ANALOG DEVICES

FAQ

One Technology Way • P.O. Box 9106 • Norwood, MA 02062-9106, U.S.A. • Tel: 781.329.4700 • Fax: 781.461.3113 • www.analog.com

AD7606B: Advanced Features Enabled in Software Mode and System Level Benefits

by Lluis Beltran Gil

INTRODUCTION

The AD7606B is an enhanced version of the AD7606 analog-to-digital, data acquisition systems (DAS). The AD7606B can be used as direct pin for pin replacement for the AD7606, which is called hardware mode, with the benefits stated in Table 1. However, the AD7606B in software mode includes the following advanced features:

- Independent, per channel basis, range selection, including a ±2.5 V range option
- System gain/phase/offset on-chip calibration
- Analog input open-circuit detection
- Additional oversampling ratios (OSR = 128 and 256)
- Optional 1, 2, or 4 serial data output configurations
- Diagnostics and monitoring

Software or register mode is enabled by tying the three oversampling digital inputs high: OS2, OS1, and OS0, allowing access to configure the register map to take advantages of the previously stated features. See the AN-1559 Application Note for full details on backwards compatibility and how to migrate from the AD7606 to the AD7606B in both hardware mode and software mode, as well as how to write the register map in both serial (SDI pin) and parallel interface ($\overline{WR}$ pin).

This document describes each of the advanced features the AD7606B has available in software mode and their system level impact. For full details on the AD7606B, see the AD7606B data sheet. Consult this data sheet in conjunction with this document.

NOTE

Table 1. Summary of the Differences Between the AD7606 and the AD7606B

Parameter	AD7606	AD7606B (Hardware Mode)	AD7606B (Software Mode)
Typical Input Impedance	1 MΩ	5 MΩ	5 MΩ
Maximum Throughput Rate	200 kSPS	800 kSPS	800 kSPS
Temperature Range	−40°C to +85°C	−40°C to +125°C	−40°C to +125°C
V_{DRIVE} Range	2.3 V to 5.25 V	1.71 V to 3.6 V	1.71 V to 3.6 V
Absolute Maximum Input Voltage	±16.5 V	±21 V	±21 V
Analog Input Range	±10 V or ±5 V[1]	±10 V or ±5 V[1]	±10 V, ±5 V, or ±2.5 V[2]
System Gain, Phase, and Offset On-Chip Calibration	Not applicable	Not accessible	Available[2]
Oversampling Ratio (OSR)	From no oversampling to OSR = 64	From no OS to OSR = 64	From no OS to OSR = 256
Open-Circuit Detection	Not applicable	Not accessible	Available[2]
Serial Data Output lines	2	2	Selectable: 1, 2, or 4
Diagnostics	Not applicable	Not accessible	Available

[1] Not on a per channel basis.
[2] On a per channel basis.

TABLE OF CONTENTS

INITIALIZE THE AD7606B

ENTER INTO SOFTWARE MODE AT POWER-UP

To initialize the AD7606B in software mode:

- All three oversampling pins (OS2, OS1, and OS0) must be tied to logic high.
- PAR/SER pin to be tied low or high depending on the data interface to use, parallel or serial, respectively.
- RANGE can be tied either low or high because it is ignored in software mode. The range is selected through the corresponding register, on a per channel basis.

After the initialization process shown above, the AD7606B is configured in software mode with registers in default status. In this document, and the firmware example referred on the note section, the AD7606B is used in software serial mode as an example.

After the initialization, the voltage pins are typically as follows:

- AVCC = 5.0 V
- V_{DRIVE} = 3.3 V, 2.5 V or 1.8 V
- REGCAP = 1.9 V
- REFIN/REFOUT = 2.5 V
- REFCAPA, REFCAPB = 4.4 V

When these voltages are verified, issuing a reset ensures the default parameters are properly loaded on the AD7606B. The following code gives an example for issuing a reset:

```
int32_t ad7606_reset(struct ad7606_dev *dev)
{
        int32_t ret;
        ret = gpio_set_value(dev->gpio_reset, 1);
                udelay(3);
        ret = gpio_set_value(dev->gpio_reset, 0);
        ad7606_reset_settings(dev);
        return ret;
}
```

AD7606B CHECK BY REGISTER READ

After the AD7606B initialization, register values are all in default, as shown in the AD7606B data sheet. Users may check the DEVICE_ID, SILICON_REVISION, or RESET_DETECT bits within the memory map, for example, to confirm a valid full or partial reset is received. A single AD7606B register read function is shown below as a reference, just replace the register address by those containing the DEVICE_ID, SILICON_REVISION, or RESET_DETECT. SPI mode 2 can be used with direct connection between the MCU/DSP and the AD7606B.

```
int32_t ad7606_spi_reg_read(struct ad7606_dev *dev,
                            uint8_t reg_addr,
                            uint8_t *reg_data)
{
        uint8_t buf[3];
        uint32_t sz = 2;
        int32_t ret;
        buf[0] = AD7606_RD_FLAG_MSK(reg_addr);
        buf[1] = 0x00;
        ret = spi_write_and_read(dev->spi_desc, buf, sz);
        if (ret < 0)
                return ret;
        dev->reg_mode = true;
        buf[0] = AD7606_RD_FLAG_MSK(reg_addr);
        buf[1] = 0x00;

        ret = spi_write_and_read(dev->spi_desc, buf, sz);
        if (ret < 0)
                return ret;
        if (reg_data)
                *reg_data = buf[1];
        return ret;
}
```

REGISTER CONFIGURATION

After verified the device is up and running, registers must be written to load the working configuration, for example, parameters such as the range, the oversampling, or the diagnostics. Some functions included on the No-OS drivers, like ad7606_set_oversampling() or ad7606_set_ch_range(), can get this configuration written. These functions in turn uses the ad7606_spi_reg_write() function, used in general for writing any device register. The following code is an example for writing any register:

```
int32_t ad7606_spi_reg_write(struct ad7606_dev *dev,
                             uint8_t reg_addr,
                             uint8_t reg_data)
{
        uint8_t buf[3];
```

```
        int32_t ret;
        uint32_t sz = 2;
        /* Dummy read to place the chip in register mode. */
        if (!dev->reg_mode) {
                ret = ad7606_spi_reg_read(dev, reg_addr,
NULL);
        }
        buf[0] = AD7606_WR_FLAG_MSK(reg_addr);
        buf[1] = reg_data;

        ret = spi_write_and_read(dev->spi_desc, buf, sz);
        return ret;
}
```

CALIBRATION FEATURES

While hardware mode is selected on the AD7606B, this device is fully compatible regarding pins and function with the AD7606. The benefits the AD7606B brings are higher input impedance, higher throughput rate, wider temperature range, and lower digital supply voltage operating range, as described in the AN-1559 Application Note. However, entering software mode, by tying the three oversampling pins high (Pin 3, Pin 4, and Pin 5), enable many advanced features that bring system level benefits, as described in the following sections.

ANALOG INPUT RANGE

On the AD7606, the polarity on the digital input RANGE pin (Pin 8) determines the input range on the eight analog input channels. If this pin is tied to a logic high, the analog input range is ±10 V for all channels. If this pin is tied to a logic low, the analog input range is ±5 V for all channels. A logic change on this pin has an immediate effect on the analog input range.

Likewise, on the AD7606B in hardware mode, the analog input range for all channels is selected through the RANGE pin. However, in software mode the analog input range is selected on a per channel basis. To do so, Register 0x03 through Register 0x06 are used to individually set the range on each channel.

For example, in the RANGE_CH1_CH2 register (Address 0x03) shown in Figure 1, the four MSBs determine the analog input range on Channel 2 and the four LSBs determine the analog input range on Channel 1. The analog input voltage can be chosen to be either ±2.5 V, ±5 V, or ±10 V single-ended.

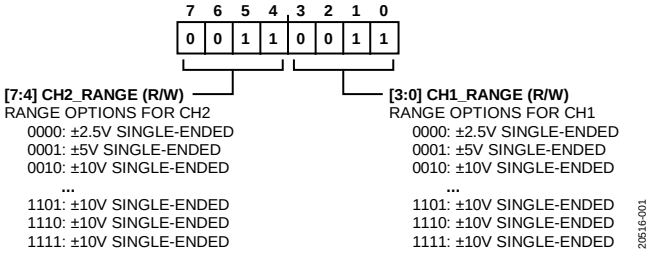

Figure 1. RANGE_CH1_CH2 Register, Address 0x03

This per channel analog input range selection allows a different range on each channel to be set, which may be beneficial if, for example, the voltage and current channels needed a different range on an energy monitoring application. The per channel analog input range selection can also serve for setting different ranges on measurement and protection channels, for example, in energy protection applications.

SYSTEM OFFSET ERROR CALIBRATION

The AD7606B in software mode has registers to compensate for offset errors present in the system, whether these are due to the inherent offset error of the sensor or a mismatch between the front-end resistors on each analog input (Vx and VxGND), for example, R_{FILTER} on Figure 2.

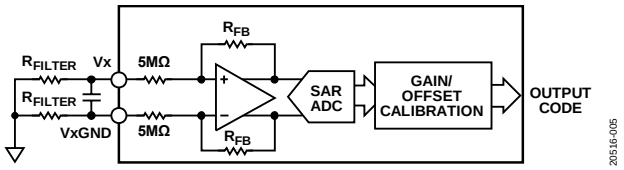

Figure 2. Gain/Offset On-Chip Calibration

An 8-bit long register per each channel, Address 0x11 through Address 0x18, allows the AD7606B to digitally add an offset from −128 LSBs (writing 0x00 to the register) to +127 LSBs (writing 0xFF to the register) to the ADC conversion data, to compensate for such system offset error.

In other words, the offset added to the ADC code is, in LSBs, equal to the CHx_OFFSET registers minus 128. Some examples for CHx_OFFSET register data written and the corresponding offset added/subtracted from the ADC code are shown in Table 2.

$$Output\ Code\ (LSBs) = ADC\ Code + CHx\_OFFSET - 128$$

Table 2. CH1_OFFSET Register Bit Decoding

CH1_OFFSET Register	Offset Calibration
0x00	−128 LSBs
0x45	−59 LSBs
0x80 (Default)	0 LSBs
0x83	+3 LSBs
0xFF	+127 LSBs

By connecting the two inputs together and tying them to AGND, as shown in Figure 2, the system offset can be obtained by reading the ADC code. This system offset can then be calibrated by writing the corresponding register.

For example, if the sensor connected on Channel 1 has an offset equivalent to 10 LSBs (average value out of multiple ADC data reads to remove distribution caused by noise) or the mismatch on the front-end resistors causes an ADC code of 10 LSBs when the inputs are tied to ground, 10 LSBs then must be subtracted from the ADC code on each conversion to remove that offset. Because 128 is the default value on the offset registers, writing the CH1_OFFSET register (Address 0x11) with 0x76 (118d (decimal)), the AD7606B effectively subtracts 10 LSBs from each ADC converted code. Therefore, this system offset error can be internally compensated through on-chip registers, eliminating the burden of having to compensate at the back-end through software. To learn more on how to program the CHx_OFFSET register, refer to the AD7606x software model from AD7606B product page.

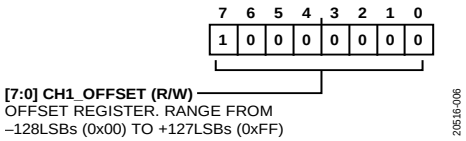

Figure 3. CH1_OFFSET Register, Address 0x11, Default Value 0x80

The following code is used to perform offset error calibration:

```
int32_t ad7606_set_ch_offset(struct ad7606_dev *dev, uint8_t
ch, int8_t offset)
{
        int ret;
        uint8_t value = (uint8_t)(offset - 0x80);
        if (ch >= dev->num_channels)
                return -EINVAL;
        ret = ad7606_spi_reg_write(dev,
AD7606_REG_OFFSET_CH(ch), value);
        dev->offset_ch[ch] = offset;
        return ret;
}
```

SYSTEM GAIN ERROR CALIBRATION

Depending on the analog input range selected on each channel, the programmable gain amplifier (PGA) is configured so that the proper gain (feedback resistor (R_{FB})/input resistor (R_{IN})) is selected to scale the analog input signal to meet the input range of the analog-to-digital converter (ADC).

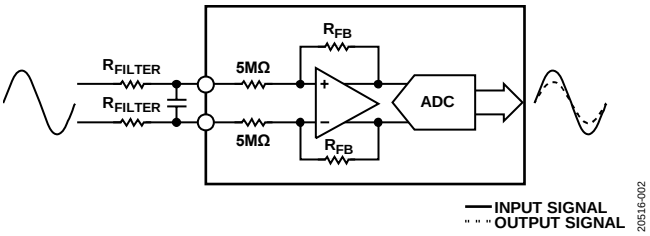

Figure 4. Analog Input Circuitry, R_{IN} of AD7606 is 1 MΩ while R_{IN} of AD7606B is 5 MΩ

Internally, the AD7606B has tight control over R_{FB} and R_{IN}, such that this ideal gain is accurately set by factory trimming, assuming no resistance on the front-end. However, if an external resistor is placed in the front end, for example, for protection or for extra filtering, the actual gain is no longer the ideal R_{FB}/R_{IN}.

$$IdealGain = \frac{R_{FB}}{R_{IN}}$$

$$ActualGain = \frac{R_{FB}}{R_{FILTER} + R_{IN}}$$

The addition of a filter resistor (R_{FILTER}) in series to R_{IN} modifies the divisor, which changes the actual system gain by adding an extra gain error. The bigger the R_{FILTER} used, the bigger the gain error introduced becomes, as shown in Figure 5. For a given R_{FILTER}, because the R_{IN} of the AD7606B is much bigger (5 MΩ) than that of the AD7606 (1 MΩ), the gain error introduced by R_{FILTER} is much smaller on the AD7606B compared to the AD7606. Therefore, the AD7606B is less sensitive to the gain error introduced by placing an external R_{FILTER} to the device.

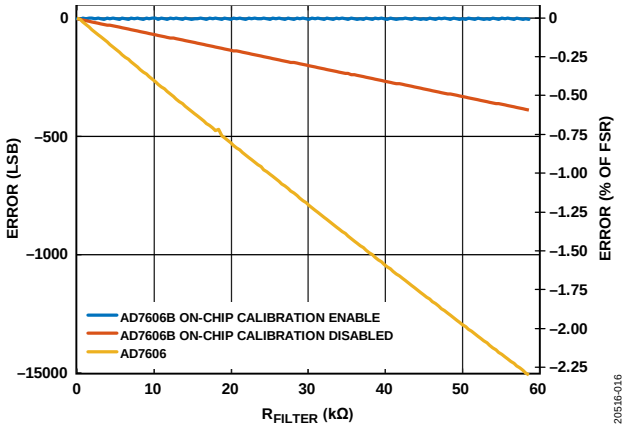

Figure 5. System Gain Error Introduced by R_{FILTER} for Either the AD7606, AD7606B, or AD7606B through On-Chip Calibration

Knowing the typical input resistance, provided in both the AD7606 data sheet and the AD7606B data sheet, a back-end calibration on the controller side (field programmable gate array (FPGA), digital signal processing (DSP)) can be programmed if there are resources available. However, if using the AD7606B in software mode, this system gain error can be automatically compensated by an on-chip register, on a per channel basis. As the input impedance of the AD7606B is internally known accurately on each device, its automatic gain calibration is always more precise than any back-end calibration programmed based on the input impedance typicial numbers, reducing the burden of doing all calibration computation on the controller side. See Figure 6 for an example of the calibration performed on the AD7606B for a certain range of R_{FILTER} values.

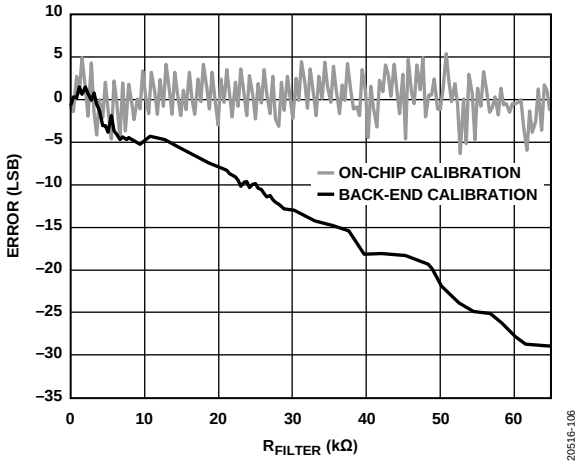

Figure 6. Total Error after Applying Back-End Calibration (Assuming R_{IN} Equal

to Typical Input Impedance Value) Compared to Applying the On-Chip Calibration

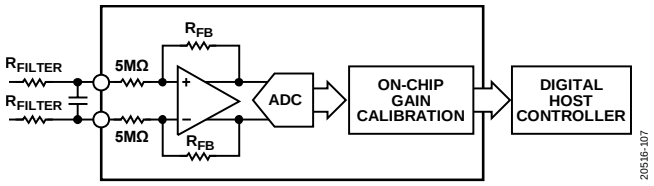

Figure 7. Analog Signal Chain with per Channel Gain Calibration, to Remove Burden on Back-End Digital Controller

To use the gain calibration feature, Register 0x09 through Register 0x10 are allocated to program the R_{FILTER} used on each channel, up to 65 kΩ with a resolution of 1024 Ω. Using this feature, the system gain error can be kept lower than 0.02% independently of what R_{FILTER} is used, as seen in Figure 5. If back-end calibration was used instead, take extra error margin into account due to input impedance uncertainty.

For example, the system gain error introduced by a 27 kΩ R_{FILTER} used in the front end of Channel 5 is automatically compensated by writing 0x1B (27d) into the CH5_GAIN register (Address 0x0D), shown in Figure 8. To learn more on how to program the CHx_GAIN register, refer to the AD7606x software model from AD7606B product page.

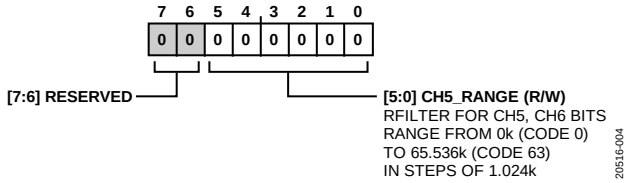

Figure 8. CH5_GAIN Register, Address 0x0D

The following code is an example for configuring the channel´s gain:

```
int32_t ad7606_set_ch_gain(struct ad7606_dev *dev, uint8_t ch,
                           uint8_t gain)
{
        int ret;
        if (ch >= dev->num_channels)
                return -EINVAL;
        gain = field_get(AD7606_GAIN_MSK, gain);
        ret = ad7606_spi_reg_write(dev,
AD7606_REG_GAIN_CH(ch), gain);
        dev->gain_ch[ch] = gain;
        return ret;
}
```

SYSTEM PHASE ERROR CALIBRATION

When measuring voltages and currents on the same power line, through a current transformer, there is a phase mismatch between the two channels. Something similar may happen if there is a mismatch between R_{FILTER} and filter capacitance (C_{FILTER}) between two channels that need to be sampled simultaneously.

This system phase error can be internally compensated on the AD7606B by delaying the internal sampling instant. On the example shown in Figure 9, there is a slight delay (t_{PHASE_REG}) between Channel 1 and Channel 4. If this time delay is known, by programming it back to the CH4_PHASE register (Address 0x1C), Channel 4 is effectively a sampled t_{PHASE_REG} time after V1, compensating for the phase mismatch between the two channels and providing an in-phase output.

Note that delaying any channel implies lengthening the BUSY pin high time, that is, conversion time (t_{CONV}) extends by as much as the most delayed channel is programmed at and, therefore, the maximum throughput rate may be compromised.

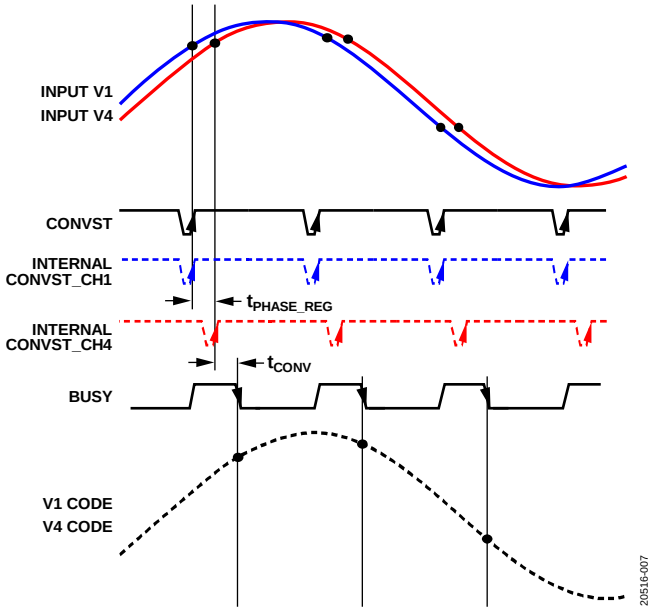

Figure 9. System Phase Calibration Functionality

There is one phase register per channel, Address 0x19 through Address 0x20, allowing compensating the phase error on each channel individually with a resolution of 1.25 µs, and up to a 318.75 µs time delay in each channel with regards to the CONVST pin signal.

To check this feature, these steps might be followed:

- Apply one AC source (or multi-channel calibrator with accurate phase control), like 50/60 Hz, to all the signal chain inputs.
- Capture enough channel codes to calculate the channel to channel group delay matching by FFT, DFT or some other method.
- Calculate the CHx_PHASE_OFFSET register values in steps of 1.25 µs.
- Write the phase matching error registers into AD7606B.

The following code is used to write the phase matching error registers:

```
int32_t ad7606_set_ch_phase(struct ad7606_dev *dev, uint8_t ch,  uint8_t phase)
{
        int ret;
        if (ch >= dev->num_channels)
                return -EINVAL;
        ret = ad7606_spi_reg_write(dev, AD7606_REG_PHASE_CH(ch), phase);
        dev->phase_ch[ch] = phase;
        return ret;
}
```

OPEN-CIRCUIT DETECTION

As explained in the AN-1559 Application Note, because the AD7606B has a larger R_{IN} than the AD7606, that allows for more easy detection of when the sensor has disconnected by using a given pull-down resistor (R_{PD}) in parallel with the sensor. The increase of R_{IN} results in a lower ADC output code when the sensor disconnects.

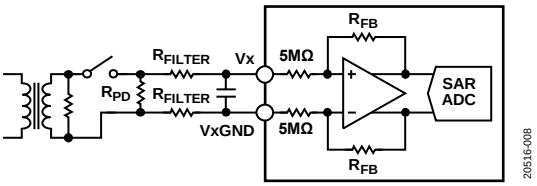

Figure 10. Analog Front End with R_{PD}

When entering software mode of the AD7606B, there is an open-circuit detection feature, disabled by default, with two possible modes, manual and automatic open-circuit detection.

Manual Mode

The open-circuit detection feature is disabled by default, but writing 0x01 into the OPEN_DETECT_QUEUE register (Address 0x2C) enables the manual open-circuit detection mode.

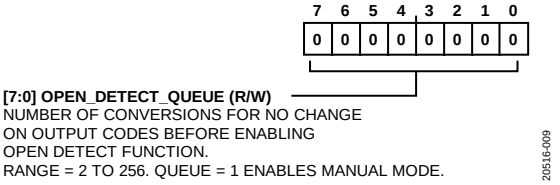

Figure 11. OPEN_DETECT_QUEUE Register, Address 0x2C

In manual mode, if the ADC code repeats for a number of conversions, before assuming the sensor has been disconnected, this can be verified through changing the PGA common mode. The common-mode voltage of PGA is controlled by the corresponding CHx_OPEN_DETECT_EN bit in the OPEN_DETECT_ENABLE register (Address 0x23), on a per channel basis, as shown in Figure 12. If changing the PGA common mode on a given channel changes the ADC output code, this indicates that the sensor has been disconnected. Otherwise, the sensor is still connected to the analog input.

This method still requires some back-end effort. Detect N repeated samples (below a threshold), change the PGA common mode, and then verify if the ADC output code changes with the common-mode change or not. This burden can be eliminated by enabling the automatic mode, which lets the AD7606B automatically detect when an open circuit occurs. The following code implements the open circuit detection algorithm, in manual mode:

```
/* Enter into manual open circuit detection mode */
do {
        if (ad7606_spi_reg_write(device, AD7606_REG_OPEN_DETECT_QUEUE, 1) == SUCCESS) {
                /* Read the ADC on selected chnnel (first reading post open circuit detection start) */
                prev_adc_code = single_data_read(device,
                                                 channel->ch_num - 1,
                                                 attr_polarity_val[channel->ch_num - 1]);

                /* Perform N conversions and monitor the code delta */
                for (cnt = 0; cnt < MANUAL_OPEN_DETECT_CONV_CNTS; cnt++) {
                        /* Check if code is within 350LSB (nearest ZS code) */
                        if (prev_adc_code >= 0 && prev_adc_code < MANUAL_OPEN_DETECT_ENTRY_TRHLD) {
                                /* Perform next conversion and read the result */
```

```
                        curr_adc_code = single_data_read(device,
                                                         channel->ch_num - 1,
                                                         attr_polarity_val[channel->ch_num - 1]);

                        /* Check if delta b/w current and previus reading is within 10 LSB code */
                        if (abs(curr_adc_code - prev_adc_code) > MANUAL_OPEN_DETECT_CONV_TRSHLD)
{
                                open_detect_done = true;
                                break;
                        }

                        /* Get the previous code */
                        prev_adc_code = curr_adc_code;
                } else {
                        open_detect_done = true;
                        break;
                }
        }

        /* Break if open circuit detection aborted (in case above conditions not met) */
        if (open_detect_done)
                break;

        /* Set common mode high (enabling open circuit detect on selected channel) */
        if (ad7606_spi_reg_write(device,
                                 AD7606_REG_OPEN_DETECT_ENABLE,
                                 (1 << ((channel->ch_num) - 1))) == SUCCESS) {

                /* Perform next conversions (~2-3) and read the result (with common mode set high) */
                for (cnt = 0; cnt < MANUAL_OPEN_DETECT_CM_CNV_CNT; cnt++) {
                        udelay(100);
                        curr_adc_code = single_data_read(device,
                                                         channel->ch_num - 1,
                                                         attr_polarity_val[channel->ch_num - 1]);
                }

                /* Check if delta b/w common mode high code and previous N conversion code is > threshold */
                if ((curr_adc_code - prev_adc_code) < MANUAL_OPEN_DETECT_THRESHOLD_RPD50K) {
                        open_detect_done = true;
```

```
                        break;
                }
        } else {
                return -EINVAL;
        }

        /* Break if open circuit detection aborted (in case above conditions not met) */
        if (open_detect_done)
                break;

        /* Set common mode low (disabling open circuit detect on channel) */
        if (ad7606_spi_reg_write(device,
                                 AD7606_REG_OPEN_DETECT_ENABLE,
                                 0) == SUCCESS) {
                /* Perform next conversion and read the result (with common mode set low) */
                curr_adc_code = single_data_read(device,
                                                 channel->ch_num - 1,
                                                 attr_polarity_val[channel->ch_num - 1]);

                /* Check if delta b/w common mode low code and previous N conversion code is < threshold */
                if (abs(curr_adc_code - prev_adc_code) < MANUAL_OPEN_DETECT_THRESHOLD_RPD50K) {
                        open_detect_flag = true;
                        open_detect_done = true;
                }
        } else {
                return -EINVAL;
        }
    } else {
        return -EINVAL;
    }
} while (0);
```

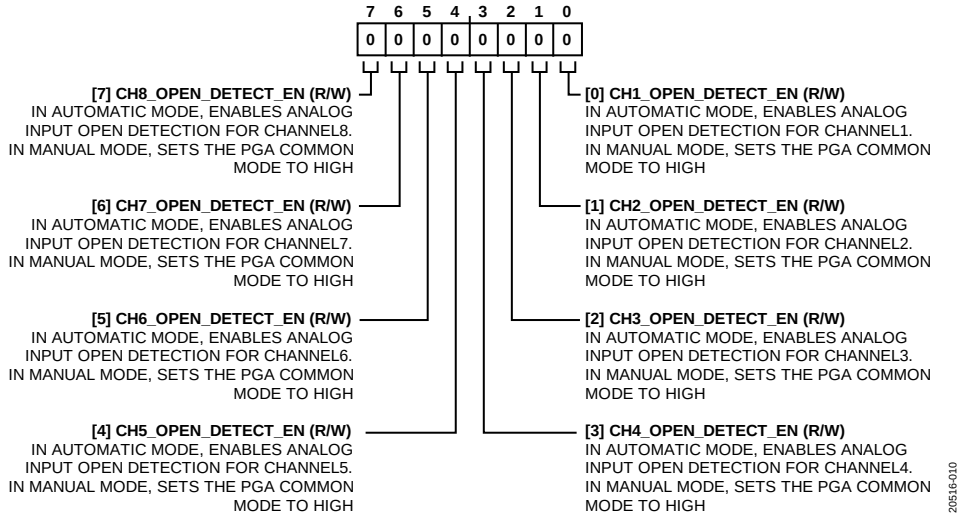

Figure 12. OPEN_DETECT_ENABLE Register, Address 0x23

Automatic Mode

In automatic mode, the previous method is implemented on chip, eliminating the burden on the back-end software that detects the sensor disconnection. The automatic mode is enabled through writing the number of conversions (N) before the flag triggers, as shown in Figure 11. Then, each channel checker can be enabled/disabled individually through the OPEN_DETECT_ENABLE register (Figure 12).

The algorithm, simplified in the diagram in Figure 13, runs in the background, and if in a particular channel the error flag triggers, the algorithm can be checked in the OPEN_ DETECTED register (Address 0x24) or in the STATUS header if appended to the ADC data, or in the STATUS register.

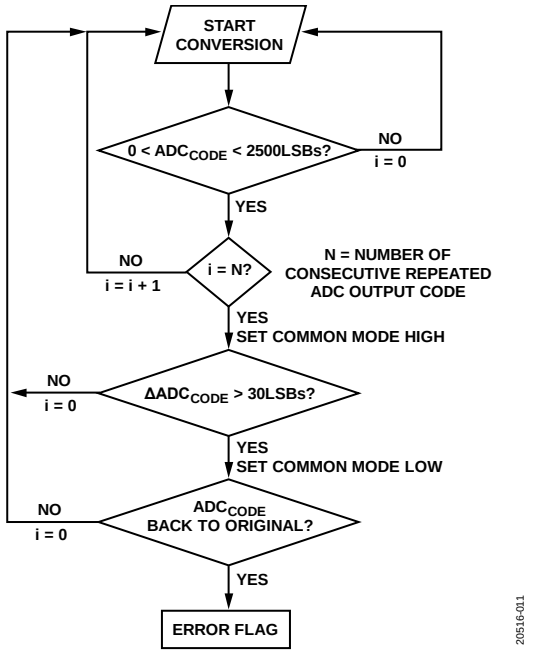

Figure 13. Automatic Open Circuit Detection Algorithm

The following code implements the open circuit detection algorithm, in automatic mode :

/ Enter into open circuit auto open detect mode */*

```
if (ad7606_spi_reg_write(device,
                         AD7606_REG_OPEN_DETECT_QUEUE,
                         open_detect_queue_cnts[channel->ch_num - 1]) == SUCCESS) {
    /* Enable open circuit detection on selected channel */
    if (ad7606_spi_reg_write(device,
                             AD7606_REG_OPEN_DETECT_ENABLE,
                             (1 << ((channel->ch_num) - 1))) == SUCCESS) {
        /* Monitor the open detect flag for max N+15 (open detect queue count) conversions.
         * Note: In ideal scenario, the open detect flash should be monitored continuously while
         * background N conversions are in progress */
        for (conv_cnts = 0;
             conv_cnts < (open_detect_queue_cnts[channel->ch_num - 1] +
                          AUTO_OPEN_DETECT_QUEUE_EXTRA_CONV_CNT);
             conv_cnts++) {
            if (ad7606_convst(device) == SUCCESS) {
                udelay(100);

                /* Monitor the open detect flag */
                if (ad7606_spi_reg_read(device,
                                        AD7606_REG_OPEN_DETECTED,
                                        &open_detect_flag) == SUCCESS) {
                    open_detect_flag >>= (channel->ch_num - 1);
                    open_detect_flag &= 0x1;

                    rw_status = SUCCESS;
                    if (open_detect_flag) {
                        break;
                    }
                } else {
                    rw_status = FAILURE;
                    break;
                }
            } else {
                rw_status = FAILURE;
                break;
            }
        }
    }
```

MONITORING AND DIAGNOSTICS FEATURES

OVERVOLTAGE (OV)/UNDERVOLTAGE (UV)

Each analog input on the AD7606B, V1 to V8 and V1GND to V8GND, have built in comparator circuitry to monitor for overvoltage and undervoltage events. This circuitry is disabled by default. Any of these comparators can be individually enabled through the AIN_OV_UV_DIAG_ENABLE register (Address 0x25), such that the analog input voltage is monitored and triggers an alert if either the OV threshold or UV threshold is surpassed.

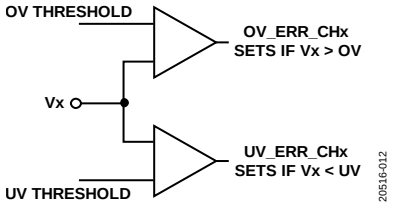

Figure 14. Overvoltage and Undervoltage Circuitry on Each Analog Input

When the circuitry is enabled, when the voltage on any analog input pin goes above the OV threshold shown in the AD7606B data sheet, the AIN_OV_DIAG_ERROR register (Address 0x26) shows which channel or channels have an overvoltage event.

When the voltage on any analog input pin goes below the UV threshold shown in the AD7606B data sheet, the AIN_UV_DIAG_ERROR register (Address 0x27) shows which channel or channels have an undervoltage event.

These monitoring flags allow the user or the controller to rapidly know when an analog input, on a circuit protection application, for example, has gone beyond the full-scale voltage and beyond a voltage level that may represent a potential risk for the rest of the circuitry, allowing the controller to take any measure to help protect the system.

Interface Check

Use the interface check to ensure that the data is properly transmitted from the AD7606B to the digital controller. Enable interface check through Bit 7 on the DIGITAL_DIAG_ENABLE register (Address 0x21), either periodically or when there is an event that may potentially cause errors on the data transmission, to make the AD7606B to clock out fixed data values instead of conversion results. That way, if the data received matches the fixed data sent from the AD7606B, it verifies that the communication is performed without data corruption.

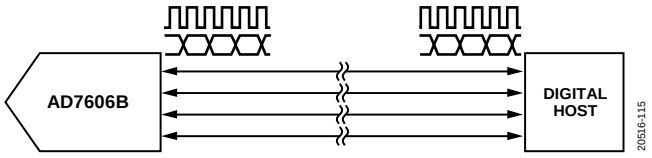

Figure 15. Data Transmission Lines Between AD7606B (Slave) and Host Controller (Master)

For more information on the digital interface check, see the AD7606B data sheet.

SPI Invalid Read/Write

When attempting to read back an invalid register address, the SPI_READ_ERR bit (Address 0x22, Bit 4) is set. The invalid readback address detection can be enabled by setting the SPI_READ_ERR_EN bit (Address 0x21, Bit 4). If an SPI read error is triggered, it is cleared by overwriting that bit or disabling the checker.

When attempting to write to an invalid register address or a read only register, the SPI_WRITE_ERR bit (Address 0x22, Bit 3) is set. The invalid write address detection can be enabled by setting the SPI_WRITE _ERR bit (Address 0x21, Bit 3). If an SPI write error is triggered, it is cleared by overwriting that bit or disabling the checker.

BUSY Stuck High

When enabled, through the DIGITAL_DIAG_ENABLE register (Address 0x21, Bit 5), this monitoring feature allows the AD7606B to alert to the controller if, by any unlikely reason, the AD7606 stopped performing conversions (BUSY line is stuck) by asserting the BUSY_STUCK_HIGH_ERR bit (Address 0x22, Bit 5). If so, the assertion of this bit indicates that a partial reset was automatically issued to recover normal operation.

For more information on the BUSY stuck high monitoring feature, see the AD7606B data sheet.

Diagnostics Multiplexer

All eight input channels contain a diagnostics multiplexer in front of the PGA that allows monitoring of the internal nodes described in Figure 16 to ensure the correct operation of the AD7606B. Figure 16 shows the bit decoding for the diagnostic mux register on Channel 1, as an example. When an internal node is selected, the input voltage at the input pins are deselected from the PGA.

Table 30. Diagnostic Mux Register Bit Decoding of Channel 1

Address 0x28			
Bit 2	**Bit 1**	**Bit 0**	**Signal on Channel 1**
0	0	0	V1
0	0	1	Temperature sensor
0	1	0	V_{REF}
0	1	1	4 × ALDO
1	0	0	4 × DLDO
1	0	1	V_{DRIVE}
1	1	0	AGND
1	1	1	AV_{CC}

Figure 16. Diagnostic Mux Register Bit Decoding of Channel 1

Temperature Sensor

The temperature sensor can be selected through the diagnostic multiplexer and converted with the ADC. The temperature sensor voltage is measured and is proportional to the die temperature, as per the following equation

$$Temperature\,(°C) = \frac{ADC_{OUT}\,(V) - 0.69068\,(V)}{0.019328\,(V/°C)} + 25\,(°C)$$

Typically, the sampled voltage of the temperature sensor is about 0.69 V in room ambient temperature.

The reference, ALDO, DLDO, V_{DRIVE}, AGND, and AVCC can be self-detected in the same way.

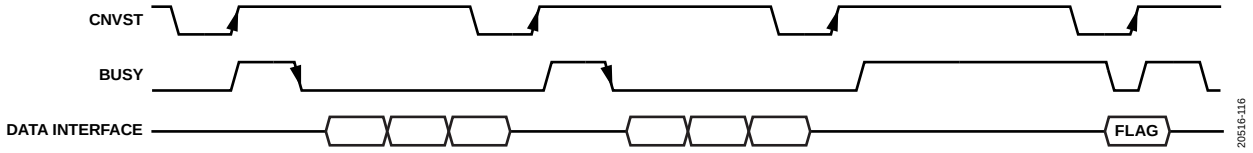

Figure 17. BUSY Stuck High

The following code is used for reading the temperature sensor:

```
ssize_t get_chn_temperature(void *device,
                            char *buf,
                            size_t len,
                            const struct iio_ch_info *channel)
{
        int32_t adc_chn_data = 0;
        float temperature;
        float voltage;

        /* Configure the channel multiplexer to select temperature read */
        if (ad7606_spi_write_mask(device,
                                  AD7606_REG_DIAGNOSTIC_MUX_CH(channel->ch_num - 1),
                                  AD7606_DIAGN_MUX_CH_MSK(channel->ch_num - 1),
                                  AD7606_DIAGN_MUX_CH_VAL((channel->ch_num - 1),
                                                          TEMPERATURE_MUX)) == SUCCESS) {

                /* Allow to settle Mux channel */
                udelay(100);

                /* Sample the channel and read conversion result */
                adc_chn_data = single_data_read(device, channel->ch_num - 1,
                                                attr_polarity_val[channel->ch_num - 1]);

                /* Convert ADC data into equivalent voltage */
                voltage = convert_adc_raw_to_voltage(adc_chn_data,
                                                     attr_scale_val[channel->ch_num - 1]);
```

```
                /* Obtain the temperature using equation specified in device datasheet */
                temperature = ((voltage - 0.69068) / 0.019328) + 25.0;
                return (ssize_t)sprintf(buf, "%f", temperature);
        }

        return -EINVAL;
}
```