ZMOTION® Engine Library for the F6482 Series

User Manual

UM027501-0816

www.zilog.com

Warning: DO NOT USE THIS PRODUCT IN LIFE SUPPORT SYSTEMS.

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concerning the devices, applications, or technology described is intended to suggest possible uses and may be superseded.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™ and Z8 Encore! XP™ are trademarks or registered trademarks of Zilog, Inc., an IXYS Company. All other product or service names are the property of their respective owners.

Revision History

Each instance in the following Revision History table reflects a change to this document from its previous version. For more details, refer to the corresponding pages or appropriate links listed in the table below.

Date	Revision Level	Description	Page Number
Aug 2016	01	Original Issue.	All

Overview

Zilog's ZMOTION Engine Library provides an integrated and flexible solution for Passive Infrared (PIR)-based motion detection applications. The software library consists of the PIR signal processing algorithms for motion detection, transient and noise detection, white light detection, and several other motion-related functions, and is integrated with the user's application code.

An Application Programming Interface (API) allows the application code to configure, control, and monitor the library in real time. API configuration parameters enable the Engine operation to be optimized for the particular lens and pyroelectric sensor being used in the application. This allows designers to create their own application-specific software while taking advantage of Zilog's ZMOTION Motion Detection Technology.

Features

The features of the ZMOTION Engine library include:

- Operates on Zilog's Z8F6482 series of microcontrollers
- Software-based Motion Detection Engine (ZMOTION Engine) library controlled and monitored through software API registers
- Low power modes for battery-operated applications
- Sensitivity, range, and frequency control
- Capable of pet immunity with appropriate lens
- Directionality detection
- Control over transient and noise immunity
- No temperature compensation required
- White light detection using status LED reduces system cost (eliminates CDS photocell)
- MCU resources remain available for other functions

Related Documentation

Additional information can be found in the following documents. These are available on the Zilog website.

Table 1. Related Documentation

Document Number	Description
PS0294	Z8F6482 MCU Series Product Specification
PB0246	Z8F6482 MCU Series Product Brief
PS0285	ZMOTION Detection and Control Product Specification
PB0225	ZMOTION Detection and Control Product Brief
PS0288	ZMOTION Intrusion Product Specification
PB0230	ZMOTION Intrusion Product Brief
PB0223	ZMOTION Detection Module Product Brief
PS0284	ZMOTION Detection Module Product Specification

Supported MCU Part Numbers

The ZMOTION Engine library operates on Zilog's Z8F6482 series of microcontrollers. **A specific version of the device must be used for the Library to operate correctly.** This version is identified by the 2247 suffix on the device part number. The following table shows the supported MCU part numbers and features.

Table 2. Supported MCU Part Numbers

MCU Part Number	Package	Memory
Z8F1681QK024XK2247	32-Pin QFN	16KB Flash, 2KB RAM
Z8F1681QN024XK2247	44-Pin QFN	16KB Flash, 2KB RAM
Z8F1681AN024XK2247	44-Pin LQFP	16KB Flash, 2KB RAM
Z8F6481QN024XK2247	44-Pin QFN	64KB Flash, 3.75KB RAM
Z8F6481AN024XK2247	44-Pin LQFP	64KB Flash, 3.75KB RAM
Z8F6481AT024XK2247	80-Pin LQFP	64KB Flash, 3.75KB RAM

Refer to the F6482 Series MCU Product Specification (PS0294) for a full description of features for each device type.

ZMOTION Engine and API

The ZMOTION Engine library is composed of four major components:

- Normal Detection Engine
- Extended Detection Engine
- Transient and Noise Detection and Immunity
- White Light Detection and Immunity

Each of these components is separately controlled and monitored through an API by the user's application code.

The library is compiled using the ZDSII Zilog Developer Studio Integrated Development Environment. This tool is available for free from the Zilog Store. The library is linked in with the user's application code.

Normal Detection Engine

The Normal Detection Engine provides the primary motion detection function in the system. This Engine is designed to detect typical motion events while ignoring signals from typical false motion events like air flow and temperature changes. Various levels of qualification are configurable in the API, which allows it to be tuned to the particular lens being used in the system.

Extended Detection Engine

The Extended Motion Engine acts as a secondary motion detection check that runs in parallel with the Normal Engine. The Extended Detection Engine looks for signals that extend beyond typical motion events, effectively improving the detection of minor motion events (micro-motion). Examples include very fast moving targets, which generate very small, high frequency motion signals, and very slow moving targets, which generate very small low frequency motion signals. However, these types of events are also common signs of false motion events, so extended detection must be used carefully.

Transient, Noise, and Spark Detection and Immunity

Because the ZMOTION Engine processes the raw/unfiltered signal from the pyroelectric sensor, signal characteristics other than normal motion signals like EMI and power supply disturbances can be detected and removed from further processing. The Transient, Noise, and Spark Detectors perform this function. A transient signal is a signal that rises quicker

than a valid motion signal would, while noise is a signal that does not have the correct shape. The sensitivity of each detector can be controlled individually.

White Light Detection and Immunity

Visible light sources can contain energy in the infrared spectrum. When visible light is shined on a pyroelectric sensor, it can generate a signal that looks like a valid motion event. The White Light Detection feature is used to eliminate this occurrence. Sensor data from a light sensing device located near the pyroelectric sensor is passed to the Engine's White Light function. When a significant change in light level is detected (configurable in the API), the Motion Detection Engines are compensated in anticipation of the simultaneous signal variation from the pyroelectric sensor. Any light sensing element can be used, including the status LED.

Notes: 1. Some intrusion/security motion detector standards require that the device be immune to this sort of false motion event.

2. The White Light Detection and Immunity API function is only available with the ZMOTION_Engine_WL_Lib library.

General Operation

During operation, the user's code first initializes the API registers, then calls the initialization function `ZMOTION_Init(void)`, which initializes all Engine parameters. The `ZMOTION_Init()` function only needs to be called once after reset. All API parameters can be updated in real time by the user's application.

After initialization, samples from the PIR sensor are acquired and passed to the Engine through the `ZMOTION_Engine(unsigned int)` function as an unsigned 16-bit value. This is done for each sensor sample at a normal rate of 1 sample per 1.5 ms. The API registers are updated accordingly after each constructed sample is created. A constructed sample comprises of 1, 2, or 4 samples as defined in the `ZM_SAMPLE_SIZE` API register.

If the application requires White Light Detection and Immunity, samples of the ambient light level are passed to the White Light Engine through the `ZMOTION_White_Light(unsigned int)` function as an unsigned 16-bit value. The White Light Engine looks for sudden changes in signal level and compensates the Normal Detection Engine to eliminate a possible false motion event. Refer to the White Light Detection section on page 37for more information.

The application monitors the API for validated motion and other events while performing other application functions. Once per second, the application must set the Engine Timer Tick bit in the ZM_CTRL0 register. All Engine parameters can be modified in real time.

ZMOTION Engine Entry Points

The three function entry points to the ZMOTION Engine are:

```
void ZMOTION_Init(void);
```

This function initializes the ZMOTION Engine. After Reset or when the Engine is restarted, the application should first initialize all API registers, and then call this function.

```
void ZMOTION_Engine(unsigned int);
```

PIR sensor samples are passed to the Engine through this function. This function performs motion detection and ancillary processing. API registers are updated after each constructed sample. The number of samples comprising a constructed sample is defined by the API register setting ZM_SAMPLE_SIZE.

```
void ZMOTION_White_Light(unsigned int);
```

This function performs all white light immunity processing. Its purpose is to determine if the system is being exposed to changes in light levels that could cause the PIR sensor to generate signals that resemble human motion and reject such signals. Samples from the light sensor are passed to this function.

Engine Timer Tick

The Engine Timer Tick bit (ZM_CTRL0[0]) must be set to 1 once per second to provide a 1-second time base for the ZMOTION Engine. The Engine uses this bit to perform housekeeping operations. The bit is checked and cleared by the ZMOTION_Engine() function after each constructed sample. The timing of this bit can be ±10%.

ZMOTION Engine CPU Stack Usage

The ZMOTION Engine shares the stack with the user application. There are no special requirements on the placement of the stack in memory; however, it is essential that the user provide enough stack space for both the user application and the ZMOTION Engine.

The ZMOTION Engine requires 4 bytes of stack space, including the call in to the library function.

ZMOTION API Register Set

The API Register Set is a series of registers reserved in memory that are used to monitor and control the ZMOTION Engine in real time.

Table 3. ZMOTION API Register Set

API Register Name	Library API Name	Description
ZMOTION Engine Status and Control		
ZMOTION Status 0	ZM_STATUS0	Engine status
ZMOTION Control 0	ZM_CTRL0	Engine operation mode control
ZMOTION Sample Size	ZM_SAMPLE_SIZE	Controls amount of PIR sensor signal averaging
ZMOTION Buffer Control 0	ZM_BUFF_CTRL0	Fast buffer refill control
ZMOTION Process Rate	ZM_PROCESS_RATE	Processed samples per second
ZMOTION Library and Version	ZM_LIB_VERSION	Library Type and Version
ZMOTION Normal Engine Configuration		
ZMOTION Normal Detection Sensitivity	ZM_N_SENSE	Sensitivity control for Normal Detection Engine
ZMOTION Control 1	ZM_CTRL1	Engine Frequency and Range settings
ZMOTION Control 2	ZM_CTRL2	Controls window configuration parameters
ZMOTION Debounce Timeout	ZM_DEB_TIMEOUT	Time to de-bounce initial motion signal
ZMOTION Debounce Batch Size	ZM_DEB_BATCH	Controls out of window samples needed for de-bounce
ZMOTION Extended Engine Configuration		
ZMOTION Extended Detection Control Register 0	ZM_EXT_CTRL0	Sensitivity settings for Extended Engine
ZMOTION Extended Detection Control Register 1	ZM_EXT_CTRL1	Range and Debounce settings for Extended Engine
Sensor Signal Information		
ZMOTION Signal	ZM_SIGNAL	Current calculated sensor constructed sample
ZMOTION DC Signal Level	ZM_SIGNAL_DC	Current calculated sensor DC offset
EMC Immunity Settings		
ZMOTION Transient Sensitivity Level	ZM_TRANSIENT_SENSE	Engine sensitivity to transient events

Table 3. ZMOTION API Register Set (Continued)

API Register Name	Library API Name	Description
ZMOTION Noise Sensitivity Level	ZM_NOISE_SENSE	Engine sensitivity to noise events
ZMOTION Spark Detection	ZM_SPARK_DETECT	Engine Sensitivity to ESD and related events
White Light Immunity Control		
ZMOTION White Light Threshold	ZM_WL_THRESHOLD	Sensitivity setting for White Light immunity
ZMOTION White Light Control 0	ZM_WL_CTRL0	White Light Debounce and Control

ZMOTION Status 0

The ZMOTION Status 0 Register, shown in Table 4, reports the status of the Engine.

Table 4. ZMOTION Status 0 (ZM_STATUS0)

Bit	7	6	5	4	3	2	1	0
Field	Motion Detected	MD Origin	Motion Direction	PIR Stable	New Sample	EM Spark Detected	EM Noise Detected	EM Transient Detected
Control	R/W	R	R	R	R/W	R/W	R/W	R/W

Bit	Description
[7] Motion Detected	Motion event detected. Set by ZMOTION Engine; cleared by application. This bit indicates that the ZMOTION Engine has detected and validated a motion event. The user application should routinely check this bit to determine if motion has been detected. This bit is set by the Engine and must be cleared by the user application. 0 = No motion detected by the Engine 1 = Motion has been detected by the Engine
[6] MD Origin	Origin of last motion detection event. Controlled by ZMOTION Engine. This bit indicates the ZMOTION Engine that detected the last Motion Detected Event. When the Engine sets the Motion Detected bit, it also sets this bit according to which detection engine detected the event. 0 - Normal Motion Detector 1 - Extended Motion Detector

Bit	Description
[5] Motion Direction	Relative Direction of Last Motion Event. Controlled by ZMOTION Engine. When Motion Direction Control is enabled (ZM_CTRL0[6] = 1), this bit is set or cleared by the ZMOTION Engine to indicate the relative direction of the last motion event. The bit state is maintained until the user application clears the Motion Detected bit. 0 = Last detected motion was negative 1 = Last detected motion was positive This status bit is undefined when Motion Direction Control is disabled.
[4] PIR Stable	Passive Infrared (PIR) Sensor signal stabilized bit. Controlled by ZMOTION Engine. At power on or after a reset, the PIR sensor takes some time to stabilize its DC offset before it can be used reliably. The amount of time taken can range from a few seconds up to a minute and is dependent on the PIR sensor itself and environmental conditions. To relieve the application software from having to assume the worst case stabilization time, the ZMOTION Engine monitors the DC offset of the PIR sensor and sets this bit when it determines that the offset has become stable and the sensor is usable. This bit indicates that the PIR sensor has stabilized after one of the following conditions: – After initial power on (cold start – After re-enabling the Engine via PIR Enable Register – After returning from sleep mode 0 = PIR sensor signal is not stable; motion detected events are not valid 1 = PIR sensor signal is stable; motion detected events are valid. **Note:** After the application detects that this bit is set (i.e., the sensor is stable), it is not necessary to continue checking the state. This bit may occasionally reset after motion events. This is not an indication that the sensor is unstable.
[3] New Sample	New sample available from the ZMOTION Signal (ZM_SIGNAL) register. Set by ZMOTION Engine, cleared by application. This bit indicates that the ZMOTION Engine has a new constructed sample available that may be read by the application. This status is available as an advanced feature because the application is not normally required to read the sampled PIR sensor signal. The application must clear this bit when the sample has been read.
[2] EM Spark Detected	Set by ZMOTION Engine; cleared by application. This bit indicates that the Engine has detected a Spark event from the PIR signal and associated motion event(s) may have been suppressed by the engine. This bit does not have to be read for normal operation and is provided as status only. The application must clear this bit after it has been read. The sensitivity to Spark events can be adjusted with the ZMOTION Spark Detection (ZM_SPARK_DETECT) register.

Bit	Description
[1] EM Noise Detected	Set by ZMOTION Engine; cleared by application. This bit indicates if the Engine has detected noise on the PIR signal. This event is provided to the user application to indicate that an EM noise event has occurred and associated motion event(s) may have been suppressed by the engine. This bit does not have to be read for normal operation and is provided as status only. The application must clear this bit after it has been read. The sensitivity to Noise events can be adjusted with the ZMOTION Noise Sensitivity Level (ZM_NOISE_SENSE) register.
[0] EM Transient Detected	Set by ZMOTION Engine; cleared by application. This bit indicates if the Engine has detected a transient on the PIR signal. This event is provided to the user application to indicate that an EM transient event has occurred and associated motion event(s) may have been suppressed by the engine. This bit does not have to be read for normal operation and is provided as status only. The application must clear this bit after it has been read. The sensitivity to Transient events can be adjusted with the ZMOTION Transient Sensitivity Level (ZM_TRANSIENT_SENSE) register.

ZMOTION Normal Detection Sensitivity

The ZMOTION Status 0 Register, shown in Table 5, is used to control the sensitivity of the Normal Motion-Engine.

Table 5. ZMOTION Normal Detection Sensitivity (ZM_N_SENSE)

Bit	7	6	5	4	3	2	1	0
Field	Normal Sensitivity							
Control	R/W							

Bit	Description
[7:0] Normal Sensitivity	ZMOTION Normal Detection Sensitivity. Controlled by application. The ZMOTION Normal Sensitivity Register is used to adjust the sensitivity of the Normal Motion Engine to target motion. Specifically, the value in this register controls the required duration that a partially validated motion signal must remain valid for. Lower values provide higher sensitivity to motion with `00h` being the most sensitive and `FFh` being the least sensitive. The user application should load this register with the appropriate value to give the desired sensitivity. **Note:** The value in this register will also have an effect on the detection distance/range. Lower values increase range and higher values decrease range.

ZMOTION Control 0

The ZMOTION Control 0 register, shown in Table 6, controls the operation mode of the Engine.

Table 6. ZMOTION Control 0 (ZM_CTRL0)

Bit	7	6	5	4	3	2	1	0
Field	MD Suspend	Motion Direction Enable	2-Pulse Mode		–	–	–	Engine Timer Tick
Control	R/W	R/W	R/W		–	–	–	R/W

Bit	Description
[7] MD Suspend	Motion Detection Suspend. Controlled by application. Temporarily suspends the ZMOTION Engine from running. This puts it in a very low processing overhead state and can be used when the application requires significant CPU processing power. While suspended, motion detection is disabled. Samples passed through the `ZMOTION_Engine()` function are not processed, but held in a circular buffer to provide a faster recovery from this mode. When the application clears this bit, suspend mode is exited upon the next call of the `ZMOTION_Engine()` function. 0 = Normal Motion Detection 1 = Suspended Motion Detection
[6] Motion Direction Enable	Motion Direction Control Enable. Controlled by application. This bit enables directional motion detection. When this bit is set and motion is detected (ZMOTION Status 0 [7] = 1), the relative direction of the detected motion event is indicated in the Motion Direction bit of the ZMOTION Status 0 register. 0 = Directional Motion Detection Disabled 1 = Directional Motion Detection Enabled The directional polarity of PIR sensors is arbitrary at the time of manufacturing. Therefore, it is necessary for the user application to calibrate to each individual PIR sensor using a controlled target (i.e., moving in a known direction) and internally record the polarity to identify which polarity represents that direction. **Note:** Enabling Motion Direction Control also disables Extended Detection. Only the Normal motion detector runs while directional detection is enabled.

Bit	Description
[5:4] 2-Pulse Mode	2-Pulse Detection Enable and Mode. Controlled by application. These bits determine if motion detection requires one or two motion pulses as the target passes across the beams created by the lens. With 2-Pulse Mode disabled, a single motion edge from the target is required to qualify as valid motion. This is the normal mode of operation. With 2-Pulse enabled, two motion edges in opposite directions are required from the target within the selected time window in order to generate a valid motion event. A pulse count of 2 is typically used in harsher environmental conditions to decrease the chances of false triggers from sources such as fast heating and cooling. When 2-Pulse is enabled, Extended Detection is automatically disabled. 00 = 2-Pulse Disabled, motion is qualified with one edge 01 = 2-Pulse Enabled, time window is two seconds 10 = 2-Pulse Enabled, time window is three seconds 11 = 2-Pulse Enabled, time window is four seconds
[0] Engine Timer Tick	One Second Engine Timer Tick. Set by Application; cleared by ZMOTION Engine. This bit must be set once per second by the application, which provides the Engine with a one-second time base to perform housekeeping operations relating to motion detection. The Engine routinely polls this bit to obtain a one-second tick. A tolerance of ± 10% on the timing of this bit is acceptable. This bit is cleared by the Engine. 0 = Cleared by ZMOTION Engine 1 = One-second interval has occurred.

ZMOTION Control 1

The ZMOTION Control 1 register, shown in Table 7, is used to control the Engine Frequency and Range settings.

Table 7. ZMOTION Control 1 (ZM_CTRL1)

Bit	7	6	5	4	3	2	1	0
Field	Frequency Response				Range Control			
Control	R/W				R/W			

Bit	Description
[7:4] Frequency Response	Frequency Response of ZMOTION Engine. Controlled by application. Range: 0h - Fh This value determines the frequency response of the motion detection system. Higher values allow lower frequencies to be accepted by the ZMOTION Engine. Lower values cause the Engine to ignore targets that generate lower frequencies. These targets typically include horizontally oriented objects such as pets. Specifically, the Frequency Response value controls the accepted rate of change in the PIR signal, causing it to act as a motion signal high-pass filter check. A larger value means a valid motion signal's rate of change (i.e. slope) can be slower. A smaller frequency response value means a valid motion signal's rate of change (slope) must be faster. To increase detection performance for smaller motion events, make this number larger, thereby allowing lower speed targets to be detected. To improve stability, particularly to environmental temperature changes and airflow (outdoor or around HVAC systems), make this number smaller. Valid values are 0h to Fh (the maximum value depends on the window size selected in ZMOTION Control 2). In most applications, values between 4h and Ch are typical, with 8h to Ah being the most common for non-pet immune applications. The maximum Frequency Response values for each window size can be summarized as: **Window Size** — **Max Frequency Response Value** Small — Fh Medium — Fh Large — Ch **Notes:** 1. Lower programmed values also have the effect of reducing the relative range of detection because PIR lenses cause targets to produce lower frequencies at greater distances. 2. The frequency of the signal is largely dependent on the structure of the PIR lens being used (number and dispersion of beams). A lens with several evenly distributed beams provides better frequency response performance than a lens with an uneven beam distribution.
3:0] Range Control	Motion Detection Range Control. Controlled by application. These bits determine the relative range of motion detection by performing a motion signal relative to amplitude check. The amplitude is relative to the average DC level of the sensor signal. A smaller range value indicates the amplitude of a valid motion signal can be lower, thereby increasing the range of detection, and vice versa. Valid values are 0-F. In most applications, values between 1 and 6 are typical, with 3 being the most common. Differences in lower Range values are more pronounced than differences in higher Range values. For example, changing from 3 to 1 may produce a more noticeable difference than changing from 9 to 7. Typical values used for Range are dependent on the lens and pyroelectric sensor being used. Range is also dependent on target size, speed, and relative temperature. For example, a range control setting that rejects one target of a particular size at a given distance does not guarantee that a larger target will be rejected at the same distance.

ZMOTION Control 2

The ZMOTION Control 2 register, shown in Table 8, is used to control the window configuration parameters.

Table 8. ZMOTION Control 2 (ZM_CTRL2)

Bit	7	6	5	4	3	2	1	0
Field	Lock Level			Window Size		Window Update Rate		
Control	R/W			R/W		R/W		

Bit	Description
[7:5] Lock Level	Control Limit Lock Level. Controlled by application. This parameter sets the minimum slope change in the signal that can be considered valid motion. It automatically adapts to spurious noise in the motion signal and ensures that a motion event has the proper profile. This prevents small signal changes caused by environmental or V_{CC} shifts from causing a false detection. Use this value in combination with ZMOTION Normal Detection Sensitivity (ZM_N_SENSE) and Range Control [3:0] settings to balance sensitivity and stability to the particular lens and pyroelectric sensor being used. A smaller Lock Level value indicates that a valid motion signal's change in amplitude can be lower whereas a larger Lock Level value indicates that the change in amplitude is higher. To increase range, make this number smaller. To increase stability, make this number larger. Valid values are 0-7. In most applications, values between 1 and 3 are typical, with 2 being the most common. **Notes:** 1. Smaller values allow subtle signals with lower slopes to be considered motion events at the expense of potential false motion events. 2. Larger values allow the system to ignore smaller signal slope changes at the expense of potentially missing smaller motion events.

Bit	Description
[4:3} Window Size	Control Limit Window Size. Controlled by application. This register determines the size of the control limit window. A larger window size produces more stable control limits at the cost of additional CPU usage. If a smaller window size is used, the window can be calculated more frequently, which allows it to track the signal better. 00 - Reserved 01 - Small window 02 - Medium window 03 - Large window
[2:0] Window Update Rate	Control Limit Update Rate. Controlled by application. This register determines how frequently the control limits are calculated. A smaller number produces more frequent calculations, which allow the control limits to track the signal better, at the cost of increased CPU usage. The valid range is from 0 to 7. The window control limits are updated every 4 + (Window Update Rate * 2) constructed samples.

ZMOTION Sample Size

The ZMOTION Sample Size register, shown in Table 9, controls the amount of PIR sensor signal averaging.

Table 9. ZMOTION Sample Size (ZM_SAMPLE_SIZE)

Bit	7	6	5	4	3	2	1	0
Field	Reserved					Sample Size 4	Sample Size 2	Sample Size 1
Control	–					R/W	R/W	R/W

Bit	Description
[2:0] Sample Size	Constructed Sample Size. Controlled by application. This register controls the amount of averaging that the engine performs on the PIR samples. More averaging improves signal noise immunity at the cost of higher processor overhead. Valid values are 1, 2, and 4. Only one bit should be set at a time. If multiple bits are set, the highest set bit is used.

ZMOTION Buffer Control 0

The ZMOTION Buffer Control Register, shown in Table 10, is used for fast buffer refill control.

Table 10. ZMOTION Buffer Control (ZM_BUFF_CTRL0)

Bit	7	6	5	4	3	2	1	0
Field	Buffer Refresh	–	–	–	Fast Buffer Refill Rate			
Control	R/W	–	–	–	R/W			

Bit	Description
[7] Buffer Refresh	Force fast fill algorithm to quickly refill the motion detection buffers. Controlled by application. This bit is used to restart motion detection by quickly re-initializing and refilling the motion detection constructed sample buffers. This can be used as a method to restore motion detection after waking up from sleep mode. Alternatively, it can be used to help ignore external events that may cause false detections. **Waking up from Sleep Mode** If this bit is set when the `ZMOTION_Engine()` function is called, the Engine refills the constructed sample buffers using a fast fill algorithm that allows it to quickly restore motion detection. This is typically used for low power applications with a wake-up circuit that provides an unqualified motion detection signal to wake up the MCU from Stop mode (SMR - Stop Mode Recovery). Refer to Sample Projects for a sample application. Upon SMR, the application sets the Buffer Refresh bit, executes the `ZMOTION_Engine()` function, and then continues with other functions for some period of time, typically 2 seconds, while polling the Motion Detected bit in ZMOTION Status 0 (ZM_STATUS0) before returning to Stop mode. By setting this bit prior to calling the `ZMOTION_Engine()` function, the Engine buffers are filled much faster, enabling it to analyze the original signal seen by the external wake up circuit and determine if it is actual motion. **Ignoring False Detection Events** The Buffer Refresh bit can be used to help ignore any PIR signal variations that could be created from power supply fluctuations. These can be caused when the MCU controls external components such as LEDs, relays, lights, TRIACs, etc. When the external device is turned on or off, the application can set the Buffer Refresh bit to effectively reset the motion detection history and therefore ignore any effect on the PIR signal from the external device.
[3:0] Fast Buffer Refill Rate	Rate at which to refill the buffers when the Buffer Refresh bit is set. Controlled by application. This value controls the additional number of times the current constructed sample is copied to the buffer when the Buffer Refresh bit is set. The buffer size is 256 bytes. For example, if Fast Buffer Refill Rate is set to 1, it will take 128 constructed samples for the Engine to fill the buffer and start looking for motion.

ZMOTION Debounce Timeout

The ZMOTION Debounce Timeout register, shown in Table 11, controls the time that the Normal Motion Detection Engine will wait to debounce the initial motion signal.

Table 11. ZMOTION Debounce Timeout (ZM_DEB_TIMEOUT)

Bit	7	6	5	4	3	2	1	0
Field	ZMOTION Debounce Timeout							
Control	R/W							

Bit	Description
[7:0] Debounce Timeout	Controlled by application. This register controls the amount of time that the Normal Motion Detection Engine will wait to fully de-bounce a motion signal. Longer times result in detection of subtle motion at the cost of additional potential false motion detections. Valid range is from 01h to FFh. Using a value less than or equal to the value in the Normal Sensitivity Register will result in no motion detection.

ZMOTION Debounce Batch Size

The ZMOTION Debounce Batch Size register, shown in Table 12, is a mask used to control the number of out-of-window samples required to initially debounce the signal.

Table 12. ZMOTION Debounce Batch (ZM_DEB_BATCH)

Bit	7	6	5	4	3	2	1	0
Field	ZMOTION Debounce Batch Size							
Control	R/W							

Bit	Description
[7:0] Debounce Batch Size	Controlled by application. This register determines the number of consecutive out-of-window samples required in order to consider the sequence a valid de-bounce count. The field works as a mask. Increasing the mask size (i.e. more bits set to 1) increases the noise immunity of the engine but results in lower sensitivity to subtle motion signals. This parameter provides glitch immunity by ignoring spurious noise on the motion signal, preventing initiation of a motion event qualification process. Valid values are 01h, 03h, 07h, 0Fh, 1Fh, 3Fh, 7Fh, and FFh.

ZMOTION Extended Detection Control Register 0

The ZMOTION Extended Detection Control Register 0, shown in Table 13, sets the sensitivity level for the Extended Engine.

Table 13. ZMOTION Extended Detection Control Register 0 (ZM_EXT_CTRL0)

Bit	7	6	5	4	3	2	1	0
Field	Extended Level			Extended Sensitivity				
Control	R/W			R/W				

Bit	Description
[7:5] Extended Level	Sets the overall sensitivity level of extended detector. Controlled by application. Extended Level is used to set the qualification level of motion events detected by the Extended Motion Engine. Extended event qualification looks for an extended motion signal to be significant enough to be considered motion. The Extended Detection Engine is capable of detecting very small signals, including some that may not be actual motion. For example, long term temperature drift or other environmental interference may cause small deviations in the signal that the extended detector could consider as motion. By adding the qualification level, these can be filtered out. The Extended Detection level is selected to provide a balance between additional sensitivity while maintaining stability (no false detections). In certain applications such as combined intrusion/lighting controls, the Extended Detection level can be increased temporarily if one or more Normal motion events are detected, verifying that the area is occupied. This method can help provide micro-motion detection when it is required, while offering good stability when the area is unoccupied. Extended detection is dependent on the lens pattern used. Smaller lens beams tend to provide more subtle motion detection. Valid values are 0-7 with 0 providing the highest sensitivity and 7 providing the lowest sensitivity. A setting of 0 disables extended detection. Typical values are 3 to 6. 000 = Extended Detection Level 0 - Highest Sensitivity 001 = Extended Detection Level 1 … 111 = Extended Detection Level 7 - Lowest Sensitivity.
[4:0] Extended Detection Sensitivity	Controlled by application. This register determines how sensitive the Extended Detection part of the engine is to fast or slow speed targets in the PIR field of view. A lower number makes the extended detector more sensitive, at the cost of potential false motion events. The valid range is 1 (most sensitive) to 31 (least sensitive) with typical values between 6 and 25. A value of 0 disables the Extended Detection Engine. 00000 = Extended Detection Disabled 00001 = Extended Detection Level 1 - Highest Sensitivity … 11111 = Extended Detection Level 31 - Lowest Sensitivity.

ZMOTION Extended Detection Control Register 1

The ZMOTION Extended Detection Control Register 1 is used to adjust the range and debounce settings for the Extended Engine.

Table 14. ZMOTION Extended Detection Control Register 1 (ZM_EXT_CTRL1)

Bit	7	6	5	4	3	2	1	0
Field	Extended Range			Extended Debounce Timeout				
Control	R/W			R/W				

Bit	Description
[7:5] Extended Range	Controlled by application. The Extended Range field is used to adjust the effective range of the extended detection engine. Lower values increase the effective range of the extended detection engine, with 0 being the most sensitive, and 7 being the least sensitive. These bits determine the relative range of motion detection by performing a motion signal relative amplitude check. The amplitude is relative to the average DC level of the sensor signal. A smaller range value means the amplitude of a valid motion signal can be lower, thereby increasing the range of detection. A larger range value means the amplitude of a valid motion signal must be higher decreasing the range of detection. Valid values are 0-7. In most applications, values between 1 and 3 are typical, with 2 being the most common. Differences in lower range values are more pronounced than differences in higher range values. For example, changing from 3 to 1 may cause a very noticeable difference. However, changing from 5 to 7 may produce a less noticeable effect. Typical values used for Range are dependent on the lens and pyroelectric sensor being used. Range is also dependent on target size, speed, and relative temperature. For example, a range control setting that rejects one target of a particular size at a given distance does not guarantee that a larger target will be rejected at the same distance.
[4:0] Extended Debounce Timeout	Controlled by application. This setting determines the length of time the Extended Detection part of the engine waits to fully de-bounce a fast-moving or subtle motion signal. Longer times result in detection of subtle motion at the cost of higher number of potential false motion detections. A valid range is from 01h to 1Fh. The value used must be greater than 50% of the value used for Extended Detection Sensitivity. Using a value less than 50% of the Extended Detection Sensitivity will prevent the Extended Engine from completing a debounce.

ZMOTION White Light Threshold

The ZMOTION White Light Threshold register, shown in Table 15, determines the sensitivity settings for white light immunity.

Table 15. ZMOTION White Light Threshold (ZM_WL_THRESHOLD)

Bit	7	6	5	4	3	2	1	0
Field	White Light Threshold							
Control	R/W							

Bit	Description
[7:0] White Light Threshold	White Light Detection Threshold. Controlled by application. These bits determine how sensitive the white light detector is to changes in detected white light. Larger values cause the white light detector to be less sensitive. A value of 0x00 disables White Light Detection. See the White Light Detection section on page 37 for a description of White Light operation.

ZMOTION White Light Control 0

The ZMOTION White Light Control 0 register, shown in Table 16, is used for white light debounce and control.

Table 16. ZMOTION White Light Control 0 (ZM_WL_CTRL0)

Bit	7	6	5	4	3	2	1	0
Field	White Light Debounce				White Light Anti-Jam	–	–	White Light Detected
Control	R/W				R/W	–	–	R/W

Bit	Description
[7:4] White Light Debounce	White light debounce time. Controlled by application. This value determines the amount of debouncing applied to White Light detection. White Light Debounce is the number of sequential WL samples above the threshold value set in ZM_WL_THRESHOLD required to consider it a WL event. Larger numbers result in stable White Light detection at the cost of slower White Light response.
[3] White Light Anti-Jam	Stop malicious jamming of the detector via continuous white light events. Controlled by application. White Light Anti-Jam is used to prevent a repetitive white light signal from jamming motion detection. Without this feature, constant source of white light events can indefinitely inhibit motion detection. With Anti-Jam enabled, after 12 White Light events within a short duration, the Engine begins to ignore them and returns to regular motion detection. After a period of no White Light events, the Engine begins looking for White Light events again. This feature is intended to prevent malicious jamming of the detector. 0 - White Light Anti-Jam disabled 1 - White Light Anti-Jam enabled.
[0] White Light Detected	White Light Detected. Set by ZMOTION Engine; cleared by application. This bit is set by the ZMOTION Engine when a White Light event has been detected. This status is provided to the user application to indicate that a White Light event has occurred and associated false motion event(s) may have been suppressed by the engine. This bit is set by the engine and must be cleared by the user application. The threshold or amount of light change required to trigger a White Light event is controlled by the ZMOTION White Light Control 0 (ZM_WL_CTRL0) register.

ZMOTION Signal

The ZMOTION Signal register, shown in Table 17, contains the current calculated sensor constructed sample.

Table 17. ZMOTION Signal (ZM_SIGNAL)

Bit	7	6	5	4	3	2	1	0
Field	ZMOTION Signal							
Control	R							

Bit	Description
[15:0] ZMOTION Signal	Controlled by ZMOTION Engine. These registers contain the last constructed PIR signal calculated by the engine. Each time the engine generates a new PIR signal sample, it places it in these registers and sets the New Sample bit in the ZM_STATUS0 register. This gives the application direct visibility to the PIR-generated signal for debugging purposes.

ZMOTION DC Signal Level

The ZMOTION DC Signal Level register, shown in Table 18, contains the current calculated sensor DC offset.

Table 18. ZMOTION DC Signal Level (ZM_SIGNAL_DC)

Bit	7	6	5	4	3	2	1	0
Field	ZMOTION Signal DC							
Control	R							

Bit	Description
[15:0] Sensor DC Signal Level	Controlled by ZMOTION Engine. These registers contain the last PIR signal DC Level calculated by the engine. Each time the engine generates new control limits, it places the DC component level in these registers.

ZMOTION Process Rate

The ZMOTION Process Rate register, shown in Table 19, indicates the processed samples per second.

Table 19. ZMOTION Process Rate (ZM_PROCESS_RATE)

Bit	7	6	5	4	3	2	1	0
Field	ZMOTION Process Rate							
Control	R							

Bit	Description
[15:0] Process Rate	Controlled by ZMOTION Engine. The ZMOTION Process Rate gives an indication of the number of constructed samples per second that the Engine is processing. If the Engine process rate drops significantly, its ability to detect motion can be significantly reduced. This value is typically used at the application development stage. Higher numbers are better. Generally, the process rate should be maintained above 0100h.

ZMOTION Transient Sensitivity Level

The ZMOTION Transient Sensitivity Level register, shown in Table 20, controls the Engine's sensitivity to transient events.

Table 20. ZMOTION Transient Sensitivity Level (ZM_TRANSIENT_SENSE)

Bit	7	6	5	4	3	2	1	0
Field	Reserved	ZMOTION Transient Sensitivity						
Control	0	R/W						

Bit	Description
[6:0] Transient Sensitivity	Controlled by application. This parameter controls the sensitivity of the transient detector. The transient detector monitors the motion signal for sudden dramatic changes in the signal and prevents them from causing false motion events. Sources of transient events include EMI from nearby radio transmitters or other circuitry on the PCB. Valid values are 0-100, with 0 disabling the Transient Detector. Typical values are from 6 to 40. Lower values provide more protection from transient events at the cost of potentially rejecting larger motion signals. Choose a value high enough to allow all desired motion events to pass through while still rejecting unwanted transient events. When a transient event is detected, the bit in ZM_STATUS0 is set, which the application reads during testing to determine the correct sensitivity level. The valid range is `0` (disabled) to `64h`.

ZMOTION Noise Sensitivity Level

The ZMOTION Noise Sensitivity Level register, shown in Table 21, controls the Engine's sensitivity to noise events.

Table 21. ZMOTION Noise Sensitivity Level (ZM_NOISE_SENSE)

Bit	7	6	5	4	3	2	1	0
Field	Reserved	ZMOTION Noise Sensitivity						
Control	0	R/W						

Bit	Description
[6:0] Noise Sensitivity	Controlled by application. This parameter controls the sensitivity of the noise detector. The noise detector monitors the motion signal for random broadband noise and prevents it from causing false motions events. Sources of broadband noise include EMI from nearby radio transmitters or other circuitry on the PCB. Valid values are determined by the window size selected (0 disables the Noise Detector). When a noise event is detected, the bit in ZM_STATUS0 (EM Noise Detected) is set, which the application reads during testing to determine the correct sensitivity level.

Window Size	Max ZMOTION Noise Sensitivity Value	Typical Value
Small	0Ch	08h
Medium	1Dh	12h
Large	46h	2D

ZMOTION Spark Detection

The ZMOTION Spark Detection register, shown in Table 22, controls the algorithm used to detect PIR signal events that have a very short duration such as ESD occurrences. When a qualified Spark event is detected, motion detection is de-sensitized for a period of 10 constructed samples (typically 15ms), allowing the event to pass without causing a false motion detection. The ZMOTION sample buffer is flushed and motion detection begins again.

Table 22. ZMOTION Spark Detection (ZM_SPARK_DETECT)

Bit	7	6	5	4	3	2	1	0
Field	Spark Gap		Spark Sensitivity					
Control	R/W		R/W					

Bit	Description
[7:6] Spark Gap	Controlled by application. This parameter controls the gap between the leading edge and trailing edge of the signal profile. It is stated in units of constructed samples. See Table 23.
[5:0] Spark Sensitivity	Controlled by application This parameter controls the sensitivity of the spark detector. It sets the minimum signal change that must be seen on both the leading edge and trailing edge of the signal. Lower values cause the Spark Detector to become more sensitive and provide additional protection to ESD events. Setting this field to 0 disables the Spark Detector. Choose a value high enough to allow all desired motion events to pass through while still rejecting unwanted ESD events. When a spark event is detected, the `ZMS0_EM_SPARK_DETECTED` bit in `ZM_STATUS0` is set. The application can read this during testing to determine the correct sensitivity level. The valid range is `0` (disabled) to `3Fh`.

Table 23. Spark G

Spark Gap	Constructed Samples	Pulse Width (1.5ms constructed sample rate)
00	±1	3ms
01	±2	6ms
10	±3	9ms
11	±4	12ms

ZMOTION Library and Version

The ZMOTION Library and Version register, shown in Table 24, indicates the library type and version.

Table 24. ZMOTION Library and Version (ZM_LIB_VERSION)

Bit	7	6	5	4	3	2	1	0
Field	Library Type				Library Version			
Control	R							

Bit	Description
[7:4] Library Type	ZMOTION Engine Library Type. Controlled by ZMOTION Engine. This field allows the library type to be identified by the application code. At present, four Library Types are defined, as shown in Table 25.
[3:0] Library Version	ZMOTION Engine Library Version. Controlled by ZMOTION Engine. The value stored in this register indicates the software version of the ZMOTION Engine Library.

Table 25. ZMOTION Library Types

Value	Library Name	Library Type	Description
1	ZMOTION_Engine_WL_Lib	ZMOTION Engine Library + White Light	Used for single and multi-sensor applications
2	ZMOTION_Engine_Lib	ZMOTION Engine Library (no White Light)	Used for single and multi-sensor applications
3	ZMOTION_Engine1_Lib	ZMOTION Library 1	Used for two and three sensor applications
4	ZMOTION_Engine2_Lib	ZMOTION Library 2	Used for three sensor applications

MCU Configuration

This section describes the MCU configurations required to support the ZMOTION Engine Library. Sample projects for each configuration are included with the ZMOTION Library installation. Please refer to these Sample Projects for recommended examples of software and hardware implementations.

The following three general system configurations are supported:

- ZMOTION Single PIR Sensor (standard ZMOTION)
- ZMOTION Multi-PIR Sensor (up to 3 sensors)
- Low Power Analog Wake-up with ZMOTION Digital Validation

Application software and hardware examples are available from Zilog for each of these configurations.

ZMOTION Single PIR Sensor

The PIR sensor is connected to the ADC through an internal operational amplifier (op amp). The op amp is configured to re-bias the PIR signal at 1.25V (half of the ADC V_{REF}) and introduce gain to the PIR signal. Low-pass filtering is not required. The amount of gain is dependent on the peak-peak noise of the PIR sensor plus the signal shift from motion events and should be set so the resultant signal is less than 2.5V peak-peak. Gain values from 5 to 20 are typical. The power supply for the PIR sensor is provided either from system power (AVdd) or from the MCU via the internally generated 2.5V VBIAS. To use VBIAS from the MCU, AVdd must be at least 3.0V (500mV above VBIAS).

A timer is used to automatically start ADC conversions every 1.5ms. The ADC is configured to its highest resolution of 14-bits using a 2-pass mode and the internal translation buffer (differential mode), and set to perform hardware averaging of 16 samples. After the 16 samples have been taken and averaged, the ADC issues an interrupt. The application code reads the ADC result and passes the value to the `ZMOTION_Engine()` function for processing. The API is monitored by the application to determine if motion has been detected. The API can be modified in real time.

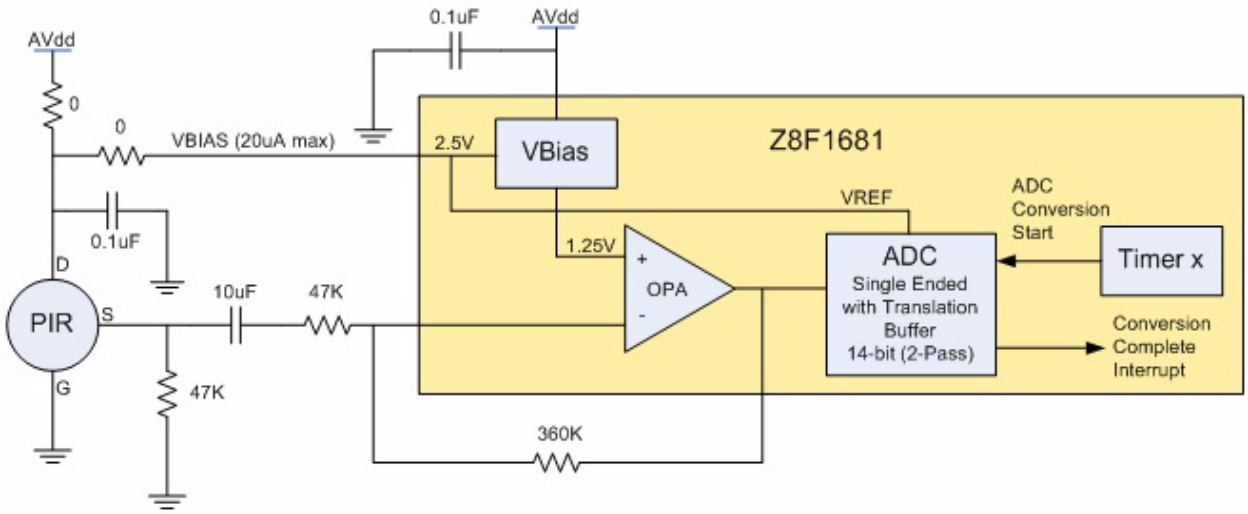

Figure 1. ZMOTION Single PIR Sensor Configuration

Observe the following procedure to initialize the MCU and ZMOTION Engine:

1. To initialize the system clock, select the desired clock input and frequency.
2. Initialize any application specific I/O and peripherals.
3. Set up the Engine API control registers with initial values and execute the `ZMOTION_Init()` function.
4. Set up op amp A and internal/external bias generation.
5. Configure ADC for single ended input with translation buffer and 2-pass 14-bit mode
6. Configure timer to generate ADC conversion such that one constructed sample is created every 1.5ms.
7. Enable ADC interrupts.
8. After each ADC interrupt, pass the converted ADC value to the ZMOTION Engine using the `ZMOTION_Engine(unsigned int)` function. Zilog recommends that you perform this call outside any interrupt.
9. Monitor the Motion Detection bit in ZM_STATUS0.
10. Update API registers as required.

ZMOTION Multi-PIR Sensor

The Z8F6481 MCU can support up to 3 PIR sensor inputs. Each sensor is processed separately by its own Engine Library.

Two PIR sensors are connected to the ADC through the internal op amps. If a third sensor is required, an external op amp is used. The op amps are configured in the same manner as for a single PIR sensor described above. VBIAS is set to 2.5 volts and is used internally as the ADC reference voltage and externally to generate the 1.25V reference for the external op amp. The VBIAS output has a source limit of 20uA and should therefore not be used as a power source for the PIR sensors.

A timer is used to automatically start ADC conversions every 1.5ms/(number of sensors). The ADC can be configured to automatically switch channels after each conversion. Alternatively, this can be handled by the software in the ADC interrupt. The ADC is set to its highest resolution of 14-bits using a 2-pass mode, using the internal translation buffer (differential mode) and set to perform hardware averaging of 16 samples. After the 16 samples have been taken and averaged, the ADC issues an interrupt where the converted value is read and saved in a buffer. After each ADC interrupt, the application code should read the ADC result and pass the value to the appropriate Engine.

Three separate Engine libraries should be included with the application project, one for each sensor. The additional libraries are called `ZMOTION_Engine_Lib_1` and `ZMOTION_Engine_Lib_2`.

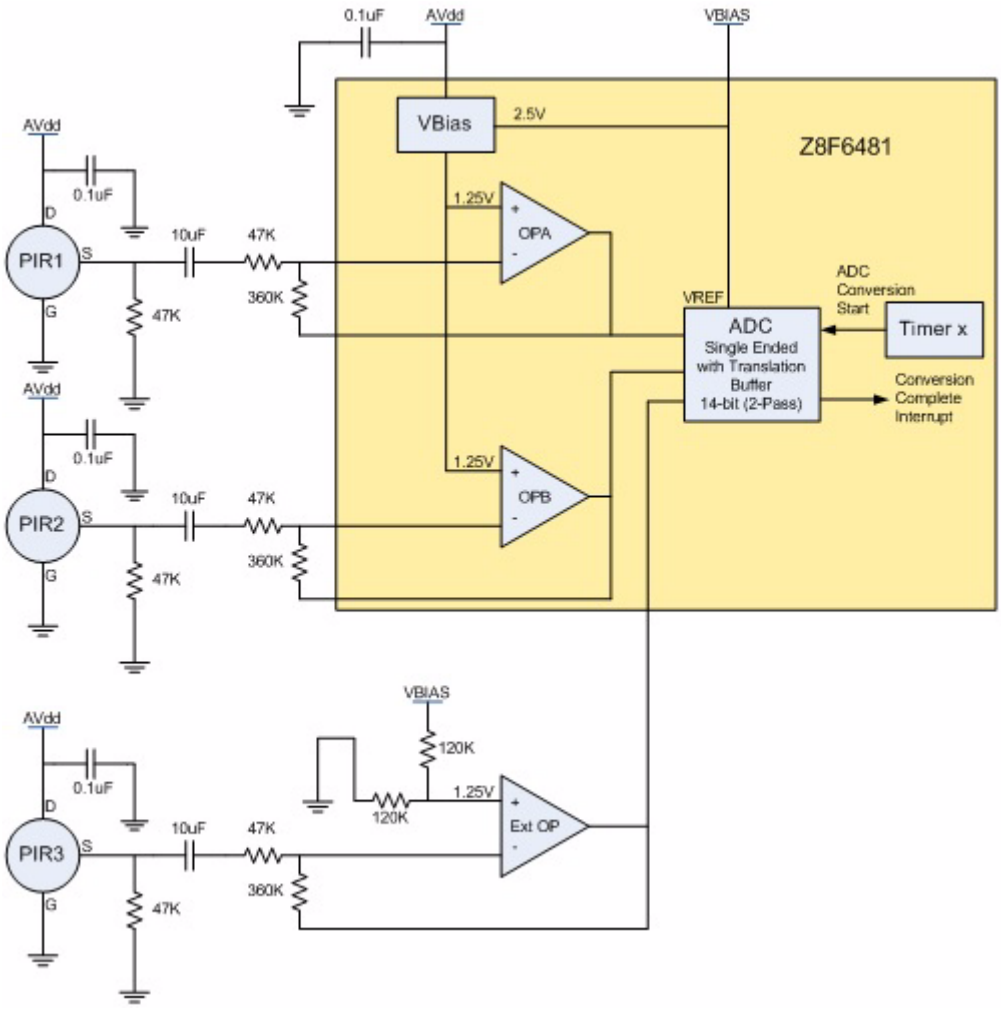

Figure 2. ZMOTION Multi-Sensor Configuration

Low Power Analog Wake-up with ZMOTION Digital Validation

For battery-operated or other low power applications, the ZMOTION Engine can provide digital validation of an analog-generated motion event. The Z8F1681 MCU internal low power op amps, comparators reference system, and one timer are configured to provide a traditional analog PIR sensor interface. Op amp A and B provide gain and filtering of the PIR signal. Component values depend on the lens being used and the required range. The two comparators and internal reference system are configured to provide a window comparator function whose output drives a timer which is used to validate/debounce the resulting signal.

This interface remains active while the MCU is in low power Stop mode. The Timer output generates a Stop Mode Recovery (SMR) when it detects a motion-like event. The ZMOTION Engine then runs and processes the direct PIR signal for a short period of time (typically up to 2 seconds) and validates the motion event.

This allows external noise sources and other causes of false detection events to be effectively ignored, thereby creating a more stable and reliable motion detector. Sample application code is available to configure and run the MCU in this mode.

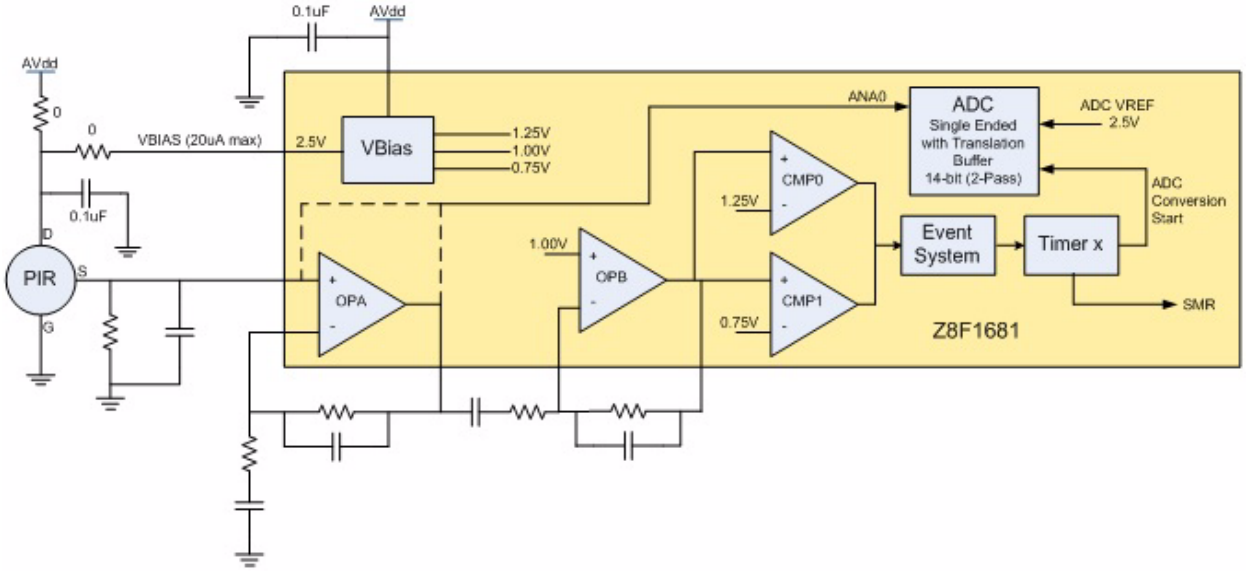

Figure 3. Low Power Analog Wake-up with ZMOTION Digital Validation

Observe the following procedure to initialize the ZMOTION Engine:

1. To initialize the system clock, select the desired clock input and frequency.
2. Initialize any application-specific I/O and peripherals.

3. Set up the Engine API control registers with initial values.
4. Execute the `ZMOTION_Init()` function.
5. Set up op amps, comparators, and internal/external bias generation as analog PIR interface.
6. Set up event system and timer to generate SMR when the analog interface detects motion.
7. Go to sleep mode and wait for the timer to generate an SMR.
8. Upon SMR wake up, configure ADC for single ended input with translation buffer and 2-pass 14-bit mode.
9. Configure another timer to generate an ADC conversion so that one constructed sample is created every 1.5ms.
10. Set the Buffer Refresh bit in ZM_BUFF_CTRL0.

Enable ADC interrupts. After each ADC interrupt, pass the converted ADC value to the ZMOTION Engine using the `ZMOTION_Engine(unsigned int)` function. This can be done inside or outside the ADC interrupt. Zilog recommends you perform this call outside any interrupt.

11. Monitor the Motion Detection bit in ZM_STATUS0.
12. Continue looking for motion for up to ~2 seconds.
13. If after ~2 seconds, no motion is detected, the event was not valid.
14. Return to sleep mode(Step 7).

ZMOTION Library Software Package

The ZMOTION Library Software Package comes with the relevant library files and several sample projects.

The ZMOTION Engine Library and Sample Projects are provided in a compressed zip file and can be downloaded from the Zilog website at www.zilog.com.

File Naming Convention

ZMOTION_Engine_Library_VX.Yz.zip

– ZMOTION Engine Library - Version X.Y

– Sample Projects - Revision z

Installation Folder Structure

[ZMOTION_Engine_Library_V1.0a]

[Sample Projects] - Contains all sample projects. Each project has a local copy of the ZMOTION Library.

[ZMOTION_Engine_V1.0] - Contains Engine Libraries and support files

[include] Header files supporting the Engine Library

[library] Engine Library files

ZMOTION Engine Library Files

[ZMOTION_Engine_V1.0]

[library]

The ZMOTION Engine Library must be included with the application project as a Library Module.

File Name:ZMOTION_Engine_Lib.lib

If using more than one PIR sensor in a multi-sensor application, one additional library must be include for each additional sensor. These additional libraries are used:

File Name:ZMOTION_Engine_Lib1.lib - For first additional sensor

File Name:ZMOTION_Engine_Lib2.lib - For second additional sensor

[include]

The following header files are to be included with the project:

Engine_API.h:
API Register definitions and Engine entry point definitions.

Engine_API1.h:
API Register definitions and Engine entry point definitions for first additional sensor.

Engine_API2.h:
API Register definitions and Engine entry point definitions for second additional sensor.

Engine_API_Def.h:
Engine API register bit definitions

API_INIT_xx.h:
This header file contains the default API settings specific to the lens and pyro-electric sensor being used. The application code loads the API registers with these values prior to executing ZMOTION_Init(). Several versions of this file are available from the Zilog website with tested configurations supporting various lenses and pyroelectric sensors.

One library is required for each PIR sensor being used in the application (i.e. a single library cannot be used to service more than 1 PIR sensor).

Table 26 shows the library and header files that must be included in the project, based on the number of PIR sensors used in the design.

Table 26. Library and Header Files

Number of PIR Sensors	White Light Support	Library Files Included with Project	Header Files
1	No	ZMOTION_Engine_Lib_Vx.x.lib	Engine_API.h Engine_API_Def.h
1	Yes	ZMOTION_Engine_WL_Lib_Vx.x.lib	Engine_API.h Engine_API_Def.h
2	No	ZMOTION_Engine_Lib_Vx.x.lib ZMOTION_Engine1_Lib_Vx.x.lib	Engine_API.h Engine_1_API.h Engine_API_Def.h
3	No	ZMOTION_Engine_Lib_Vx.x.lib ZMOTION_Engine1_Lib_Vx.x.lib ZMOTION_Engine2_Lib_Vx.x.lib	Engine_API.h Engine_1_API.h Engine_API_Def Engine_2_API.h

Note: Vx.x in the library file name indicates the library version number.

API Register Names

When two or more PIR sensors are used in the application, the API register names are modified slightly to indicate the Library for which they are associated. For example, ZM_N_SENSE becomes ZM1_N_SENSE for ZMOTION_Engine_Lib1 or ZM2_N_SENSE for ZMOTION_Engine_Lib2. Refer to the associated Engine_1_API.h file for all register definitions. The register bit definitions remain the same for all libraries.

Example:

In a design using 3 PIR sensors and without support for White Light Detection and Immunity, the following Libraries must be included:

ZMOTION_Engine_Lib First PIR Sensor

ZMOTION_Engine1_Lib Second PIR Sensor

ZMOTION_Engine2_Lib Third PIR Sensor

To access the ZM_N_SENSE API register for the first PIR Sensor, write to ZM_N_SENSE

To access the ZM_N_SENSE API register for the second PIR Sensor, write to ZM1_N_SENSE

To access the ZM_N_SENSE API register for the third PIR Sensor, write to ZM2_N_SENSE

All other API registers follow the same format.

Sample Projects

Several sample projects are included with the ZMOTION Library package. These projects demonstrate how to set up a ZMOTION project in ZDS-II and implement various applications.

Refer to the **ZMOTION Sample Projects** file included in the **Sample Projects** folder for a description of each project.

ZDSII Project Settings

Code Generation

Memory Model: Large

Frames: Dynamic

Parameter Passing: Register

ZDS-II Device Selection

From **General** settings, select the CPU Family and CPU that matches the device you are using. Table 27 lists the MCU part numbers and their details.

Table 27. Device Selection

MCU Part Number	Package	CPU Family	CPU
Z8F1681QK024XK2247	32-QFN	Z8Encore_XP_F6482_Series_16K	Z8F1681XK
Z8F1681QN024XK2247	44-QFN	Z8Encore_XP_F6482_Series_16K	Z8F1681XN
Z8F1681AN024XK2247	44-LQFP	Z8Encore_XP_F6482_Series_16K	Z8F1681XN
Z8F6481QN024XK2247	44-QFN	Z8Encore_XP_F6482_Series_64K	Z8F6481XN
Z8F6481AN024XK2247	44-LQFP	Z8Encore_XP_F6482_Series_64K	Z8F6481XN
Z8F6481AT024XK2247	80-LQFP	Z8Encore_XP_F6482_Series_64K	Z8F6481XT

White Light Detection

Due to the nature of the pyroelectric sensor, sudden large changes in white light will cause a DC shift in the signal output. If the pattern and intensity of the light is just right, the resulting signal can appear the same as a real motion event. Sources such as car headlights have been known to cause this issue. The `ZMOTION_White_Light()` function can be used to monitor this occurrence and automatically compensate the motion detection algorithms to ignore the event. This API function is only available in the ZMOTION_Engine_WL_Lib Library.

The application should pass samples from a light detecting source such as a high efficiency LED or ambient light senor to the `ZMOTION_White_Light()` function as an unsigned 16-bit value. The system should be physically designed such that if light is shining on the light sensor, it is also shining on the PIR sensor. Do not place the light sensor behind a lens with white light filtering.

When there is a sudden change in light level greater than the programmed White Light Threshold value in the ZM_WL_THRESHOLD register, the ZMOTION Engine compensates its detection algorithms for the accompanying signal shift from the PIR sensor. This allows the Engine to suppress false motion events while still detecting real motion events. The application is notified of this event though the White Light Detected bit in ZM_STATUS0. The Engine handles the event and the application code does not need to act on it.

Using an LED for White Light Detection

If an LED is being used in the system in relatively close proximity to the PIR sensor, it can be effectively used as a light sensor to help reduce system cost. White light detection looks for relative changes in the light level and not absolute light levels.

The LED Anode is connected to an ADC input through a resistor. The LED Cathode is connected to system ground. To turn on the LED, configure the pin as General Purpose output and drive it high. To turn off the LED, drive the pin low. To take an ADC measurement, configure the pin as an ADC input, measure the voltage level and pass it to the ZMOTION Engine as an unsigned 16-bit value via the `ZMOTION_White_Light()` function.

Pass the ADC sample to the `ZMOTION_White_Light()` function once every 50 ms.

While most high efficiency LEDs will perform well for this function, the following requirements are placed on the specifications of the LED used in the system:

- Do not place the LED behind any white light filtering material. If it is behind a lens or a light pipe, these materials should be transparent to white light.

- Ensure the light source for the LED comes from the same general direction as the PIR sensor. It is important that the PIR sensor and the LED receive the light at the same time.
- LEDs are available with a large range of electrical specifications. The White Light Threshold register gives the ZMOTION Engine flexibility to work with many LED types; however, LEDs that are more efficient at generating a voltage from a light source typically perform better as a white light detector.
- Most high efficiency LEDs in red, yellow, or green with a forward voltage drop less than 2V @2mA are well suited for white light detection.

Customer Support

To share comments, get your technical questions answered, or report issues you may be experiencing with our products, please visit Zilog's Technical Support page at http://support.zilog.com.

To learn more about this product, find additional documentation, or to discover other facets about Zilog product offerings, please visit the Zilog Knowledge Base at http://zilog.com/kb or consider participating in the Zilog Forum at http://zilog.com/forum.

This publication is subject to replacement by a later edition. To determine whether a later edition exists, please visit the Zilog website at http://www.zilog.com.