Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!

Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832
Email & Skype: info@chipsmall.com Web: www.chipsmall.com
Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China

Keypad Decoder and I/O Expansion

Data Sheet **ADP5589**

FEATURES

- 16-element FIFO for event recording
- 19 configurable I/Os allowing functions such as
  - Keypad decoding for matrix up to 11 × 8
  - Key press/release interrupts
  - Key pad lock/unlock
  - 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
  - Dual programmable logic blocks
  - PWM generator
    - Internal PWM generation
    - External PWM with internal PWM AND function
  - Clock divider
  - Reset generators
- I²C interface with fast-mode plus (Fm+) support up to 1 MHz
- Open-drain interrupt output
- 24-lead LFCSP 3.5 mm × 3.5 mm
- 25-ball WLCSP 1.99 mm × 1.99 mm

APPLICATIONS

- Devices requiring keypad entry and I/O expansion capabilities

FUNCTIONAL BLOCK DIAGRAM

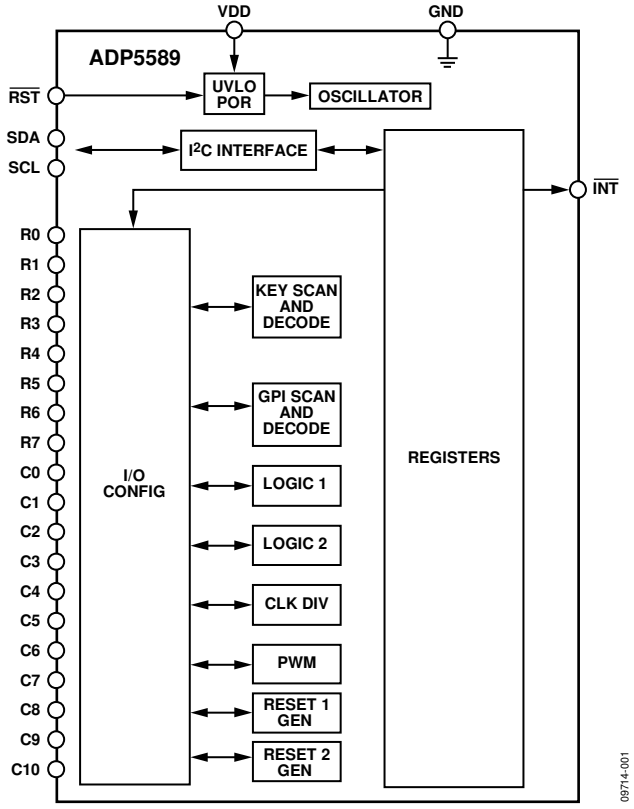


Figure 1.

GENERAL DESCRIPTION

The ADP5589 is a 19 I/O port expander with built-in keypad matrix decoder, programmable logic, reset generator, and PWM generator. I/O 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 ADP5589, which handles all key scanning and decoding, can flag the main processor via an interrupt line when new key events have occurred. In addition, GPI changes and logic changes can be tracked as events via the FIFO, eliminating the need to monitor different registers for event changes. The ADP5589 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 ADP5589 frees up the main processor from having to monitor the keypad, thereby 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, saving board area and cost.

Rev. B

One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, U.S.A.
Tel: 781.329.4700
Technical Support www.analog.com

ADP5589* PRODUCT PAGE QUICK LINKS

Last Content Update: 02/23/2017

COMPARABLE PARTS

View a parametric search of comparable parts.

EVALUATION KITS

- ADP5589 Evaluation Board

DOCUMENTATION

Data Sheet

- ADP5589: Keypad Decoder and I/O Expansion Data Sheet

SOFTWARE AND SYSTEMS REQUIREMENTS

- ADP5589 - No-OS Driver for Microchip Microcontroller Platforms
- ADP5589 - No-OS Driver for Renesas Microcontroller Platforms
- ADP5589 Input Keyboard and GPIO Linux Driver
- ADP5589 Pmod Xilinx FPGA Reference Design

TOOLS AND SIMULATIONS

- ADIsimPower™ Voltage Regulator Design Tool

DESIGN RESOURCES

- ADP5589 Material Declaration
- PCN-PDN Information
- Quality And Reliability
- Symbols and Footprints

DISCUSSIONS

View all ADP5589 EngineerZone Discussions.

SAMPLE AND BUY

Visit the product page to see pricing options.

TECHNICAL SUPPORT

Submit a technical question or find your regional support number.

DOCUMENT FEEDBACK

Submit feedback for this data sheet.

TABLE OF CONTENTS

REVISION HISTORY

1/13—Rev. A to Rev. B

Changes to Detailed Register Descriptions Section and Table 7 ... 22
Changes to Table 33 and Table 34 ... 29
Changes to Table 36 ... 30
Changes to Table 37 ... 31
Changes to Table 69 ... 41
Changes to Table 84 ... 46
Changes to Figure 31 ... 48

8/11—Revision A: Initial Version

SPECIFICATIONS

VDD = 1.8 V to 3.3 V, T_A = −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)	$I_{SCAN = 10 ms}$	CORE_FREQ = 50 kHz, scan active, 300 kΩ pull-up, VDD = 1.65 V		30	40	µA
	$I_{SCAN = 10 ms}$	CORE_FREQ = 50 kHz, scan active, 100 kΩ pull-up, VDD = 1.65 V		35	45	µA
	$I_{SCAN = 10 ms}$	CORE_FREQ = 50 kHz, scan active, 300 kΩ pull-up, VDD = 3.3 V		75	85	µA
	$I_{SCAN = 10 ms}$	CORE_FREQ = 50 kHz, scan active, 100 kΩ pull-up, VDD = 3.3 V		80	90	µA
PULL-UP, PULL-DOWN RESISTANCE						
Pull-Up Option 1			50	100	150	kΩ
Pull-Up 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, R6, R7, C0, C1, C2, C3, C4, C5, C6, C7, C8, C9, C10)						
Logic Low Input Voltage	V_{IL}				0.3 × VDD	V
Logic High Input Voltage	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, R6, R7, C0, C1, C2, C3, C4, C5, C6, C7, C8, C9, C10)						
Logic Low Output Voltage[2]	V_{OL}	Sink current = 10 mA			0.4	V
Logic Low Output Voltage[3]	V_{OL}	Sink current = 10 mA			0.5	V
Logic High Output Voltage	V_{OH}	Source current = 5 mA	0.7 × VDD			V
Logic High Leakage Current (Per Pin)	$V_{OH-Leak}$			0.1	1	µA
OPEN-DRAIN OUTPUT LOGIC LEVEL ($\overline{INT}$, SDA)						
Logic Low Output Voltage ($\overline{INT}$)	V_{OL}	I_{SINK} = 10 mA			0.4	V
Logic Low Output Voltage (SDA)	V_{OL}	I_{SINK} = 20 mA			0.4	V
Logic High Leakage Current (Per Pin)	$V_{OH-Leak}$			0.1	1	µA
Logic Propagation Delay				125	300	ns
FF1 Hold Time[4]				0		ns
FF1 Setup Time[4]				175		ns
FF2 Hold Time[4]				0		ns
FF2 Setup Time[4]				175		ns
GPIO Debounce[4]					70	µs
Internal Oscillator Frequency[5]	OSC_{FREQ}		900	1000	1100	kHz
I²C TIMING SPECIFICATIONS						
Delay from UVLO/Reset Inactive to I²C Access					60	µs
SCL Clock Frequency	f_{SCL}		0		1000	kHz
SCL High Time	t_{HIGH}		0.26			µs
SCL Low Time	t_{LOW}		0.5			µs
Data Setup Time	$t_{SU; DAT}$		50			ns
Data Hold Time	$t_{HD; DAT}$		0			µs
Setup Time for Repeated Start	$t_{SU; STA}$		0.26			µs

Parameter	Symbol	Test Conditions/Comments	Min	Typ	Max	Unit
Hold Time for Start/Repeated Start	$t_{HD;\ STA}$		0.26			µs
Bus Free Time for Stop and Start Condition	t_{BUF}		0.5			µs
Setup Time for Stop Condition	$t_{SU;\ STO}$		0.26			µs
Data Valid Time	$t_{VD;\ DAT}$				0.45	µs
Data Valid Acknowledge	$t_{VD;\ ACK}$				0.45	µs
Rise Time for SCL and SDA	t_R				120	ns
Fall Time for SCL and SDA	t_F				120	ns
Pulse Width of Suppressed Spike	t_{SP}		0		50	ns
Capacitive Load for Each Bus Line	C_B[6]				550	pF

[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] Maximum of five GPIOs active simultaneously.
[3] All GPIOs active simultaneously.
[4] Guaranteed by design.
[5] All timers are referenced from the base oscillator and have the same ±10% accuracy.
[6] C_B is the total capacitance of one bus line in picofarads.

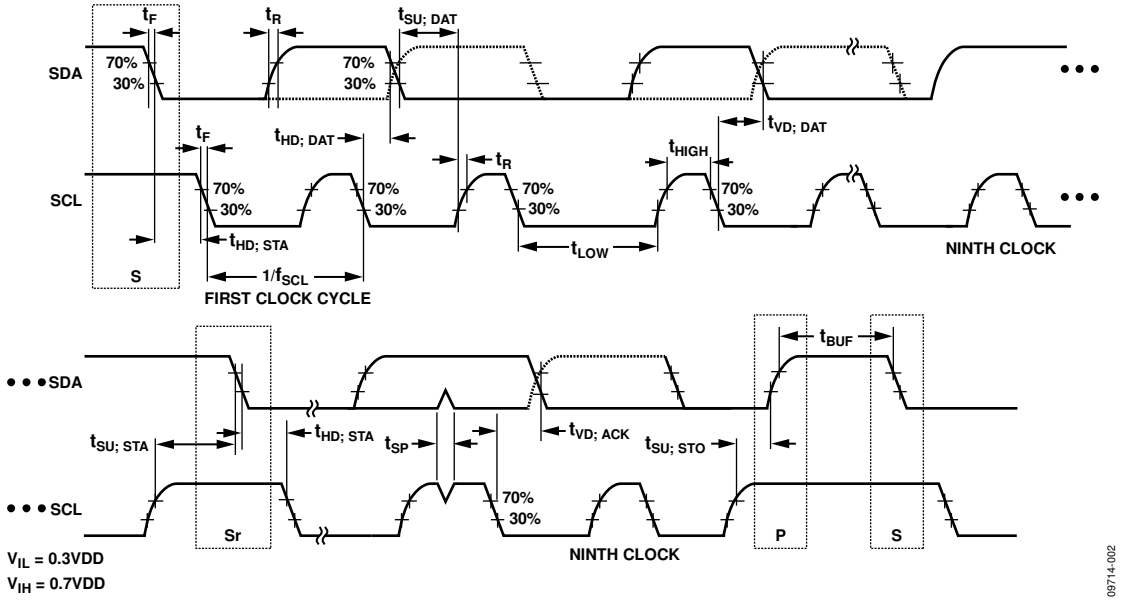


Figure 2. I²C Interface Timing Diagram

ABSOLUTE MAXIMUM RATINGS

Table 2.

Parameter	Rating
VDD to Ground	−0.3 V to 4 V
SCL, SDA, $\overline{RST}$, $\overline{INT}$, R0, R1, R2, R3, R4, R5, R6, R7, C0, C1, C2, C3, C4, C5, C6, C7, C8, C9, C10 to Ground	−0.3 V to (VDD + 0.3 V)
Operating Ambient Temperature Range	−40°C to +85°C[1]
Operating Junction Temperature Range	−40°C to +125°C
Storage Temperature Range	−65°C to +150°C
Soldering Conditions	JEDEC J-STD-020

[1] In applications where high power dissipation and poor thermal resistance are present, the maximum ambient temperature may have 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 part/package in the application (θ_{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 ground.

THERMAL RESISTANCE

θ_{JA} is specified for the worst-case conditions, that is, a device soldered in a circuit board for surface-mount packages.

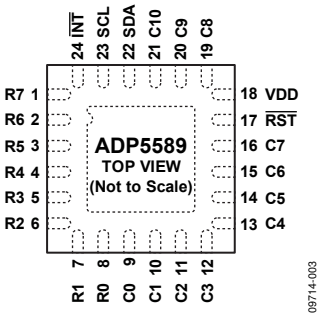
Table 3.

Thermal Resistance	θ_{JA}	Unit
24-Lead LFCSP	43.83	C/W
Maximum Power Dissipation	120	mW
25-Ball WLCSP	43	C/W
Maximum Power Dissipation	120	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



NOTES
1. THE EXPOSED PAD MUST BE CONNECTED TO GROUND.

Figure 3. LFCSP Pin Configuration

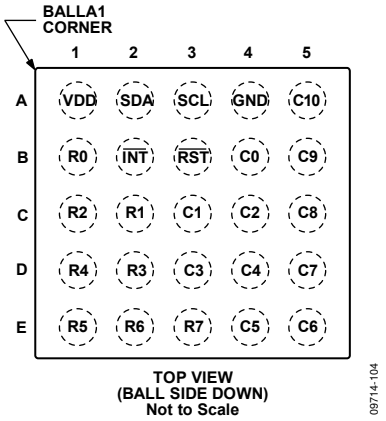


Figure 4. WLCSP Pin Configuration

Table 4. Pin Function Descriptions

Pin No. (LFCSP)	Pin No. (WLCSP)	Mnemonic	Description
1	E3	R7	GPIO 8. This pin functions as Row 7 if used as keypad.
2	E2	R6	GPIO 7. This pin functions as Row 6 if used as keypad.
3	E1	R5	GPIO 6. This pin functions as Row 5 if used as keypad.
4	D1	R4	GPIO 5 (GPIO alternate function: RESET1). This pin functions as Row 4 if used as keypad.
5	D2	R3	GPIO 4 (GPIO alternate function: LC1, PWM_OUT, or CLK_OUT. This pin functions as Row 3 if used as keypad.
6	C1	R2	GPIO 3 (GPIO alternate function: LB1). This pin functions as Row 2 if used as a keypad.
7	C2	R1	GPIO 2 (GPIO alternate function: LA1). This pin functions as Row 1 if used as a keypad.
8	B1	R0	GPIO 1 (GPIO alternate function: LY1). This pin functions as Row 0 if used as a keypad.
9	B4	C0	GPIO 9. This pin functions as Column 0 if used as keypad.
10	C3	C1	GPIO 10. This pin functions as Column 1 if used as keypad.
11	C4	C2	GPIO 11. This pin functions as Column 2 if used as keypad.
12	D3	C3	GPIO 12. This pin functions as Column 3 if used as keypad.
13	D4	C4	GPIO 13 (GPIO alternate function: RESET2). This pin functions as Column 4 if used as keypad.
14	E4	C5	GPIO 14. This pin functions as Column 5 if used as keypad.
15	E5	C6	GPIO 15 (GPIO alternate function: LC2, PWM_IN, or CLK_IN). This pin functions as Column 6 if used as keypad.
16	D5	C7	GPIO 16 (GPIO alternate function: LB2). This pin functions as Column 7 if used as keypad.
17	B3	$\overline{\text{RST}}$	Input Reset Signal.
18	A1	VDD	Supply Voltage Input.
19	C5	C8	GPIO 17 (GPIO alternate function: LA2). This pin functions as Column 8 if used as keypad.
20	B5	C9	GPIO 18 (GPIO alternate function: LY2). This pin functions as Column 9 if used as keypad.
21	A5	C10	GPIO 19. This pin functions as Column 10 if used as keypad.
22	A2	SDA	I^2C Data Input/Output.
23	A3	SCL	I^2C Clock Input.
24	B2	$\overline{\text{INT}}$	Open-Drain Interrupt Output.
EP (pad)	A4	GND	Ground. The exposed pad of the LFCSP package must be connected to ground.

QUICK DEVICE OVERVIEW

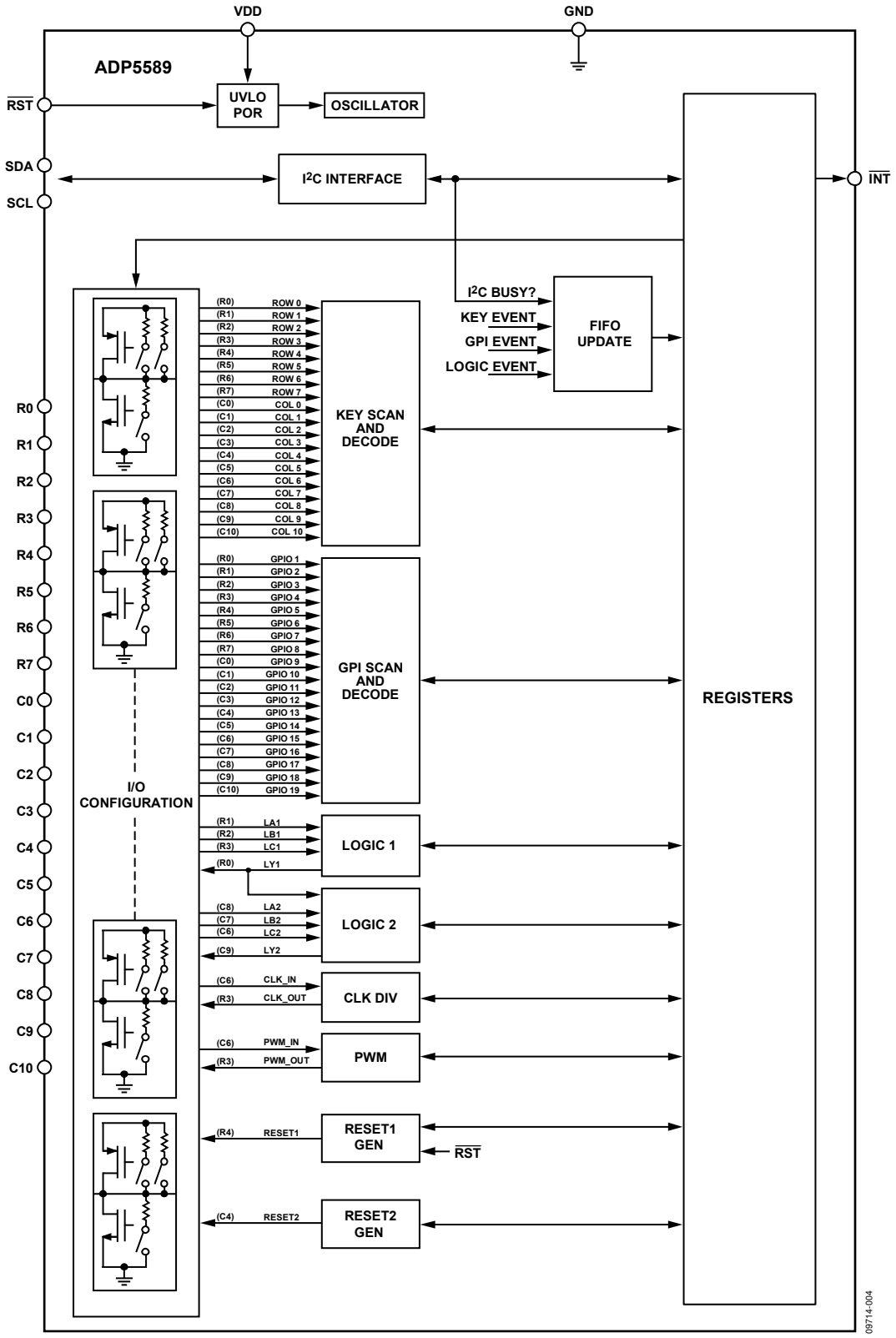


Figure 5. Internal Block Diagram

DEVICE ENABLE

When sufficient voltage is applied to VDD and the $\overline{\text{RST}}$ pin is driven with a logic high level, the ADP5589 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 ADP5589 enters a reset state and all settings return to default. The $\overline{\text{RST}}$ pin features a debounce filter.

DEVICE OVERVIEW

The ADP5589 contains 19 multiconfigurable input/output pins. Each pin can be programmed to enable the device to carry out its various functions, as follows:

- Keypad matrix decoding (11-column by 8-row matrix maximum).
- General-purpose I/O expansion (up to 19 inputs/outputs).
- PWM generation.
- Clock division of externally supplied source.
- Dual logic function building blocks (up to three inputs, one output).
- Two reset generators.

All 19 input/output pins have an I/O structure, as shown in Figure 6.

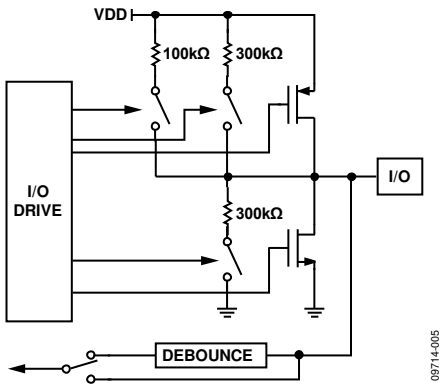


Figure 6. I/O Structure

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 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 6 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 keypad row pins and the 10 mA NMOS sinks for grounding 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 via the $\overline{\text{INT}}$ pin.

The ADP5589 is offered with three feature sets. Table 5 lists the options that are available for each model of the ADP5589.

Table 5. Available Options

Models	Description
ADP5589ACPZ-00-R7 ADP5589ACBZ-00-R7	All GPIOs pulled up (default option)
ADP5589ACPZ-01-R7 ADP5589ACBZ-01-R7	Reset pass-through[1]
ADP5589ACPZ-02-R7 ADP5589ACBZ-02-R7	Pull-down on special function pins[2]

[1] Reset pass-through implies that the RESET1 output (R4) follows the logic level of the reset input pin, $\overline{\text{RST}}$, after the oscillator has been enabled.
[2] Special function pins are defined as R0 (Row 0), R3 (Row 3), R4 (Row 4), C4 (Column 4), C6 (Column 6), and C9 (Column 9).

DETAILED DESCRIPTION

EVENT FIFO

It is important to understand the function of the event FIFO. The ADP5589 features an event FIFO that 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], is composed of five bits and works in tandem with the FIFO so that the user knows how much of the FIFO must be read back at any given time.

The FIFO is composed of 16 eight-bit sections that the user accesses by reading the FIFO_x registers. The actual FIFO is not in user accessible registers until a read occurs. The FIFO can be thought of as a "first in, first out" buffer used to fill Register 0x03 to Register 0x12.

The event FIFO is made up of 16 eight-bit registers. In each register, Bits[6:0] hold the event identifier, and Bit 7 holds the event state. With seven bits, 127 different events can be identified. See Table 11 for event decoding.

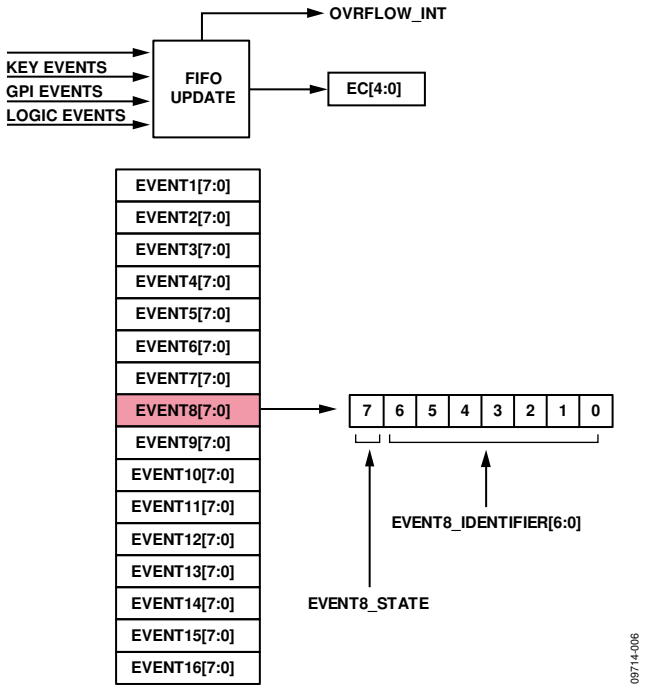


Figure 7. Breakdown of Eventx[7:0] Bits

When events are available on the FIFO, the user should first read back the event count, EC[4:0], to determine how many events must be read back. Events can be read from the top of the FIFO only. When an event is read back, all remaining events in the FIFO are shifted up one location, and the EC[4:0] count is decremented.

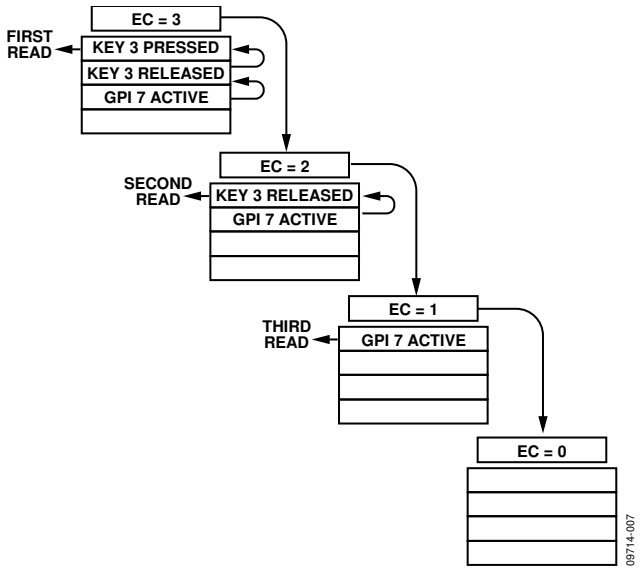


Figure 8. FIFO Operation

The FIFO registers (0x03 to 0x12) 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 only be read back in the order in which they occurred, thus ensuring the integrity of the FIFO system.

Some of the onboard functions of ADP5589 can be programmed to generate events on the FIFO. A FIFO update control block manages updates to the FIFO. If an I^2C transaction is accessing any of the FIFO address locations, updates are paused until the I^2C transaction has completed.

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 status bit is set. An interrupt is generated if overflow interrupt is enabled, signaling to the processor that more than 16 events have occurred.

KEY SCAN CONTROL

General

The 19 input/output pins can be configured to decode a keypad matrix up to a maximum size of 88 switches (11 × 8 matrix). Smaller matrices can also be configured, freeing up the unused row and column pins for other I/O functions.

The R0 through R7 I/O pins comprise the rows of the keypad matrix. The C0 through C10 I/O pins comprise the columns of the keypad matrix. Pins used as rows are pulled up via the internal 300 kΩ (or 100 kΩ) resistors. Pins used as columns are driven low via the internal NMOS current sink.

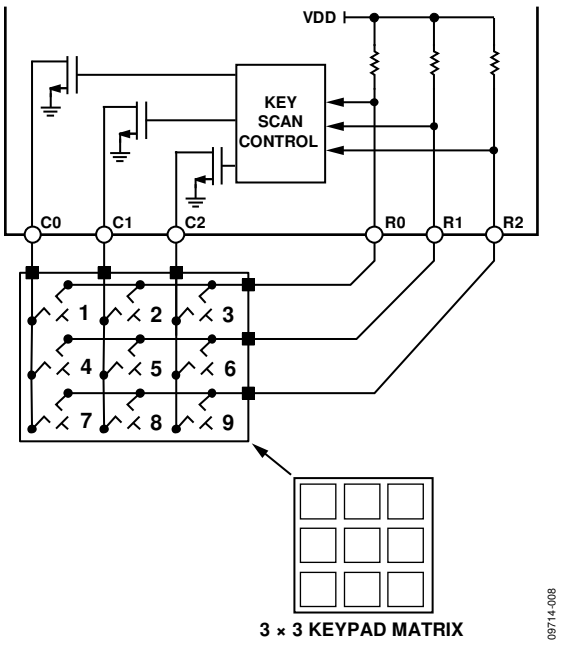


Figure 9. Simplified Key Scan Block

Figure 9 shows a simplified representation of the key scan block using three row 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 is pulled low, and a key scan cycle begins. Key scanning involves driving all column pins high, then driving each column pin, one at a time, low and sensing whether a row pin is low or not. All row/column pairs are scanned; therefore, if multiple keys are pressed, they are detected.

To prevent glitches or narrow press times being registered as a valid key press, the key scanner requires 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, wait, scan, wait, and so forth.

If Switch 6 is released, the connection between R1 and C2 breaks, and R1 is pulled up 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, it may take up to two full wait/scan cycles for a key to register as released. When the key is registered as released, and no other keys are pressed, the key scanner returns to idle mode.

For the remainder of this document, 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 represents released. This eliminates the need to draw individual row/column signals when describing key events.

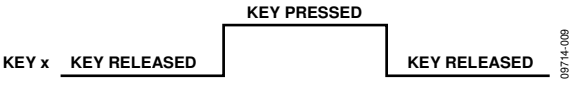


Figure 10. Logic Low: Released; Logic High: Pressed

Figure 11 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 88 unique keys can be scanned.

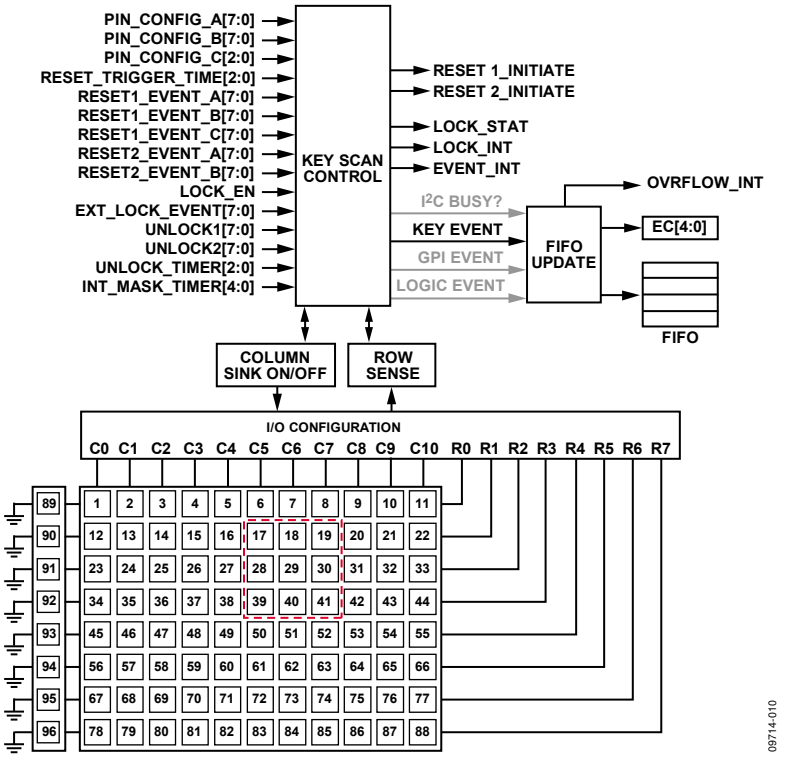


Figure 11. Detailed Key Scan Block

Use Registers PIN_CONFIG_A[7:0] (0x49), PIN_CONFIG_B[7:0] (0x4A), and PIN_CONFIG_C[2:0] (0x4B) to configure 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 88 key identifiers on the FIFO.

If a smaller 3 × 3 matrix is configured, for example, using the C5, C6, and C7 column pins and the R1, R2, and R3 row pins, only the nine event identifiers (17, 18, 19, 28, 29, 30, 39, 40, and 41) can possibly be observed on the FIFO, as shown in Figure 11.

By default, the ADP5589 records key presses and releases on the FIFO. Figure 12 illustrates what happens when a single key is pressed and released. Initially, the key scanner is idle. When Key 32 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 32 has remained pressed, which sets the EVENT_INT interrupt. The event counter, EC[4:0], is incremented to 1, EVENT1[7:0] of the FIFO is updated with its event identifier set to 32, and its Event1_State bit is set to 1, indicating a press.

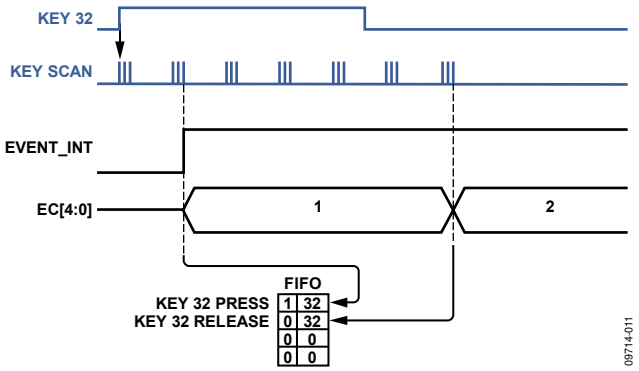


Figure 12. Press and Release Event

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[7:0] of the FIFO is updated with its event identifier set to 32. Its Event2_State bit is set to 0, indicating a release. The key scanner goes back to idle mode because no other keys are pressed.

The EVENT_INT interrupt can be triggered by both press and release key events. As shown in Figure 13, if Key 32 is pressed, EVENT_INT is asserted, EC[4:0] is updated, and the FIFO is updated. During the time that the key is still 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 incremented, and the FIFO updated with the release event information.

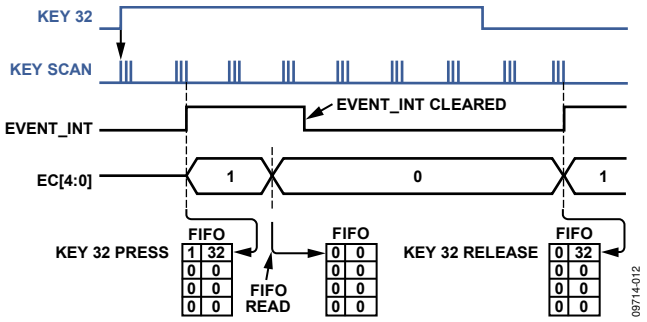


Figure 13. Asserting the EVENT_INT Interrupt

Key Pad Extension

As shown in Figure 11, 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 whole row is grounded. When the key scanner completes scanning, it normally detects Key 1 to Key 11 as being pressed; however, this unique condition is decoded by the ADP5589, and Key Event 89 is assigned to it. Up to eight more key event assignments are possible, allowing the keypad size to extend up to 96. However, if one of the extended keys is pressed, none of the keys on that row is detectable. Activation of a ground key causes all other keys sharing that row to be undetectable.

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 can cause ghosting.

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 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 that are likely to be pressed together in the same row or column. Some examples of keys that are likely to be pressed together are as follows:

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

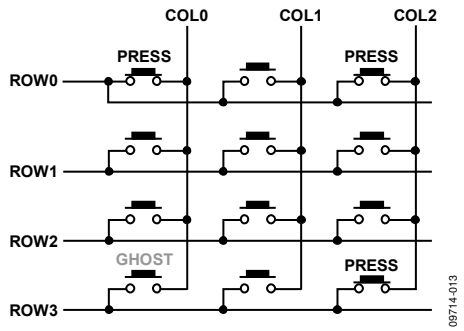


Figure 14. COL0-ROW3 is a Ghost Key Due to Short Between ROW0, COL0, COL2 and ROW3 During Key Press

FIFO Lock/Unlock

The ADP5589 features a lock mode, whereby events are prevented from updating the FIFO or the event counter or from generating EVENT_INT interrupts until an unlock event is detected.

The lock feature is enabled by setting the LOCK_EN (0x37[0]) bit or, alternatively, by a user programmable key or GPI event (set via EXT_LOCK_EVENT[7:0], Address 0x35). If the lock feature is enabled by the LOCK_EN bit, the LOCK_STAT (0x02[5]) bit is set. If the lock feature is enabled by an external event, then the LOCK_STAT bit is set, and a LOCK_INT interrupt is generated.

Unlock events are programmed via the UNLOCK1[7:0] (0x33) and UNLOCK2[7:0] (0x34) registers. Bits[6:0] comprise the even number. Bit 7 determines the active/inactive event (see the UNLOCK1 Register 0x33 (Table 59) and the UNLOCK2 Register 0x34 (Table 60).

If the user chooses to use only one unlock event, only the UNLOCK1[7:0] register should be programmed. Unlock events can be key press events (Event 1 to Event 88). Key release events are ignored when the keypad is locked and should not be used as unlock events.

GPIs configured to generate FIFO updates can also be used as unlock events (Event 97 to Event 115, either active or inactive). If either UNLOCKx register is programmed with Value 127 (Event 127), this means that any allowable event (key or GPI) is the unlock event. For example, if UNLOCK1[6:0] is programmed with 17, and UNLOCK2[6:0] is programmed with 127, the unlock sequence is Key 17 press followed by any other allowable event.

If the first unlock event is detected, partial unlock has occurred. If the next event after the first unlock event is not the second unlock event, then a full lock state is entered again. If the next event after the first unlock event is the second unlock event, then LOCK_STAT is cleared, and a LOCK_INT interrupt is generated. The user can at any stage clear LOCK_EN. This clears the LOCK_STAT bit but does not cause a LOCK_INT interrupt to be generated.

When full unlock is achieved, FIFO and event count updates resume. Note that if a key press is used as the second unlock event, the release of that key is captured on the FIFO after unlocking is completed.

The ADP5589 features an unlock timer, UNLOCK_TIMER[2:0] (0x36[2:0]). When enabled, after the first unlock event occurs, the unlock timer begins counting, and the second unlock event must occur before the unlock timer expires. If the unlock timer expires, the first unlock event must occur again to restart the unlock process. Figure 15 shows a simple state diagram of the unlocking process.

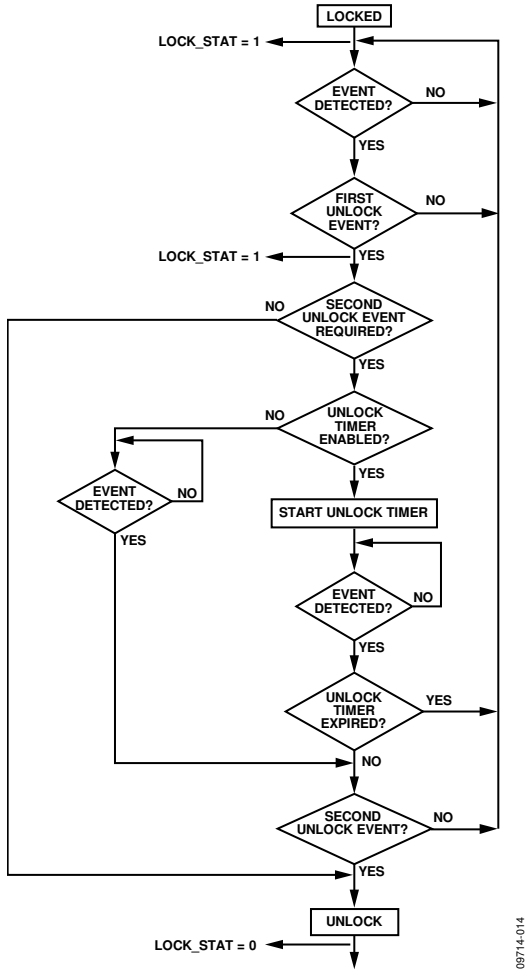


Figure 15. State Diagram of Unlocking Process

When lock mode is enabled, no EVENT_INT interrupts can be generated until the unlock events occur.

The ADP5589 features an interrupt mask timer, INT_MASK_TIMER[4:0] (0x36[7:3]). When this timer and lock mode are enabled, a single EVENT_INT is generated if any key is pressed or any GPI (programmed to update the FIFO) is active. When the EVENT_INT is generated, the mask timer begins counting. No additional EVENT_INT interrupts are generated until the mask timer expires and a new key is pressed or any GPI (programmed to update the FIFO) is active, unless the unlock events occur, in which case, normal operation is resumed.

Allowing a single EVENT_INT interrupt is useful to alert the processor to turn on its screen and display an unlock message to the user. Blanking out additional key presses ensures that the processor is not unnecessarily interrupted until the unlock events occur. Figure 16 shows the unlock sequence when the interrupt mask timer is enabled.

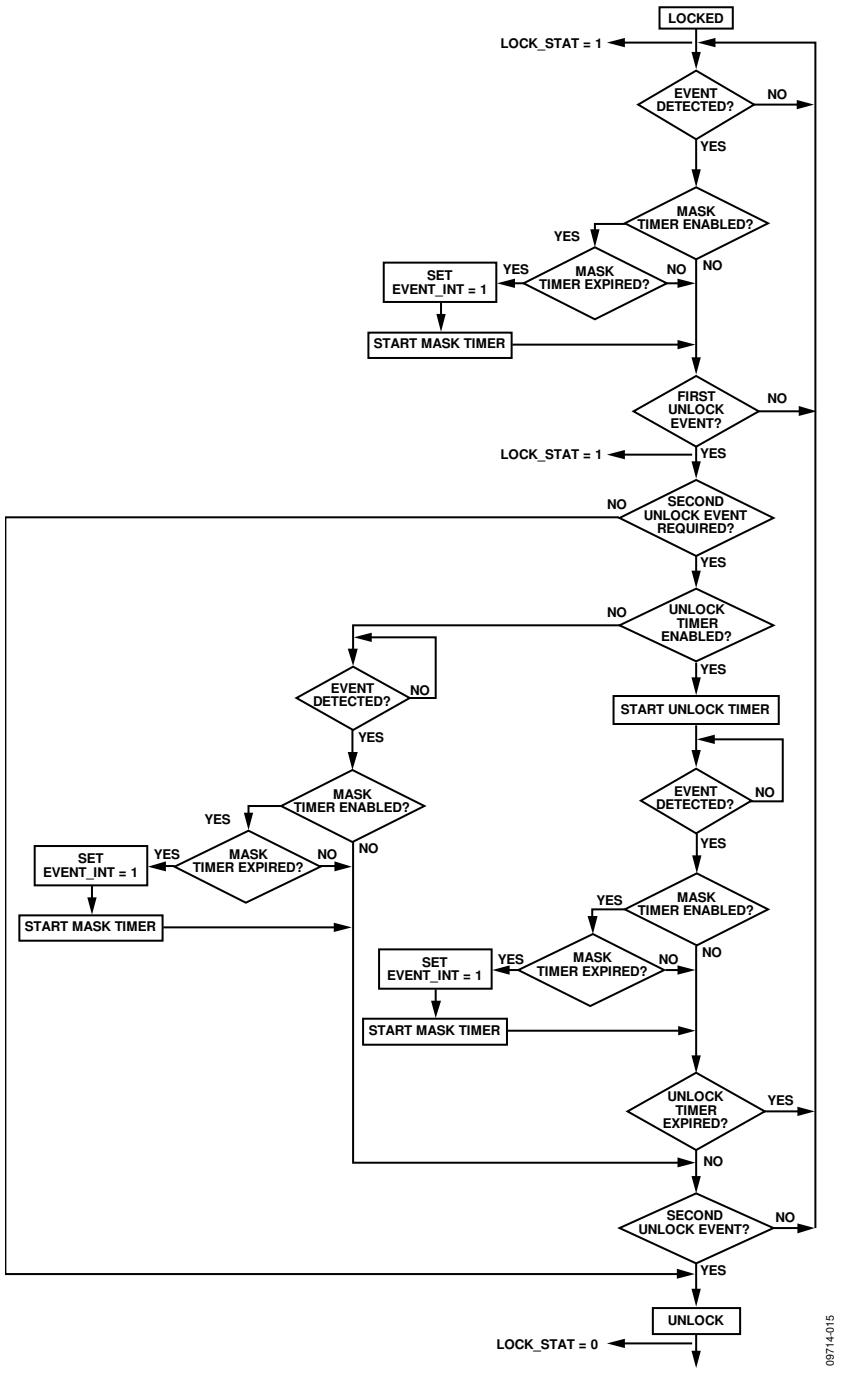


Figure 16. Unlock Sequence

GPI Input

Each of the 19 I/O lines can be configured as a general-purpose logic input line. Figure 17 shows a detailed representation of the GPI scan and detect block and all its associated control and status signals.

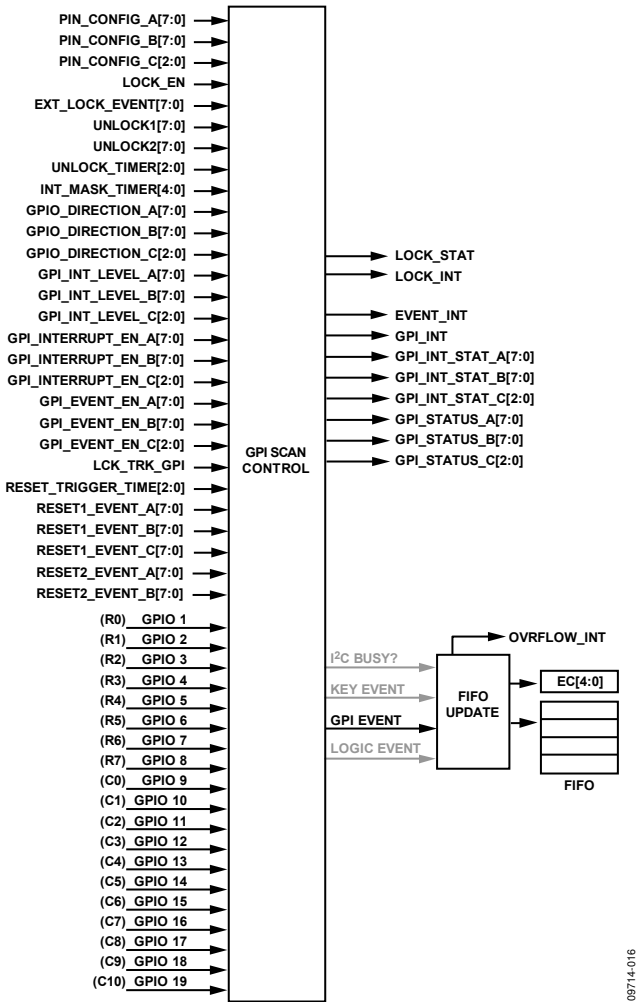


Figure 17. GPI Scan and Detect Block

The current input state of each GPI can be read back using the GPI_STATUS_x registers. Each GPI can be programmed to generate an interrupt via the GPI_INTERRUPT_EN_x registers. The interrupt status is stored in the GPI_INT_STAT_x registers. GPI interrupts can be programmed to trigger on inputs being high or on inputs being low via the GPI_INT_LEVEL_x registers. If any of the GPI interrupts is triggered, the master GPI_INT interrupt is also triggered.

Figure 18 demonstrates a single GPI and how it affects its corresponding status and interrupt status bits.

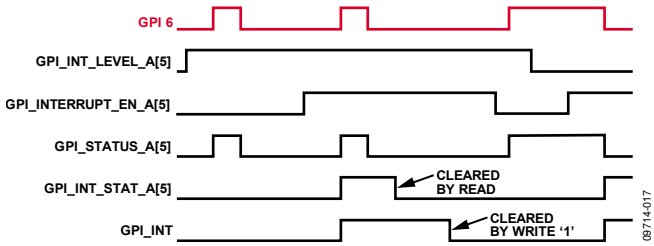


Figure 18. Single GPI Example

GPIs can be programmed to generate FIFO events via the GPI_EVENT_EN_x registers. GPIs in this mode do not generate GPI_INT interrupts and instead generate EVENT_INT interrupts. Figure 19 shows several GPI lines and their effects on the FIFO and event count, EC[4:0].

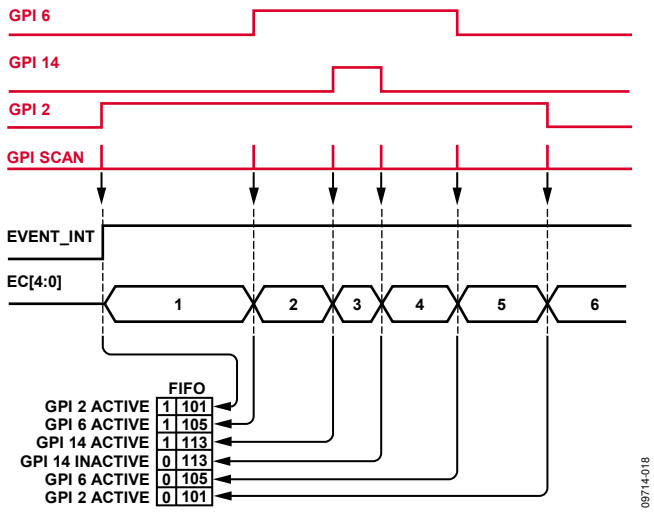


Figure 19. Multiple GPI Lines Example

The GPI scanner is idle until it detects a level transition. It scans the GPI inputs and updates accordingly. It then returns to idle immediately; it does not scan/wait, like the key scanner. As such, the GPI scanner can detect narrow pulses once they get past the 50 µs input debounce filter.

GPIs (programmed for FIFO updating) can be used as keypad unlock events via the UNLOCKx registers (see the FIFO Lock/Unlock section). The LCK_TRK_GPI bit can be used to allow GPIs (programmed for FIFO updating) to be tracked when the keypad is locked.

GPO OUTPUT

Each of the 19 I/O lines can be configured as a general-purpose output (GPO) line. Figure 6 shows a detailed diagram of the I/O structure. See the Detailed Register Descriptions section for GPO configuration and usage.

LOGIC BLOCKS

Several of the ADP5589 I/O lines can be used as inputs and outputs for implementing some common logic functions.

The R1, R2, and R3 I/O pins can be used as inputs, and the R0 I/O pin can be used as an output for Logic Block 1.

The C8, C7, and C6 I/O pins can be used as inputs, and the C9 I/O pin can be used as an output, for Logic Block 2. It is also possible to cascade the output of Logic Block 1 as an alternate input for Logic Block 2 (LY1 is used instead of LA2).

The outputs from the logic blocks can be configured to generate interrupts. They can also be configured to generate events on the FIFO. The LCK_TRK_LOGIC (0x4D[4]) bit can be used to allow logic events (programmed for FIFO updating) to be tracked when the keypad is locked.

Figure 21 and Figure 22 show detailed diagrams of the internal make-up of each logic block, illustrating the possible logic functions that can be implemented.

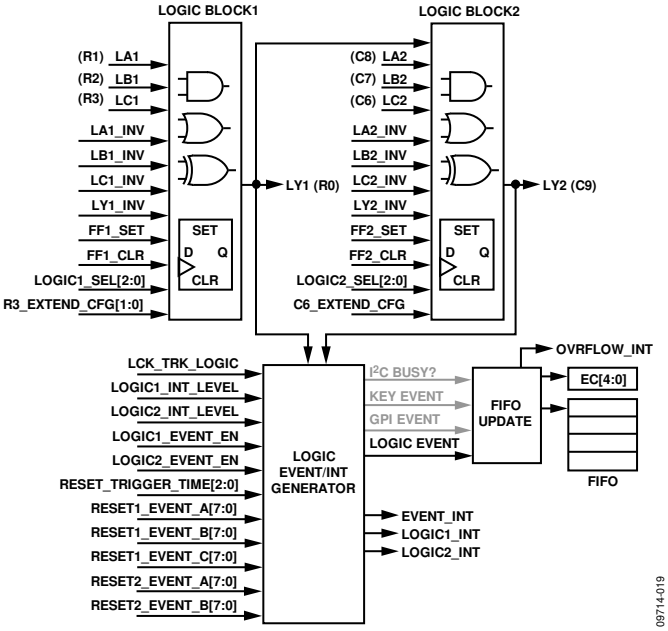


Figure 20. Logic Blocks Overview

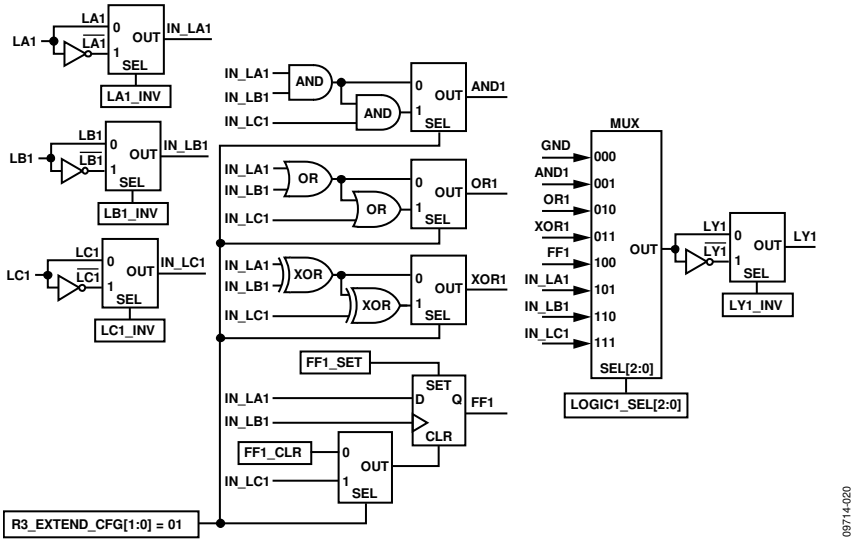


Figure 21. Logic Block 1

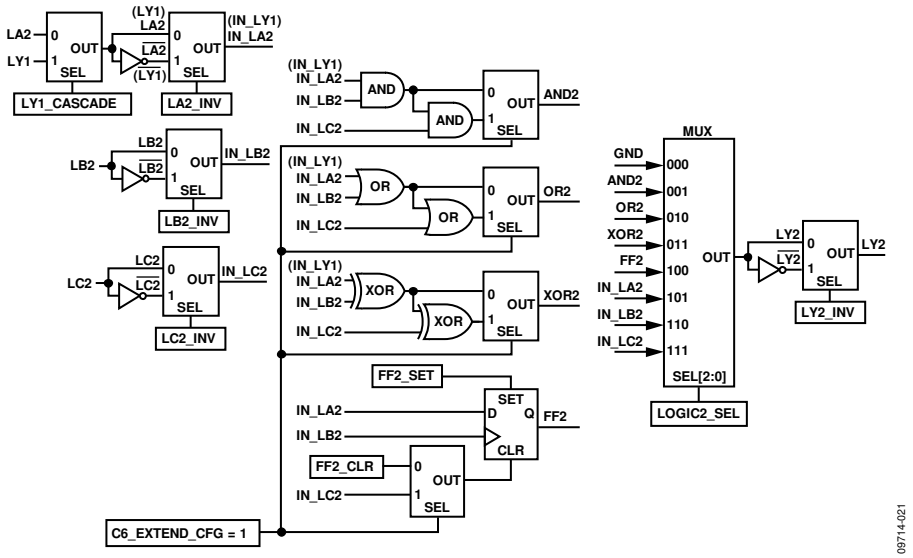


Figure 22. Logic Block 2

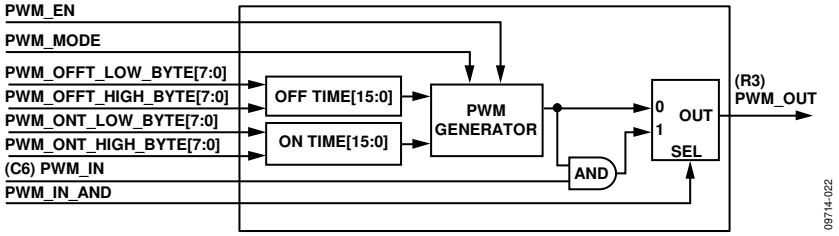


Figure 23. PWM Block Diagram

PWM BLOCK

The ADP5589 features a PWM generator whose output can be configured to drive out on I/O Pin R3. PWM on/off times are programmed via four 8-bit registers.

Newly programmed values are not latched until the final byte, PWM_ONT_HIGH_BYTE (Address 0x41, Bits[7:0]), is written to (see Figure 23).

The highest frequency obtainable from the PWM is performed by setting the least significant bit (LSB) of both the on and off bit patterns, resulting in a 500 kHz signal with a 50% duty cycle. Each LSB respresents 1 µs of on or off time.

The PWM block provides support for continuous PWM mode as well as a one-shot mode (see Table 74). Additionally, an external signal can be AND'ed with the internal PWM signal. This option can be selected by writing a 1 to PWM_IN_AND, PWM_CFG[2]. The input to the external AND is the C6 I/O pin. C6 should be set to GPI (GPIO15). Note that the debounce for C6 will result in a delay of the AND'ing, and can be controlled using register GPI_15_DEB_DIS (Address 0x28, Bit[6]).

Newly programmed values are not latched until the final byte, PWM_ONT_HIGH_BYTE (Address 0x41, Bits[7:0]), is written.

CLOCK DIVIDER BLOCK

The ADP5589 features a clock divider block that divides down the frequency of an externally supplied source via I/O Pin C6. The output of the divider is driven out on I/O Pin R3.

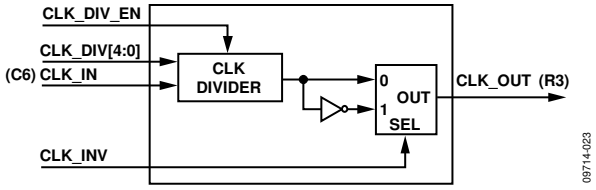


Figure 24. Clock Divider Block

RESET BLOCKS

The ADP5589 features two reset blocks that can generate reset conditions if certain events are detected at the same time. Up to three reset trigger events can be programmed for RESET1. Up to two reset trigger events can be programmed for RESET2. The event scan control blocks monitor whether these events are present for the duration of RESET_TRIGGER_TIME[2:0] (0x3D[4:2]). If they are, reset-initiate signals are sent to the reset generator blocks. The generated reset signal pulse width is programmable.

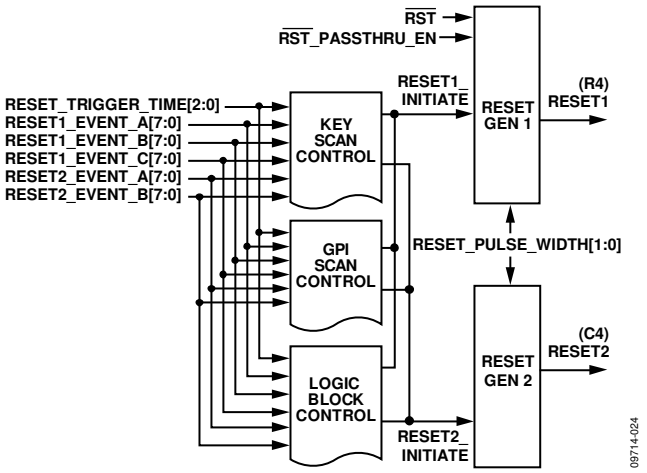


Figure 25. Reset Blocks

The RESET1 signal uses I/O Pin R4 as its output. A pass-through mode allows the main $\overline{\text{RST}}$ pin to be output on the R4 pin also.

The RESET2 signal uses I/O Pin C4 as its output.

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. This alleviates the need for removing the battery from the system and performing a hard reset.

It is not recommended to use the immediate trigger time (see the details of the RESET_CFG Register, 0x3D, in Table 69) because this setting may cause false triggering.

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 INT_EN (Address 0x4E, Bits[7:0]) (refer to Table 86). allows the user to choose whether the external interrupt pin remains asserted, or deasserts for 50 µs, then reasserts, in the case that there are multiple internal interrupts asserted, and one is cleared (refer to Table 85).

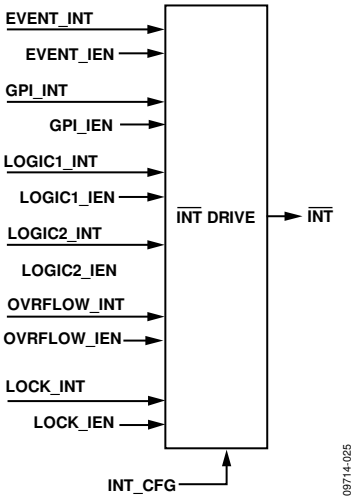


Figure 26. Asserting $\overline{\text{INT}}$ Low

REGISTER INTERFACE

Register access of the ADP5589 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 27 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 ADP5589 is 0x34, followed by the R/$\overline{W}$ bit set to 0 for a write cycle. The ADP5589 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 ADP5589 acknowledges the register pointer byte by pulling the data line low. The data byte to be written is sent next. The ADP5589 acknowledges the data byte by pulling the data line low. A stop condition completes the sequence.

Figure 28 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 set to 0 for a write cycle. The ADP5589 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 ADP5589 acknowledges the register pointer byte by pulling the data line low. The data byte to be written is sent next. The ADP5589 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 ADP5589 pulls the data line low after every byte, and a stop condition completes the sequence.

Figure 29 shows a typical byte 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 ADP5589 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 ADP5589 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 ADP5589 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 27. I²C Single-Byte Write Sequence

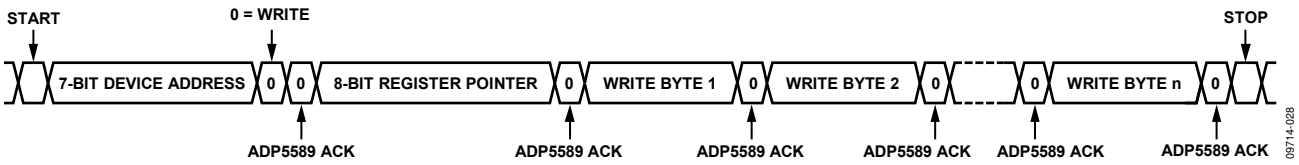


Figure 28. I²C Multibyte Write Sequence

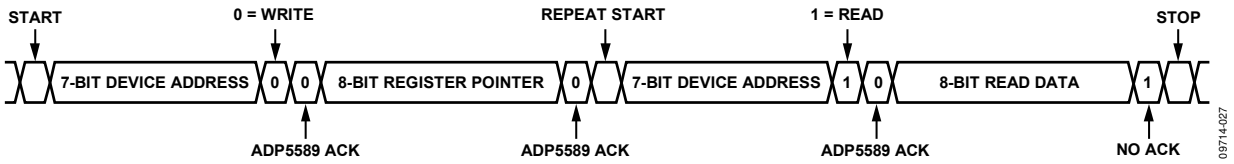


Figure 29. I²C Single-Byte Read Sequence

Figure 30 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 ADP5589 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 ADP5589 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 ADP5589 acknowledges the address byte by pulling the data line low. 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.

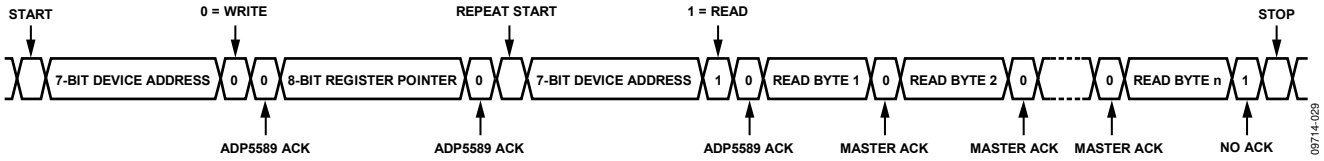


Figure 30. I²C Multibyte Read Sequence

REGISTER MAP

Table 6.

Addr.	R/W	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x00	R	MAN_ID				REV_ID			
0x01	R/W	Reserved		LOGIC2_INT	LOGIC1_INT	LOCK_INT	OVRFLOW_INT	GPI_INT	EVENT_INT
0x02	R	LOGIC2_STAT	LOGIC1_STAT	LOCK_STAT	EC[4:0]				
0x03	R	Event1_State	EVENT1_IDENTIFIER[6:0]						
0x04	R	Event2_State	EVENT2_IDENTIFIER[6:0]						
0x05	R	Event3_State	EVENT3_IDENTIFIER[6:0]						
0x06	R	Event4_State	EVENT4_IDENTIFIER[6:0]						
0x07	R	Event5_State	EVENT5_IDENTIFIER[6:0]						
0x08	R	Event6_State	EVENT6_IDENTIFIER[6:0]						
0x09	R	Event7_State	EVENT7_IDENTIFIER[6:0]						
0x0A	R	Event8_State	EVENT8_IDENTIFIER[6:0]						
0x0B	R	Event9_State	EVENT9_IDENTIFIER[6:0]						
0x0C	R	Event10_State	EVENT10_IDENTIFIER[6:0]						
0x0D	R	Event11_State	EVENT11_IDENTIFIER[6:0]						
0x0E	R	Event12_State	EVENT12_IDENTIFIER[6:0]						
0x0F	R	Event13_State	EVENT13_IDENTIFIER[6:0]						
0x10	R	Event14_State	EVENT14_IDENTIFIER[6:0]						
0x11	R	Event15_State	EVENT15_IDENTIFIER[6:0]						
0x12	R	Event16_State	EVENT16_IDENTIFIER[6:0]						
0x13	R	GPI_INT_STAT_A[7:0]							
0x14	R	GPI_INT_STAT_B[7:0]							
0x15	R	Reserved					GPI_INT_STAT_C[2:0]		
0x16	R	GPI_STATUS_A[7:0]							
0x17	R	GPI_STATUS_B[7:0]							
0x18	R	Reserved					GPI_STATUS_C[2:0]		
0x19	R/W	RPULL_CONFIG_A[7:0]							
0x1A	R/W	RPULL_CONFIG_B[7:0]							
0x1B	R/W	RPULL_CONFIG_C[7:0]							
0x1C	R/W	RPULL_CONFIG_D[7:0]							
0x1D	R/W	Reserved		RPULL_CONFIG_E[5:0]					
0x1E	R/W	GPI_INT_LEVEL_A[7:0]							
0x1F	R/W	GPI_INT_LEVEL_B[7:0]							
0x20	R/W	Reserved					GPI_INT_LEVEL_C[2:0]		
0x21	R/W	GPI_EVENT_EN_A[7:0]							
0x22	R/W	GPI_EVENT_EN_B[7:0]							
0x23	R/W	Reserved					GPI_EVENT_EN_C[2:0]		
0x24	R/W	GPI_INTERRUPT_EN_A[7:0]							
0x25	R/W	GPI_INTERRUPT_EN_B[7:0]							
0x26	R/W	Reserved					GPI_INTERRUPT_EN_C[2:0]		
0x27	R/W	DEBOUNCE_DIS_A[7:0]							
0x28	R/W	DEBOUNCE_DIS_B[7:0]							
0x29	R/W	Reserved					DEBOUNCE_DIS_C[2:0]		
0x2A	R/W	GPO_DATA_OUT_A[7:0]							
0x2B	R/W	GPO_DATA_OUT_B[7:0]							
0x2C	R/W	Reserved					GPO_DATA_OUT_C[2:0]		
0x2D	R/W	GPO_OUT_MODE_A[7:0]							
0x2E	R/W	GPO_OUT_MODE_B[7:0]							
0x2F	R/W	Reserved					GPO_OUT_MODE_C[2:0]		
0x30	R/W	GPIO_DIRECTION_A[7:0]							
0x31	R/W	GPIO_DIRECTION_B[7:0]							
0x32	R/W	Reserved					GPIO_DIRECTION_C[2:0]		

Addr.	R/W	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x33	R/W	UNLOCK1_ STATE	UNLOCK1[6:0]						
0x34	R/W	UNLOCK2_ STATE	UNLOCK2[6:0]						
0x35	R/W	EXT_LOCK_ STATE	EXT_LOCK_EVENT[6:0]						
0x36	R/W	INT_MASK_TIMER[4:0]					UNLOCK_TIMER[2:0]		
0x37	R/W	Reserved							LOCK_EN
0x38	R/W	RESET1_ EVENT_A Level	RESET1_EVENT_A[6:0]						
0x39	R/W	RESET2_ EVENT_B Level	RESET1_EVENT_B[6:0]						
0x3A	R/W	RESET1_ EVENT_B Level	RESET1_EVENT_C[6:0]						
0x3B	R/W	RESET1_ EVENT_B Level	RESET2_EVENT_A[6:0]						
0x3C	R/W	RESET1_ EVENT_B Level	RESET2_EVENT_B[6:0]						
0x3D	R/W	RESET2_POL	RESET1_POL	$\overline{\text{RST}}$_ PASSTHRU_EN	RESET_TRIGGER_TIME[2:0]			RESET_PULSE_WIDTH[1:0]	
0x3E	R/W	PWM_OFFT_LOW_BYTE[7:0]							
0x3F	R/W	PWM_OFFT_HIGH_BYTE[7:0]							
0x40	R/W	PWM_ONT_LOW_BYTE[7:0]							
0x41	R/W	PWM_ONT_HIGH_BYTE[7:0]							
0x42	R/W	Reserved					PWM_IN_AND	PWM_MODE	PWM_EN
0x43	R/W	Reserved	CLK_INV	CLK_DIV[4:0]					CLK_DIV_EN
0x44	R/W	Reserved	LY1_INV	LC1_INV	LB1_INV	LA1_INV	LOGIC1_SEL[2:0]		
0x45	R/W	LY1_CASCADE	LY2_INV	LC2_INV	LB2_INV	LA2_INV	LOGIC2_SEL[2:0]		
0x46	R/W	Reserved				FF2_SET	FF2_CLR	FF1_SET	FF1_CLR
0x47	R/W	Reserved		LY2_DBNC_DIS	LOGIC2_ EVENT_EN	LOGIC2_INT_ LEVEL	LY1_DBNC_ DIS	LOGIC1_ EVENT_EN	LOGIC1_INT_ LEVEL
0x48	R/W	Reserved						KEY_POLL_TIME[1:0]	
0x49	R/W	PIN_CONFIG_A[7:0]							
0x4A	R/W	PIN_CONFIG_B[7:0]							
0x4B	R/W	Reserved					PIN_CONFIG_C[2:0]		
0x4C	R/W	PULL_SELECT	C4_EXTEND_ CFG	R4_EXTEND_ CFG	C6_ EXTEND_CFG	R3_EXTEND_CFG[1:0]		C9_EXTEND_ CFG	R0_EXTEND_ CFG
0x4D	R/W	OSC_EN	CORE_FREQ[1:0]		LCK_TRK_ LOGIC	LCK_TRK_GPI		$\overline{\text{INT}}$_CFG	$\overline{\text{RST}}$_CFG
0x4E	R/W	Reserved		LOGIC2_IEN	LOGIC1_ IEN	LOCK_IEN	OVRFLOW_ IEN	GPI_IEN	EVENT_IEN

DETAILED REGISTER DESCRIPTIONS

Note: N/A throughout this section means not applicable.

Note: All registers default to 0000 0000 unless otherwise specified.

ID Register 0x00

Table 7. ID Bit Descriptions

Bits	Name	R/W	Description
[7: 4]	MAN_ID	R	Manufacturer ID, default = 0001.
[3:0]	REV_ID	R	Rev ID.
Default = 0001 XXXX			

INT_STATUS Register 0x01

Table 8. INT_STATUS Bit Descriptions

Bits	Name	R/W	Description
[7: 6]	N/A		Reserved.
5	LOGIC2_INT	R/W	0 = no interrupt. 1 = interrupt due to a general Logic 2 condition. Write a 1 to this bit to clear it.
4	LOGIC1_INT	R/W	0 = no interrupt. 1 = interrupt due to a general Logic 1 condition. Write a 1 to this bit to clear it.
3	LOCK_INT	R/W	0 = no interrupt. 1 = interrupt due to a lock/unlock condition. The user can read LOCK_STAT (0x02[5]) to determine if LOCK_INT is due to a lock or unlock event. If LOCK_STAT = 1, LOCK_INT is due to a lock event. If LOCK_STAT = 0, LOCK_INT is due to an unlock event. Write a 1 to this bit to clear it. If lock mode is enabled via the software bit LOCK_EN (0x37[0]), a LOCK_INT is not generated because the processor knows it just enabled lock mode. If lock mode is disabled (while locked) via the software bit LOCK_EN, a LOCK_INT is not generated because the processor knows it just disabled lock mode.
2	OVRFLOW_INT	R/W	0 = no interrupt. 1 = interrupt due to an overflow condition. Write a 1 to this bit to clear it.
1	GPI_INT	R/W	0 = no interrupt. 1 = interrupt due to a general GPI condition. This bit is not set by a GPI that has been configured to update the FIFO and event count. Write a 1 to this bit to clear it. This bit cannot be cleared until all GPI_x_INT bits are cleared.
0	EVENT_INT	R/W	0 = no interrupt. 1 = interrupt due to key event (press/release), GPI event (GPI programmed for FIFO updates), or Logic 1/Logic 2 event (programmed for FIFO updates).Write a 1 to this bit to clear it.

Status Register 0x02

Table 9. Status Bit Descriptions

Bits	Name	R/W	Description
7	LOGIC2_STAT	R	0 = output from Logic Block 2. (LY2) is low. 1 = output from Logic Block 2. (LY2) is high.
6	LOGIC1_STAT	R	0 = output from Logic Block 1 (LY1) is low. 1 = output from Logic Block 1 (LY1) is high.
5	LOCK_STAT	R	0 = unlocked. 1 = locked.
[4:0]	EC[4:0]	R	Event count value. Indicates how many events are currently stored on the FIFO.

FIFO_1 Register 0x03

Table 10. FIFO_1 Bit Descriptions

Bits	Name	R/W	Description
7	Event1_State	R	The seven lower bits of each FIFO location contain the event identifier, which can be decoded to reveal the event recorded. Table 11 outlines each event number, what it represents, and the I/O pins associated with it. Bit 7 is the Event 1 state.
[6:0]	EVENT1_IDENTIFIER[6:0]		This bit represents the state of the event that is recorded in EVENT1_IDENTIFIER[6:0]. For key events (Event 1 to Event 96). 1 = key is pressed. 0 = key is released. For GPI and logic events (Event 97 to Event 117). 1 = GPI/logic is active. 0 = GPI/logic is inactive. Active and inactive states are programmable.

Table 11. Event Decoding

Event No.	Meaning	Event No.	Meaning	Event No.	Meaning	Event No.	Meaning
0	No event	32	Key 32 (R2, C9)	64	Key 64 (R5, C8)	96	Key 96 (R7, GND)
1	Key 1 (R0, C0)	33	Key 33 (R2, C10)	65	Key 65 (R5, C9)	97	GPI 1 (R0)
2	Key 2 (R0, C1)	34	Key 34 (R3, C0)	66	Key 66 (R5, C10)	98	GPI 2 (R1)
3	Key 3 (R0, C2)	35	Key 35 (R3, C1)	67	Key 67 (R6, C0)	99	GPI 3 (R2)
4	Key 4 (R0, C3)	36	Key 36 (R3, C2)	68	Key 68 (R6, C1)	100	GPI 4 (R3)
5	Key 5 (R0, C4)	37	Key 37 (R3, C3)	69	Key 69 (R6, C2)	101	GPI 5 (R4)
6	Key 6 (R0, C5)	38	Key 38 (R3, C4)	70	Key 70 (R6, C3)	102	GPI 6 (R5)
7	Key 7 (R0, C6)	39	Key 39 (R3, C5)	71	Key 71 (R6, C4)	103	GPI 7 (R6)
8	Key 8 (R0, C7)	40	Key 40 (R3, C6)	72	Key 72 (R6, C5)	104	GPI 8 (R7)
9	Key 9 (R0, C8)	41	Key 41 (R3, C7)	73	Key 73 (R6, C6)	105	GPI 9 (C0)
10	Key 10 (R0, C9)	42	Key 42 (R3, C8)	74	Key 74 (R6, C7)	106	GPI 10 (C1)
11	Key 11 (R0, C10)	43	Key 43 (R3, C9)	75	Key 75 (R6, C8)	107	GPI 11 (C2)
12	Key 12 (R1, C0)	44	Key 44 (R3, C10)	76	Key 76 (R6, C9)	108	GPI 12 (C3)
13	Key 13 (R1, C1)	45	Key 45 (R4, C0)	77	Key 77 (R6, C10)	109	GPI 13 (C4)
14	Key 14 (R1, C2)	46	Key 46 (R4, C1)	78	Key 78 (R7, C0)	110	GPI 14 (C5)
15	Key 15 (R1, C3)	47	Key 47 (R4, C2)	79	Key 79 (R7, C1)	111	GPI 15 (C6)
16	Key 16 (R1, C4)	48	Key 48 (R4, C3)	80	Key 80 (R7, C2)	112	GPI 16 (C7)
17	Key 17 (R1, C5)	49	Key 49 (R4, C4)	81	Key 81 (R7, C3)	113	GPI 17 (C8)
18	Key 18 (R1, C6)	50	Key 50 (R4, C5)	82	Key 82 (R7, C4)	114	GPI 18 (C9)
19	Key 19 (R1, C7)	51	Key 51 (R4, C6)	83	Key 83 (R7, C5)	115	GPI 19 (C10)
20	Key 20 (R1, C8)	52	Key 52 (R4, C7)	84	Key 84 (R7, C6)	116	Logic 1
21	Key 21 (R1, C9)	53	Key 53 (R4, C8)	85	Key 85 (R7, C7)	117	Logic 2
22	Key 22 (R1, C10)	54	Key 54 (R4, C9)	86	Key 86 (R7, C8)	118	Unused
23	Key 23 (R2, C0)	55	Key 55 (R4, C10)	87	Key 87 (R7, C9)	119	Unused
24	Key 24 (R2, C1)	56	Key 56 (R5, C0)	88	Key 88 (R7, C10)	120	Unused
25	Key 25 (R2, C2)	57	Key 57 (R5, C1)	89	Key 89 (R0, GND)	121	Unused
26	Key 26 (R2, C3)	58	Key 58 (R5, C2)	90	Key 90 (R1, GND)	122	Unused
27	Key 27 (R2, C4)	59	Key 59 (R5, C3)	91	Key 91 (R2, GND)	123	Unused
28	Key 28 (R2, C5)	60	Key 60 (R5, C4)	92	Key 92 (R3, GND)	124	Unused
29	Key 29 (R2, C6)	61	Key 61 (R5, C5)	93	Key 93 (R4, GND)	125	Unused
30	Key 30 (R2, C7)	62	Key 62 (R5, C6)	94	Key 94 (R5, GND)	126	Unused
31	Key 31 (R2, C8)	63	Key 63 (R5, C7)	95	Key 95 (R6, GND)	127	Wildcard for unlock