Awake

Table 13. Application Configuration Macro Definitions

Define	Value	Description
MOD_STAT1_DEF	08h	cModuleStatus1 register POR Default value
MOD_STAT2_DEF	00h	cModuleStatus2 register POR Default value
MOD_STAT3_DEF	08h	cModuleStatus3 register POR Default value

Table 14. Delay Pot Voltage Threshold (Hardware Interface Mode)

Define	Value	Description
DLY_POT_000MV	10	0 mV
DLY_POT_200MV	31	200 mV
DLY_POT_400MV	51	400 mV
DLY_POT_600MV	71	600 mV
DLY_POT_800MV	92	800 mV
DLY_POT_1000MV	112	1000 mV
DLY_POT_1200MV	132	1200 mV
DLY_POT_1400MV	153	1400 mV
DLY_POT_1600MV	173	1600 mV

Table 15. Delay Values (Hardware Interface Mode)

Define	Value	Description
DELAY_2SEC	2	Delay the program by 2 sec
DELAY_5SEC	5	Delay the program by 5 sec
DELAY_10SEC	10	Delay the program by 10 sec
DELAY_30SEC	30	Delay the program by 30 sec
DELAY_60SEC	60	Delay the program by 60 sec
DELAY_2MIN	120	Delay the program by 2 minutes
DELAY_3MIN	0x80+3	Delay the program by 3 minutes
DELAY_5MIN	0x80+5	Delay the program by 5 minutes
DELAY_10MIN	0x80+10	Delay the program by 10 minutes
DELAY_15MIN	0x80+15	Delay the program by 15 minutes

Table 16. Serial Commands (Serial Interface Mode)

Define	Value	Description
CMD_MD_STATUS_READ	'a'	Read Motion Status
CMD_LG_AMBIENT_READ	'b'	Read Current Light Gate Input Level
CMD_MD_CONFIG_STATUS_READ	'c'	Read /MD//RST Pin Configuration
CMD_MD_CONFIG_STATUS_WRITE	'C'	Write /MD//RST Pin Configuration
CMD_DELAY_TIME_READ	'd'	Read /MD Activation Time
CMD_DELAY_TIME_WRITE	'D'	Write /MD Activation Time
CMD_HYPERSENSE_READ	'e'	Read Hyper Sense Setting
CMD_HYPERSENSE_WRITE	'E'	Write Hyper Sense Setting
CMD_FREQ_RESP_READ	'f'	Read Frequency Response Setting
CMD_FREQ_RESP_WRITE	'F'	Write Frequency Response Setting
CMD_MD_SUSPEND_READ	'h'	Read Motion Detection Suspend Setting
CMD_MD_SUSPEND_WRITE	'H'	Write Motion Detection Suspend Setting
CMD_VERSION_READ	'i'	Read Module S/W Version
CMD_SER_INTERFACE_READ	'k'	Read Serial Interface Command Mode
CMD_SER_INTERFACE_WRITE	'K'	Write Serial Interface Command Mode
CMD_LG_THRESH_READ	'l'	Read Light Gate Threshold
CMD_LG_THRESH_WRITE	'L'	Write Light Gate Threshold
CMD_REAL_TIME_MD_READ	'm'	Read Motion Detected Unsolicited Mode
CMD_REAL_TIME_MD_WRITE	'M'	Write Motion Detected Unsolicited Mode
CMD_MD_OUT_STATE_READ	'o'	Read /MD Current Output Active Time
CMD_MD_OUT_STATE_WRITE	'O'	Write /MD Output State
CMD_PING_READ	'p'	Read Pulse Count
CMD_PING_WRITE	'P'	Write Pulse Count
CMD_RANGE_CONTROL_READ	'r'	Read Range Setting
CMD_RANGE_CONTROL_WRITE	'R'	Write Range Setting
CMD_SENS_READ	's'	Read Sensitivity
CMD_SENS_WRITE	'S'	Write Sensitivity
CMD_DUAL_DIRECTION_READ	'u'	Read Dual Directional Mode
CMD_DUAL_DIRECTION_WRITE	'U'	Write Dual Directional Mode
CMD_DIRECTION_READ	'v'	Read Single Directional Mode
CMD_DIRECTION_WRITE	'V'	Write Single Directional Mode
CMD_RESET_REQUEST	'X'	Module Reset
CMD_SLEEP_TIME_READ	'y'	Read Sleep Time
CMD_SLEEP_TIME_WRITE	'Y'	Write Sleep Time
CMD_SLEEP_REQUEST	'Z'	Sleep Mode

Table 17. Serial Command States (Serial Interface Mode)

Define	Value	Description
ZDOT_IDLE	0	Idle State
ZDOT_SENS_SET	1	Handle the Sensitivity Change state
ZDOT_DELAY_SET	2	Handle the Delay Change state
ZDOT_2PULSE_SET	3	UNUSED
ZDOT_FREQ_SET	4	Handle the Freq. Response Change state
ZDOT_SLEEP_SET	5	Handle the Sleep Mode state
ZDOT_LG_THRESH_SET	6	Handle the Light Gate Threshold Change state
ZDOT_MD_STATE_SET	7	Handle the /MD Output State Change state
ZDOT_DIR_SET	8	Handle the Direction Change state
ZDOT_RESET_SET	9	Handle the Reset Request state
ZDOT_REAL_TIME_MD_SET	10	Handle the Real-Time MD Status Change state
ZDOT_MD_CONFIG_SET	11	Handle the /MD Config Change state
ZDOT_HYPER_SET	12	Handle the Hypersense Change state
ZDOT_MD_SUSPEND_SET	13	Handle the MD Suspend Status Change state
ZDOT_SLEEP_PIN_SET	14	UNUSED
ZDOT_SLEEP_TIME_SET	15	Handle the Sleep Duration Change state
ZDOT_PING_WRITE	16	Handle the Ping write request
ZDOT_RANGE_CONTROL_SET	17	Handle the Range Control Change state
ZDOT_DUAL_DIR_SET	18	Handle the Dual Direction Change state
ZDOT_SER_INTERFACE_SET	19	Handle the Serial Interface Mode state

Table 18. Motion Alarm Output Macro Definitions

Define	Value	Description
ALARM_PORT	PAOUT	Motion Alarm Port
ALARM_ON	FBh	Motion Alarm Activation mask (/MD is on PA2 - active low)
ALARM_OFF	~FBh	Motion Alarm Deactivation mask

Table 19. Inline Assembly Macros

Define	Function	Description
Z8_NOP	asm("NOP")	Do nothing
Z8_WDT	asm("WDT")	Refresh the watch-dog timer
Z8_HALT	asm("HALT")	Enter "Halt" mode
Z8_STOP	asm("STOP")	Enter "Stop" mode
Z8_ATM	asm("ATM")	DI for the next 3 assembly instructions

Table 20. Main Flag for cModuleStatus1 Register

Flag Name : cModuleStatus1
Data Size : UINT8
Bit Names :

b0	SER_BYTE_RX	1:Serial byte receive
b1	SER_MOTION_DETECTED	1:Motion detected copy for Serial mode
b2	SER_REAL_TIME_MD	1:Serial sends real-time Motion Detected Status
b3	SER_MD_IS_RESET	1:MD is reset in serial mode
b4	SER_BYTE_TX	1:Serial Byte to Transmit
b5	SER_PIR_STABLE	1:PIR sensor is stable after power-up
b6	SER_SLEEP_REQ	1:Sleep Mode request
b7	SER_MODE_ENABLE	1:Enable Serial Interface Mode

Table 21. Main Flag for cModuleStatus2 Register

Flag Name : cModuleStatus2
Data Size : UINT8
Bit Names :

b0	SCAN_SENSE_POT	1:Sensing Enable current channel scanned
b1	SCAN_DELAY_POT	1:Delay Enable current channel scanned
b2	MOD2_FFU04	0:RESERVE-Must be 0
b3	SCAN_LIGHT_GATE	1:Light Gate Enable current channel scanned
b4	MOD2_FFU10	0:RESERVE-Must be 0
b5	MOD2_FFU20	0:RESERVE-Must be 0
b6	SCAN_REQUEST_NEW	1:Enable Request new ADC channel scan
b7	REQUEST_SCAN	1:Enable Time to request a scan

Table 22. Main Flag for cModuleStatus3 Register

Flag Name : cModuleStatus3
Data Size : UINT8
Bit Names :

b0	DUAL_DIR_ENABLED	1:Dual Direction Mode Enable
b1	SINGLE_DIR_ENABLED	1:Single Direction Mode Enable
b2	POSITIVE_DIRECTION	1:Motion is positive direction
b3	LOW_FREQ	1:Low frequency response enabled
b4	ASCII_MODE_ON	1:Serial Interface Command Mode Enabled
b5	RXDATA_OK	1:Indicates that the data being returned by cSerialRx is valid
b6	GEN_ERROR_FLAG	1:General error flag passed between functions

b7	RESERVE	0: RESERVE-Must be 0

Appendix F—API_INIT_06.h

This appendix describes the contents of `API_INIT_06.h`.

Table 23. Serial Command States

Define	Value	Description
EPIR_SENSITIVITY_DEF	16	ePIR Sensitivity Register Default Configuration for Normal Scan Rate. The lower the value, the greater sensitivity.
EPIR_SC0_DEF	00h	ePIR Serial Command Register 0 Default Configuration for Normal Scan Rate Bit 6-7 - Extended Detection - Level 0 (00) Bit 5 - Engine Disabled - Engine Controlled (0) Bit 4 - MD Suspend - OFF (0) Bit 3 - Motion Direction Control – OFF (0) Bit 2 - Motion Direction - Engine Controlled (0) Bit 1 - Motion Detected - Engine Controlled (0) Bit 0 - PIR Stable - Engine Controlled (0)
EPIR_SC1_DEF	28h	ePIR Serial Command Register 1 Default Configuration for Normal Scan Rate Bit 7 - Engine Timer Tick Bits 6-3 - Frequency Response (0101) Bit 2 - PIR Scan Rate - Normal Mode (0) Bit 1 - Reserved (0) Bit 0 - Dual Pyro Enable – OFF (0)
EPIR_SC2_DEF	02h	ePIR Serial Command Register 2 Default Configuration for Normal Scan Rate. Lower values provide greater range. Bits 7-3 - Reserved (00000) Bits 2-0 – Range (010)
EPIR_SC3_DEF	00h	ePIR Serial Command Register 3 Default Configuration for Normal Scan Rate Bits 7-0 - ANAx Scan Request – None (00000000) No ADC Scan requests made during Init
EPIR_ASC0_DEF	00h	ePIR Analog Serial Command Register 0 Default Configuration for Normal Scan Rate Bits 7-5 - Reserved (000) Bit 4 - Buffer Refresh - OFF 0) Bit 3 - New Sample - Engine Controlled (0) Bit 2 - MD Origin - Engine Controlled (0) Bit 1 - EM Noise - Engine Controlled (0) Bit 0 - EM Transient - Engine Controlled (0)
EPIR_ASC2_DEF	5Ah	ePIR Analog Serial Command Register 2 Default Configuration for Normal Scan Rate Bits 7-5 - Lock Level - 2 (010) Bits 4-3 - Window Size - 3 (11) Bits 2-0 - Window Update Rate – 2 (010)
EPIR_SAMPLE_SIZE_DEF	20h	ePIR Sample Size Default Value for Normal Scan Rate ePIR Sample Size - 32 (00100000)
EPIR_DEBOUNCE_DEF	64h	ePIR Debounce Time Default Value for Normal Scan Rate ePIR Debounce Time - 0x64 (01100100)

Define	Value	Description
EPIR_DEBOUNCE_BATCH_DEF	FFh	ePIR Debounce_Batch Default Value for Normal Scan Rate ePIR Debounce Batch Size – 255 (11111111)
EPIR_TRANSIENT_SENSE_DEF	00h	ePIR Transient_Sense Default Value for Normal Scan Rate ePIR Transient Sensitivity – Disabled (00000000)
EPIR_NOISE_SENSE_DEF	00h	ePIR Noise Sense Default Value for Normal Scan Rate ePIR Noise Sensitivity – Disabled (00000000)

Appendix G—ePIR_API.h

This appendix describes the contents of `ePIR_API.h`.

Table 24. ePIR Enable Register (ePIR_Enable) Enable/Disable Patterns

Define	Value	Description
EPIR_DISABLE_PATTERN	00h	Disables all primary engine functions, including motion detection. Used to temporarily or permanently shut down the engine.
EPIR_ENABLE_PATTERN	11h	Enables the ePIR engine. All primary engine functions as configured in Engine Status/Control Registers are enabled. Confirmation of enabled status is provided through Engine Disabled bit in Status/Control Register 0.

Table 25. ePIR Sensitivity Register (ePIR_Sensitivity

Define	Value	Description
ePIR_SENSITIVITY	FFh	(R/W) Register used to adjust the sensitivity of the ePIR

Table 26. ePIR API Status/Control Register 0 (ePIR_SC0)

Define	Value	Description
SC0_PIR_STABLE	01h	(R) Status/Control Register 0 PIR Stable
SC0_MOTION_DETECTED	02h	(R/W) Status/Control Register 0 Motion Detected
SC0_MOTION_DIRECTION	04h	(R) Status/Control Register 0 Motion Direction
SC0_DIRECTION_CONTROL	08h	(R/W) Status/Control Register 0 Direction Control
SC0_MD_SUSPEND	10h	(R/W)Status/Control Register 0 Motion Detection Suspend
SC0_ENGINE_DISABLED	20h	(R) Status/Control Register 0 Engine Disabled
SC0_EXTENDED	C0h	(R/W) Status/Control Register 0 Extended

Table 27. ePIR API Status/Control Register 1 (ePIR_SC1)

Define	Value	Description
SC1_DUAL_PYRO_ENABLE	01h	(R/W) Status/Control Register 1 Dual Pyro Enable
SC1_BIT1_RESERVED	02h	(R) Status/Control Register 1 Bit1 Reserved
SC1_PIR_SCAN_RATE	04h	(R/W) Status/Control Register 1 PIR Scan Rate
SC1_FREQUENCY_RESPONSE	78h	(R/W) Status/Control Register 1 Frequency Response
SC1_ENGINE_TIMER_TICK	80h	(R/W) Status/Control Register 1 Engine Timer Tick

Table 28. ePIR API Status/Control Register 2 (ePIR_SC2)

Define	Value	Description
SC2_BIT34567_RESERVED	F8h	(Reserve) Status/Control Register 2 Bit34567 Reserved
SC2_RANGE_CONTROL	07h	(R/W) Status/Control Register 2 Range Control

Table 29. ePIR API Status/Control Register 3 (ePIR_SC3)

Define	Value	Description
SC3_ANA0_SCAN_REQUEST	01h	(R/W) Status/Control Register 3 Analog0 Scan Request
SC3_ANA1_SCAN_REQUEST	02h	(R/W) Status/Control Register 3 Analog1 Scan Request
SC3_BIT2_RESERVED	04h	(Reserve) Status/Control Register 3 BIT2 Reserve
SC3_ANA3_SCAN_REQUEST	08h	(R/W) Status/Control Register 3 Analog3 Scan Request
SC3_ANA4_SCAN_REQUEST	10h	(R/W) Status/Control Register 3 Analog4 Scan Request
SC3_ANA5_SCAN_REQUEST	20h	(R/W) Status/Control Register 3 Analog5 Scan Request
SC3_ANA6_SCAN_REQUEST	40h	(R/W) Status/Control Register 3 Analog6 Scan Request
SC3_ANA7_SCAN_REQUEST	80h	(R/W) Status/Control Register 3 Analog7 Scan Request

Advanced API Registers

The following tables describe the advanced API registers.

Table 30. ePIR Advanced API Status/Control Register 0 (ePIR_ASC0)

Define	Value	Description
ASC0_TRANSIENT_DETECTED	01h	(R/W) API Status/Control Register 0 Transient Detected
ASC0_NOISE_DETECTED	02h	(R/W) API Status/Control Register 0 Noise Detected
ASC0_MD_ORIGIN	04h	(R) API Status/Control Register 0 Motion Detection Origin
ASC0_NEW_SAMPLE	08h	(R/W) API Status/Control Register 0 New Sample
ASC0_BUFFER_REFRESH	10h	(R/W) API Status/Control Register 0 Buffer Refresh
ASC0_EXTENDED_DETECTION	20h	(R/W) API Status/Control Register 0 Extended Detection
ASC0_BIT6_RESERVED	40h	(Reserve) API Status/Control Register 0 Bit6 Reserve
ASC0_BIT7_RESERVED	80h	(Reserve) API Status/Control Register 0 Bit7 Reserve

Table 31. ePIR Advanced API Status/Control Register 2 (ePIR_ASC2)

Define	Value	Description
ASC2_WINDOW_UPDATE_RATE	07h	(R/W) API Status/Control Reg. 2 Window Update Rate
ASC2_WINDOW_SIZE	18h	(R/W) API Status/Control Reg.2 Window Size
ASC2_LOCK_LEVEL	E0h	(R/W) API Status/Control Reg.2 Lock Level

Table 32. ePIR Process Rate Register (ePIR_Process_Rate)

Define	Value	Description
ePIR_PROCESS_RATE	FFFFh	(R) ePIR Process Rate

Table 33. ePIR Sample Size Register (ePIR_Sample_Size)

Define	Value	Description
ePIR_SAMPLE_SIZE	FFh	(R/W) ePIR Sample Size

Table 34. ePIR Debounce Time Register (ePIR_Debounce)

Define	Value	Description
ePIR_DEBOUNCE_TIME	FFh	(R/W) ePIR Debounce Time

Table 35. ePIR Transient Sensitivity Level Register (ePIR_Transient_Sense)

Define	Value	Description
ePIR_TRANSIENT_SENSE	7Fh	(R/W) ePIR Transient Sense
ePIR_TRANSIENT_SENSE_BIT7_RESERVED	80h	(Reserve)

Table 36. ePIR Noise Sensitivity Register (ePIR_Noise_Sense)

Define	Value	Description
ePIR_NOISE_SENSE	7Fh	(R/W) ePIR Noise Sense
ePIR_NOISE_SENSE_BIT7_RESERVED	80h	(Reserve)

Table 37. ePIR Signal Register (ePIR_Signal)

Define	Value	Description
ePIR_SIGNAL	FFFFh	(R) ePIR Signal

Table 38. ePIR DC Signal Register (ePIR_DC_Signal)

Define	Value	Description
ePIR_DC_SIGNAL	FFFFh	(R) ePIR DC Signal

Table 39. ePIR Engine Entry Macro Definitions

Define	Description
EPIR_INIT	Initialized the peripherals such as ADC and GPIO; ADC IRQ is set to medium priority.
EPIR_ADC_ISR	Initialized Interrupt for ADC. All the motion detected is executed by this macro.

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

 Information in this publication concerning the devices, applications, or technology described is intended to suggest possible uses and may be superseded. ZILOG, INC. DOES NOT ASSUME LIABILITY FOR OR PROVIDE A REPRESENTATION OF ACCURACY OF THE INFORMATION, DEVICES, OR TECHNOLOGY DESCRIBED IN THIS DOCUMENT. ZILOG ALSO DOES NOT ASSUME LIABILITY FOR INTELLECTUAL PROPERTY INFRINGEMENT RELATED IN ANY MANNER TO USE OF INFORMATION, DEVICES, OR TECHNOLOGY DESCRIBED HEREIN OR OTHERWISE. The information contained within this document has been verified according to the general principles of electrical and mechanical engineering.

ZMOTION is a trademark of Zilog, Inc. All other product or service names are the property of their respective owners.