ZMOTION® Engine Library
User Manual

UM027503-0621

www.zilog.com

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concerning the devices, applications, or technology described is intended to suggest possible uses and may be superseded.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.

Revision History

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

Date	Revision Level	Description	Page Number
June 2021	03	Complete update. Added descriptions, clarified operation, corrected typos	All
Apr 2018	02	Updated the following sections: Features, Related Documents and Files, Supported ZMOTION MCU Devices, MCU Configurations, ZMOTION Engine Library Software Package, Installation Files and ZDS-II Project Settings.	1, 2, 2, 6, 11, 12, 15
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 specific 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

- Operates on all Zilog ZMOTION Series Microcontrollers
- Low power modes (validation) for battery powered applications
- Two independent motion detection engines running in parallel
  - Normal Engine and Extended Engine
- Support for up to three PIR sensors
- Sensitivity, range, and frequency control
- Pet immunity
- Micro Motion detection
- 2-Pulse mode
- Direction detection
- Transient, Noise and Spark detection and immunity
- White light detection and immunity
- MCU resources remain available for other application functions

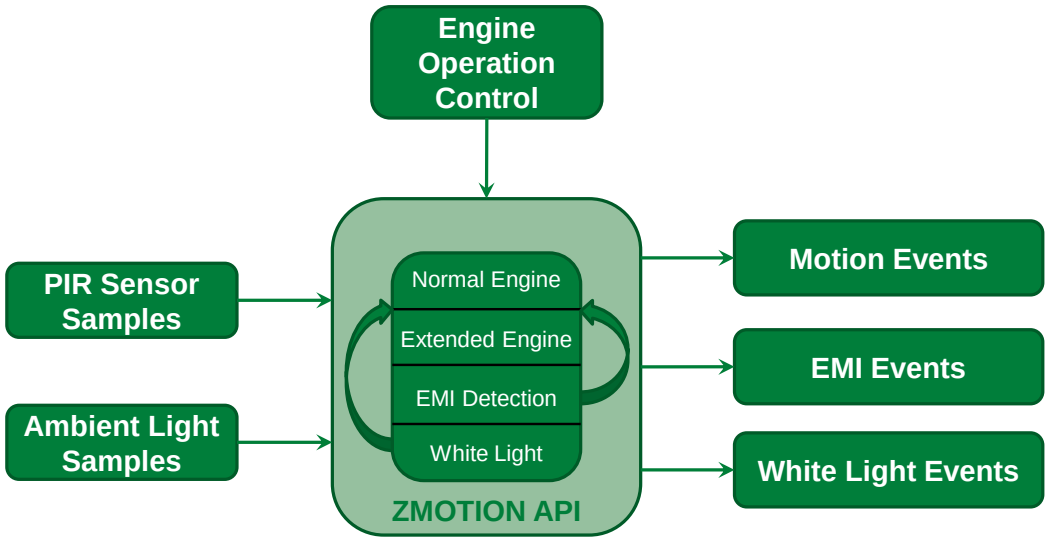

Figure 1. ZMOTION Engine Library

Related Documents and Files

Additional information can be found in the following documents and files. These are available from the Zilog website (www.zilog.com).

Table 1. ZMOTION Library Documentation and Files

Document Number	Description
PS0294	Z8F6482 MCU Series Product Specification
PS0381	Z8F3224 MCU Series Product Specification
PB0258	ZMOTION MCU Product Brief
ZMOTIONL300ZCOG	Z8F6481 ZMOTION Development Kit
ZMOTIONL300ZCOG.exe	Z8F6481 ZMOTION Library and Sample Projects
ZMOTIONL400ZCOG	Z8F3224 ZMOTION Development Kit
ZMOTIONL400ZCOG.exe	Z8F6481 ZMOTION Library and Sample Projects

Supported ZMOTION MCU Devices

The ZMOTION Engine Library is compatible with the Z8F6482 and Z8F3224 series of microcontrollers. The following table shows the supported MCU part numbers.

Table 2. ZMOTION MCU Devices

MCU Part Number	Package	Memory	Product Specification
Z8F1681QK024XK2247	32-Pin QFN	16KB Flash, 2KB RAM	PS0294
Z8F1681QN024XK2247	44-Pin QFN		
Z8F1681AN024XK2247	44-Pin LQFP		
Z8F6481QK024XK2247	32-Pin QFN	64KB Flash, 3.75KB RAM	
Z8F6481QN024XK2247	44-Pin QFN		
Z8F3224QK020XK2258	44-Pin QFN	32KB Flash, 3.75KB RAM	PS0381
Z8F3224QN020XK2258	44-Pin LQFP		

Note:

A specific version of the device must be used with the ZMOTION Library. This is identified by the 2247 or 2258 suffix included in the part number. Device versions without the suffix do not support the ZMOTION Library.

ZMOTION Engine & API

The ZMOTION Engine Library is composed of 4 major components:

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

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

The Library is compiled using the ZDS-II Zilog Developer Studio Integrated Development Environment. This tool is available from the Zilog Website. The library is linked in with the user's application code.

Normal Engine

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

Extended Engine

The Extended Engine runs in parallel with the Normal Engine and is tuned to detect signals that extend beyond typical motion events. Examples are fast-moving targets (generating small, high frequency motion signals) and slow-moving targets (generating small, low frequency motion signals). The Extended Engine is very useful for detection of minor motion events (micro-motion) in lighting control applications. These types of motion signals are also common signatures of false motion sources, so the detection parameters of this engine are controlled independent of the Normal Engine.

Transient, Noise and Spark Detection and Immunity

Signal characteristics that fall outside of 'normal' motion signals like fast moving EMI events and power supply disturbances can be detected and removed from further processing. The Transient, Noise and Spark Detectors perform this function. Transients and Sparks are signals that rise quicker than a valid motion signal would. 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

Light from external sources like headlights or sunlight can cause Pyro Electric Sensors to generate signals that look identical to valid motion signals. This is typically caused by the localized heating effect from the light shining on the lens or pyro electric sensor itself. The White Light Detection and Immunity function is used to eliminate false detections caused by these events.

A light sensing device is placed near the pyroelectric sensor and sampled at regular intervals. When a significant change in light level is detected (configurable in the API), the Motion Detection Engine is compensated in anticipation of the simultaneous signal variation from the pyroelectric sensor.

Some Intrusion/Security motion detector standards require that the device be immune to this source of false motion events.

ZMOTION Engine Entry Points

There are three entry point functions to the ZMOTION Engine. None of the functions return a value – they are all void type.

ZMOTION_Init (void);

This routine has a dual function depending on the setting of the Buffer Refresh Bit in ZMOTION Buffer Control 0 register.

ZM_BUFF_CTRL0[Buffer Refresh] = 0

This is used for full initialization of the Engine and must be executed one time after power-on once the API registers have been initialized. This prepares the Engine to monitor the PIR sensor for stability and all other functions.

ZM_BUFF_CTRL0[Buffer Refresh] = 1

This causes the Engine to refresh its sample buffer and restart any validation stages currently in process. The Engine will refill its sample buffer at a faster rate defined by the Fast Buffer Refill Rate field in ZMOTION Buffer Control 0 register. This is used in low power mode to allow the Engine to quickly validate a motion signal.

ZMOTION_Engine (unsigned int);

PIR sensor samples are passed to the Engine through this function. All motion detection functions (normal and extended engines), and Transient, Noise and Spark detection processing are performed by this function. API registers are updated after each constructed sample. These values are used to create the constructed sample processed by the Engine. The constructed sample is a 16-bit value created by adding the PIR sensor sample values, so the maximum value passed to the Engine depends on the ZM_SAMPLE_SIZE selected in the ZMOTION Sample Size API register.

ZM_SAMPLE_SIZE	Maximum value passed to Engine from each PIR sample
1	0xFFFF (16-bit value)
2	0x7FFF (15-bit value)
4	0x3FFF (14-bit value)

ZMOTION_White_Light (unsigned int);

Samples from a light sensing device are passed through this function. 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 these signals.

Engine Timer Tick

The Engine Timer Tick bit (ZM_CTRL0[0]) must be set once per second to provide a 1 second time base for the ZMOTION Engine. The Engine uses this to perform house-keeping 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 Usage

This section describes how to use the ZMOTION Engine Library. Sample application projects are available for each of the supported configurations.

The application code first initializes the ZMOTION API registers then calls the initialization function ZMOTION_Init(void).

After initialization, samples from the PIR sensor are acquired and passed to the Engine through the ZMOTION_Engine(unsigned int) function. This is done for each PIR sensor sample. The Engine creates a Constructed Sample which is comprised of 1, 2 or 4 PIR samples as defined in the ZM_SAMPLE_SIZE API register. PIR samples should be passed to the engine such that a Constructed Sample is generated every 1.5ms to 3.0ms.

Each constructed sample is processed by the Normal and Extended engines for motion detection, and examined for Transient, Noise and Spark events. The API status registers are updated after each constructed sample.

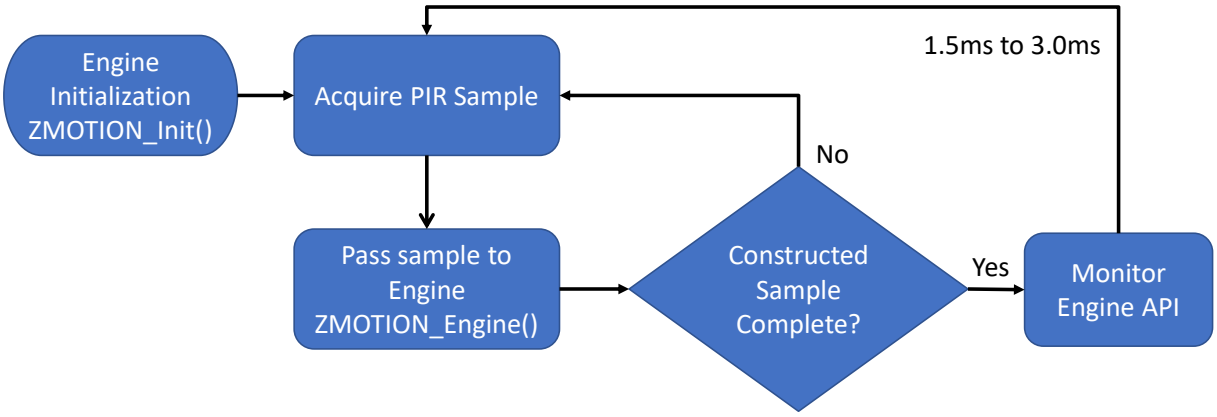

Figure 2. NP Mode Operation

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. This is done at a typical rate of 1 sample every 50ms. The White Light Engine looks for sudden changes in signal level and compensates the Normal Detection Engine to eliminate the possible false motion event. Refer to the

White Light Detection section for more information.

The application must set the Engine Timer Tick bit in the ZM_CTRL0 register once per second.

All Engine parameters can be modified in real time. The application monitors the API for validated motion and other events while performing its other application functions.

MCU Configurations

This section describes the MCU configuration supported by the ZMOTION Engine Library.

The ZMOTION library can be used with two MCU configurations called Normal Power (NP) and Low Power (LP). Generally, NP configuration runs the Engine continuously while LP only runs the Engine as a validation of a motion event detected by a separate low power circuit.

Sample application projects for both NP and LP configurations are included with the Library installation.

Normal Power (NP) Configuration

In this configuration, PIR samples are passed to the ZMOTION Engine continuously at a rate that provides a constructed sample every 1.5m to 3.0ms. Using the NP configuration, 1, 2 or 3 PIR sensors can be supported by passing samples from each PIR sensor to individual Engines included as separate libraries.

Refer to the Application Configuration section for more information on which libraries files are required for 1, 2 or 3 PIR sensor configurations.

Normal Power Example

Referring to Figure 3, the PIR Sensor is AC coupled to an Op Amp (inverting amplifier) with programmable gain and referenced to 1.0V. The AC coupling removes the PIR sensor's DC offset which typically ranges from a few hundred millivolts up to 1.5V. The Op Amp output is connected to the input of a 14-bit ADC. A timer is used to initiate conversions from the ADC at regular intervals. When the ADC completes a conversion, it generates an interrupt, and the sample is stored in a buffer – the sample is not passed to the Engine in the ADC interrupt to minimize processing in the interrupt context. The PIR sensor samples are passed to the ZMOTION Engine in main() or some other non-interrupt context. After the Engine processes the sample, the API is monitored for a detected motion event.

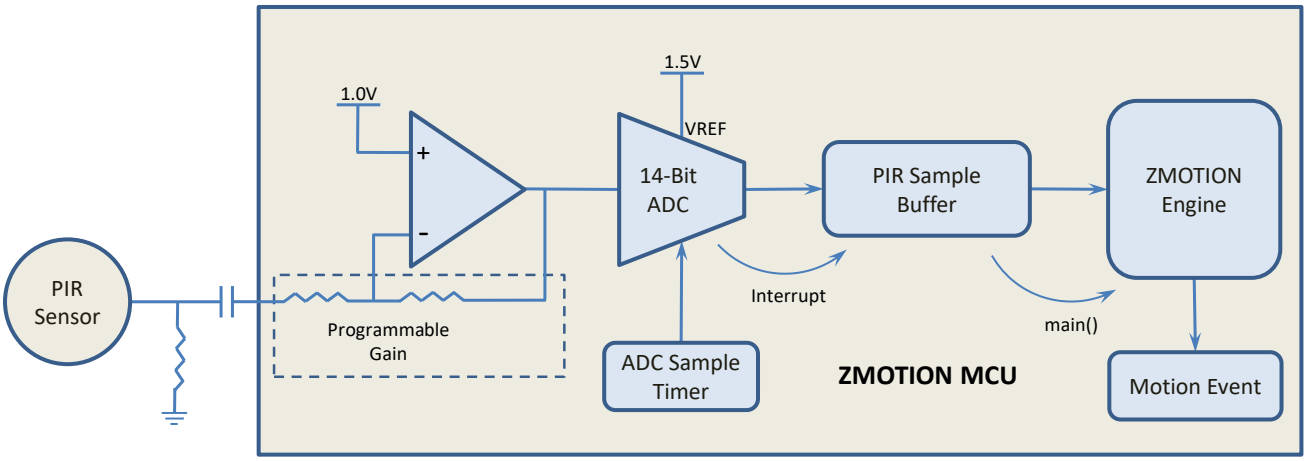

Figure 3. NP Mode MCU Configuration

To reduce power consumption, the MCU stays in Halt mode until the ADC, or some other interrupt wakes it up to perform the necessary processing.

Low Power (LP) Configuration

To support Low Power operation, the ZMOTION Engine supports a mode where the Engine is used to validate a motion event initially detected by a separate detection source. Typically, this separate detection source would be a traditional low power analog PIR detection circuit which would act as a wakeup signal for the MCU. These analog circuits are prone to false motion events from EMI, pets, temperature changes and air currents. The ZMOTION Engine can be used to help eliminate these false motion event sources.

The MCU remains in a low power idle state while waiting for a wake-up event from the PIR wake-up circuit as shown below.

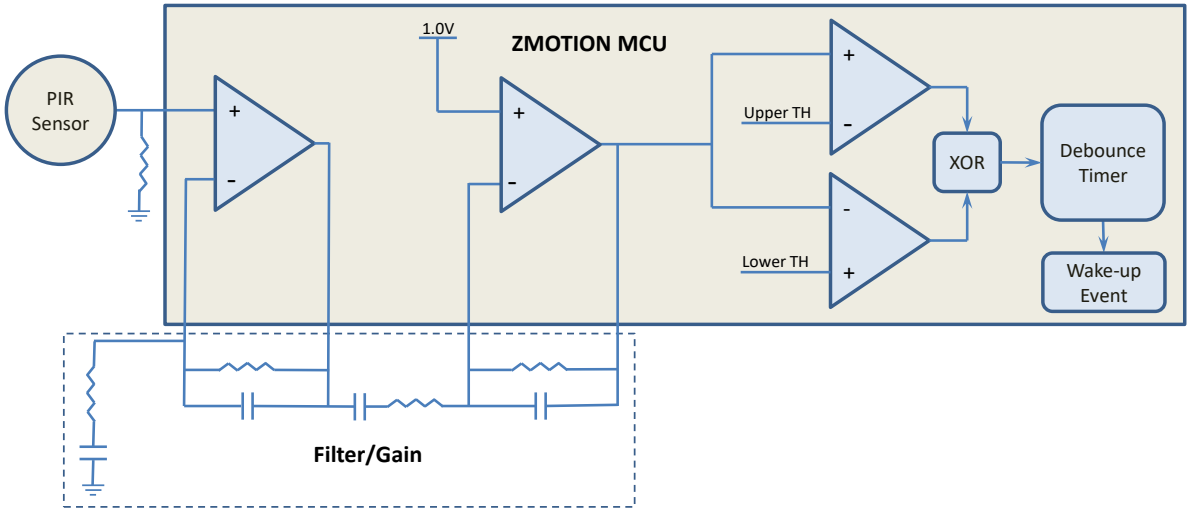

Figure 4. LP Mode MCU Configuration

When the MCU wakes up, it reconfigures the PIR sensor interface to the same configuration as used in NP mode but using unity gain for the Op Amp. Unity gain must be used since the PIR sensor is not AC coupled. The Fast Buffer Refill bit is set in the API causing the Engine to quickly fills its sample buffer and performs a validation on the signal to eliminate false sources and provide improved pet immunity.

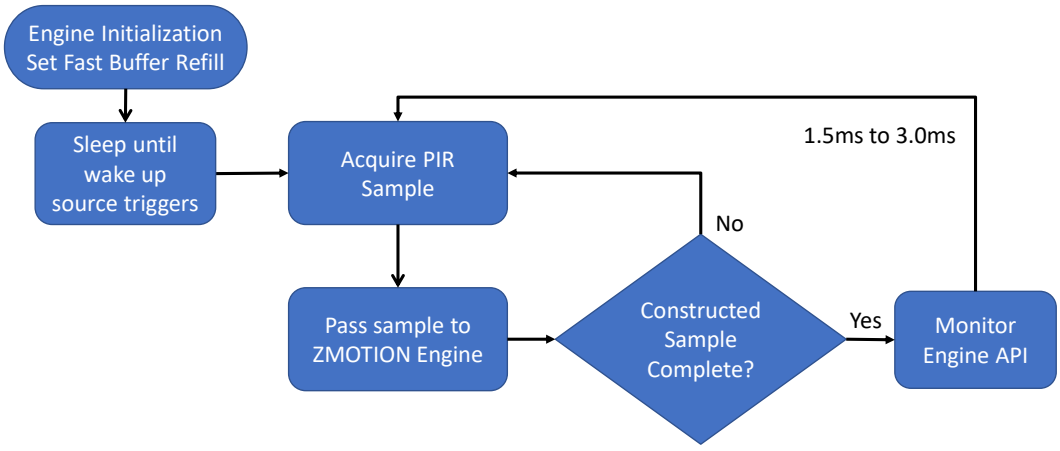

Figure 5. LP Mode Operation

White Light Detection

Sudden large changes in the light energy shining on a PIR sensor 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 valid motion event. The *ZMOTION_White_Light*() function can be used to monitor this occurrence and automatically compensate the motion detection algorithms to ignore the event.

The application should pass samples from a light detecting source such as a high efficiency LED or photo diode to the *ZMOTION_White_Light()* function. The system should be designed such that any light shining on the PIR sensor also shines on the light detecting device.

When there is a sudden change in light level greater than the programmed White Light Threshold value in ZMOTION White Light Threshold register, the Engine compensates its motion detection algorithms for the accompanying signal shift from the PIR sensor. This allows it to suppress any false motion events while still detecting valid motion events. The application is notified of this event though the White Light Detected bit in the ZMOTION Status 0 register. The application code does not need to act on this event – the status bit is provided to aid with development and tuning.

Using an LED for White Light Detection

If an LED is being used by the system and that LED is in relatively close to the PIR sensor, it can be used effectively as a light sensor. The white light detection looks for relative changes in the light level and not absolute light levels.

The design should have the LED Anode connected to an ADC input through a resistor. The LED Cathode is connected to system ground. To turn on the LED, the pin should be configured as an output and driven high. To turn off the LED, the pin is driven low. To take an ADC measurement, the pin is configured as an ADC input, the voltage level is measured and passed to the ZMOTION Engine via the *ZMOTION_White_Light()* function.

The ADC sample should be passed to the *ZMOTION_White_Light()* function once every 50ms.

While most high efficiency LEDs will perform well for this function, there are certain requirements 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 is coming 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.
- LED's are available with a large range of electrical specifications. The ZMOTION White Light Threshold register gives the user flexibility to work with many LED types, but generally any high-efficiency LED in red, yellow or green with a forward voltage drop less than ~2V @2mA is well suited for white light detection.

ZMOTION Engine CPU Stack Usage

The ZMOTION Engine shares the stack with the user application. There are no special requirements for the placement of the stack in memory, but 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 Engine Operation

The ZMOTION Engine processes PIR sensor samples passed to it through the ZMOTION_Engine() function. There are 2 motion detection Engines that run in parallel and process the same sample. The purpose of using 2 engines is to detect different types of motion which can be controlled separately.

The Normal Engine is the primary motion detection engine used in the application. It is designed to detect all normal motion events with very good false event rejection. It is also capable of providing pet immunity.

The Extended Engine is a secondary engine designed to detect motion that extends beyond typical motion events - small movement, fast or very slow movements. It is intended for micro-motion detection. Having separate control over its operation means that the application can desensitize or disable the Extended Engine when it's not needed, helping to reduce susceptibility to false motion events.

The PIR sensor samples are also processed by the transient, noise and spark detectors which then automatically compensate the Normal and Extended motion detection Engines.

Ambient light samples passed through the ZMOTION_White_Light() function are processed separately from the PIR sensor samples. The White Light detection process compensates the Normal and Extended motion detection Engines when white light events are detected.

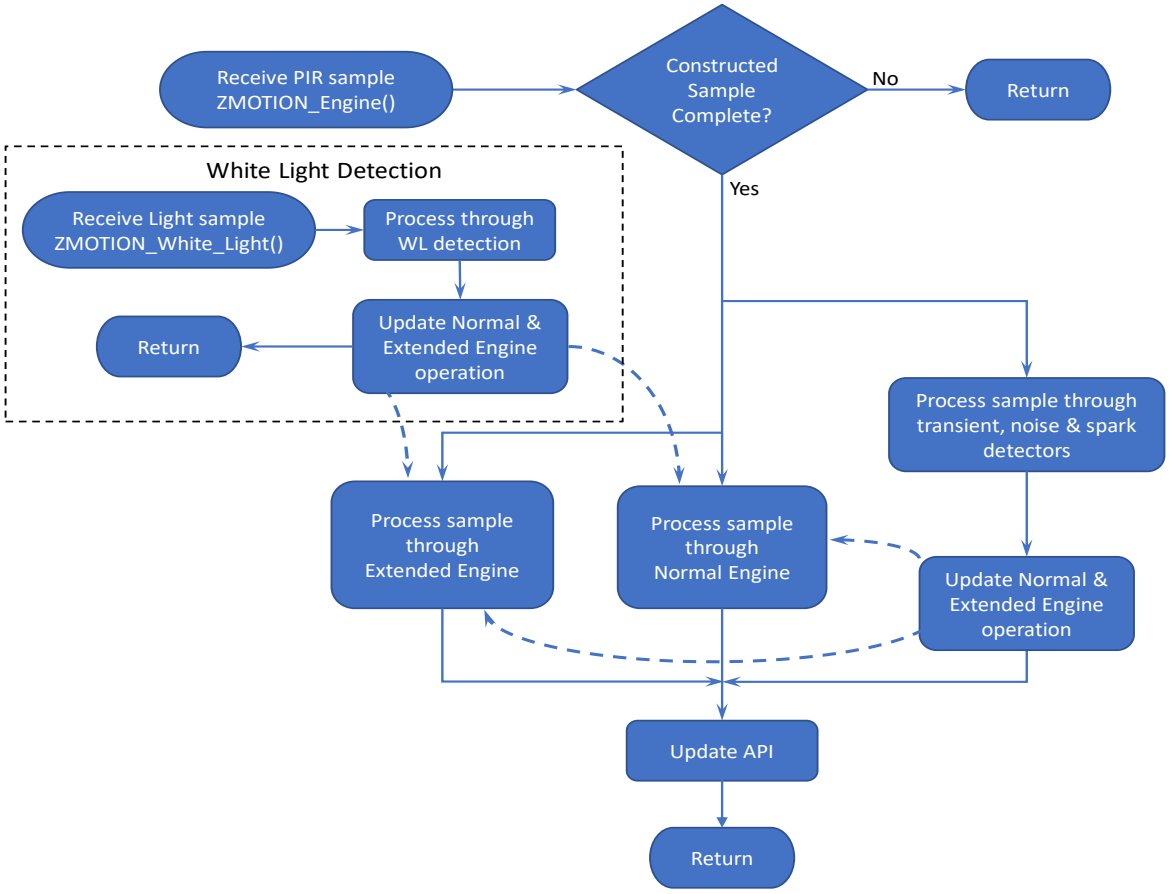

Figure 6. ZMOTION Engine Flow

Engine Overhead

The Normal Engine has 3 phases of operation: PIR Sample Phase, Constructed Sample Phase and Window Calculation Phase. The processing time of each phase can vary based on the API settings.

PIR Sample Phase

Each time a sample is passed to the Engine through the ZMOTION_Engine() function, it is processed into the current constructed sample.

CPU Time: 68 clocks

Constructed Sample Phase

When a constructed sample is generated (every ZM_SAMPLE_SIZE PIR samples), it is processed through the motion detection engines, and if enabled, transient, noise and spark detection is also processed at this time. The Normal Engine always runs, so the processing time associated with it is always incurred. If other functions are enabled, their associated processing time is added.

CPU Time:

Normal Engine: 504 clocks

Extended Engine: 250 clocks

Transient Detection: 192 clocks

Noise Detection: 12 clocks

Spark Detection: 185 clocks

Window Calculation Phase

The control limits and other motion detection parameters are updated based on the Window Size and Update Rate fields in ZM_CTRL2. The time taken to perform these calculations depends on the Window Size value and the rate at which it will occur depends on the Update Rate value.

CPU Time:

Window Size Large: 10,027 clocks

Window Size Medium: 5,765 clocks

Window Size Small: 4,010 clocks

Engine Overhead Example

In a typical configuration, the Engine API would be configured as follows:

A PIR sample is passed to the Engine every 485us.

Constructed Sample Size = 4

Window Size = Large

Window Update Rate = 1

Transient Detection = Disabled

Extended Engine = Disabled

In Figure 7, a Constructed Sample is calculated every 4th PIR Sample (Constructed Sample Size = 4) and the Window parameters are updated every 7th constructed sample (5 + Window Update Rate * 2).

In this example, the CPU clock rate is set to 2.785MHz – this is the lowest recommended operating frequency. The Engine uses approximately 5.2ms every 16.4ms = 31.7% overhead. At higher CPU clock frequencies, the CPU overhead is reduced.

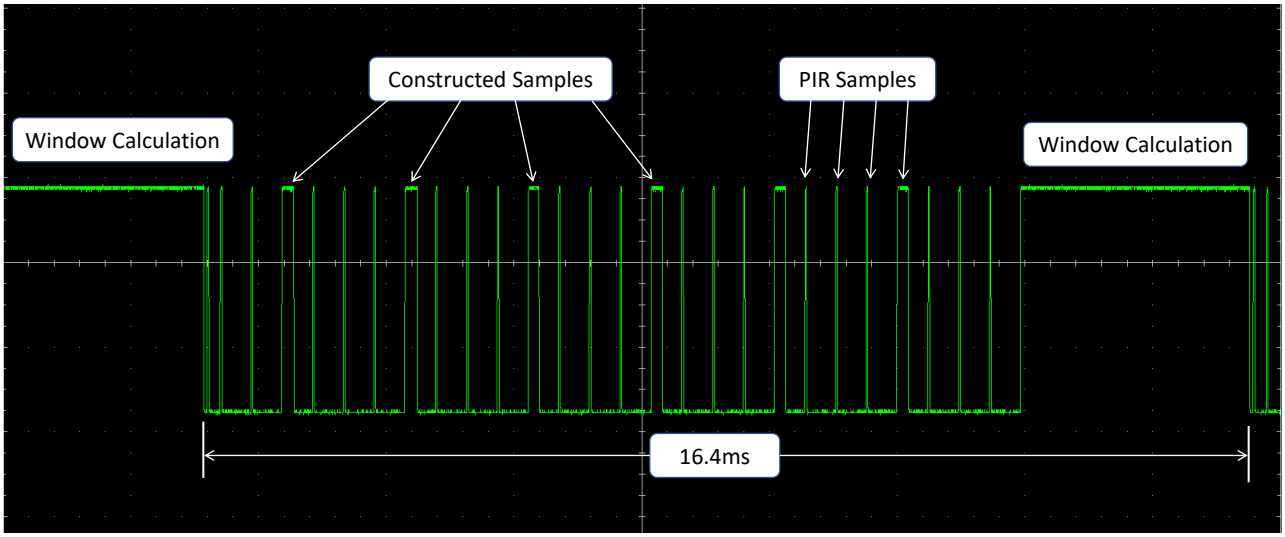

Figure 7. Engine Overhead

ZMOTION Engine Library Software Package

The ZMOTION Library Software Package comes with the relevant library files, sample projects, documentation, and reference schematics. These are provided in an installation file and can be downloaded from the ZMOTION Development Software section of the Zilog website.

Installation Files

The installation files are associated to the MCU development kits.

Table 3. ZMOTION Library Installation File

Development Kit	Supported MCU	Library Installation File Name
ZMOTIONL300ZCOG	Z8F6481 Series	ZMOTIONL300ZCOG_x.y.exe
ZMOTIONL400ZCOG	Z8F3224 Series	ZMOTIONL400ZCOG_x.y.exe

The "x.y" indicates the Library Installation File version number. The version of the ZMOTION Engine Library is separate from this.

Executing the installation file will create a folder that includes the ZMOTION Engine Library, reference schematics, sample applications, related documentation and a VCOM USB driver to be used with the associated development kit.

After installation the following folder structure will be created (ZMOTIONL300ZCOG_x.y.exe):

\Zilog\ZMOTIONL300ZCOG_x.y

Documentation

Drivers

Z8F6481 ZMOTION Library

Reference Schematics

Sample Applications

ZMOTION Engine Library

ZMOTION Engine Library Files

The files included with the ZMOTION Engine Library are shown in Table 4. The “Vx” in the file name indicates the library version number.

Table 4. ZMOTION Engine Library Files

File Name	Description
ZMOTION_Engine_Vx.lib	ZMOTION Engine Library supporting first PIR sensor – Version x
ZMOTION_Engine_WL_Vx.lib	ZMOTION Engine Library with white light immunity supporting first PIR sensor – Version x
ZMOTION_Engine_1_Vx.lib	ZMOTION Engine Library supporting second PIR sensor – Version x
ZMOTION_Engine_2_Vx.lib	ZMOTION Engine Library supporting third PIR sensor – Version x
Engine_API_Def.h	API register bit definitions – common to all libraries
Engine_API.h	API register and entry point definitions for ZMOTION_Engine_Lib_Vx.lib and ZMOTION_Engine_WL_Lib_Vx.lib
Engine_1_API.h	API register and entry point definitions for ZMOTION_Engine_Lib1_Vx.lib
Engine_2_API.h	API register and entry point definitions for ZMOTION_Engine_Lib2_Vx.lib

Application Configuration

Depending on the number of PIR sensors being supported by the application, the following files should be included in the project.

Table 5. Library Files to Include in Project

1 PIR Sensor	
Library File Name	**Header File**
ZMOTION_Engine_Vx.lib	Engine_API.h
	Engine_API_Def.h
2 PIR Sensors	
Library File Name	**Header File**
ZMOTION_Engine_Vx.lib	Engine_API.h
ZMOTION_Engine1_Vx.lib	Engine1_API.h
	Engine_API_Def.h
3 PIR Sensors	
Library File Name	**Header File**
ZMOTION_Engine_Vx.lib	Engine_API.h
ZMOTION_Engine1_Vx.lib	Engine1_API.h
ZMOTION_Engine2_Vx.lib	Engine2_API.h
	Engine_API_Def.h

ZMOTION Engine Memory Usage

The memory used by each library is shown in Table 6.

Table 6. ZMOTION Library Memory Requirements

File Name	Code Space	RAM – Near/Far
ZMOTION_Engine_Vx.lib	3,561 Bytes	59 Bytes/641 Bytes
ZMOTION_Engine_WL_Vx.lib	4,089 Bytes	68 Bytes/647 Bytes
ZMOTION_Engine_1_Vx.lib	3,561 Bytes	59 Bytes/641 Bytes
ZMOTION_Engine_2_Vx.lib	3,561 Bytes	59 Bytes/641 Bytes

ZDS-II Project Settings

ZDS-II should be set up with the following Project Settings:

Objects and Libraries

Add the library files required by the application to the project:

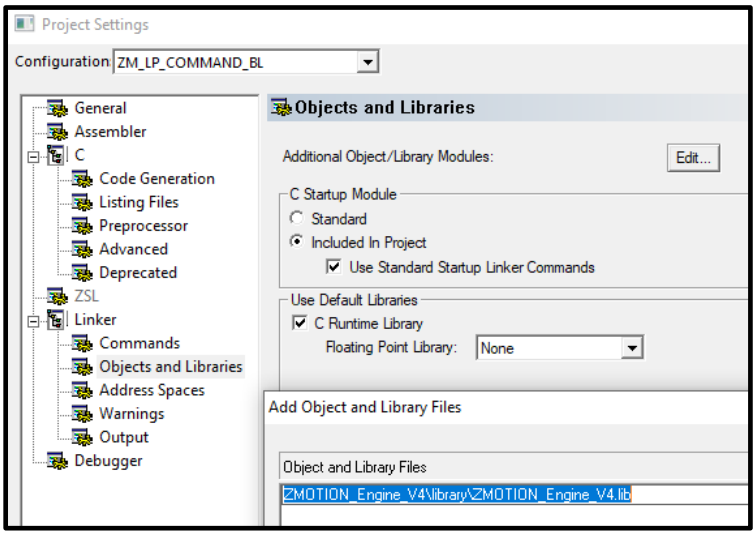

Figure 8. Adding Library to Project

Code Generation

Memory Model: Large

Frames: Dynamic

Parameter Passing: Register

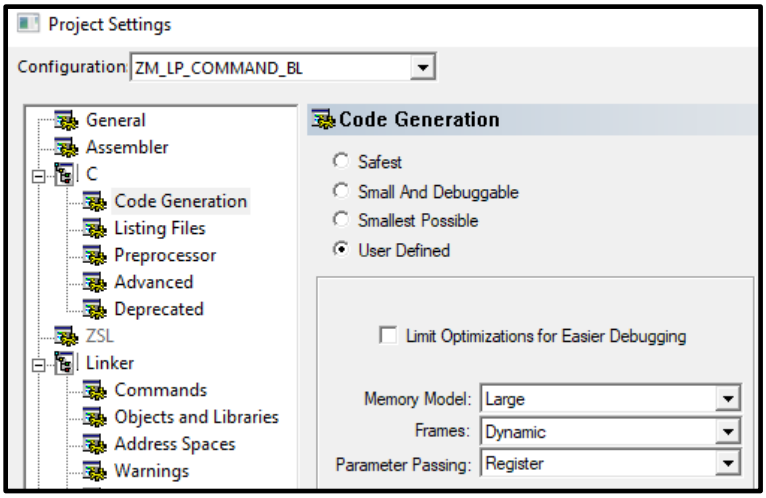

Figure 9. Code Generation

Preprocessor Definitions

Include a definition ZMOTION_ENGINE_LIB in the preprocessor definitions:

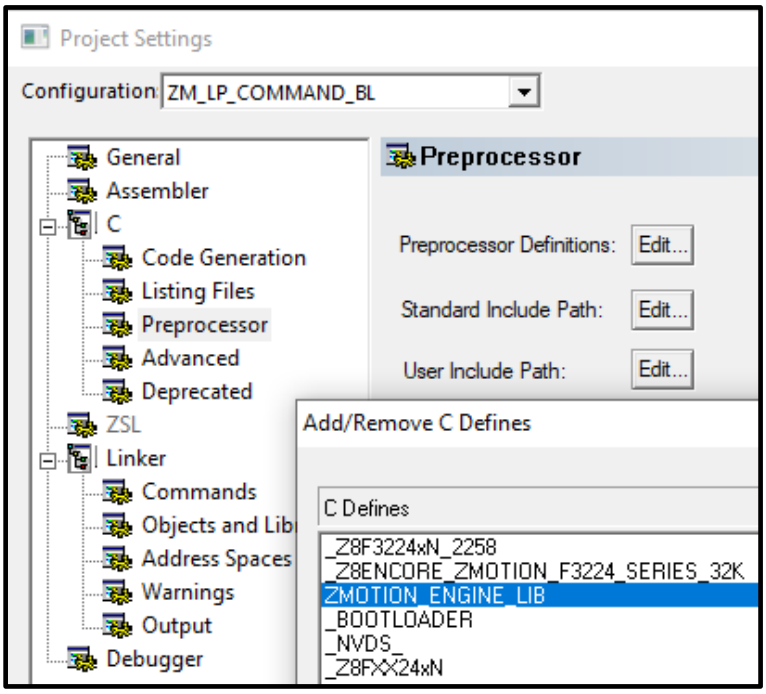

Figure 10. Preprocessor Definitions

ZMOTION API

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

API Register Names

The API register names used in Table 7 and the register descriptions following are based on a design using a single PIR sensor.

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, the ZM_N_SENSE API register is defined as follows:

Register Name	Library
ZM_N_SENSE	ZMOTION_Engine_Vx.lib
ZM1_N_SENSE	ZMOTION_Engine1_Vx.lib
ZM2_N_SENSE	ZMOTION_Engine2_Vx.lib

Refer to the associated Engine_API.h, Engine_1_API.h or Engine_2_API.h files for all register name definitions. The register bit definitions remain the same for all libraries.

Table 7. ZMOTION API Register Summary

API Register Name	Library API Name	Description
ZMOTION Engine Status and Control		
Status 0	ZM_STATUS0	Engine status
Control 0	ZM_CTRL0	Engine operation mode control
Sample Size	ZM_SAMPLE_SIZE	Number of PIR sensor samples used to create a full constructed sample
Buffer Control 0	ZM_BUFF_CTRL0	Fast buffer refill control
Process Rate	ZM_PROCESS_RATE	Processed samples per second
Library Type and Version	ZM_LIB_VERSION	Library Type and Version
ZMOTION Normal Engine Configuration		
Normal Detection Sensitivity	ZM_N_SENSE	Sensitivity control
Control 1	ZM_CTRL1	Frequency and Range settings
Control 2	ZM_CTRL2	Controls window configuration parameters
Debounce Timeout	ZM_DEB_TIMEOUT	Time to de-bounce initial motion signal
Debounce Batch	ZM_DEB_BATCH	Out of window samples needed validate initial de-bounce
ZMOTION Extended Engine Configuration		
Extended Detection Control Register 0	ZM_EXT_CTRL0	Sensitivity control
Extended Detection Control Register 1	ZM_EXT_CTRL1	Range and Debounce settings
Sensor Signal Information		
Signal Level	ZM_SIGNAL	Current constructed sample
DC Signal Level	ZM_SIGNAL_DC	Sensor signal DC offset
EMC Immunity Settings		
Transient Sensitivity Level	ZM_TRANSIENT_SENSE	Transient detection sensitivity
Noise Sensitivity Level	ZM_NOISE_SENSE	Noise detection sensitivity
Spark Sensitivity Level	ZM_SPARK_DETECT	Spark detection gap and sensitivity
White Light Immunity Control		
White Light Threshold	ZM_WL_THRESHOLD	White Light immunity sensitivity
White Light Control 0	ZM_WL_CTRL0	White Light Debounce and Control

ZMOTION Status 0

The Status 0 Register, shown in Table 8, reports the status of the Engine.

Table 8. 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 event detected.
Motion Detected	Set by ZMOTION Engine; cleared by application.
	This bit indicates that the ZMOTION Engine has detected or 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]	Origin of last motion detection event.
MD Origin	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
[5]	Relative direction of last motion event.
Motion Direction	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.

Bit	Description
[4] **PIR Stable**	Passive infrared (PIR) sensor stabilized. 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. 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. Note: This bit may occasionally reset after motion events or other disturbances. This is not an indication that the sensor is unstable. 0 = PIR sensor signal is not stable; motion detected events are not valid 1 = PIR sensor signal is stable; motion detected events are valid.
[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 is not needed for normal operation but is available to allow the application to monitor the constructed samples.
[2] **EM Spark Detected**	Spark event 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 sensitivity to Spark events can be adjusted with the ZMOTION Spark Detection (ZM_SPARK_DETECT) register.
[1] **EM Noise Detected**	Noise event detected. Set by ZMOTION Engine; cleared by application. This bit indicates if the Engine has detected noise on 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 sensitivity to Noise events can be adjusted with the ZMOTION Noise Sensitivity Level (ZM_NOISE_SENSE) register.
[0] **EM Transient Detected**	Transient event detected. Set by ZMOTION Engine; cleared by application. This bit indicates if the Engine has detected a transient on 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 sensitivity to Transient events can be adjusted with the ZMOTION Transient Sensitivity Level (ZM_TRANSIENT_SENSE) register.

ZMOTION Normal Motion Sensitivity

The Normal Motion Sensitivity Register, shown in Table 9, is used to control the sensitivity of the Normal Motion Engine.

Table 9. ZMOTION Normal Motion Sensitivity (ZM_N_SENSE)

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

Bit	Description
[7:0]	Normal motion detector sensitivity level.
Normal Motion Sensitivity	Controlled by application. The Normal Motion 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 (in constructed samples) that a partially validated motion signal must remain valid. Lower values provide higher sensitivity to motion with 0x00 being the most sensitive and 0xFF being the least sensitive. The user application should load this register with the appropriate value to give the desired sensitivity. In most applications values between 6 and 40 are typical, with 8 to 20 being the most common. Changes to low values affect motion sensitivity more than changes to high values. For example, changing from 10 to 8 may make a very noticeable difference, while a change from 30 to 24 will be less noticeable. Values larger than the ZMOTION Debounce Timeout will result in no motion detection. Note: The value in this register may also affect the detection distance/range.

ZMOTION Control 0

The Control 0 register, shown in Table 10, controls the operation mode of the Engine.

Table 10. ZMOTION Control 0 (ZM_CTRL0)

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

Bit	Description
[7] **Motion Detection Suspend**	Suspend motion detection. Controlled by application. Temporarily suspends motion detection. When this bit is set, samples passed through the ZMOTION_Engine() function are held in a circular buffer, but not processed. This puts it in a low processing overhead state and can be used when the application needs to disable motion detection. When the application clears this bit, suspend mode is exited upon the next call of the ZMOTION_Engine() function and the samples in the buffer are processed. 0 = Normal Motion Detection 1 = Suspended Motion Detection
[6] **Motion Direction Enable**	Motion direction detection control. Controlled by application. This bit enables directional motion detection. When this bit is set, 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 direction of motion is determined by the direction of the PIR signal (rising/falling) when the motion event is detected and is dependent on the polarity of the PIR sensor. Zilog recommends using a polarity matched PIR sensor or performing a factory calibration during manufacture to use this feature. Note: Enabling motion direction control also disables extended detection. Only the normal motion detector runs while directional detection is enabled.

Bit	Description
[5]	2-Pulse detection control.
2-Pulse 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 to generate a valid motion event. 2-Pulse mode is typically used in harsh environmental conditions to decrease the chances of false triggers from sources such as fast heating and cooling or air movement. 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]	One second engine timer tick.
Engine Timer Tick	Set by Application; cleared by ZMOTION Engine. This bit must be set once per second by the application to provide the Engine with a one-second time base used to perform housekeeping operations. The Engine routinely polls this bit to obtain a one-second tick. A tolerance of ± 10% on the timing of this bit is acceptable. 0 = Cleared by ZMOTION Engine 1 = One-second interval has occurred.

ZMOTION Control 1

The Control 1 Register, shown in Table 11, reports the status of the Engine.

Table 11. 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 of ZMOTION Engine.
Frequency Response	Controlled by application. 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, it determines the location of the sample window used to calculate the control limits relative to the current sample. By using a smaller frequency response value, the window is closer to the current sample reducing sensitivity to signals with slower rates of change. By using a larger frequency response value, the window is moved farther from the current sample increasing the sensitivity signals with slower rates of change. Valid values are 0h to Fh or Ch depending on the window size selected in ZM_CTRL2. In most applications, values between 3h and Ch are typical, with 8h to Ah being the most common for non-pet immune applications. Use lower values to improve motion stability, particularly to environmental temperature changes, airflow, and pets. The maximum Frequency Response values for each window size can be summarized as:

ZM_CTRL2 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.

Bit	Description
[3:0]	Motion detection range control.
Range Control	Controlled by application. These bits determine the relative range of motion detection by performing a relative amplitude check on the PIR signal. The amplitude is relative to the average DC level of the sensor signal. A smaller range value allows motion signals with smaller relative amplitudes to be qualified. Since targets further from the sensor produce smaller relative signals, this has the effect of increasing the range of detection, while larger values decrease the range of detection. Valid values are 0-F. Values used for this setting are dependent on the lens and pyroelectric sensor being used, but values between 2h and Fh are typical. Differences in lower Range values are more pronounced than differences in higher Range values. For example, changing the Range value from 3 to 2 may produce a more noticeable difference than changing from 9 to 7. 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 smaller target will be rejected at the same distance under different conditions.

ZMOTION Control 2

The Control 2 register, shown in Table 12, controls the Control Limit Window operation.

Table 12. 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]	Control limit lock level.
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 VCC 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 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 detection range, make this number smaller. To increase stability, make this number larger. Valid values are 0-7. Values between 1 and 3 are typical. 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.
[4:3]	Control limit window size.
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. A large window size is typically used. 00 - Reserved 01 - Small window 10 - Medium window 11 - Large window

Bit	Description
[2:0]	Control limit window update rate.
Window 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. Typical values used are between 0 and 3. The window control limits are updated every 4 + (Window Update Rate * 2) constructed samples.

ZMOTION Sample Size

The Sample Size Register, shown in controls the number of individual PIR samples needed to create a constructed sample.

Table 13. 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
[7]	Number of PIR samples used to generate a Constructed Sample.
Sample Size	Controlled by application. This register controls the number of samples used to create a constructed sample. 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 Buffer Control 0 Register, shown in Table 8, is used for refreshing the buffer of constructed samples.

Table 14. ZMOTION Buffer Control 0 (ZM_BUFF_CTRL0)

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

Bit	Description
[7]	Force fast fill algorithm to quickly refill the motion detection buffers.
Buffer Refresh	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 the MCU from Stop mode (SMR - Stop Mode Recovery). Upon SMR, the application sets the Buffer Refresh bit, executes the ZMOTION_Engine() function, and then continues with other functions, while polling the Motion Detected bit in ZMOTION Status 0 (ZM_STATUS0) for no more than 2 seconds 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.

Bit	Description
[6]	Rate at which to refill the buffers when the Buffer Refresh bit is set.
Fast Buffer Refill Rate	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 holds 256 constructed samples. For example, when the buffer refresh bit is set and the Fast Buffer Refill Rate is set to 1, each constructed sample will be saved to the buffer twice so that only 128 constructed samples are needed to fill the buffer and start looking for motion. If Fast Buffer Refill is set to 3, the constructed sample is saved to 4 locations in the buffer and only 64 constructed samples are needed to fill the buffer and start looking for motion.

ZMOTION Debounce Timeout

The Debounce Timeout register, shown in Table 15, controls the time that the Normal Motion Detection Engine will wait to debounce the initial motion signal.

Table 15. ZMOTION Debounce Timeout (ZM_DEB_TIMEOUT)

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

Bit	Description
[7:0]	Time to wait for the Debounce Batch to qualify.
Debounce Timeout	Controlled by application. This value works in conjunction with the Debounce Batch Size register and controls the time that the Normal Motion Detection Engine will wait to fully debounce a motion signal as part of the pre-qualification phase. Longer times result in detection of subtle motion at the cost of additional potential false motion detections. Valid range is from 01h to FFh and uses constructed samples as the time base. 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 Debounce Batch Size register, shown in Table 16, is a mask used to control the number of out-of-window samples required to initially debounce the signal.

Table 16. ZMOTION Debounce Batch Size (ZM_DEB_BATCH)

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

Bit	Description
[7:0]	Prequalification count of constructed samples outside of control limits
Debounce Batch Size	Controlled by application. This register determines the number of consecutive samples required to be outside of the adjusted controls limits 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 Control 0

The Extended Control Register 0, shown in Table 17, sets the sensitivity levels for the Extended Engine.

The Extended Detection Engine can detect very small signals and is therefore useful for applications requiring micro-motion detection. This sensitivity to small signals has a drawback that it can generate false detections in response to environmental influences like temperature drift, airflow, or sunlight.

Table 17. ZMOTION Extended Control 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]	Sets the overall sensitivity level of extended detector.
Extended Level	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 level is selected to provide a balance between additional sensitivity while maintaining stability (no false detections). The valid range is 1 (most sensitive) to 7 (least sensitive) with typical values between 3 and 6. A value of 0 disables the Extended Detection Engine. 000 = Extended Detection Level 0 - Highest Sensitivity … 111 = Extended Detection Level 7 - Lowest Sensitivity.
[6]	Extended Motion Engine Sensitivity
Extended 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 Control 1

The Extended Control Register 1, shown in Table 18, sets the range and debounce time for the Extended Engine.

Table 18. ZMOTION Extended Control 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]	Sets the range of the extended engine.
Extended Range	Controlled by application. The Extended Range field is used to adjust the effective range of the extended detection engine. This value controls how much the current constructed sample must exceed the average DC level prior to starting the debounce phase. A smaller extended range value allows a smaller amplitude signal to begin debounce, thereby increasing the range of detection. A larger extended range value decreases the range of detection. The valid range of this field is 0 (farthest range) to 7 (shortest range) with values between 2 and 4 being typical. 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. 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.
[6]	Time allowed for the extended engine to debounce a valid signal.
Extended Debounce Timeout	Controlled by application. This field controls the amount of time the Extended Detection engine will wait to fully debounce a fast-moving or subtle motion signal. Higher values result in detection of subtle motion at the cost of higher number of potential false motion detections. The valid range of this filed is from 1 to 31 with typical values between 10 and 26. The value used must be greater than 50% of the value used for Extended Sensitivity in ZM_EXT_CTRL0. Using a value less than 50% of the Extended Sensitivity will prevent the Extended Engine from completing a debounce.

ZMOTION White Light Threshold

The ZMOTION White Light Threshold register, shown in Table 19, determines the sensitivity settings for white light immunity. See the

White Light Detection section on page 8 for a description of White Light operation.

Table 19. 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 detection threshold.
White Light 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.

ZMOTION White Light Control 0

The ZMOTION White Light Control 0 register, shown in Table 20. ZMOTION White Light Control 0 (ZM_WL_CTRL0)Table 20, is used to control the white light debounce time, anti-jam feature provides white light event status. See the

White Light Detection section on page 8 for a description of White Light operation.

Table 20. ZMOTION White Light Control 0 (ZM_WL_CTRL0)

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

Bit	Description
[7:4]	White Light debounce time.
White Light Debounce	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]	Enable/disable Anti-Jam feature.
Anti-Jam	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 for intrusion/security applications. 0 - White Light Anti-Jam disabled 1 - White Light Anti-Jam enabled.
[0]	White Light event status
White Light Event Detected	Set by ZMOTION Engine; Cleared by application. This bit is set by the ZMOTION Engine when a White Light event has been detected and associated false motion event(s) may have been suppressed. This bit is set by the engine and must be cleared by the user application.

ZMOTION PIR Signal

The ZMOTION Signal register, shown in Table 21, contains the current calculated sensor constructed sample.

Table 21. ZMOTION Signal (ZM_SIGNAL)

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	2	2	1	0
Field:	Constructed Sample															
Control:	R															

Bit	Description
[15:0]	Current constructed sample.
ZMOTION Signal	Controlled by ZMOTION Engine. This 16-bit register provides access to the last constructed PIR sample calculated by the engine. Each time the engine generates a new PIR constructed sample (based on the Constructed Sample Size set in ZMOTION Sample Size (ZM_SAMPLE_SIZE) register, it places it in this register and sets the New Sample bit in the ZMOTION Status 0 (ZM_STATUS0) register. This gives the application direct visibility to the PIR-generated signal for development purposes.

ZMOTION DC Signal Level

The ZMOTION DC Signal Level register, shown in Table 22, contains the current calculated PIR sensor DC level.

Table 22. ZMOTION DC Signal Level (ZM_SIGNAL_DC)

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	2	2	1	0
Field:	DC Signal Level															
Control:	R															

Bit	Description
[15:0]	PIR signal DC level.
ZMOTION Signal DC	Controlled by ZMOTION Engine. This 16-bit register contains 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 this register.

ZMOTION Process Rate

The ZMOTION Process Rate register, shown in Table 23, indicates the rate at which the engine is processing constructed samples. The value is in samples per second.

Table 23. ZMOTION Process Rate (ZM_PROCESS_RATE)

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	2	2	1	0
Field:	Process Rate															
Control:	R															

Bit	Description
[15:0]	Constructed sample processing rate.
ZMOTION Process Rate	Controlled by ZMOTION Engine. This 16-bit register indicates 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 values indicate more processing. Generally, the process rate should be maintained above 0100h.

ZMOTION Transient Sensitivity Level

The ZMOTION Transient Sensitivity Level register, shown in Table 24, controls the Engine's sensitivity to transient events.

Table 24. ZMOTION Transient Sensitivity Level (ZM_TRANSIENT_SENSE)

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

Bit	Description
[6:0]	PIR sensor signal transient rejection.
Transient Sensitivity	Controlled by application. This parameter controls the sensitivity of the transient detector. The transient detector monitors the motion signal for fast changes in the PIR signal and prevents them from generating false motion events. Typical sources of transient events include EMI from nearby radio transmitters. Typical values are in the range of 6 to 28h. 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 EM Transient Detected bit in the ZMOTION Status 0 (ZM_STATUS0) register is set. The application can monitor this bit during testing to determine the correct sensitivity level required. The valid range is 0 (disabled) to 64h.

ZMOTION Noise Sensitivity Level

The ZMOTION Noise Sensitivity Level register, shown in Table 25, controls the Engine's sensitivity to noise events.

Table 25. ZMOTION Noise Sensitivity Level (ZM_NOISE_SENSE)

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

Bit	Description
[6:0]	PIR sensor signal noise rejection.
Noise Sensitivity	Controlled by application.
	This parameter controls the sensitivity of the noise detector. The noise detector monitors the motion signal for changes in the PIR signal noise range/amplitudes and prevents them from causing false motion events.
	Typical sources of broadband noise include EMI from nearby radio transmitters.
	The valid range of values is determined by the Window Size selected in the ZMOTION Control 2 register as shown in the table below. Lower values provide more protection from noise 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 noise events.
	When a noise event is detected, the EM Noise Detected bit in ZM_STATUS0 (EM Noise Detected) is set. The application can monitor this bit during testing to determine the correct sensitivity level required.

Window Size	Maximum 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 26, 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 to 20ms), allowing the event to pass without detecting a false motion event. The ZMOTION sample buffer is flushed, and motion detection begins again.

Table 26. ZMOTION Transient Sensitivity Level (ZM_TRANSIENT_SENSE)

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

Bit	Description
[7:6]	Spark detection gap size.
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.

Spark Gap	Constructed Samples	Pulse Width with 1.5ms constructed sample rate
00	+/-1	3ms
01	+/-2	6ms
10	+/-3	9ms
11	+/-4	12ms

[5:0]	Sensitivity to spark events.
Spark Sensitivity	Controlled by application This parameter controls the sensitivity of the spark detector. It sets the minimum signal change required on both the leading edge and trailing edge of the signal to generate a Spark event. Typical values are in the range of 08h to 20h. Lower values cause the Spark Detector to become more sensitive and provide additional protection to ESD events. Choose a value high enough to allow all desired motion events to pass through while still rejecting unwanted ESD events. Setting this field to 0 disables the Spark Detector. When a spark event is detected, the ZMS0_EM_SPARK_DETECTED bit in the ZMOTION Status 0 (ZM_STATUS0) register is set. The application can read this during testing to determine the correct sensitivity level. The valid range is 0 (disabled) to 3Fh.

ZMOTION Library and Version

The ZMOTION Library and Version register, shown in Table 27, indicates the library type used and version of the library.

Table 27. ZMOTION Library and Version (ZM_LIB_VERSION)

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

Bit	Description
[7:4]	ZMOTION Engine library type.
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 28.
[3:0]	ZMOTION Engine library version.
Library Version	Controlled by ZMOTION Engine. The value stored in this register indicates the software version of the ZMOTION Engine Library.

Table 28. ZMOTION Library Types

Library Type Field	Library Name	Library Function	Description
1	ZMOTION_Engine_WL_Lib	ZMOTION Engine Library + White Light	Library used to support single and multi-sensor applications needing white light immunity
2	ZMOTION_Engine_Lib	ZMOTION Engine Library	Library used to support single and multi-sensor applications not requiring white light immunity
3	ZMOTION_Engine1_Lib	ZMOTION Engine Library 1	Library used to support a second PIR sensor.
4	ZMOTION_Engine2_Lib	ZMOTION Engine Library 2	Library used to support a third PIR sensor.

Customer Support

To receive answers to your technical questions or report issues you may be experiencing with our products, please visit Zilog's Technical Support page.

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