# Keypad Decoder and I/O Port Expander

Data Sheet

# ADP5586

## FEATURES

- 16-element FIFO for event recording
- 10 configurable I/Os allowing for such functions as
  - Keypad decoding for a matrix of up to 5 × 5
  - Key press/release interrupts
  - GPIO functions
  - GPI with selectable interrupt level
  - 100 kΩ or 300 kΩ pull-up resistors
  - 300 kΩ pull-down resistors
  - GPO with push-pull or open drain
  - Programmable logic block
  - Pulse generators
    - Periods and on times
      - Above 30 sec in 125 ms increments
      - Up to 255 ms in 1 ms increments
  - Reset generator
- I²C interface with Fast-mode Plus (Fm+) support of up to 1 MHz
- Open-drain interrupt output
- 16-ball WLCSP, 1.59 mm × 1.59 mm

## APPLICATIONS

- Keypad entries and input/output expansion capabilities
- Smartphones, remote controls, and cameras
- Healthcare, industrial, and instrumentation

## FUNCTIONAL BLOCK DIAGRAM

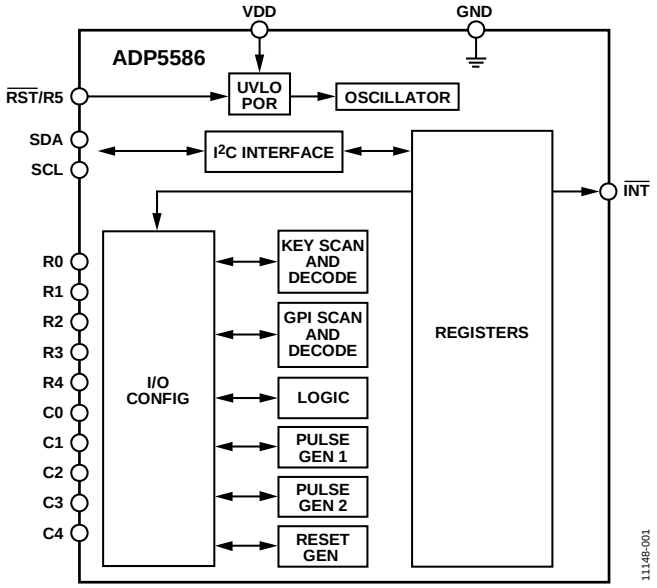



*Figure 1.*

## GENERAL DESCRIPTION

The ADP5586 is a 10-input/output port expander with a built-in keypad matrix decoder, programmable logic, reset generator, and pulse generators. Input/output expander ICs are used in portable devices (phones, remote controls, and cameras) and nonportable applications (healthcare, industrial, and instrumentation). I/O expanders can be used to increase the number of I/Os available to a processor or to reduce the number of I/Os required through interface connectors for front panel designs.

The ADP5586 handles all key scanning and decoding and can flag the main processor, via an interrupt line, that new key events have occurred. GPI changes and logic changes can also be tracked as events via the FIFO, eliminating the need to monitor different registers for event changes. The ADP5586 is equipped with a FIFO to store up to 16 events. Events can be read back by the processor via an I²C-compatible interface.

The ADP5586 eliminates the need for the main processor to monitor the keypad, thus reducing power consumption and/or increasing processor bandwidth for performing other functions.

The programmable logic functions allow common logic requirements to be integrated as part of the GPIO expander, thus saving board area and cost.

Rev. 0 Document Feedback

Information furnished by Analog Devices is believed to be accurate and reliable. However, no responsibility is assumed by Analog Devices for its use, nor for any infringements of patents or other rights of third parties that may result from its use. Specifications subject to change without notice. No license is granted by implication or otherwise under any patent or patent rights of Analog Devices. Trademarks and registered trademarks are the property of their respective owners.

One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, U.S.A.
Tel: 781.329.4700
©2013 Analog Devices, Inc. All rights reserved.
Technical Support www.analog.com

# TABLE OF CONTENTS

Features ..... 1
Applications ..... 1
Functional Block Diagram ..... 1
General Description ..... 1
Revision History ..... 2
Specifications ..... 3
I²C Timing Specifications ..... 4
Absolute Maximum Ratings ..... 5
Thermal Resistance ..... 5
ESD Caution ..... 5
Pin Configuration and Function Descriptions ..... 6
Theory of Operation ..... 7
Device Enable ..... 8
Device Overview ..... 8
Functional Description ..... 9
Event FIFO ..... 9
Key Scan Control ..... 10
GPI Input ..... 13
GPO Output ..... 13
Logic Block ..... 14
Reset Block ..... 15
Interrupts ..... 15
Pulse Generators ..... 16
Register Interface ..... 17
Register Map ..... 19
Detailed Register Descriptions ..... 21
Applications Schematic ..... 41
Outline Dimensions ..... 42
Ordering Guide ..... 42

## REVISION HISTORY

**3/13—Revision 0: Initial Version**

# SPECIFICATIONS

VDD = 1.8 V to 3.3 V, $T_A = T_J = -40°C$ to +85°C, unless otherwise noted.[1]

**Table 1.**

| Parameter | Symbol | Test Conditions/Comments | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| SUPPLY VOLTAGE | | | | | | |
| VDD Input Voltage Range | VDD | | 1.65 | | 3.6 | V |
| Undervoltage Lockout Threshold | $UVLO_{VDD}$ | UVLO active, VDD falling | 1.2 | 1.3 | | V |
| | | UVLO inactive, VDD rising | | 1.4 | 1.6 | V |
| SUPPLY CURRENT | | | | | | |
| Standby Current | $I_{STNBY}$ | VDD = 1.65 V | | 1 | 4 | µA |
| | | VDD = 3.3 V | | 1 | 10 | µA |
| Operating Current (One Key Press) | $I_{SCAN1}$ | Scan = 10 ms, CORE_FREQ = 50 kHz, scan active, 300 kΩ pull-up, VDD = 1.65 V | | 30 | 40 | µA |
| | $I_{SCAN2}$ | Scan = 10 ms, CORE_FREQ = 50 kHz, scan active, 300 kΩ pull-up, VDD = 3.3 V | | 75 | 85 | µA |
| PULL-UP, PULL-DOWN RESISTANCE | | | | | | |
| Pull-Up | | | | | | |
| Option 1 | | | 50 | 100 | 150 | kΩ |
| Option 2 | | | 150 | 300 | 450 | kΩ |
| Pull-Down | | | 150 | 300 | 450 | kΩ |
| INPUT LOGIC LEVEL ($\overline{RST}$, SCL, SDA, R0, R1, R2, R3, R4, R5, C0, C1, C2, C3, C4) | | | | | | |
| Input Voltage | | | | | | |
| Logic Low | $V_{IL}$ | | | | 0.3 × VDD | V |
| Logic High | $V_{IH}$ | | 0.7 × VDD | | | V |
| Input Leakage Current (Per Pin) | $V_{I-LEAK}$ | | | 0.1 | 1 | µA |
| PUSH-PULL OUTPUT LOGIC LEVEL (R0, R1, R2, R3, R4, R5, C0, C1, C2, C3, C4 | | | | | | |
| Output Voltage | | | | | | |
| Logic Low | $V_{OL1}$ | Sink current = 10 mA, maximum of five GPIOs active simultaneously | | | 0.4 | V |
| | $V_{OL2}$ | Sink current = 10 mA, all GPIOs active simultaneously | | | 0.5 | V |
| Logic High | $V_{OH}$ | Source current = 5 mA | 0.7 × VDD | | | V |
| Logic High Output Leakage Current (Per Pin) | $V_{OH-LEAK}$ | | | 0.1 | 1 | µA |
| OPEN-DRAIN OUTPUT LOGIC LEVEL ($\overline{INT}$, SDA) | | | | | | |
| Output Voltage | | | | | | |
| Logic Low | | | | | | |
| $\overline{INT}$ | $V_{OL3}$ | $I_{SINK}$ = 10 mA | | | 0.4 | V |
| SDA | $V_{OL4}$ | $I_{SINK}$ = 20 mA | | | 0.4 | V |
| Logic High Output Leakage Current (Per Pin) | $V_{OH-LEAK}$ | | | 0.1 | 1 | µA |
| Logic Propagation Delay | | | | 125 | 300 | ns |
| Flip-Flop (FF) Hold Time[2] | | | | 0 | | ns |
| FF Setup Time[2] | | | | 175 | | ns |
| GPIO Debounce[2] | | | | | 70 | µs |
| Internal Oscillator Frequency[3] | $OSC_{FREQ}$ | | 720 | 800 | 880 | kHz |

[1] All limits at temperature extremes are guaranteed via correlation, using standard statistical quality control (SQC). Typical values are at $T_A$ = 25°C, VDD = 1.8 V.
[2] Guaranteed by design.
[3] All timers are referenced from the base oscillator and have the same ±10% accuracy.

## I²C TIMING SPECIFICATIONS

**Table 2.**

| Parameter | Description | Min | Max | Unit |
|---|---|---|---|---|
| I²C TIMING SPECIFICATIONS | | | | |
| Delay from UVLO/$\overline{RST}$ Inactive to I²C Access | | | 60 | µs |
| $f_{SCL}$ | SCL clock frequency | 0 | 1000 | kHz |
| $t_{HIGH}$ | SCL high time | 0.26 | | µs |
| $t_{LOW}$ | SCL low time | 0.5 | | µs |
| $t_{SU;\ DAT}$ | Data setup time | 50 | | ns |
| $t_{HD;\ DAT}$ | Data hold time | 0 | | µs |
| $t_{SU;\ STA}$ | Setup time for repeated start | 0.26 | | µs |
| $t_{HD;\ STA}$ | Hold time for start/repeated start | 0.26 | | µs |
| $t_{BUF}$ | Bus free time for stop and start conditions | 0.5 | | µs |
| $t_{SU;\ STO}$ | Setup time for stop condition | 0.26 | | µs |
| $t_{VD;\ DAT}$ | Data valid time | | 0.45 | µs |
| $t_{VD;\ ACK}$ | Data valid acknowledge | | 0.45 | µs |
| $t_R$ | Rise time for SCL and SDA | | 120 | ns |
| $t_F$ | Fall time for SCL and SDA | | 120 | ns |
| $t_{SP}$ | Pulse width of suppressed spike | 0 | 50 | ns |
| $C_B$[1] | Capacitive load for each bus line | | 550 | pF |

[1] $C_B$ is the total capacitance of one bus line in picofarads (pF).

### *Timing Diagram*

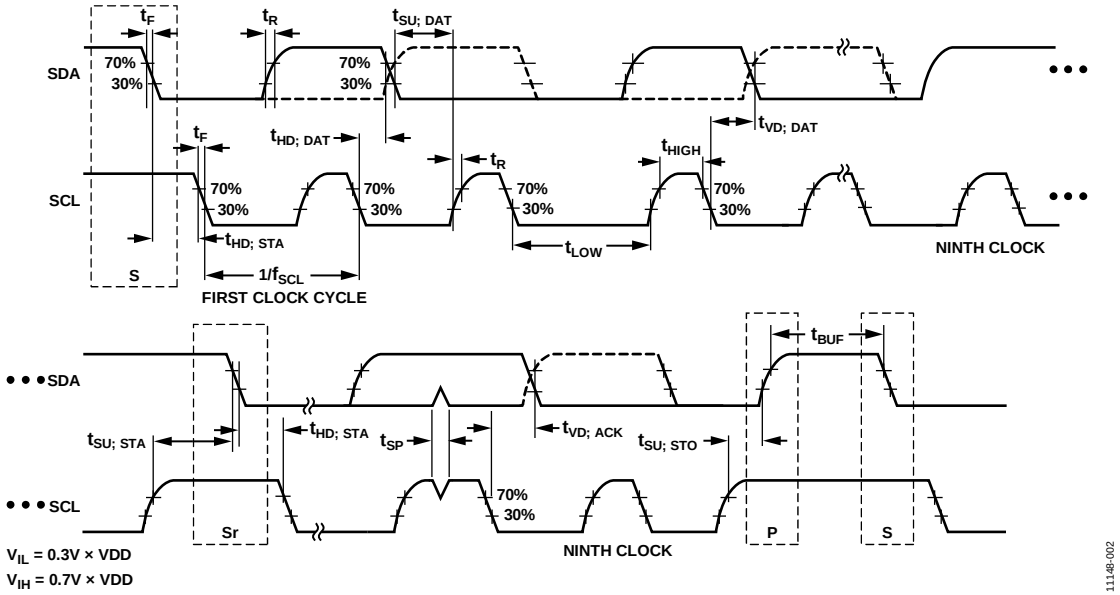



*Figure 2. I²C Interface Timing Diagram*

# ABSOLUTE MAXIMUM RATINGS

**Table 3.**

| Parameter | Rating |
|---|---|
| VDD to GND | −0.3 V to +4 V |
| SCL, SDA, $\overline{RST}$, $\overline{INT}$, R0, R1, R2, R3, R4, C0, C1, C2, C3, C4 | −0.3 V to (VDD + 0.3 V) |
| Temperature Range | |
| Operating (Ambient) | −40°C to +85°C[1] |
| Operating (Junction) | −40°C to +125°C |
| Storage | −65°C to +150°C |

[1] In applications where high power dissipation and poor thermal resistance are present, the maximum ambient temperature may need to be derated. Maximum ambient temperature ($T_{A\,(MAX)}$) is dependent on the maximum operating junction temperature ($T_{J\,(MAXOP)}$ = 125°C), the maximum power dissipation of the device ($P_{D\,(MAX)}$), and the junction-to-ambient thermal resistance of the device/package in the application ($\theta_{JA}$), using the following equation: $T_{A\,(MAX)} = T_{J\,(MAXOP)} - (\theta_{JA} \times P_{D\,(MAX)})$.

Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; functional operation of the device at these or any other conditions above those indicated in the operational section of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Absolute maximum ratings apply individually only, not in combination. Unless otherwise specified, all other voltages are referenced to GND.

## THERMAL RESISTANCE

$\theta_{JA}$ is specified for the worst-case conditions, that is, a device soldered in a printed circuit board (PCB) for surface-mount packages.

**Table 4.**

| Thermal Resistance | $\theta_{JA}$ | Unit |
|---|---|---|
| 16-Ball WLCSP | 62 | °C/W |
| Maximum Power Dissipation | 70 | mW |

## ESD CAUTION



**ESD (electrostatic discharge) sensitive device.** Charged devices and circuit boards can discharge without detection. Although this product features patented or proprietary protection circuitry, damage may occur on devices subjected to high energy ESD. Therefore, proper ESD precautions should be taken to avoid performance degradation or loss of functionality.

# PIN CONFIGURATION AND FUNCTION DESCRIPTIONS

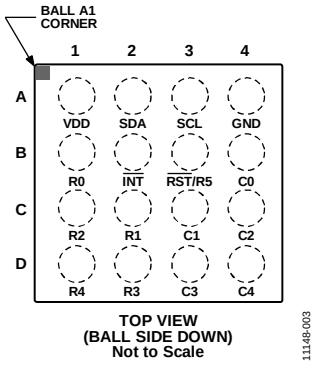
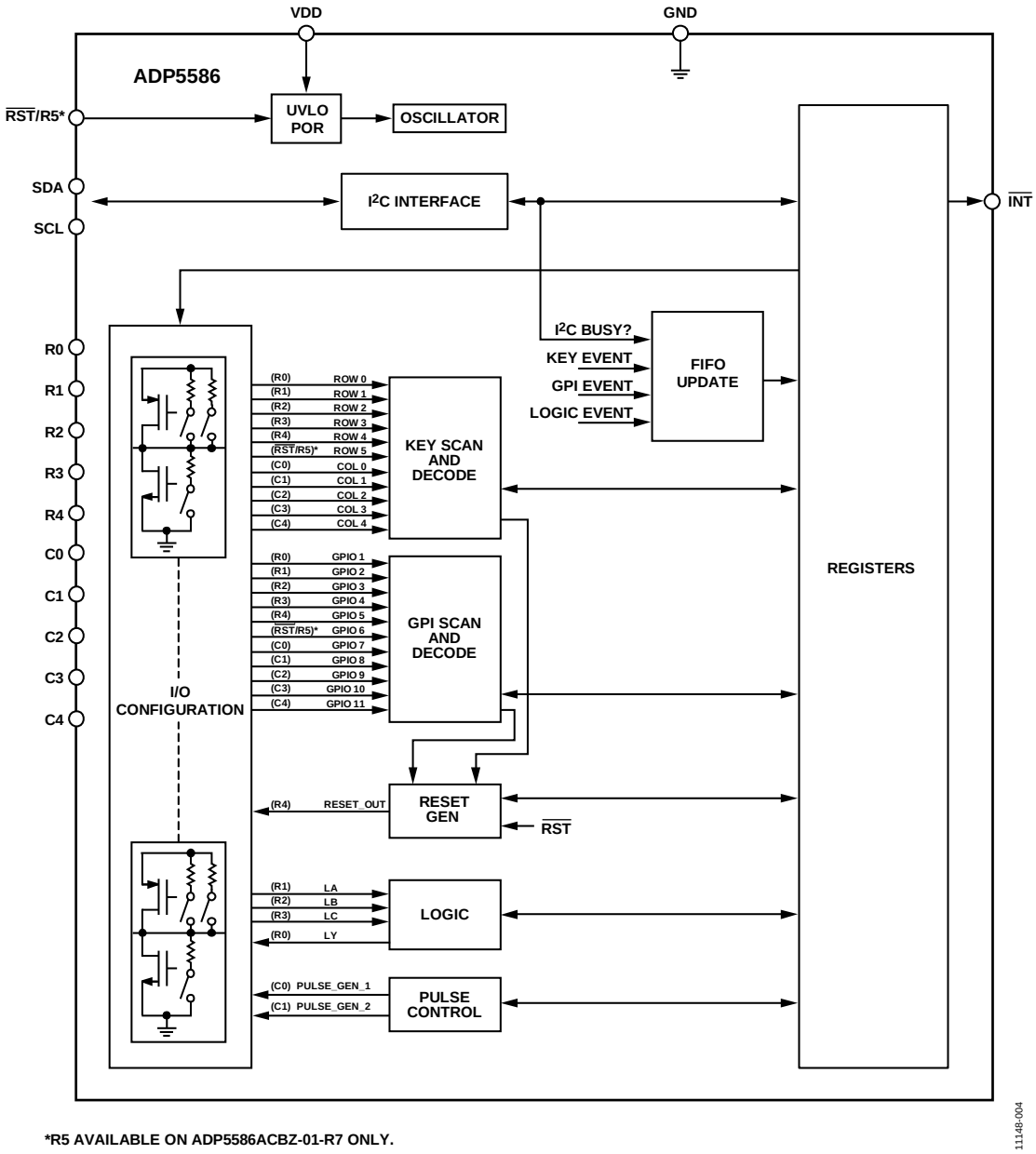



*Figure 3. Pin Configuration*

**Table 5. Pin Function Descriptions**

| Pin No. | Mnemonic | Description |
|---|---|---|
| A1 | VDD | Supply Voltage Input. |
| A2 | SDA | $I^2C$ Data Input/Output. |
| A3 | SCL | $I^2C$ Clock Input. |
| A4 | GND | Ground. |
| B1 | R0 | GPIO 1 (GPIO Alternate Function: Logic Block Output LY). This pin functions as Row 0 when configured in keypad mode. |
| B2 | $\overline{INT}$ | Open-Drain Interrupt Output. |
| B3 | $\overline{RST}$/R5 | Input Reset Signal ($\overline{RST}$). The reset signal function applies to all models except the ADP5586ACBZ-01-R7.<br>GPIO 6/Row 5 (R5). This function applies only to the ADP5586ACBZ-01-R7 model. |
| B4 | C0 | GPIO 7 (GPIO Alternate Function: PULSE_GEN_1). This pin functions as Column 0 when configured in keypad mode. |
| C1 | R2 | GPIO 3 (GPIO Alternate Function: Logic Block Input LB). This pin functions as Row 2 when configured in keypad mode. |
| C2 | R1 | GPIO 2 (GPIO Alternate Function: Logic Block Input LA). This pin functions as Row 1 when configured in keypad mode. |
| C3 | C1 | GPIO 8 (GPIO Alternate Function: PULSE_GEN_2). This pin functions as Column 1 when configured in keypad mode. |
| C4 | C2 | GPIO 9. This pin functions as Column 2 when configured in keypad mode. |
| D1 | R4 | GPIO 5 (GPIO Alternate Function: RESET_OUT). This pin functions as Row 4 when configured in keypad mode. |
| D2 | R3 | GPIO 4 (GPIO Alternate Function: Logic Block Input LC). This pin functions as Row 3 when configured in keypad mode. |
| D3 | C3 | GPIO 10. This pin functions as Column 3 when configured in keypad mode. |
| D4 | C4 | GPIO 11. This pin functions as Column 4 when configured in keypad mode. |

# THEORY OF OPERATION




*R5 AVAILABLE ON ADP5586ACBZ-01-R7 ONLY.

*Figure 4. Internal Block Diagram*

## DEVICE ENABLE

When sufficient voltage is applied to VDD and the $\overline{\text{RST}}$ pin is driven with a logic high level, the ADP5586 starts up in standby mode with all settings at default. The user can configure the device via the I²C interface. When the $\overline{\text{RST}}$ pin is low, the ADP5586 enters a reset state and all settings return to default. The $\overline{\text{RST}}$ pin features a debounce filter.

If the ADP5586ACBZ-01-R7 device model is used, the $\overline{\text{RST}}$ pin acts as an additional row pin (R5). To reset the part without a reset pin, either bring VDD below the UVLO threshold, or set the SW_RESET bit to 1 (Register 0x3D, Bit 2).

## DEVICE OVERVIEW

The ADP5586 contains 10 multiconfigurable input/output pins. Each pin can be programmed to enable the device to carry out its various functions, as follows:

- Keypad matrix decoding (five-column by five-row matrix maximum)
- General-purpose I/O expansion (up to 10 inputs/outputs)
- Reset generator
- Logic function building blocks (up to three inputs and one output)
- Two pulse generators

All 10 input/output pins have an I/O structure as shown in Figure 5.

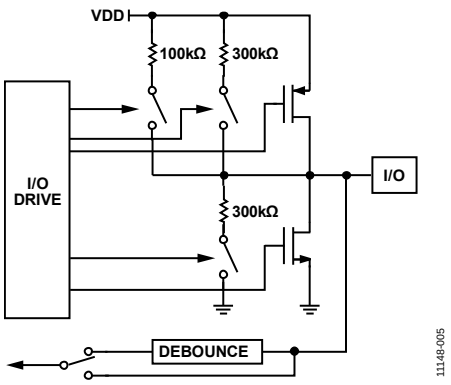



*Figure 5. I/O Structure*

Each I/O can be pulled up with a 100 kΩ or 300 kΩ resistor or pulled down with a 300 kΩ resistor. For logic output drive, each I/O has a 5 mA PMOS source and a 10 mA NMOS sink for a push-pull type output. For open-drain output situations, the 5 mA PMOS source is not enabled. For logic input applications, each I/O can be sampled directly or, alternatively, sampled through a debounce filter.

The I/O structure shown in Figure 5 allows for all GPI and GPO functions, as well as PWM and clock divide functions. For key matrix scan and decode, the scanning circuit uses the 100 kΩ or 300 kΩ resistor for pulling up the keypad row pins and the 10 mA NMOS sinks for grounding the keypad column pins (see the Key Scan Control section for details about key decoding).

Configuration of the device is carried out by programming an array of internal registers via the I²C interface. Feedback of device status and pending interrupts can be flagged to an external processor by using the $\overline{\text{INT}}$ pin.

The ADP5586 is offered with three feature sets. Table 6 lists the options that are available for each model of the ADP5586. Contact your local Analog Devices, Inc., field applications engineers for availability and/or alternate configurations.

**Table 6. Matrix Options by Device Model[1]**

| Model | Description |
|---|---|
| ADP5586ACBZ-00-R7 | GPIO pull-down on startup<br>5-row × 5-column matrix |
| ADP5586ACBZ-01-R7 | Row 5 added to GPIOs<br>6-row × 5-column matrix |
| ADP5586ACBZ-03-R7 | Alternate I²C address (0x30)<br>5-row × 5-column matrix |

[1] Contact Analog Devices for availability of configurations not shown here.

# FUNCTIONAL DESCRIPTION

## EVENT FIFO

Before going into detail on the various blocks of the ADP5586, it is important to understand the function of the event FIFO that is featured in the ADP5586. The event FIFO (Register 0x03 to Register 0x12) can record as many as 16 events. By default, the FIFO primarily records key events, such as key press and key release. However, it is possible to configure the general-purpose input (GPI) and logic activity to generate event information on the FIFO, as well. An event count, EC[4:0] (Register 0x02, Bits[4:0]), is composed of five bits and works in tandem with the FIFO so that the user knows how many events are stored in the FIFO.

The FIFO consists of sixteen 8-bit elements. Bits[6:0] of each element store the event identifier, and Bit 7 stores the event state. The user can read the top element of the FIFO from any of the FIFO_1 through FIFO_16 registers. The ADP5586 has multiple copies of the FIFO register to allow reading of the complete FIFO with a single I²C burst read.

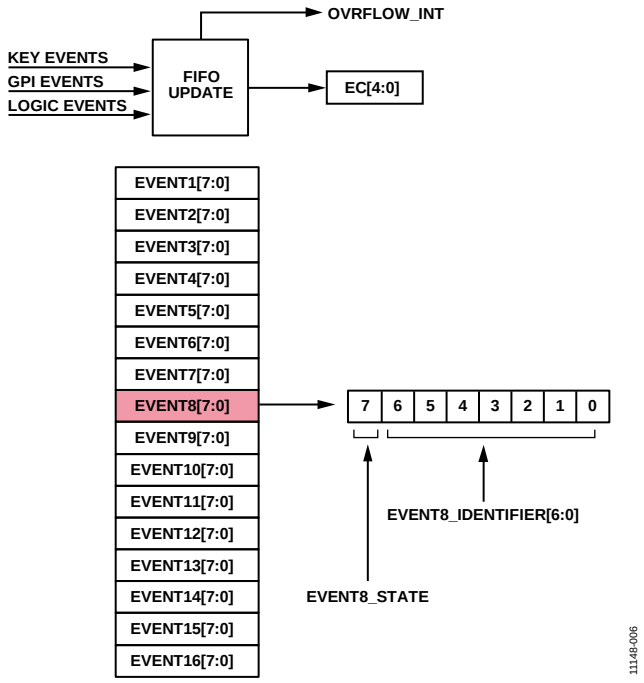



*Figure 6. Breakdown of Eventx[7:0] Bits*

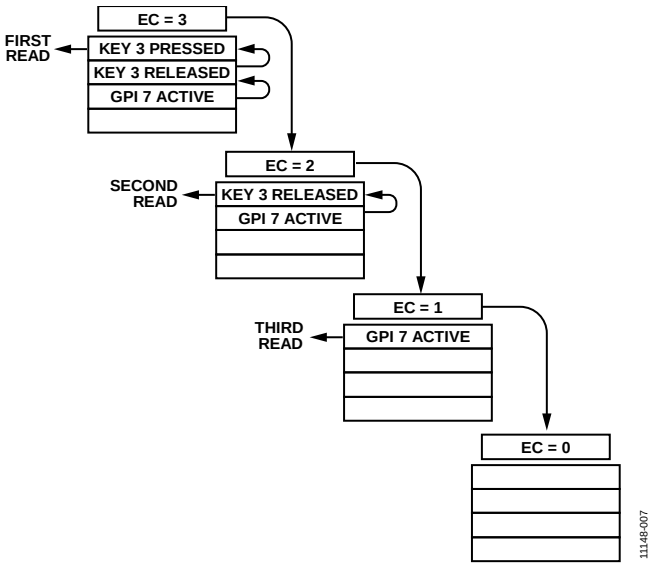



*Figure 7. FIFO Operation*

The FIFO registers always point to the top of the FIFO (that is, the location of EVENT1[7:0]). If the user tries to read back from any location in a FIFO, data is always obtained from the top of that FIFO. This ensures that events can be read back only in the order in which they occurred, thereby ensuring the integrity of the FIFO system.

As stated previously, some of the on-board functions of the ADP5586 can be programmed to generate events on the FIFO. A FIFO update control block manages updates to the FIFO. If an I²C transaction is accessing any of the FIFO address locations, updates are paused until the I²C transaction is complete.

A FIFO overflow event occurs when more than 16 events are generated prior to an external processor reading a FIFO and clearing it.

If an overflow condition occurs, the overflow interrupt status bit is set (OVRFLOW_INT, Register 0x01, Bit 2). An interrupt is generated if an overflow interrupt is enabled, signaling to the processor that more than 16 events have occurred.

## KEY SCAN CONTROL

### *General*

The 10 input/output pins can be configured to decode a keypad matrix up to a maximum size of 25 switches (5 × 5 matrix) using the PIN_CONFIG_A, PIN_CONFIG_B, and PIN_CONFIG_C registers (Registers 0x3A through 0x3C). Smaller matrices can also be configured, making the unused row and column pins available for other I/O functions.

The R0 through R4 I/O pins comprise the rows of the keypad matrix. The C0 through C4 I/O pins comprise the columns of the keypad matrix. Pins that are used as rows are pulled up via the internal 300 kΩ (or 100 kΩ) resistors. Pins that are used as columns are driven low via the internal NMOS current sink.

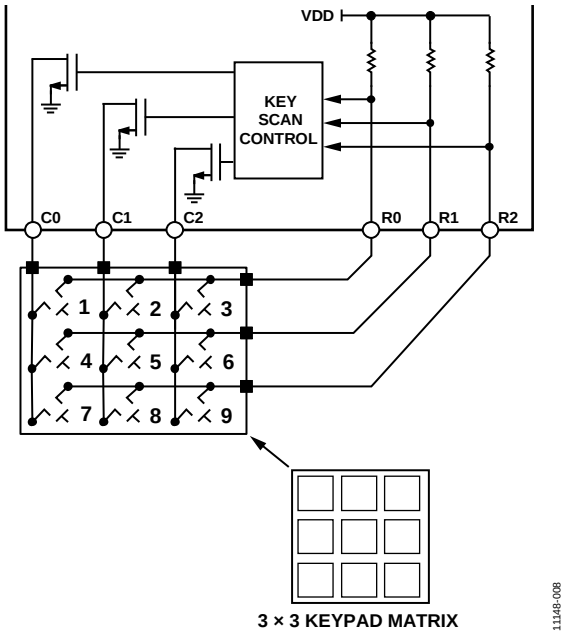



*Figure 8. Simplified Key Scan Block*

Figure 8 shows a simplified representation of the key scan block using three row pins and three column pins connected to a small 3 × 3, nine-switch keypad matrix. When the key scanner is idle, the row pins are pulled high and the column pins are driven low. The key scanner operates by checking the row pins to see if they are low.

If Switch 6 in the matrix is pressed, R1 connects to C2. The key scan circuit senses that one of the row pins has been pulled low, and a key scan cycle begins. Key scanning involves driving all column pins high, then driving each column pin low, one at a time, and sensing whether a row pin is low. All row/column pairs are scanned; therefore, if multiple keys are pressed, they are detected.

To prevent a glitch or narrow press time from being registered as a valid key press, the key scanner requires that the key be pressed for two scan cycles. The key scanner has a wait time between each scan cycle; therefore, the key must be pressed and held for at least this wait time to register as being pressed. If the key is continuously pressed, the key scanner continues to scan and wait for as long as the key is pressed.

If Switch 6 is released, the connection between R1 and C2 breaks, and R1 is pulled high. The key scanner requires that the key be released for two scan cycles because the release of a key is not necessarily in sync with the key scanner. Up to two full wait/scan cycles may be required for a key to register as released. When the key registers as released, and no other keys are pressed, the key scanner returns to idle mode.

For the remainder of this data sheet, the press/release status of a key is represented as simply a logic signal in the figures. A logic high level represents the key status as pressed, and a logic low level represents released. This eliminates the need to draw individual row/column signals when describing key events.

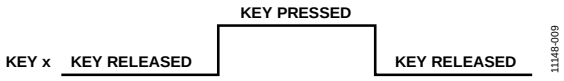



*Figure 9. Logic Low: Key Released; Logic High: Key Pressed*

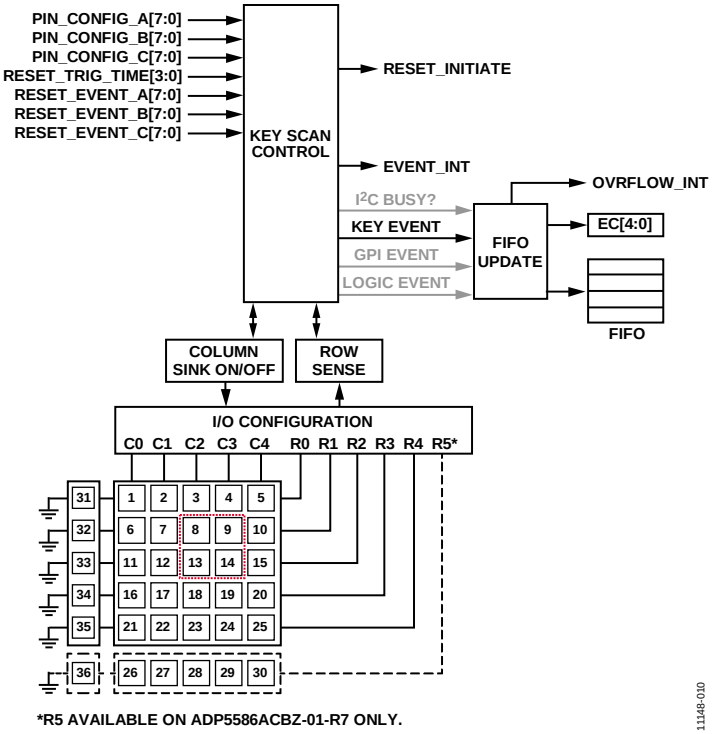



*Figure 10. Detailed Key Scan Block*

Figure 10 shows a detailed representation of the key scan block and its associated control and status signals. When all row and column pins are used, a matrix of 25 unique keys can be scanned.

Use the PIN_CONFIG_A[5:0] and PIN_CONFIG_B[4:0] registers (Register 0x3A and Register 0x3B, respectively) to configure the I/Os for keypad decoding. The number label on each key switch represents the event identifier that is recorded if that switch is pressed. If all row/column pins are configured, it is possible to observe all 25 key identifiers on the FIFO.

If a smaller 2 × 2 matrix is configured, for example, by using the C2 and C3 column pins and the R1 and R2 row pins, only four event identifiers (8, 9, 13, and 14) can possibly be observed on the FIFO, as shown in Figure 10.

By default, the ADP5586 records key presses and releases on the FIFO. Figure 11 illustrates what happens when a single key is pressed and released. Initially, the key scanner is idle. When Key 3 is pressed, the scanner begins scanning through all configured row/column pairs. After the scan wait time, the scanner again scans through all configured row/column pairs and detects that Key 3 has remained pressed, which sets the EVENT_INT interrupt bit (Register 0x01, Bit 0). The event counter, EC[4:0] (Register 0x02, Bits[4:0]), is then incremented to 1; EVENT1_IDENTIFIER[6:0] of the FIFO is updated with its event identifier set to 3; and its EVENT1_STATE bit is set to 1, indicating a key press.

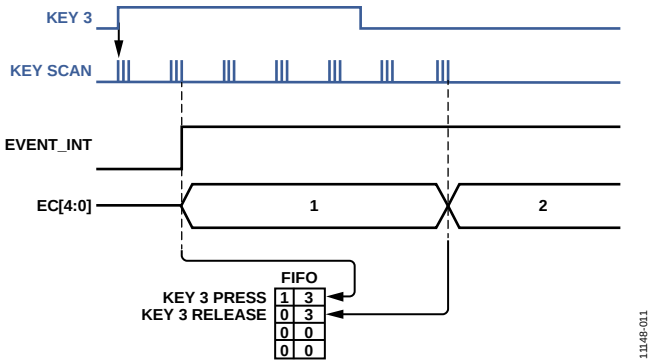



*Figure 11. Press and Release Event*

The key scanner continues the scan/wait cycles while the key remains pressed. If the scanner detects that the key has been released for two consecutive scan cycles, the event counter, EC[4:0], is incremented to 2, and EVENT2_IDENTIFIER[6:0] of the FIFO is updated with its event identifier set to 3. The EVENT2_STATE bit is set to 0, indicating a release. The key scanner returns to idle mode because no other keys are pressed.

The EVENT_INT interrupt (Register 0x01, Bit 0) can be triggered by both press and release key events. As shown in Figure 12, if Key 3 is pressed, EVENT_INT is asserted, EC[4:0] is updated, and the FIFO is updated. During the time that the key remains pressed, it is possible for the FIFO to be read, the event counter decremented to 0, and EVENT_INT cleared. When the key is finally released, EVENT_INT is asserted, the event counter is incremented, and the FIFO is updated with the release event information.

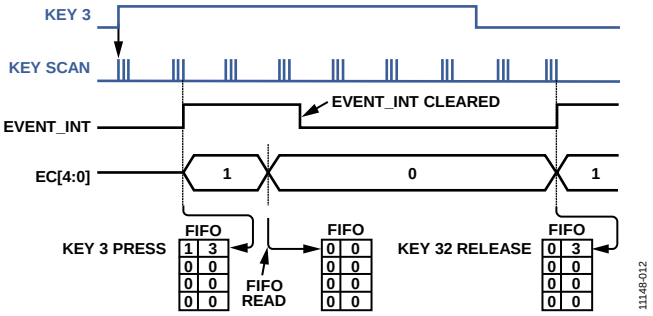



*Figure 12. Asserting the EVENT_INT Interrupt Keypad Extension*

As shown in Figure 10, the keypad can be extended if each row is connected directly to ground by a switch. If the switch placed between R0 and ground is pressed, the entire row is grounded. When the key scanner completes scanning, it normally detects Key 1 to Key 5 as being pressed; however, this unique condition is decoded by the ADP5586, and Key Event 31 is assigned to it. Up to five more key event assignments are possible, allowing the keypad size to extend up to 30. However, if one of the extended keys is pressed, none of the keys on that row is detectable. The activation of a ground key causes all other keys sharing that row to be undetectable.

### Precharge Time

During a scan sequence, a row scans through the columns sequentially. Each row/column combination is tested at a rate that is defined by the KEY_POLL_TIME bits (Register 0x39, Bits[1:0]). Within each of these scan times, each column is scanned for a time defined by the PRECHARGE_TIME bit (Register 0x39, Bit 3). As shown in Figure 13, the resistance capacitance (RC) time constant, which is defined by the series resistance (from pull-up/pull-down, for example) and parallel capacitance that is seen on the individual columns, affects the sampling of a key press event.

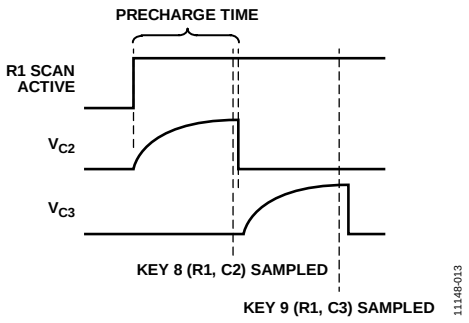



*Figure 13. Precharge Time*

The ADP5586 samples the state of the row/column pairs near the end of the precharge time. By extending this time, higher RC time constants can be accommodated. For applications that use physical buttons, the RC time constant is usually not an issue, but if external relay switches or multiple external muxes are attached to columns, the RC constant may increase. Using a smaller pull-up resistor on the rows (Register 0x3C, Bit 7) reduces the RC time constant.

### Ghosting

Ghosting is an occurrence where, given certain key press combinations on a keypad matrix, a false positive reading of an additional key is detected. Ghosting is created when three or more keys are pressed simultaneously on multiple rows or columns (see Figure 14). Key combinations that form a right angle on the keypad matrix may cause ghosting.

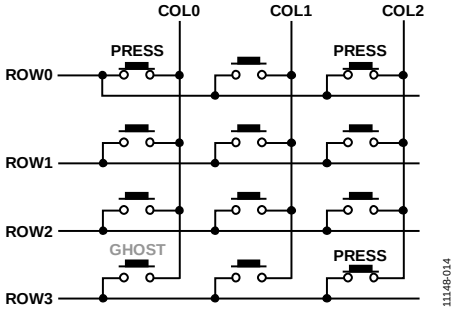



*Figure 14. Ghosting Example: Column 0/Row 3 is a Ghost Key Due to a Short Among Row 0, Column 0, Column 2, and Row 3 During Key Press*

The solution to ghosting is to select a keypad matrix layout that takes into account three key combinations that are most likely to be pressed together. Multiple keys that are pressed across one row or across one column do not cause ghosting. Staggering keys so that they do not share a column also avoids ghosting. The most common practice is to place keys in the same row or column that are likely to be pressed at the same time. Some examples of keys that are likely to be pressed at the same time are as follows:

- The navigation keys in combination with the Select key
- The navigation keys in combination with the space bar
- The reset combination keys, such as CTRL + ALT + DEL

## GPI INPUT

Each of the 10 input/output lines can be configured as a general-purpose logic input line using the GPIO_INP_EN_A and GPIO_INP_EN_B registers (Register 0x29 and Register 0x2A). GPIO lines can be configured to allow both input and output at the same time. Figure 15 shows a detailed representation of the GPI scan and detect block and its associated control and status signals.

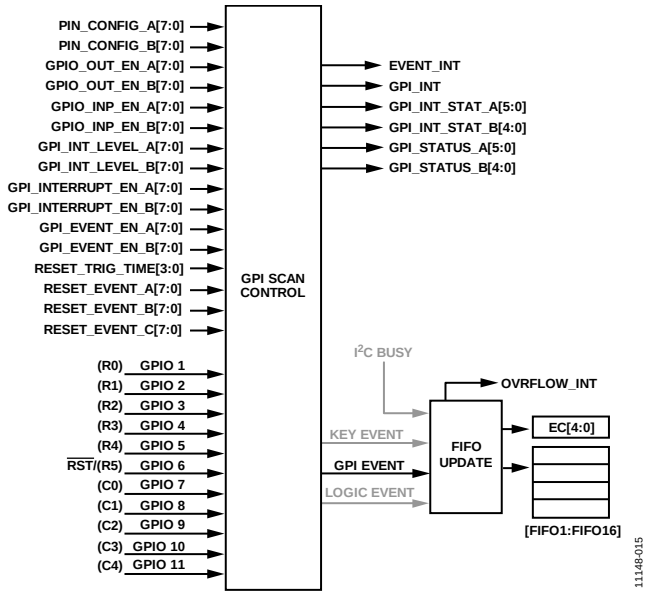



*Figure 15. GPI Scan and Detect Block*

The current input state of each GPI can be read back using the GPI_STATUS_x registers (Register 0x15 and Register 0x16). Each GPI can be programmed to generate an interrupt via the GPI_INTERRUPT_EN_x registers (Register 0x1F and Register 0x20). The interrupt status is stored in the GPI_INT_STAT_x registers (Register 0x13 and Register 0x14). GPI interrupts can be programmed to trigger on the positive or negative edge by configuring the GPI_INT_LEVEL_x registers (Register 0x1B and Register 0x1C). If any GPI interrupt is triggered, the master GPI_INT interrupt bit (Register 0x01, Bit 1) is also triggered. Figure 16 shows a single GPI and how it affects its corresponding status and the interrupt status bits.

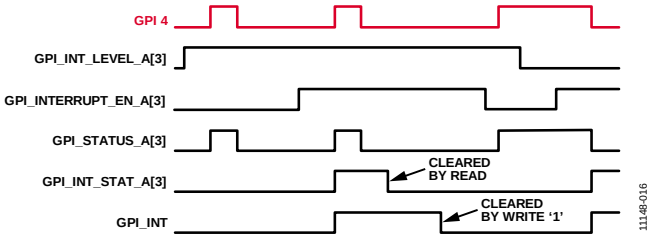



*Figure 16. Single GPI Example*

GPIs can be programmed to generate FIFO events via the GPI_EVENT_EN_x registers (Register 0x1D and Register 0x1E). GPIs in this mode do not generate GPI_INT interrupts. Instead, they generate EVENT_INT interrupts (Register 0x01, Bit 0). Figure 17 shows several GPI lines and their effects on the FIFO and event count, EC[4:0].

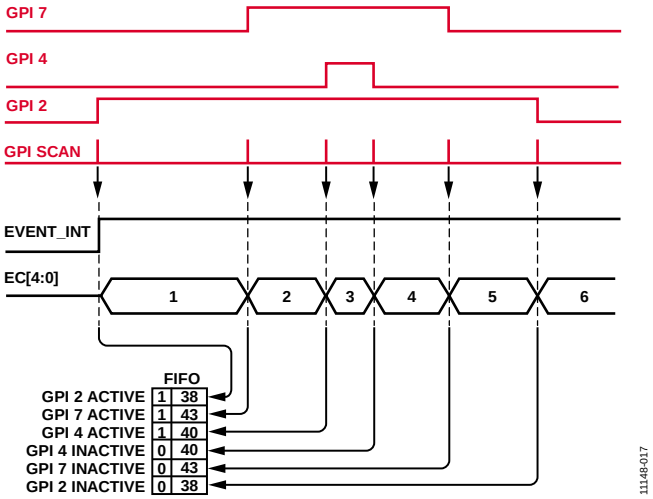



*Figure 17. Multiple GPI Example*

The GPI scanner is idle until it detects a level transition. It then scans the GPI inputs and updates accordingly. After updating, it returns immediately to idle; it does not scan/wait, like the key scanner. As a result, the GPI scanner can detect both edges of narrow pulses after they pass the 70 µs input debounce filter.

## GPO OUTPUT

Each of the 10 input/output lines can be configured as a general-purpose output (GPO) line using the GPIO_OUT_EN_A and GPIO_OUT_EN_B registers (Register 0x27 and Register 0x28). GPIO lines can be configured to allow both input and output at the same time (see Figure 5 for a detailed diagram of the I/O structure). GPO configuration and usage are programmed in the GPO_DATA_OUT_x and GPO_OUT_MODE_x registers (Register 0x23 to Register 0x26). See the Detailed Register Descriptions section for more information.

## LOGIC BLOCK

Several of the ADP5586 input/output lines can be used as inputs and outputs for implementing some common logic functions.

The R1, R2, and R3 input/output pins can be used as inputs, and the R0 input/output pin can be used as an output for the logic block. When the R1, R2, and R3 input lines are used, the GPIO_4_INP_EN, GPIO_3_INP_EN, and GPIO_2_INP_EN bits (Register 0x29, Bits[3:1]) must be enabled to accept inputs.

When the R0 pin is used as an output for the logic block, the GPIO_1_OUT_EN bit (Register 0x27, Bit 0) must be enabled.

The outputs from the logic block can be configured to generate interrupts. They can also be configured to generate events on the FIFO.

Figure 19 shows a detailed diagram of the internal makeup of the logic block, illustrating the possible logic functions that can be implemented.

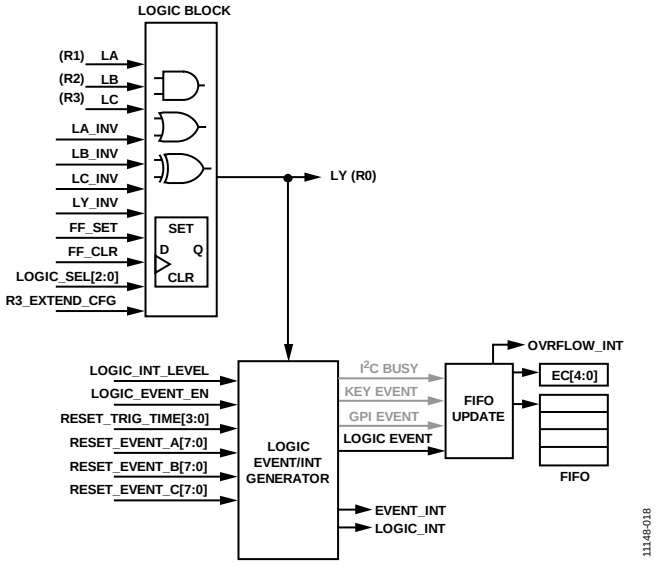



*Figure 18. Logic Block Overview*

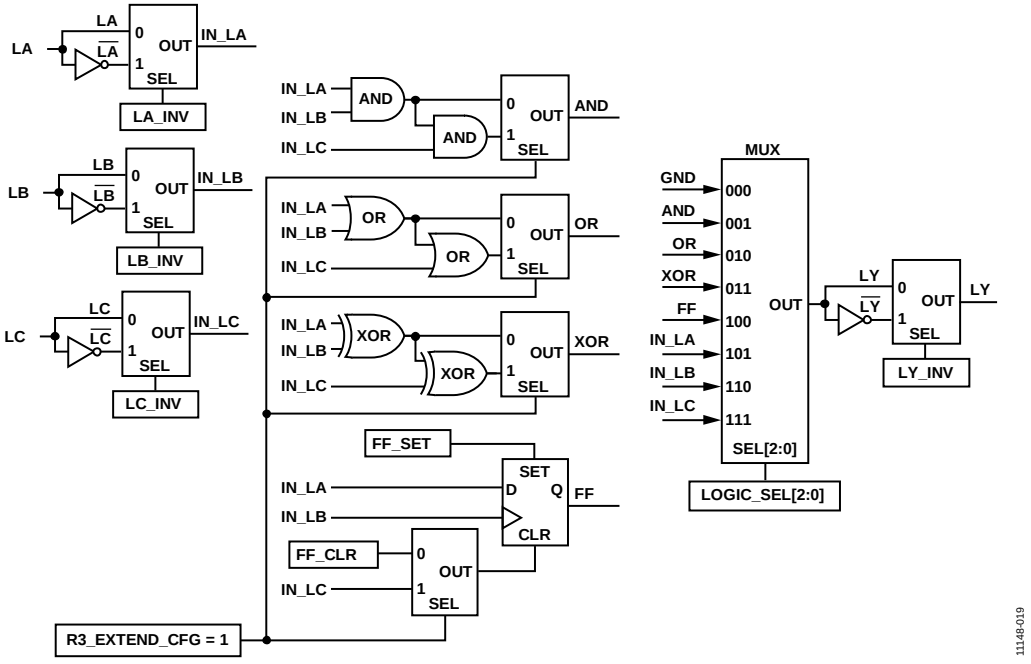



*Figure 19. Logic Block Internal Makeup*

## RESET BLOCK

The ADP5586 features a reset block that can generate reset conditions if certain events are detected simultaneously. Up to three reset trigger events can be programmed for RESET_OUT. The event scan control blocks monitor whether these events are present for the duration of RESET_TRIG_TIME[3:0] (Register 0x2E, Bits[5:2]). If they are present, reset-initiate signals are sent to the reset generator blocks. The generated reset signal pulse width is programmable.

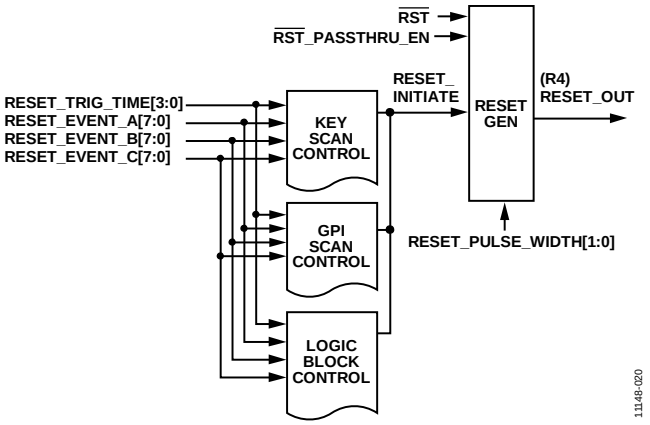



*Figure 20. Reset Blocks*

The RESET_OUT signal uses the R4 I/O pin as its output, which must be configured via the GPIO_5_OUT_EN bit (Register 0x27, Bit 4) to enable the output function. A pass-through mode also allows the $\overline{\text{RST}}$ pin function to be output on the R4 pin.

The reset generation signals are useful in situations where the system processor has locked up and the system is unresponsive to input events. The user can press one of the reset event combinations and initiate a system-wide reset, which eliminates the need to remove the battery from the system and perform a hard reset.

The use of the immediate trigger time setting (see Table 55) is recommended only in very low noise conditions with good debounce; otherwise, false triggering may occur.

## INTERRUPTS

The $\overline{\text{INT}}$ pin can be asserted low if any of the internal interrupt sources is active. The user can select which internal interrupts interact with the external interrupt pin in Register 0x3E (see Table 71). Register 0x3D allows the user to choose whether the external interrupt pin remains asserted, or deasserts for 50 µs and then reasserts, as in the case where multiple internal interrupts are asserted and one is cleared (see Table 70).

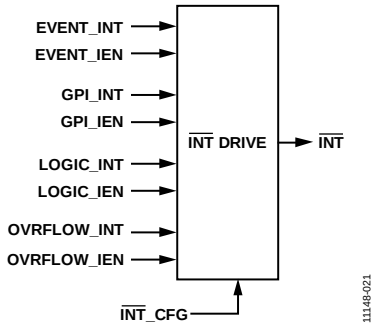



*Figure 21. Asserting $\overline{\text{INT}}$ Low*

## PULSE GENERATORS

The ADP5586 contains two pulse generators that are suitable for driving indicator LED drive signals, as well as watchdog timers and other extended time pulsed applications. The ADP5586 allows for eight bits of definition for both the on time and period of the generated pulse. To allow for extended timings, the user can choose between a 1 ms clock and a 125 ms clock to increment these timers. The PULSE_GEN_1_PERIOD and PULSE_GEN_2_PERIOD registers (Register 0x30 and Register 0x33, respectively) define the periods of the two pulse generators. Choosing a clock period of 125 ms in the PULSE_GEN_CONFIG register (Register 0x35, Bit 1 and Bit 5) allows for the setting of pulse generator periods of up to 31.875 sec. Setting the PULSE_GEN_x_ON_CLK bit to a step size of 125 ms and the PULSE_GEN_x_PRD_CLK bit to a step size of 1 ms is not a supported configuration.

To support active low applications, a signal inversion can be programmed in the PULSE_GEN_CONFIG register, using Bit 7 and Bit 3 (PULSE_GEN_x_INV). Delays can be introduced to create synchronized offsets between the channels. If both channels are enabled at the same time (that is, enabled from the same I²C write), the difference in delays is the offset between the channels. If a single channel is active and delays are to be synchronized, the user must first disable both pulse generators before enabling both pulse generators with the same I²C write command. The delay counter uses the same clock selection as the period counter. See Table 56 through Table 61 for more details. To enable pulse generator output on C1 and/or C0, the GPIO_8_OUT_EN bit and/or the GPIO_7_OUT_EN bit (Register 0x28, Bits[1:0]) must be enabled.

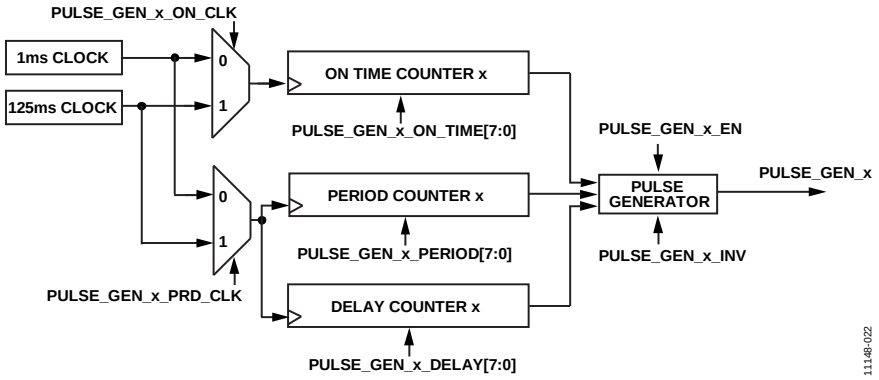



*Figure 22. Pulse Generator Block Diagram*

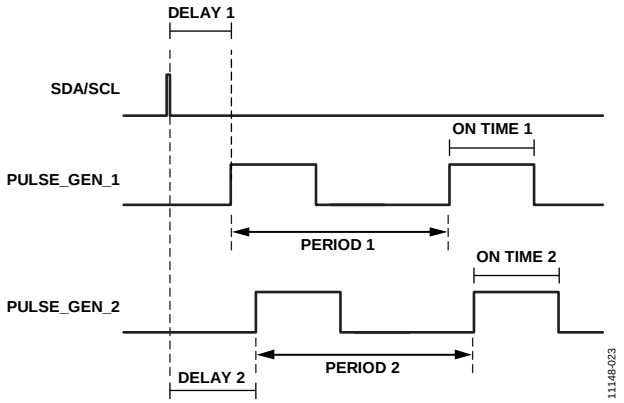



*Figure 23. Example Pulse Generator Timing*

# REGISTER INTERFACE

Register access to the ADP5586 is acquired via its I²C-compatible serial interface. The interface can support clock frequencies of up to 1 MHz. If the user is accessing the FIFO or key event counter (KEC), FIFO/KEC updates are paused. If the clock frequency is very low, events may not be recorded in a timely manner. FIFO or KEC updates can happen up to 23 µs after an interrupt is asserted because of the number of I²C cycles required to perform an I²C read or write. This delay should not present an issue to the user.

Figure 24 shows a typical write sequence for programming an internal register. The cycle begins with a start condition, followed by the hard coded 7-bit device address, which for the ADP5586 is 0x34, followed by the R/$\overline{W}$ bit set to 0 for a write cycle. The ADP5586 acknowledges the address byte by pulling the data line low. The address of the register to which data is to be written is sent next. The ADP5586 acknowledges the register pointer byte by pulling the data line low. The data byte to be written is sent next. The ADP5586 acknowledges the data byte by pulling the data line low. A stop condition completes the sequence.

Figure 25 shows a typical multibyte write sequence for programming internal registers. The cycle begins with a start condition followed by the 7-bit device address (0x34), followed by the R/$\overline{W}$ bit, which is set to 0 for a write cycle. The ADP55866 acknowledges the address byte by pulling the data line low. The address of the register to which data is to be written is sent next. The ADP5586 acknowledges the register pointer byte by pulling the data line low. The data byte to be written is sent next. The ADP5586 acknowledges the data byte by pulling the data line low. The pointer address is then incremented to write the next data byte, until it finishes writing the n data byte. The ADP5586 pulls the data line low after every byte, and a stop condition completes the sequence.

Figure 26 shows a typical byte read sequence for reading internal registers. The cycle begins with a start condition followed by the 7-bit device address, followed by the R/$\overline{W}$ bit set to 0 for a write cycle. The ADP5586 acknowledges the address byte by pulling the data line low. The address of the register from which data is to be read is sent next. The ADP5586 acknowledges the register pointer byte by pulling the data line low. A start condition is repeated, followed by the 7-bit device address (0x34), followed by the R/$\overline{W}$ bit set to 1 for a read cycle. The ADP5586 acknowledges the address byte by pulling the data line low. The 8-bit data is then read. The host pulls the data line high (no acknowledge), and a stop condition completes the sequence.




*Figure 24. I²C Single Byte Write Sequence*

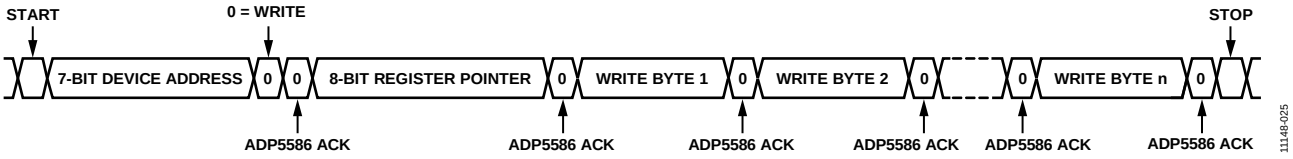



*Figure 25. I²C Multibyte Write Sequence*

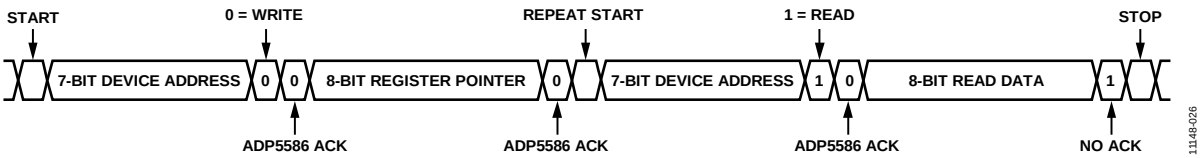



*Figure 26. I²C Single Byte Read Sequence*

Figure 27 shows a typical multibyte read sequence for reading internal registers. The cycle begins with a start condition followed by the 7-bit device address (0x34), followed by the R/$\overline{W}$ bit set to 0 for a write cycle. The ADP5586 acknowledges the address byte by pulling the data line low. The address of the register from which data is to be read is sent next. The ADP5586 acknowledges the register pointer byte by pulling the data line low. A start condition is repeated, followed by the 7-bit device address (0x34), followed by the R/$\overline{W}$ bit set to 1 for a read cycle. The ADP5586 acknowledges the address byte by pulling the data line low. Next, the 8-bit data is then read. The address pointer is then incremented to read the next data byte, and the host continues to pull the data line low for each byte (master acknowledge) until the n data byte is read. The host pulls the data line high (no acknowledge) after the last byte is read, and a stop condition completes the sequence.

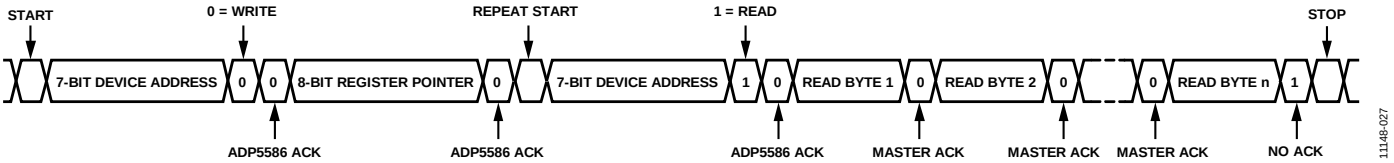



*Figure 27. I²C Multibyte Read Sequence*

# REGISTER MAP

**Table 7.**

| Reg Addr | Register Name | R/W[1] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00 | ID | R | MAN_ID | | | | REV_ID | | | |
| 0x01 | INT_STATUS | R/W | Reserved | | | LOGIC_INT | Reserved | OVRFLOW_INT | GPI_INT | EVENT_INT |
| 0x02 | Status | R | Reserved | LOGIC_STAT | Reserved | EC[4:0] | | | | |
| 0x03 | FIFO_1 | R | EVENT1_STATE | EVENT1_IDENTIFIER[6:0] | | | | | | |
| 0x04 | FIFO_2 | R | EVENT2_STATE | EVENT2_IDENTIFIER[6:0] | | | | | | |
| 0x05 | FIFO_3 | R | EVENT3_STATE | EVENT3_IDENTIFIER[6:0] | | | | | | |
| 0x06 | FIFO_4 | R | EVENT4_STATE | EVENT4_IDENTIFIER[6:0] | | | | | | |
| 0x07 | FIFO_5 | R | EVENT5_STATE | EVENT5_IDENTIFIER[6:0] | | | | | | |
| 0x08 | FIFO_6 | R | EVENT6_STATE | EVENT6_IDENTIFIER[6:0] | | | | | | |
| 0x09 | FIFO_7 | R | EVENT7_STATE | EVENT7_IDENTIFIER[6:0] | | | | | | |
| 0x0A | FIFO_8 | R | EVENT8_STATE | EVENT8_IDENTIFIER[6:0] | | | | | | |
| 0x0B | FIFO_9 | R | EVENT9_STATE | EVENT9_IDENTIFIER[6:0] | | | | | | |
| 0x0C | FIFO_10 | R | EVENT10_STATE | EVENT10_IDENTIFIER[6:0] | | | | | | |
| 0x0D | FIFO_11 | R | EVENT11_STATE | EVENT11_IDENTIFIER[6:0] | | | | | | |
| 0x0E | FIFO_12 | R | EVENT12_STATE | EVENT12_IDENTIFIER[6:0] | | | | | | |
| 0x0F | FIFO_13 | R | EVENT13_STATE | EVENT13_IDENTIFIER[6:0] | | | | | | |
| 0x10 | FIFO_14 | R | EVENT14_STATE | EVENT14_IDENTIFIER[6:0] | | | | | | |
| 0x11 | FIFO_15 | R | EVENT15_STATE | EVENT15_IDENTIFIER[6:0] | | | | | | |
| 0x12 | FIFO_16 | R | EVENT16_STATE | EVENT16_IDENTIFIER[6:0] | | | | | | |
| 0x13 | GPI_INT_STAT_A | R | Reserved | | GPI_6_INT | GPI_5_INT | GPI_4_INT | GPI_3_INT | GPI_2_INT | GPI_1_INT |
| 0x14 | GPI_INT_STAT_B | R | Reserved | | | GPI_11_INT | GPI_10_INT | GPI_9_INT | GPI_8_INT | GPI_7_INT |
| 0x15 | GPI_STATUS_A | R | Reserved | | GPI_6_STAT | GPI_5_STAT | GPI_4_STAT | GPI_3_STAT | GPI_2_STAT | GPI_1_STAT |
| 0x16 | GPI_STATUS_B | R | Reserved | | | GPI_11_STAT | GPI_10_STAT | GPI_9_STAT | GPI_8_STAT | GPI_7_STAT |
| 0x17 | R_PULL_CONFIG_A | R/W | R3_PULL_CFG | | R2_PULL_CFG | | R1_PULL_CFG | | R0_PULL_CFG | |
| 0x18 | R_PULL_CONFIG_B | R/W | Reserved | | | | R5_PULL_CFG | | R4_PULL_CFG | |
| 0x19 | R_PULL_CONFIG_C | R/W | C3_PULL_CFG | | C2_PULL_CFG | | C1_PULL_CFG | | C0_PULL_CFG | |
| 0x1A | R_PULL_CONFIG_D | R/W | Reserved | | | | | | C4_PULL_CFG | |
| 0x1B | GPI_INT_LEVEL_A | R/W | Reserved | | GPI_6_INT_LEVEL | GPI_5_INT_LEVEL | GPI_4_INT_LEVEL | GPI_3_INT_LEVEL | GPI_2_INT_LEVEL | GPI_1_INT_LEVEL |
| 0x1C | GPI_INT_LEVEL_B | R/W | Reserved | | | GPI_11_INT_LEVEL | GPI_10_INT_LEVEL | GPI_9_INT_LEVEL | GPI_8_INT_LEVEL | GPI_7_INT_LEVEL |
| 0x1D | GPI_EVENT_EN_A | R/W | Reserved | | GPI_6_EVENT_EN | GPI_5_EVENT_EN | GPI_4_EVENT_EN | GPI_3_EVENT_EN | GPI_2_EVENT_EN | GPI_1_EVENT_EN |
| 0x1E | GPI_EVENT_EN_B | R/W | Reserved | | | GPI_11_EVENT_EN | GPI_10_EVENT_EN | GPI_9_EVENT_EN | GPI_8_EVENT_EN | GPI_7_EVENT_EN |
| 0x1F | GPI_INTERRUPT_EN_A | R/W | Reserved | | GPI_6_INT_EN | GPI_5_INT_EN | GPI_4_INT_EN | GPI_3_INT_EN | GPI_2_INT_EN | GPI_1_INT_EN |
| 0x20 | GPI_INTERRUPT_EN_B | R/W | Reserved | | | GPI_11_INT_EN | GPI_10_INT_EN | GPI_9_INT_EN | GPI_8_INT_EN | GPI_7_INT_EN |
| 0x21 | DEBOUNCE_DIS_A | R/W | Reserved | | GPI_6_DEB_DIS | GPI_5_DEB_DIS | GPI_4_DEB_DIS | GPI_3_DEB_DIS | GPI_2_DEB_DIS | GPI_1_DEB_DIS |
| 0x22 | DEBOUNCE_DIS_B | R/W | Reserved | | | GPI_11_DEB_DIS | GPI_10_DEB_DIS | GPI_9_DEB_DIS | GPI_8_DEB_DIS | GPI_7_DEB_DIS |
| 0x23 | GPO_DATA_OUT_A | R/W | Reserved | | GPO_6_DATA | GPO_5_DATA | GPO_4_DATA | GPO_3_DATA | GPO_2_DATA | GPO_1_DATA |
| 0x24 | GPO_DATA_OUT_B | R/W | Reserved | | | GPO_11_DATA | GPO_10_DATA | GPO_9_DATA | GPO_8_DATA | GPO_7_DATA |
| 0x25 | GPO_OUT_MODE_A | R/W | Reserved | | GPO_6_OUT_MODE | GPO_5_OUT_MODE | GPO_4_OUT_MODE | GPO_3_OUT_MODE | GPO_2_OUT_MODE | GPO_1_OUT_MODE |
| 0x26 | GPO_OUT_MODE_B | R/W | Reserved | | | GPO_11_OUT_MODE | GPO_10_OUT_MODE | GPO_9_OUT_MODE | GPO_8_OUT_MODE | GPO_7_OUT_MODE |
| 0x27 | GPIO_OUT_EN_A | R/W | Reserved | | GPIO_6_OUT_EN | GPIO_5_OUT_EN | GPIO_4_OUT_EN | GPIO_3_OUT_EN | GPIO_2_OUT_EN | GPIO_1_OUT_EN |
| 0x28 | GPIO_OUT_EN_B | R/W | Reserved | | | GPIO_11_OUT_EN | GPIO_10_OUT_EN | GPIO_9_OUT_EN | GPIO_8_OUT_EN | GPIO_7_OUT_EN |

| Reg Addr | Register Name | R/W[1] | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x29 | GPIO_INP_EN_A | R/W | Reserved | | GPIO_6_ INP_EN | GPIO_5_ INP_EN | GPIO_4_ INP_EN | GPIO_3_ INP_EN | GPIO_2_ INP_EN | GPIO_1_ INP_EN |
| 0x2A | GPIO_INP_EN_B | R/W | Reserved | | | GPIO_11_ INP_EN | GPIO_10_ INP_EN | GPIO_9_ INP_EN | GPIO_8_ INP_EN | GPIO_7_ INP_EN |
| 0x2B | RESET_EVENT_A | R/W | RESET_EVENT_ A_LEVEL | RESET_EVENT_A, Bits[6:0] | | | | | | |
| 0x2C | RESET_EVENT_B | R/W | RESET_EVENT_ B_LEVEL | RESET_EVENT_B, Bits[6:0] | | | | | | |
| 0x2D | RESET_EVENT_C | R/W | RESET_EVENT_ C_LEVEL | RESET_EVENT_C, Bits[6:0] | | | | | | |
| 0x2E | RESET_CFG | R/W | RESET_POL | $\overline{\text{RST}}$_PASSTHRU_EN | RESET_TRIG_TIME, Bits[3:0] | | | | RESET_PULSE_WIDTH, Bits[1:0] | |
| 0x2F | PULSE_GEN_1_ DELAY | R/W | PULSE_GEN_1_DELAY, Bits[7:0] | | | | | | | |
| 0x30 | PULSE_GEN_1_ PERIOD | R/W | PULSE_GEN_1_PERIOD, Bits[7:0] | | | | | | | |
| 0x31 | PULSE_GEN_1_ ON_TIME | R/W | PULSE_GEN_1_ON_TIME, Bits[7:0] | | | | | | | |
| 0x32 | PULSE_GEN_2_ DELAY | R/W | PULSE_GEN_2_DELAY, Bits[7:0] | | | | | | | |
| 0x33 | PULSE_GEN_2_ PERIOD | R/W | PULSE_GEN_2_PERIOD, Bits[7:0] | | | | | | | |
| 0x34 | PULSE_GEN_2_ ON_TIME | R/W | PULSE_GEN_2_ON_TIME, Bits[7:0] | | | | | | | |
| 0x35 | PULSE_GEN_ CONFIG | R/W | PULSE_ GEN_1_INV | PULSE_GEN_1_ ON_CLK | PULSE_ GEN_1_ PRD_CLK | PULSE_ GEN_1_EN | PULSE_ GEN_2_INV | PULSE_ GEN_2_ ON_CLK | PULSE_ GEN_2_ PRD_CLK | PULSE_ GEN_2_EN |
| 0x36 | LOGIC_CFG | R/W | Reserved | LY_INV | LC_INV | LB_INV | LA_INV | LOGIC_SEL, Bits[2:0] | | |
| 0x37 | LOGIC_FF_CFG | R/W | Reserved | | | | | | FF_SET | FF_CLR |
| 0x38 | LOGIC_INT_ EVENT_EN | R/W | Reserved | | | | | LY_DBNC_ DIS | LOGIC_ EVENT_EN | LOGIC_ INT_LEVEL |
| 0x39 | POLL_TIME_CFG | R/W | Reserved | | | | PRECHARGE_ TIME | Reserved | KEY_POLL_TIME, Bits[1:0] | |
| 0x3A | PIN_CONFIG_A | R/W | Reserved | | R5_CONFIG | R4_CONFIG | R3_CONFIG | R2_CONFIG | R1_CONFIG | R0_CONFIG |
| 0x3B | PIN_CONFIG_B | R/W | Reserved | | | C4_CONFIG | C3_CONFIG | C2_CONFIG | C1_CONFIG | C0_CONFIG |
| 0x3C | PIN_CONFIG_C | R/W | PULL_SELECT | C0_EXTEND_CFG | R4_EXTEND_ CFG | C1_EXTEND_ CFG | R3_EXTEND_ CFG | Reserved | | R0_ EXTEND_ CFG |
| 0x3D | GENERAL_CFG | R/W | OSC_EN | OSC_FREQ, Bits[1:0] | | Reserved | | SW_RESET | $\overline{\text{INT}}$_CFG | $\overline{\text{RST}}$_CFG |
| 0x3E | INT_EN | R/W | Reserved | | | LOGIC_IEN | Reserved | OVRFLOW_ IEN | GPI_IEN | EVENT_IEN |

[1] R means read, W means write, and R/W means read/write.

## DETAILED REGISTER DESCRIPTIONS

Note that all registers default to 0000 0000, unless otherwise specified.

### *ID, Register 0x00*

Default: 0011 XXXX (where X = don't care)

**Table 8. ID Bit Descriptions**

| Bits | Bit Name | Access | Description |
|---|---|---|---|
| [7:4] | MAN_ID | Read only | Manufacturer ID, default = 0011 |
| [3:0] | REV_ID | Read only | Revision ID |

### *INT_STATUS, Register 0x01*

**Table 9. INT_STATUS Bit Descriptions**

| Bits | Bit Name | Access | Description[1] |
|---|---|---|---|
| [7:5] | Reserved | Reserved | Reserved. |
| 4 | LOGIC_INT | Read/write | 0 = no interrupt.<br>1 = interrupt due to a general logic condition. |
| 3 | Reserved | Reserved | Reserved. |
| 2 | OVERFLOW_INT | Read/write | 0 = no interrupt.<br>1 = interrupt due to an overflow condition. |
| 1 | GPI_INT | Read/write | This bit is not set by a GPI that has been configured to update the FIFO and event count. This bit cannot be cleared until all GPI_x_INT bits are cleared.<br>0 = no interrupt.<br>1 = interrupt due to a general GPI condition. |
| 0 | EVENT_INT | Read/write | 0 = no interrupt.<br>1 = interrupt due to key event (press/release), GPI event (GPI programmed for FIFO updates), or logic event (programmed for FIFO updates). |

[1] Interrupt bits are cleared by writing a 1 to the flag; writing a 0 or reading the flag has no effect.

### *Status, Register 0x02*

**Table 10. Status Bit Descriptions**

| Bits | Bit Name | Access | Description |
|---|---|---|---|
| 7 | Reserved | Reserved | Reserved. |
| 6 | LOGIC_STAT | Read only | 0 = output from logic block (LY) is low.<br>1 = output from logic block (LY) is high. |
| 5 | Reserved | Reserved | Reserved. |
| [4:0] | EC[4:0] | Read only | Event count value. Indicates how many events are currently stored on the FIFO. |

### *FIFO_1, Register 0x03*

**Table 11. FIFO_1 Bit Descriptions**

| Bits | Bit Name | Access | Description |
|---|---|---|---|
| 7 | EVENT1_STATE | Read only | This bit represents the state of the event that is recorded in the EVENT1_IDENTIFIER[6:0] bits.<br>For key events from Event 1 to Event 36, use the following settings:<br>1 = key is pressed.<br>0 = key is released.<br>For GPI and logic events from Event 37 to Event 48, use the following settings:<br>1 = GPI/logic is active.<br>0 = GPI/logic is inactive.<br>Active and inactive states for Event 37 to Event 48 are programmable. |
| [6:0] | EVENT1_IDENTIFIER[6:0] | Read only | Contains the event identifier for the pin. See Table 12 for event decoding information. |

**Table 12. Event Decoding**

| Event No. | Meaning | Event No. | Meaning |
|---|---|---|---|
| 0 | No event | 25 | Key 25 (R4, C4) |
| 1 | Key 1 (R0, C0) | 26 | Key 26 (R5, C0) |
| 2 | Key 2 (R0, C1) | 27 | Key 27 (R5, C1) |
| 3 | Key 3 (R0, C2) | 28 | Key 28 (R5, C2) |
| 4 | Key 4 (R0, C3) | 29 | Key 29 (R5, C3) |
| 5 | Key 5 (R0, C4) | 30 | Key 30 (R5, C4) |
| 6 | Key 6 (R1, C0) | 31 | Key 31 (R0, GND) |
| 7 | Key 7 (R1, C1) | 32 | Key 32 (R1, GND) |
| 8 | Key 8 (R1, C2) | 33 | Key 33 (R2, GND) |
| 9 | Key 9 (R1, C3) | 34 | Key 34 (R3, GND) |
| 10 | Key 10 (R1, C4) | 35 | Key 35 (R4, GND) |
| 11 | Key 11 (R2, C0) | 36 | Key 36 (R5, GND) |
| 12 | Key 12 (R2, C1) | 37 | GPI 1 (R0) |
| 13 | Key 13 (R2, C2) | 38 | GPI 2 (R1) |
| 14 | Key 14 (R2, C3) | 39 | GPI 3 (R2) |
| 15 | Key 15 (R2, C4) | 40 | GPI 4 (R3) |
| 16 | Key 16 (R3, C0) | 41 | GPI 5 (R4) |
| 17 | Key 17 (R3, C1) | 42 | GPI 6 (R5) |
| 18 | Key 18 (R3, C2) | 43 | GPI 7 (C0) |
| 19 | Key 19 (R3, C3) | 44 | GPI 8 (C1) |
| 20 | Key 20 (R3, C4) | 45 | GPI 9 (C2) |
| 21 | Key 21 (R4, C0) | 46 | GPI 10 (C3) |
| 22 | Key 22 (R4, C1) | 47 | GPI 11 (C4) |
| 23 | Key 23 (R4, C2) | 48 | Logic |
| 24 | Key 24 (R4, C3) | 49 to 127 | Unused |

### *FIFO_2, Register 0x04*

**Table 13. FIFO_2 Bit Descriptions**

| Bits | Bit Name | Access | Description |
|---|---|---|---|
| 7 | EVENT2_STATE | Read only | See Table 11 for bit descriptions. |
| [6:0] | EVENT2_IDENTIFIER[6:0] | Read only | See Table 11 for bit descriptions. |

### *FIFO_3, Register 0x05*

**Table 14. FIFO_3 Bit Descriptions**

| Bits | Bit Name | Access | Description |
|---|---|---|---|
| 7 | EVENT3_STATE | Read only | See Table 11 for bit descriptions. |
| [6:0] | EVENT3_IDENTIFIER[6:0] | Read only | See Table 11 for bit descriptions. |

### *FIFO_4, Register 0x06*

**Table 15. FIFO_4 Bit Descriptions**

| Bits | Bit Name | Access | Description |
|---|---|---|---|
| 7 | EVENT4_STATE | Read only | See Table 11 for bit descriptions. |
| [6:0] | EVENT4_IDENTIFIER[6:0] | Read only | See Table 11 for bit descriptions. |

### *FIFO_5, Register 0x07*

**Table 16. FIFO_5 Bit Descriptions**

| Bits | Bit Name | Access | Description |
|---|---|---|---|
| 7 | EVENT5_STATE | Read only | See Table 11 for bit descriptions. |
| [6:0] | EVENT5_IDENTIFIER[6:0] | Read only | See Table 11 for bit descriptions. |

### *FIFO_6 Register 0x08*

**Table 17. FIFO_6 Bit Descriptions**

| Bits | Bit Name | Access | Description |
|---|---|---|---|
| 7 | EVENT6_STATE | Read only | See Table 11 for bit descriptions. |
| [6:0] | EVENT6_IDENTIFIER[6:0] | Read only | See Table 11 for bit descriptions. |

### *FIFO_7, Register 0x09*

**Table 18. FIFO_7 Bit Descriptions**

| Bits | Bit Name | Access | Description |
|---|---|---|---|
| 7 | EVENT7_STATE | Read only | See Table 11 for bit descriptions. |
| [6:0] | EVENT7_IDENTIFIER[6:0] | Read only | See Table 11 for bit descriptions. |

### *FIFO_8, Register 0x0A*

**Table 19. FIFO_8 Bit Descriptions**

| Bits | Bit Name | Access | Description |
|---|---|---|---|
| 7 | EVENT8_STATE | Read only | See Table 11 for bit descriptions. |
| [6:0] | EVENT8_IDENTIFIER[6:0] | Read only | See Table 11 for bit descriptions. |

### *FIFO_9, Register 0x0B*

**Table 20. FIFO_9 Bit Descriptions**

| Bits | Bit Name | Access | Description |
|---|---|---|---|
| 7 | EVENT9_STATE | Read only | See Table 11 for bit descriptions. |
| [6:0] | EVENT9_IDENTIFIER[6:0] | Read only | See Table 11 for bit descriptions. |

### *FIFO_10, Register 0x0C*

**Table 21. FIFO_10 Bit Descriptions**

| Bits | Bit Name | Access | Description |
|---|---|---|---|
| 7 | EVENT10_STATE | Read only | See Table 11 for bit descriptions. |
| [6:0] | EVENT10_IDENTIFIER[6:0] | Read only | See Table 11 for bit descriptions. |

### *FIFO_11, Register 0x0D*

**Table 22. FIFO_11 Bit Descriptions**

| Bits | Bit Name | Access | Description |
|---|---|---|---|
| 7 | EVENT11_STATE | Read only | See Table 11 for bit descriptions. |
| [6:0] | EVENT11_IDENTIFIER[6:0] | Read only | See Table 11 for bit descriptions. |

### *FIFO_12, Register 0x0E*

**Table 23. FIFO_12 Bit Descriptions**

| Bits | Bit Name | Access | Description |
|---|---|---|---|
| 7 | EVENT12_STATE | Read only | See Table 11 for bit descriptions. |
| [6:0] | EVENT12_IDENTIFIER[6:0] | Read only | See Table 11 for bit descriptions. |

### *FIFO_13, Register 0x0F*

**Table 24. FIFO_13 Bit Descriptions**

| Bits | Bit Name | Access | Description |
|---|---|---|---|
| 7 | EVENT13_STATE | Read only | See Table 11 for bit descriptions. |
| [6:0] | EVENT13_IDENTIFIER[6:0] | Read only | See Table 11 for bit descriptions. |

### *FIFO_14, Register 0x10*

**Table 25. FIFO_14 Bit Descriptions**

| Bits | Bit Name | Access | Description |
|---|---|---|---|
| 7 | EVENT14_STATE | Read only | See Table 11 for bit descriptions. |
| [6:0] | EVENT14_IDENTIFIER[6:0] | Read only | See Table 11 for bit descriptions. |

### *FIFO_15, Register 0x11*

**Table 26. FIFO_15 Bit Descriptions**

| Bits | Bit Name | Access | Description |
|---|---|---|---|
| 7 | EVENT15_STATE | Read only | See Table 11 for bit descriptions. |
| [6:0] | EVENT15_IDENTIFIER[6:0] | Read only | See Table 11 for bit descriptions. |

### *FIFO_16, Register 0x12*

**Table 27. FIFO_16 Bit Descriptions**

| Bits | Bit Name | Access | Description |
|---|---|---|---|
| 7 | EVENT16_STATE | Read only | See Table 11 for bit descriptions. |
| [6:0] | EVENT16_IDENTIFIER[6:0] | Read only | See Table 11 for bit descriptions. |

### *GPI_INT_STAT_A, Register 0x13*

**Table 28. GPI_INT_STAT_A Bit Descriptions**

| Bits | Bit Name | Access | Description |
|---|---|---|---|
| [7:6] | Reserved | Reserved | Reserved. |
| 5 | GPI_6_INT | Read only | 0 = no interrupt<br>1 = interrupt due to GPI 6 (R5 pin). Cleared on read. |
| 4 | GPI_5_INT | Read only | 0 = no interrupt<br>1 = interrupt due to GPI 5 (R4 pin). Cleared on read. |
| 3 | GPI_4_INT | Read only | 0 = no interrupt<br>1 = interrupt due to GPI 4 (R3 pin). Cleared on read. |
| 2 | GPI_3_INT | Read only | 0 = no interrupt<br>1 = interrupt due to GPI 3 (R2 pin). Cleared on read. |
| 1 | GPI_2_INT | Read only | 0 = no interrupt<br>1 = interrupt due to GPI 2 (R1 pin). Cleared on read. |
| 0 | GPI_1_INT | Read only | 0 = no interrupt<br>1 = interrupt due to GPI 1 (R0 pin). Cleared on read. |

### *GPI_INT_STAT_B, Register 0x14*

**Table 29. GPI_INT_STAT_B Bit Descriptions**

| Bits | Bit Name | Access | Description |
|---|---|---|---|
| [7:5] | Reserved | Reserved | Reserved. |
| 4 | GPI_11_INT | Read only | 0 = no interrupt.<br>1 = interrupt due to GPI 11 (C4 pin). Cleared on read. |
| 3 | GPI_10_INT | Read only | 0 = no interrupt.<br>1 = interrupt due to GPI 10 (C3 pin). Cleared on read. |
| 2 | GPI_9_INT | Read only | 0 = no interrupt.<br>1 = interrupt due to GPI 9 (C2 pin). Cleared on read. |
| 1 | GPI_8_INT | Read only | 0 = no interrupt.<br>1 = interrupt due to GPI 8 (C1 pin). Cleared on read. |
| 0 | GPI_7_INT | Read only | 0 = no interrupt.<br>1 = interrupt due to GPI 7 (C0 pin). Cleared on read. |

### *GPI_STATUS_A, Register 0x15*

**Table 30. GPI_STATUS_A Bit Descriptions**

| Bits | Bit Name | Access | Description |
|---|---|---|---|
| [7:6] | Reserved | Reserved | Reserved. |
| 5 | GPI_6_STAT | Read only | 0 = GPI 6 (R5 pin) is low.<br>1 = GPI 6 (R5 pin) is high. |
| 4 | GPI_5_STAT | Read only | 0 = GPI 5 (R4 pin) is low.<br>1 = GPI 5 (R4 pin) is high. |
| 3 | GPI_4_STAT | Read only | 0 = GPI 4 (R3 pin) is low.<br>1 = GPI 4 (R3 pin) is high. |
| 2 | GPI_3_STAT | Read only | 0 = GPI 3 (R2 pin) is low.<br>1 = GPI 3 (R2 pin) is high. |
| 1 | GPI_2_STAT | Read only | 0 = GPI 2 (R1 pin) is low.<br>1 = GPI 2 (R1 pin) is high. |
| 0 | GPI_1_STAT | Read only | 0 = GPI 1 (R0 pin) is low.<br>1 = GPI 1 (R0 pin) is high. |

### *GPI_STATUS_B, Register 0x16*

**Table 31. GPI_STATUS_B Bit Descriptions**

| Bits | Bit Name | Access | Description |
|---|---|---|---|
| [7:5] | Reserved | Reserved | Reserved. |
| 4 | GPI_11_STAT | Read only | 0 = GPI 11 (C4 pin) is low.<br>1 = GPI 11 (C4 pin) is high. |
| 3 | GPI_10_STAT | Read only | 0 = GPI 10 (C3 pin) is low.<br>1 = GPI 10 (C3 pin) is high. |
| 2 | GPI_9_STAT | Read only | 0 = GPI 9 (C2 pin) is low.<br>1 = GPI 9 (C2 pin) is high. |
| 1 | GPI_8_STAT | Read only | 0 = GPI 8 (C1 pin) is low.<br>1 = GPI 8 (C1 pin) is high. |
| 0 | GPI_7_STAT | Read only | 0 = GPI 7 (C0 pin) is low.<br>1 = GPI 7 (C0 pin) is high. |

### *R_PULL_CONFIG_A, Register 0x17*

Default = 0101 0101

**Table 32. R_PULL_CONFIG_A Bit Descriptions**

| Bits | Bit Name | Access | Description |
|---|---|---|---|
| [7:6] | R3_PULL_CFG | Read/write | 00 = enables 300 kΩ pull-up resistor.<br>01 = enables 300 kΩ pull-down resistor.<br>10 = enables 100 kΩ pull-up resistor.<br>11 = disables all pull-up/pull-down resistors. |
| [5:4] | R2_PULL_CFG | Read/write | 00 = enables 300 kΩ pull-up resistor.<br>01 = enables 300 kΩ pull-down resistor.<br>10 = enables 100 kΩ pull-up resistor.<br>11 = disables all pull-up/pull-down resistors. |
| [3:2] | R1_PULL_CFG | Read/write | 00 = enables 300 kΩ pull-up resistor.<br>01 = enables 300 kΩ pull-down resistor.<br>10 = enables 100 kΩ pull-up resistor.<br>11 = disables all pull-up/pull-down resistors. |
| [1:0] | R0_PULL_CFG | Read/write | 00 = enables 300 kΩ pull-up resistor.<br>01 = enables 300 kΩ pull-down resistor.<br>10 = enables 100 kΩ pull-up resistor.<br>11 = disables all pull-up/pull-down resistors. |

### *R_PULL_CONFIG_B, Register 0x18*

Default = 0000 0101

**Table 33. R_PULL_CONFIG_B Bit Descriptions**

| Bits | Bit Name | Access | Description |
|---|---|---|---|
| [7:4] | Reserved | Reserved | Reserved. |
| [3:2] | R5_PULL_CFG | Read/write | Reserved except for the ADP5586ACBZ-01-R7 options.<br>00 = enables 300 kΩ pull-up resistor.<br>01 = enables 300 kΩ pull-down resistor.<br>10 = enables 100 kΩ pull-up resistor.<br>11 = disables all pull-up/pull-down resistors. |
| [1:0] | R4_PULL_CFG | Read/write | 00 = enables 300 kΩ pull-up resistor.<br>01 = enables 300 kΩ pull-down resistor.<br>10 = enables 100 kΩ pull-up resistor.<br>11 = disables all pull-up/pull-down resistors. |

### *R_PULL_CONFIG_C, Register 0x19*

Default = 0101 0001

**Table 34. R_PULL_CONFIG_C Bit Descriptions**

| Bits | Bit Name | Access | Description |
|---|---|---|---|
| [7:6] | C3_PULL_CFG | Read/write | 00 = enables 300 kΩ pull-up resistor.<br>01 = enables 300 kΩ pull-down resistor.<br>10 = enables 100 kΩ pull-up resistor.<br>11 = disables all pull-up/pull-down resistors. |
| [5:4] | C2_PULL_CFG | Read/write | 00 = enables 300 kΩ pull-up resistor.<br>01 = enables 300 kΩ pull-down resistor.<br>10 = enables 100 kΩ pull-up resistor.<br>11 = disables all pull-up/pull-down resistors. |
| [3:2] | C1_PULL_CFG | Read/write | 00 = enables 300 kΩ pull-up resistor.<br>01 = enables 300 kΩ pull-down resistor.<br>10 = enables 100 kΩ pull-up resistor.<br>11 = disables all pull-up/pull-down resistors. |
| [1:0] | C0_PULL_CFG | Read/write | 00 = enables 300 kΩ pull-up resistor.<br>01 = enables 300 kΩ pull-down resistor.<br>10 = enables 100 kΩ pull-up resistor.<br>11 = disables all pull-up/pull-down resistors. |

### *R_PULL_CONFIG_D, Register 0x1A*

Default = 0000 0001

**Table 35. R_PULL_CONFIG_D Bit Descriptions**

| Bits | Bit Name | Access | Description |
|---|---|---|---|
| [7:2] | Reserved | Reserved | Reserved. |
| [1:0] | C4_PULL_CFG | Read/write | 00 = enables 300 kΩ pull-up resistor.<br>01 = enables 300 kΩ pull-down resistor.<br>10 = enables 100 kΩ pull-up resistor.<br>11 = disables all pull-up/pull-down resistors. |

### *GPI_INT_LEVEL_A, Register 0x1B*

**Table 36. GPI_INT_LEVEL_A Bit Descriptions**

| Bits | Bit Name | Access | Description |
|---|---|---|---|
| [7:6] | Reserved | Reserved | Reserved. |
| 5 | GPI_6_INT_LEVEL | Read/write | 0 = GPI 6 interrupt is active low (GPI_6_INT sets whenever R5 is low).<br>1 = GPI 6 interrupt is active high (GPI_6_INT sets whenever R5 is high). |
| 4 | GPI_5_INT_LEVEL | Read/write | 0 = GPI 5 interrupt is active low (GPI_5_INT sets whenever R4 is low).<br>1 = GPI 5 interrupt is active high (GPI_5_INT sets whenever R4 is high). |
| 3 | GPI_4_INT_LEVEL | Read/write | 0 = GPI 4 interrupt is active low (GPI_4_INT sets whenever R3 is low).<br>1 = GPI 4 interrupt is active high (GPI_4_INT sets whenever R3 is high). |
| 2 | GPI_3_INT_LEVEL | Read/write | 0 = GPI 3 interrupt is active low (GPI_3_INT sets whenever R2 is low).<br>1 = GPI 3 interrupt is active high (GPI_3_INT sets whenever R2 is high). |
| 1 | GPI_2_INT_LEVEL | Read/write | 0 = GPI 2 interrupt is active low (GPI_2_INT sets whenever R1 is low).<br>1 = GPI 2 interrupt is active high (GPI_2_INT sets whenever R1 is high). |
| 0 | GPI_1_INT_LEVEL | Read/write | 0 = GPI 1 interrupt is active low (GPI_1_INT sets whenever R0 is low).<br>1 = GPI 1 interrupt is active high (GPI_1_INT sets whenever R0 is high). |

### *GPI_INT_LEVEL_B, Register 0x1C*

**Table 37. GPI_INT_LEVEL_B Bit Descriptions**

| Bits | Bit Name | Access | Description |
|---|---|---|---|
| [7:5] | Reserved | Reserved | Reserved. |
| 4 | GPI_11_INT_LEVEL | Read/write | 0 = GPI 11 interrupt is active low (GPI_11_INT sets whenever R10 is low).<br>1 = GPI 11 interrupt is active high (GPI_11_INT sets whenever R10 is high). |
| 3 | GPI_10_INT_LEVEL | Read/write | 0 = GPI 10 interrupt is active low (GPI_10_INT sets whenever R9 is low).<br>1 = GPI 10 interrupt is active high (GPI_10_INT sets whenever R9 is high). |
| 2 | GPI_9_INT_LEVEL | Read/write | 0 = GPI 9 interrupt is active low (GPI_9_INT sets whenever R8 is low).<br>1 = GPI 9 interrupt is active high (GPI_9_INT sets whenever R8 is high). |
| 1 | GPI_8_INT_LEVEL | Read/write | 0 = GPI 8 interrupt is active low (GPI_8_INT sets whenever R7 is low).<br>1 = GPI 8 interrupt is active high (GPI_8_INT sets whenever R7 is high). |
| 0 | GPI_7_INT_LEVEL | Read/write | 0 = GPI 7 interrupt is active low (GPI_7_INT sets whenever R6 is low).<br>1 = GPI 7 interrupt is active high (GPI_7_INT sets whenever R6 is high). |

### *GPI_EVENT_EN_A, Register 0x1D*

**Table 38. GPI_EVENT_EN_A Bit Descriptions**

| Bits | Bit Name | Access | Description |
|---|---|---|---|
| [7:6] | Reserved | Reserved | Reserved. |
| 5 | GPI_6_EVENT_EN | Read/write | 0 = disables GPI events from GPI 6.<br>1 = allows GPI 6 activity to generate events on the FIFO.[1] |
| 4 | GPI_5_EVENT_EN | Read/write | 0 = disables GPI events from GPI 5.<br>1 = allows GPI 5 activity to generate events on the FIFO.[1] |
| 3 | GPI_4_EVENT_EN | Read/write | 0 = disables GPI events from GPI 4.<br>1 = allows GPI 4 activity to generate events on the FIFO.[1] |
| 2 | GPI_3_EVENT_EN | Read/write | 0 = disables GPI events from GPI 3.<br>1 = allows GPI 3 activity to generate events on the FIFO.[1] |
| 1 | GPI_2_EVENT_EN | Read/write | 0 = disables GPI events from GPI 2.<br>1 = allows GPI 2 activity to generate events on the FIFO.[1] |
| 0 | GPI_1_EVENT_EN | Read/write | 0 = disables GPI events from GPI 1.<br>1 = allows GPI 1 activity to generate events on the FIFO.[1] |

[1] GPIs in this mode are considered FIFO events and can be used for unlock purposes. GPI activity in this mode causes EVENT_INT interrupts. GPIs in this mode do not generate GPI_INT interrupts.

### *GPI_EVENT_EN_B, Register 0x1E*

**Table 39. GPI_EVENT_EN_B Bit Descriptions**

| Bits | Bit Name | Access | Description |
|---|---|---|---|
| [7:5] | Reserved | Reserved | Reserved. |
| 4 | GPI_11_EVENT_EN | Read/write | 0 = disables GPI events from GPI 11.<br>1 = allows GPI 11 activity to generate events on the FIFO.[1] |
| 3 | GPI_10_EVENT_EN | Read/write | 0 = disables GPI events from GPI 10.<br>1 = allows GPI 10 activity to generate events on the FIFO.[1] |
| 2 | GPI_9_EVENT_EN | Read/write | 0 = disables GPI events from GPI 9.<br>1 = allows GPI 9 activity to generate events on the FIFO.[1] |
| 1 | GPI_8_EVENT_EN | Read/write | 0 = disables GPI events from GPI 8.<br>1 = allows GPI 8activity to generate events on the FIFO.[1] |
| 0 | GPI_7_EVENT_EN | Read/write | 0 = disables GPI events from GPI 7.<br>1 = allows GPI 7 activity to generate events on the FIFO.[1] |

[1] GPIs in this mode are considered FIFO events and can be used for unlock purposes. GPI activity in this mode causes EVENT_INT interrupts. GPIs in this mode do not generate GPI_INT interrupts.

### *GPI_INTERRUPT_EN_A, Register 0x1F*

**Table 40. GPI_INTERRUPT_EN_A Bit Descriptions**

| Bits | Bit Name | Access | Description |
|---|---|---|---|
| [7:6] | Reserved | Reserved | Reserved. |
| 5 | GPI_6_INT_EN | Read/write | 0 = GPI_6_INT is disabled.<br>1 = GPI_6_INT enabled. Asserts the GPI_INT bit (Register 0x01, Bit 1) if GPI_6_INT is set and the GPI 6 interrupt condition is met. |
| 4 | GPI_5_INT_EN | Read/write | 0 = GPI_5_INT is disabled.<br>1 = GPI_5_INT enabled. Asserts the GPI_INT bit (Register 0x01, Bit 1) if GPI_5_INT is set and the GPI 5 interrupt condition is met. |
| 3 | GPI_4_INT_EN | Read/write | 0 = GPI_4_INT is disabled.<br>1 = GPI_4_INT enabled. Asserts the GPI_INT bit (Register 0x01, Bit 1) if GPI_4_INT is set and the GPI 4 interrupt condition is met. |
| 2 | GPI_3_INT_EN | Read/write | 0 = GPI_3_INT is disabled.<br>1 = GPI_3_INT enabled. Asserts the GPI_INT bit (Register 0x01, Bit 1) if GPI_3_INT is set and the GPI 3 interrupt condition is met. |
| 1 | GPI_2_INT_EN | Read/write | 0 = GPI_2_INT is disabled.<br>1 = GPI_2_INT enabled. Asserts the GPI_INT bit (Register 0x01, Bit 1) if GPI_2_INT is set and the GPI 2 interrupt condition is met. |
| 0 | GPI_1_INT_EN | Read/write | 0 = GPI_1_INT is disabled.<br>1 = GPI_1_INT enabled. Asserts the GPI_INT bit (Register 0x01, Bit 1) if GPI_1_INT is set and the GPI 1 interrupt condition is met. |

### *GPI_INTERRUPT_EN_B, Register 0x20*

**Table 41. GPI_INTERRUPT_EN_B Bit Descriptions**

| Bits | Bit Name | Access | Description |
|---|---|---|---|
| [7:5] | Reserved | Reserved | Reserved. |
| 4 | GPI_11_INT_EN | Read/write | 0 = GPI_11_INT is disabled.<br>1 = GPI_11_INT enabled. Asserts the GPI_INT bit (Register 0x01, Bit 1) if GPI_11_INT is set and the GPI 11 interrupt condition is met. |
| 3 | GPI_10_INT_EN | Read/write | 0 = GPI_10_INT is disabled.<br>1 = GPI_10_INT enabled. Asserts the GPI_INT bit (Register 0x01, Bit 1) if GPI_10_INT is set and the GPI 10 interrupt condition is met. |
| 2 | GPI_9_INT_EN | Read/write | 0 = GPI_9_INT is disabled.<br>1 = GPI_9_INT enabled. Asserts the GPI_INT bit (Register 0x01, Bit 1) if GPI_9_INT is set and the GPI 9 interrupt condition is met. |
| 1 | GPI_8_INT_EN | Read/write | 0 = GPI_8_INT is disabled.<br>1 = GPI_8_INT enabled. Asserts the GPI_INT bit (Register 0x01, Bit 1) if GPI_8_INT is set and the GPI 8 interrupt condition is met. |
| 0 | GPI_7_INT_EN | Read/write | 0 = GPI_7_INT is disabled.<br>1 = GPI_7_INT enabled. Asserts the GPI_INT bit (Register 0x01, Bit 1) if GPI_7_INT is set and the GPI 7 interrupt condition is met. |

### *DEBOUNCE_DIS_A, Register 0x21*

**Table 42. DEBOUNCE_DIS_A Bit Descriptions**

| Bits | Bit Name | Access | Description |
|---|---|---|---|
| [7:6] | Reserved | Reserved | Reserved. |
| 5 | GPI_6_DEB_DIS | Read/write | 0 = debounce enabled on GPI 6.<br>1 = debounce disabled on GPI 6. |
| 4 | GPI_5_DEB_DIS | Read/write | 0 = debounce enabled on GPI 5.<br>1 = debounce disabled on GPI 5. |
| 3 | GPI_4_DEB_DIS | Read/write | 0 = debounce enabled on GPI 4.<br>1 = debounce disabled on GPI 4. |
| 2 | GPI_3_DEB_DIS | Read/write | 0 = debounce enabled on GPI 3.<br>1 = debounce disabled on GPI 3. |
| 1 | GPI_2_DEB_DIS | Read/write | 0 = debounce enabled on GPI 2.<br>1 = debounce disabled on GPI 2. |
| 0 | GPI_1_DEB_DIS | Read/write | 0 = debounce enabled on GPI 1.<br>1 = debounce disabled on GPI 1. |

### *DEBOUNCE_DIS_B, Register 0x22*

**Table 43. DEBOUNCE_DIS_B Bit Descriptions**

| Bits | Bit Name | Access | Description |
|---|---|---|---|
| [7:5] | Reserved | Reserved | Reserved. |
| 4 | GPI_11_DEB_DIS | Read/write | 0 = debounce enabled on GPI 11.<br>1 = debounce disabled on GPI 11. |
| 3 | GPI_10_DEB_DIS | Read/write | 0 = debounce enabled on GPI 10.<br>1 = debounce disabled on GPI 10. |
| 2 | GPI_9_DEB_DIS | Read/write | 0 = debounce enabled on GPI 9.<br>1 = debounce disabled on GPI 9. |
| 1 | GPI_8_DEB_DIS | Read/write | 0 = debounce enabled on GPI 8.<br>1 = debounce disabled on GPI 8. |
| 0 | GPI_7_DEB_DIS | Read/write | 0 = debounce enabled on GPI 7.<br>1 = debounce disabled on GPI 7. |

### *GPO_DATA_OUT_A, Register 0x23*

**Table 44. GPO_DATA_OUT_A Bit Descriptions**

| Bits | Bit Name | Access | Description |
|---|---|---|---|
| [7:6] | Reserved | Reserved | Reserved. |
| 5 | GPO_6_DATA | Read/write | 0 = sets output low.<br>1 = sets output high. |
| 4 | GPO_5_DATA | Read/write | 0 = sets output low.<br>1 = sets output high. |
| 3 | GPO_4_DATA | Read/write | 0 = sets output low.<br>1 = sets output high. |
| 2 | GPO_3_DATA | Read/write | 0 = sets output low.<br>1 = sets output high. |
| 1 | GPO_2_DATA | Read/write | 0 = sets output low.<br>1 = sets output high. |
| 0 | GPO_1_DATA | Read/write | 0 = sets output low.<br>1 = sets output high. |

### *GPO_DATA_OUT_B, Register 0x24*

**Table 45. GPO_DATA_OUT_B Bit Descriptions**

| Bits | Bit Name | Access | Description |
|---|---|---|---|
| [7:5] | Reserved | Reserved | Reserved. |
| 4 | GPO_11_DATA | Read/write | 0 = sets output low.<br>1 = sets output high. |
| 3 | GPO_10_DATA | Read/write | 0 = sets output low.<br>1 = sets output high. |
| 2 | GPO_9_DATA | Read/write | 0 = sets output low.<br>1 = sets output high. |
| 1 | GPO_8_DATA | Read/write | 0 = sets output low.<br>1 = sets output high. |
| 0 | GPO_7_DATA | Read/write | 0 = sets output low.<br>1 = sets output high. |

### *GPO_OUT_MODE_A, Register 0x25*

**Table 46. GPO_OUT_MODE_A Bit Descriptions**

| Bits | Bit Name | Access | Description |
|---|---|---|---|
| [7:6] | Reserved | Reserved | Reserved. |
| 5 | GPO_6_OUT_MODE | Read/write | 0 = push-pull.<br>1 = open drain. |
| 4 | GPO_5_OUT_MODE | Read/write | 0 = push-pull.<br>1 = open drain. |
| 3 | GPO_4_OUT_MODE | Read/write | 0 = push-pull.<br>1 = open drain. |
| 2 | GPO_3_ OUT_MODE | Read/write | 0 = push-pull.<br>1 = open drain. |
| 1 | GPO_2_OUT_MODE | Read/write | 0 = push-pull.<br>1 = open drain. |
| 0 | GPO_1_OUT_MODE | Read/write | 0 = push-pull.<br>1 = open drain. |

### *GPO_OUT_MODE_B, Register 0x26*

**Table 47. GPO_OUT_MODE_B Bit Descriptions**

| Bits | Bit Name | Access | Description |
|---|---|---|---|
| [7:5] | Reserved | Reserved | Reserved. |
| 4 | GPO_11_OUT_MODE | Read/write | 0 = push-pull.<br>1 = open drain. |
| 3 | GPO_10_OUT_MODE | Read/write | 0 = push-pull.<br>1 = open drain. |
| 2 | GPO_9_OUT_MODE | Read/write | 0 = push-pull.<br>1 = open drain. |
| 1 | GPO_8_OUT_MODE | Read/write | 0 = push-pull.<br>1 = open drain. |
| 0 | GPO_7_OUT_MODE | Read/write | 0 = push-pull.<br>1 = open drain. |

### *GPIO_OUT_EN_A, Register 0x27*

**Table 48. GPIO_OUT_EN_A Bit Descriptions**

| Bits | Bit Name | Access | Description |
|---|---|---|---|
| [7:6] | Reserved | Reserved | Reserved. |
| 5 | GPIO_6_OUT_EN | Read/write | 0 = GPIO 6 output disabled.<br>1 = GPIO 6 output enabled. |
| 4 | GPIO_5_OUT_EN | Read/write | 0 = GPIO 5 output disabled.<br>1 = GPIO 5 output enabled. |
| 3 | GPIO_4_OUT_EN | Read/write | 0 = GPIO 4 output disabled.<br>1 = GPIO 4 output enabled. |
| 2 | GPIO_3_OUT_EN | Read/write | 0 = GPIO 3 output disabled.<br>1 = GPIO 3 output enabled. |
| 1 | GPIO_2_OUT_EN | Read/write | 0 = GPIO 2 output disabled.<br>1 = GPIO 2 output enabled. |
| 0 | GPIO_1_OUT_EN | Read/write | 0 = GPIO 1 output disabled.<br>1 = GPIO 1 output enabled. |

### *GPIO_OUT_EN_B, Register 0x28*

**Table 49. GPIO_OUT_EN_B Bit Descriptions**

| Bits | Bit Name | Access | Description |
|---|---|---|---|
| [7:5] | Reserved | Reserved | Reserved. |
| 4 | GPIO_11_OUT_EN | Read/write | 0 = GPIO 11 output disabled.<br>1 = GPIO 11 output enabled. |
| 3 | GPIO_10_OUT_EN | Read/write | 0 = GPIO 10 output disabled.<br>1 = GPIO 10 output enabled. |
| 2 | GPIO_9_OUT_EN | Read/write | 0 = GPIO 9 output disabled.<br>1 = GPIO 9 output enabled. |
| 1 | GPIO_8_OUT_EN | Read/write | 0 = GPIO 8 output disabled.<br>1 = GPIO 8 output enabled. |
| 0 | GPIO_7_OUT_EN | Read/write | 0 = GPIO 7 output disabled.<br>1 = GPIO 7 output enabled. |

### *GPIO_INP_EN_A, Register 0x29*

**Table 50. GPIO_INP_EN_A Bit Descriptions**

| Bits | Bit Name | Access | Description |
|---|---|---|---|
| [7:6] | Reserved | Reserved | Reserved. |
| 5 | GPIO_6_INP_EN | Read/write | 0 = GPIO 6 input disabled.<br>1 = GPIO 6 input enabled. |
| 4 | GPIO_5_INP_EN | Read/write | 0 = GPIO 5 input disabled.<br>1 = GPIO 5 input enabled. |
| 3 | GPIO_4_INP_EN | Read/write | 0 = GPIO 4 input disabled.<br>1 = GPIO 4 input enabled. |
| 2 | GPIO_3_INP_EN | Read/write | 0 = GPIO 3 input disabled.<br>1 = GPIO 3 input enabled. |
| 1 | GPIO_2_INP_EN | Read/write | 0 = GPIO 2 input disabled.<br>1 = GPIO 2 input enabled. |
| 0 | GPIO_1_INP_EN | Read/write | 0 = GPIO 1 input disabled.<br>1 = GPIO 1 input enabled. |

### *GPIO_INP_EN_B, Register 0x2A*

**Table 51. GPIO_INP_EN_B Bit Descriptions**

| Bits | Bit Name | Access | Description |
|---|---|---|---|
| [7:5] | Reserved | Reserved | Reserved. |
| 4 | GPIO_11_INP_EN | Read/write | 0 = GPIO 11 input disabled.<br>1 = GPIO 11 input enabled. |
| 3 | GPIO_10_INP_EN | Read/write | 0 = GPIO 10 input disabled.<br>1 = GPIO 10 input enabled. |
| 2 | GPIO_9_INP_EN | Read/write | 0 = GPIO 9 input disabled.<br>1 = GPIO 9 input enabled. |
| 1 | GPIO_8_INP_EN | Read/write | 0 = GPIO 8 input disabled.<br>1 = GPIO 8 input enabled. |
| 0 | GPIO_7_INP_EN | Read/write | 0 = GPIO 7 input disabled.<br>1 = GPIO 7 input enabled. |

### *RESET_EVENT_A, Register 0x2B*

**Table 52. RESET_EVENT_A Bit Descriptions**

| Bits | Bit Name | Access | Description |
|---|---|---|---|
| 7 | RESET_EVENT_A_LEVEL | Read/write | Defines which level the first reset event should be to generate the RESET_OUT signal.<br>For key events, use the following settings:<br>0 = inactive event used as a reset condition.<br>1 = active event used as a reset condition.<br>For GPIs and logic outputs configured for FIFO updates, use the following settings:<br>0 = not applicable; releases not used for reset generation.<br>1 = press is used as a reset event. |
| [6:0] | RESET_EVENT_A[6:0] | Read/write | Defines an event that can be used to generate the RESET_OUT signal. Up to three events can be defined for generating the RESET_OUT signal, using RESET_EVENT_A[6:0], RESET_EVENT_B[6:0], and RESET_EVENT_C[6:0]. If one of the registers is 0, that register is not used for reset generation. All reset events must be detected at the same time to trigger the reset. |

### *RESET_EVENT_B, Register 0x2C*

**Table 53. RESET_EVENT_B Bit Descriptions**

| Bits | Bit Name | Access | Description |
|---|---|---|---|
| 7 | RESET_EVENT_B_LEVEL | Read/write | Defines which level the second reset event should be to generate the RESET_OUT signal. Refer to Table 52. |
| [6:0] | RESET_EVENT_B[6:0] | Read/write | Defines an event that can be used to generate the RESET_OUT signal. See Table 12. |

### *RESET_EVENT_C, Register 0x2D*

**Table 54. RESET_EVENT_C Bit Descriptions**

| Bits | Bit Name | Access | Description |
|---|---|---|---|
| 7 | RESET_EVENT_C_LEVEL | Read/write | Defines which level the third reset event should be to generate the RESET_OUT signal. Refer to Table 52. |
| [6:0] | RESET_EVENT_C[6:0] | Read/write | Defines an event that can be used to generate the RESET_OUT signal. See Table 12. |

### *RESET_CFG, Register 0x2E*

**Table 55. RESET_CFG Bit Descriptions**

| Bits | Bit Name | Access | Description |
|---|---|---|---|
| 7 | RESET_POL | Read/write | Sets the polarity of the RESET_OUT signal.<br>0 = RESET_OUT is active low.<br>1 = RESET_OUT is active high. |
| 6 | $\overline{\text{RST}}$_PASSTHRU_EN | Read/write | Allows the $\overline{\text{RST}}$ pin to override (OR with) the RESET_OUT signal. |
| [5:2] | RESET_TRIG_TIME[3:0] | Read/write | Defines the length of time that the reset events must be active before a RESET_OUT signal is generated. All events must be active at the same time for the same duration.<br>0000 = immediate.<br>0001 = 1.0 sec.<br>0010 = 1.5 sec.<br>0011 = 2.0 sec.<br>0100 = 2.5 sec.<br>0101 = 3.0 sec.<br>0110 = 3.5 sec.<br>0111 = 4.0 sec.<br>1000 = 5.0 sec.<br>1001 = 6.0 sec.<br>1010 = 7.0 sec.<br>1011 = 8.0 sec.<br>1100 = 9.0 sec.<br>1101 = 10.0 sec.<br>1110 = 11.0 sec.<br>1111 = 12.0 sec. |
| [1:0] | RESET_PULSE_WIDTH[1:0] | Read/write | Defines the pulse width of the RESET_OUT signal.<br>00 = 500 µs.<br>01 = 1 ms.<br>10 = 2 ms.<br>11 = 10 ms. |

### *PULSE_GEN_1_DELAY, Register 0x2F*

**Table 56. PULSE_GEN_1_DELAY Bit Descriptions**

| Bits | Bit Name | Access | Description | | |
|---|---|---|---|---|---|
| [7:0] | PULSE_GEN_1_DELAY[7:0] | Read/write | Defines initial delay from the first clock of the first enable of Pulse Generator 1. Delay is defined as the number of clock cycles of the chosen period clock speed (see Register 0x35). For example, | | |
| | | | | **PULSE_GEN_1_PRD_CLK** | |
| | | | **PULSE_GEN_1_DELAY** | **0** | **1** |
| | | | 0000 0000 | 0 | 0 ms |
| | | | 0000 0001 | 1 | 125 ms |
| | | | 0000 0010 | 2 | 250 ms |
| | | | 0000 0011 | 3 | 375 ms |
| | | | 0000 0100 | 4 | 500 ms |
| | | | … | … | … |
| | | | 1111 1110 | 254 ms | 31.750 sec |
| | | | 1111 1111 | 255 ms | 31.875 sec |

### *PULSE_GEN_1_PERIOD, Register 0x30*

**Table 57. PULSE_GEN_1_PERIOD Bit Descriptions**

| Bits | Bit Name | Access | Description | | |
|---|---|---|---|---|---|
| [7:0] | PULSE_GEN_1_PERIOD[7:0] | Read/write | Defines period of Pulse Generator 1. Period is defined as the number of clock cycles of the chosen period clock speed (see Register 0x35). For example, | | |
| | | | | PULSE_GEN_1_PRD_CLK | |
| | | | PULSE_GEN_1_PERIOD | 0 | 1 |
| | | | 0000 0000 | 0 ms | 0 ms |
| | | | 0000 0001 | 1 ms | 125 ms |
| | | | 0000 0010 | 2 ms | 250 ms |
| | | | 0000 0011 | 3 ms | 375 ms |
| | | | 0000 0100 | 4 ms | 500 ms |
| | | | … | … | … |
| | | | 1111 1110 | 254 ms | 31.750 sec |
| | | | 1111 1111 | 255 ms | 31.875 sec |

### *PULSE_GEN_1_ON_TIME, Register 0x31*

**Table 58. PULSE_GEN_1_ON_TIME Bit Descriptions**

| Bits | Bit Name | Access | Description | | |
|---|---|---|---|---|---|
| [7:0] | PULSE_GEN_1_ON_TIME[7:0] | Read/write | Defines on time of Pulse Generator 1. On time is defined as the number of clock cycles of the chosen clock speed (see Register 0x35). For example, | | |
| | | | | PULSE_GEN_1_ON_CLK | |
| | | | PULSE_GEN_1_ON_TIME | 0 | 1 |
| | | | 0000 0000 | 0 ms | 0 ms |
| | | | 0000 0001 | 1 ms | 125 ms |
| | | | 0000 0010 | 2 ms | 250 ms |
| | | | 0000 0011 | 3 ms | 375 ms |
| | | | 0000 0100 | 4 ms | 500 ms |
| | | | … | … | … |
| | | | 1111 1110 | 254 ms | 31.750 sec |
| | | | 1111 1111 | 255 ms | 31.875 sec |

### *PULSE_GEN_2_DELAY, Register 0x32*

**Table 59. PULSE_GEN_2_DELAY Bit Descriptions**

| Bits | Bit Name | Access | Description | | |
|---|---|---|---|---|---|
| [7:0] | PULSE_GEN_2_DELAY[7:0] | Read/write | Defines initial delay from the first clock of the first enable of Pulse Generator 2. Delay is defined as the number of clock cycles of the chosen period clock speed (see Register 0x35). For example, | | |
| | | | | PULSE_GEN_2_PRD_CLK | |
| | | | PULSE_GEN_2_DELAY | 0 | 1 |
| | | | 0000 0000 | 0 ms | 0 ms |
| | | | 0000 0001 | 1 ms | 125 ms |
| | | | 0000 0010 | 2 ms | 250 ms |
| | | | 0000 0011 | 3 ms | 375 ms |
| | | | 0000 0100 | 4 ms | 500 ms |
| | | | … | … | … |
| | | | 1111 1110 | 254 ms | 31.750 sec |
| | | | 1111 1111 | 255 ms | 31.875 sec |

### *PULSE_GEN_2_PERIOD, Register 0x33*

**Table 60. PULSE_GEN_2_PERIOD Bit Descriptions**

| Bits | Bit Name | Access | Description | | |
|---|---|---|---|---|---|
| [7:0] | PULSE_GEN_2_PERIOD[7:0] | Read/write | Defines period of Pulse Generator 2. Period is defined as the number of clock cycles of the chosen period clock speed (see Register 0x35). For example, | | |
| | | | | PULSE_GEN_2_PRD_CLK | |
| | | | PULSE_GEN_2_PERIOD | 0 | 1 |
| | | | 0000 0000 | 0 ms | 0 ms |
| | | | 0000 0001 | 1 ms | 125 ms |
| | | | 0000 0010 | 2 ms | 250 ms |
| | | | 0000 0011 | 3 ms | 375 ms |
| | | | 0000 0100 | 4 ms | 500 ms |
| | | | … | … | … |
| | | | 1111 1110 | 254 ms | 31.750 sec |
| | | | 1111 1111 | 255 ms | 31.875 sec |

### *PULSE_GEN_2_ON_TIME, Register 0x34*

**Table 61. PULSE_GEN_2_ON_TIME Bit Descriptions**

| Bits | Bit Name | Access | Description | | |
|---|---|---|---|---|---|
| [7:0] | PULSE_GEN_2_ON_TIME[7:0] | Read/write | Defines on time of Pulse Generator 2. On time is defined as the number of clock cycles of the chosen clock speed (see Register 0x35). For example, | | |
| | | | | PULSE_GEN_2_ON_CLK | |
| | | | PULSE_GEN_2_ON_TIME | 0 | 1 |
| | | | 0000 0000 | 0 ms | 0 ms |
| | | | 0000 0001 | 1 ms | 125 ms |
| | | | 0000 0010 | 2 ms | 250 ms |
| | | | 0000 0011 | 3 ms | 375 ms |
| | | | 0000 0100 | 4 ms | 500 ms |
| | | | … | … | … |
| | | | 1111 1110 | 254 ms | 31.750 sec |
| | | | 1111 1111 | 255 ms | 31.875 sec |

### *PULSE_GEN_CONFIG, Register 0x35*

**Table 62. PULSE_GEN_CONFIG Bit Descriptions**

| Bits | Bit Name | Access | Description |
|---|---|---|---|
| 7 | PULSE_GEN_1_INV | Read/write | 0 = no inversion on Pulse Generator 1. On time is defined as the length of time a high signal is output.<br>1 = inverted output on Pulse Generator 1. On time is defined as the length of time a high signal is output. |
| 6 | PULSE_GEN_1_ON_CLK | Read/write | Defines clock speed for the on time of Pulse Generator 1.<br>0 = 1 ms.<br>1 = 125 ms.<br>Setting PULSE_GEN_1_ON_CLK = 1 and PULSE_GEN_1_PRD_CLK = 0 is not a supported configuration. |
| 5 | PULSE_GEN_1_PRD_CLK | Read/write | Defines clock speed for the period of Pulse Generator 1.<br>0 = 1 ms.<br>1 = 125 ms.<br>Setting PULSE_GEN_1_ON_CLK = 1 and PULSE_GEN_1_PRD_CLK = 0 is not a supported configuration. |
| 4 | PULSE_GEN_1_EN | Read/write | 0 = Pulse Generator 1 is disabled. The off signal is output constantly.<br>1= Pulse Generator 1 is enabled. |
| 3 | PULSE_GEN_2_INV | Read/write | 0 = no inversion on Pulse Generator 2. On time is defined as the length of time a high signal is output.<br>1 = inverted output on Pulse Generator 2. On time is defined as the length of time a low signal is output. |
| 2 | PULSE_GEN_2_ON_CLK | Read/write | Defines clock speed for the on time of Pulse Generator 2.<br>0 = 1 ms.<br>1 = 125 ms.<br>Setting PULSE_GEN_2_ON_CLK = 1 and PULSE_GEN_2_PRD_CLK = 0 is not a supported configuration. |
| 1 | PULSE_GEN_2_PRD_CLK | Read/write | Defines clock speed for the period of Pulse Generator 2.<br>0 = 1 ms.<br>1 = 125 ms.<br>Setting PULSE_GEN_2_ON_CLK = 1 and PULSE_GEN_2_PRD_CLK = 0 is not a supported configuration. |
| 0 | PULSE_GEN_2_EN | Read/write | 0 = Pulse Generator 2 is disabled. The off signal is output constantly.<br>1 = Pulse Generator 2 is enabled. |

### *LOGIC_CFG, Register 0x36*

**Table 63. LOGIC_CFG Bit Descriptions**

| Bits | Bit Name | Access | Description |
|---|---|---|---|
| 7 | Reserved | Reserved | Reserved. |
| 6 | LY_INV | Read/write | 0 = the LY output is not inverted before passing into the logic block.<br>1 = inverts the LY output from the logic block. |
| 5 | LC_INV | Read/write | 0 = the LC input is not inverted before passing into the logic block.<br>1 = inverts LC input before passing it into the logic block. |
| 4 | LB_INV | Read/write | 0 = the LB input is not inverted before passing into the logic block.<br>1 = inverts LB input before passing it into the logic block. |
| 3 | LA_INV | Read/write | 0 = the LA input is not inverted before passing into the logic block.<br>1 = inverts LA input before passing it into the logic block. |
| [2:0] | LOGIC_SEL[2:0] | Read/write | Configures the digital mux for the logic block. Refer to Figure 19.<br>000 = off/disable.<br>001 = AND.<br>010 = OR.<br>011 = XOR.<br>100 = FF.<br>101 = IN_LA.<br>110 = IN_LB.<br>111 = IN_LC. |

### *LOGIC_FF_CFG, Register 0x37*

**Table 64. LOGIC_FF_CFG Bit Descriptions**

| Bits | Bit Name | Access | Description |
|---|---|---|---|
| [7:2] | Reserved | Reserved | Reserved. |
| 1 | FF_SET | Read/write | 0 = FF not set in the logic block. Refer to Figure 19.<br>1 = sets FF in the logic block. |
| 0 | FF_CLR | Read/write | 0 = FF not cleared in the logic block. Refer to Figure 19.<br>1 = clears FF in the logic block. |

### *LOGIC_INT_EVENT_EN, Register 0x38*

**Table 65. LOGIC_INT_EVENT_EN Bit Descriptions**

| Bits | Bit Name | Access | Description |
|---|---|---|---|
| [7:3] | Reserved | Reserved | Reserved. |
| 2 | LY_DBNC_DIS | Read/write | 0 = output of the logic block is debounced before entering the event/interrupt block.<br>1 = output of the logic block is not debounced before entering the event/interrupt block. Use with caution because glitches may generate interrupts prematurely. |
| 1 | LOGIC_EVENT_EN | Read/write | 0 = LY cannot generate interrupt.<br>1 = allows LY activity to generate events on the FIFO. |
| 0 | LOGIC_INT_LEVEL | Read/write | Configures the logic level of LY that generates an interrupt.<br>0 = LY is active low.<br>1 = LY is active high. |

### *POLL_TIME_CFG, Register 0x39*

**Table 66. POLL_TIME_CFG Bit Descriptions**

| Bits | Bit Name | Access | Description |
|---|---|---|---|
| [7:4] | Reserved | Reserved | Reserved. |
| 3 | PRECHARGE_TIME | Read/write | Defines time to allow precharge.<br>0 = 100 µs.<br>1 = 200 µs. |
| 2 | Reserved | Reserved | Reserved. |
| [1:0] | KEY_POLL_TIME[1:0] | Read/write | Configures time between consecutive scan cycles.<br>00 = 10 ms.<br>01 = 20 ms.<br>10 = 30 ms.<br>11 = 40 ms. |

### *PIN_CONFIG_A, Register 0x3A*

**Table 67. PIN_CONFIG_A Bit Descriptions**

| Bits | Bit Name | Access | Description |
|---|---|---|---|
| [7:6] | Reserved | Reserved | Reserved. |
| 5 | R5_CONFIG | Read/write | 0 = GPIO 6.<br>1 = Row 5. |
| 4 | R4_CONFIG | Read/write | 0 = GPIO 5 (see R4_EXTEND_CFG in Table 69 for alternate configuration, RESET).<br>1 = Row 4 |
| 3 | R3_CONFIG | Read/write | 0 = GPIO 4 (see R3_EXTEND_CFG in Table 69 for alternate configuration, LC).<br>1 = Row 3 |
| 2 | R2_CONFIG | Read/write | 0 = GPIO 3<br>1 = Row 2 |
| 1 | R1_CONFIG | Read/write | 0 = GPIO 2<br>1 = Row 1 |
| 0 | R0_CONFIG | Read/write | 0 = GPIO 1/LY (see R0_EXTEND_CFG in Table 69 for alternate configuration, LY).<br>1 = Row 0 |

### *PIN_CONFIG_B, Register 0x3B*

**Table 68. PIN_CONFIG_B Bit Descriptions**

| Bits | Bit Name | Access | Description |
|---|---|---|---|
| [7:5] | Reserved | Reserved | Reserved. |
| 4 | C4_CONFIG | Read/write | 0 = GPIO 11.<br>1 = Column 4. |
| 3 | C3_CONFIG | Read/write | 0 = GPIO 10.<br>1 = Column 3. |
| 2 | C2_CONFIG | Read/write | 0 = GPIO 9.<br>1 = Column 2. |
| 1 | C1_CONFIG | Read/write | 0 = GPIO 8 (see C1_EXTEND_CFG in Table 69 for alternate configuration, PULSE_GEN_2).<br>1 = Column 1. |
| 0 | C0_CONFIG | Read/write | 0 = GPIO 7 (see C0_EXTEND_CFG in Table 69 for alternate configuration, PULSE_GEN_1).<br>1 = Column 0. |

### *PIN_CONFIG_C, Register 0x3C*

**Table 69. PIN_CONFIG_C Bit Descriptions**

| Bits | Bit Name | Access | Description |
|---|---|---|---|
| 7 | PULL_SELECT | Read/write | 0 = 300 kΩ resistor used for row pull-up during key scanning.<br>1 = 100 kΩ resistor used for row pull-up during key scanning. |
| 6 | C0_EXTEND_CFG | Read/write | 0 = C0 remains configured as GPIO 7.<br>1 = C0 reconfigured as PULSE_GEN_1 output. |
| 5 | R4_EXTEND_CFG | Read/write | 0 = R4 remains configured as GPIO 5.<br>1 = R4 reconfigured as RESET_OUT output. |
| 4 | C1_EXTEND_CFG | Read/write | 0 = C1 remains configured as GPIO 8.<br>1 = C1 reconfigured as PULSE_GEN_2 output. |
| 3 | R3_EXTEND_CFG | Read/write | 0 = R3 remains configured as GPIO 4.<br>1 = R3 reconfigured as LC input for the logic block. |
| [2:1] | Reserved | Reserved | Reserved. |
| 0 | R0_EXTEND_CFG | Read/write | 0 = R0 remains configured as GPIO 1.<br>1 = R0 reconfigured as LY output from the logic block. |

### *GENERAL_CFG, Register 0x3D*

**Table 70. GENERAL_CFG Bit Descriptions**

| Bits | Bit Name | Access | Description |
|---|---|---|---|
| 7 | OSC_EN | Read/write | 0 = disables internal 800 kHz oscillator.<br>1 = enables internal 800 kHz oscillator. |
| [6:5] | OSC_FREQ[1:0] | Read/write | Sets the input clock frequency fed from the base 800 kHz oscillator to the digital core. Slower frequencies result in less quiescent current, but key and GPI scan times increase.<br>00 = 50 kHz.<br>01 = 100 kHz.<br>10 = 200 kHz.<br>11 = 400 kHz. |
| [4:3] | Reserved | Reserved | Reserved. |
| 2 | SW_RESET | Read/write | Software reset. Set to 1 to reset the ADP5586. This function is similar to bringing $\overline{RST}$ low, then high. Wait at least 200 µs before reprogramming the device. |
| 1 | $\overline{INT}$_CFG | Read/write | Configures the behavior of the $\overline{INT}$ pin if the user tries to clear it while an interrupt is pending.<br>0 = $\overline{INT}$ pin remains asserted if an interrupt is pending.<br>1 = $\overline{INT}$ pin deasserts for 50 µs and reasserts if an interrupt is pending. |
| 0 | $\overline{RST}$_CFG | Read/write | Configures the response of the ADP5586 to the $\overline{RST}$ pin and the SW_RESET bit.<br>0 = the ADP5586 resets if $\overline{RST}$ is low.<br>1 = the ADP5586 does not reset if $\overline{RST}$ is low. |

### *INT_EN, Register 0x3E*

**Table 71. INT_EN Bit Descriptions**

| Bits | Bit Name | Access | Description |
|---|---|---|---|
| [7:5] | Reserved | Reserved | Reserved. |
| 4 | LOGIC_IEN | Read/write | 0 = Logic 1 interrupt is disabled.<br>1 = asserts the $\overline{\text{INT}}$ pin if the LOGIC_INT bit is set (Register 0x01, Bit 4). |
| 3 | Reserved | Reserved | Reserved. |
| 2 | OVRFLOW_IEN | Read/write | 0 = overflow interrupt is disabled.<br>1 = asserts the $\overline{\text{INT}}$ pin if the OVRFLOW_INT bit is set (Register 0x01, Bit 2). |
| 1 | GPI_IEN | Read/write | 0 = GPI interrupt is disabled.<br>1 = asserts the $\overline{\text{INT}}$ pin if the GPI_INT is set (Register 0x01, Bit 1). |
| 0 | EVENT_IEN | Read/write | 0 = event interrupt is disabled.<br>1 = asserts the $\overline{\text{INT}}$ pin if the EVENT_INT is set (Register 0x01, Bit 0). |

# APPLICATIONS SCHEMATIC

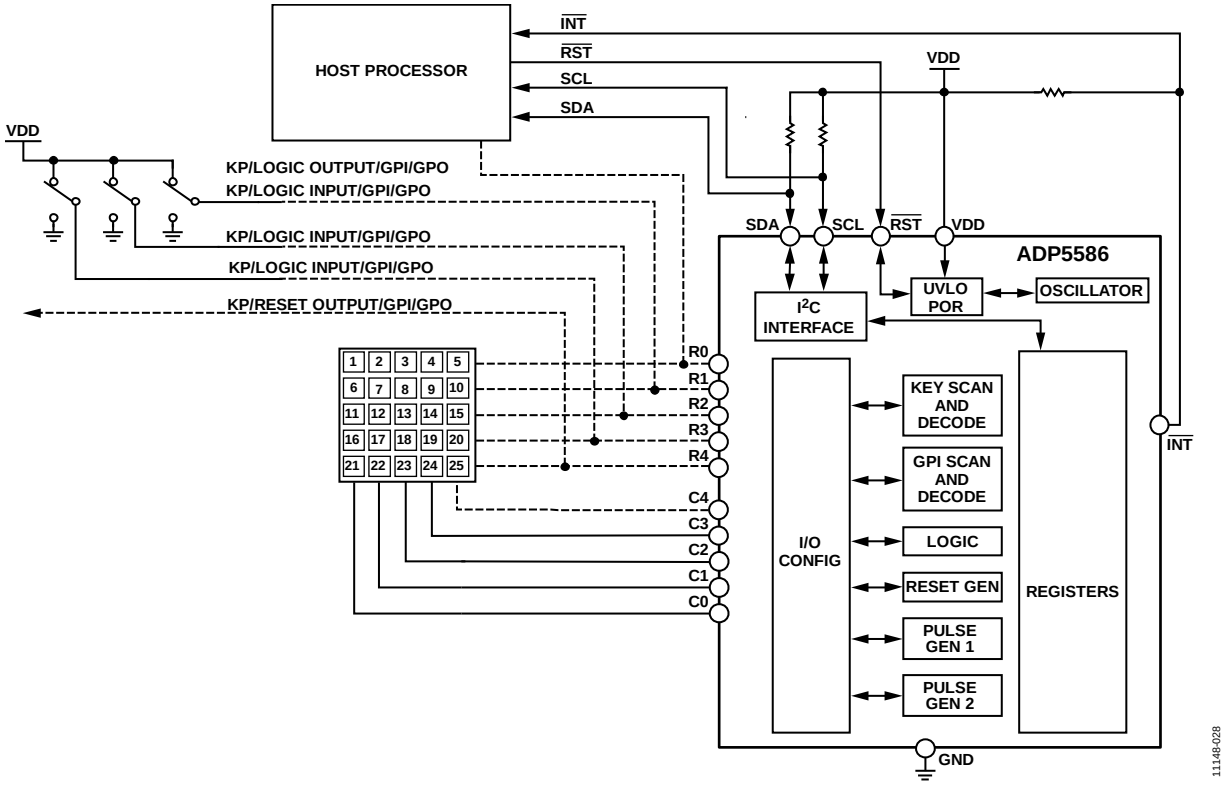



*Figure 28. Typical Applications Schematic*

# OUTLINE DIMENSIONS

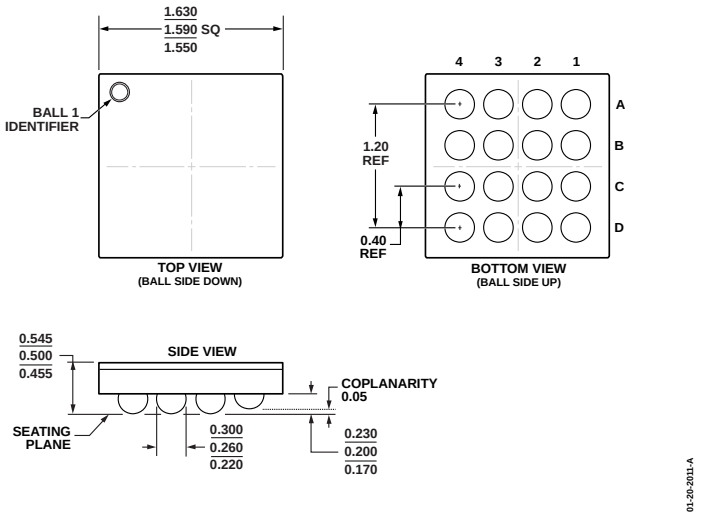



*Figure 29. 16-Ball Wafer Level Chip Scale Package [WLCSP]*
*(CB-16-10)*
*Dimensions shown in millimeters*

## ORDERING GUIDE

| Model[1] | Temperature Range | Package Description | Package Option |
|---|---|---|---|
| ADP5586ACBZ-00-R7 | −40°C to +85°C | 16-Ball Wafer Level Chip Scale Package [WLCSP] | CB-16-10 |
| ADP5586ACBZ-01-R7 | −40°C to +85°C | 16-Ball Wafer Level Chip Scale Package [WLCSP] | CB-16-10 |
| ADP5586ACBZ-03-R7 | −40°C to +85°C | 16-Ball Wafer Level Chip Scale Package [WLCSP] | CB-16-10 |
| ADP5586CB-EVALZ | | WLCSP Evaluation Board | CB-16-10 |

[1] Z = RoHS Compliant Part.

## NOTES

# NOTES

I²C refers to a communications protocol originally developed by Philips Semiconductors (now NXP Semiconductors).

©2013 Analog Devices, Inc. All rights reserved. Trademarks and registered trademarks are the property of their respective owners.
D11148-0-3/13(0)

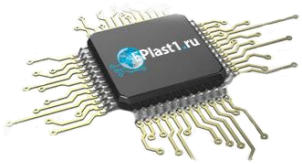
Компания «ЭлектроПласт» предлагает заключение долгосрочных отношений при поставках импортных электронных компонентов на взаимовыгодных условиях!

Наши преимущества:

- Оперативные поставки широкого спектра электронных компонентов отечественного и импортного производства напрямую от производителей и с крупнейших мировых складов;
- Поставка более 17-ти миллионов наименований электронных компонентов;
- Поставка сложных, дефицитных, либо снятых с производства позиций;
- Оперативные сроки поставки под заказ (от 5 рабочих дней);
- Экспресс доставка в любую точку России;
- Техническая поддержка проекта, помощь в подборе аналогов, поставка прототипов;
- Система менеджмента качества сертифицирована по Международному стандарту ISO 9001;
- Лицензия ФСБ на осуществление работ с использованием сведений, составляющих государственную тайну;
- Поставка специализированных компонентов (Xilinx, Altera, Analog Devices, Intersil, Interpoint, Microsemi, Aeroflex, Peregrine, Syfer, Eurofarad, Texas Instrument, Miteq, Cobham, E2V, MA-COM, Hittite, Mini-Circuits,General Dynamics и др.);

Помимо этого, одним из направлений компании «ЭлектроПласт» является направление «Источники питания». Мы предлагаем Вам помощь Конструкторского отдела:

- Подбор оптимального решения, техническое обоснование при выборе компонента;
- Подбор аналогов;
- Консультации по применению компонента;
- Поставка образцов и прототипов;
- Техническая поддержка проекта;
- Защита от снятия компонента с производства.




**Как с нами связаться**

**Телефон:** 8 (812) 309 58 32 (многоканальный)
**Факс:** 8 (812) 320-02-42
**Электронная почта:** org@eplast1.ru
**Адрес:** 198099, г. Санкт-Петербург, ул. Калинина, дом 2, корпус 4, литера А.