Technical Note

008370
Issue 1

ABP3 Board Mount Pressure Sensor
I²C Communication Protocol

I²C BUS configuration

The I²C bus is a simple, serial 8-bit oriented computer bus for efficient I²C (Inter-IC) control. It provides good support for communication between different ICs across short circuit-board distances, such as interfacing microcontrollers with various low speed peripheral devices (source: NXP Semiconductor at https://www.nxp.com/docs/en/user-guide/UM10204.pdf).

Each device connected to the bus is software addressable by a unique address and a simple Master/Sensor relationship that exists all the time. The output stages of devices connected to the bus are designed around an open collector architecture. Because of this, pull-up resistors to $+V_{DD}$ must be provided on the bus. Both SDA and SCL are bidirectional lines, and it is important to system performance to match the capacitive loads on both lines. In addition, in accordance with the I²C specification, the maximum allowable capacitance on either line is 400 pF to ensure reliable edge transitions at 400 kHz clock speeds. When the bus is free, both lines are pulled up to $+V_{DD}$. Data on the I2C bus can be transferred at a rate up to 100 kbit/s in the standard-mode, or up to 400Kbit/s in the fast mode.

I²C data transfer

ABP3 Series I²C sensors are designed to respond to requests from a master device. Following the address and read bit from the master, ABP3 Series digital output pressure sensors are designed to output up to 6 bytes of data – the last two data bytes are the Status Byte (16 bit), the third and the fourth bytes are the compensated pressure output in unsigned format (16 bit) and the first two bytes are the compensated temperature output in unsigned format (16 bit).

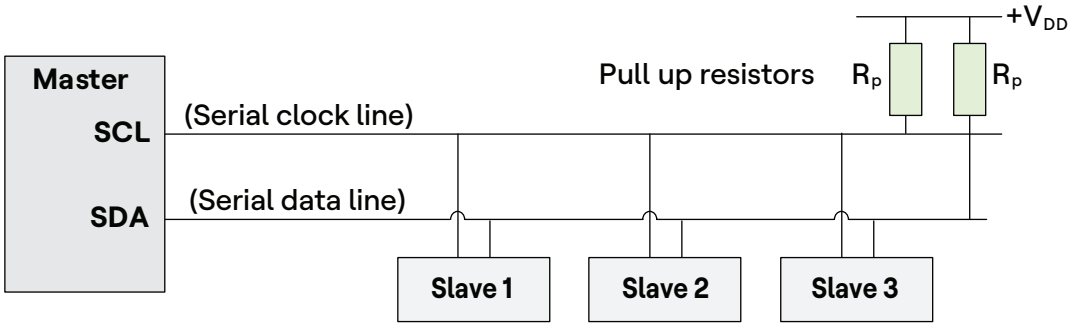
I²C pressure and temperature reading

To read out the compensated pressure and temperature reading, the master generates a START condition[VB3.1] and sends the sensor address followed by a read bit ①. After the sensor generates an acknowledge, it will transmit up to 6 bytes of data. The last two data bytes are the Status Byte (16 bit), the third and fourth bytes are the compensated pressure output in unsigned format (16 bit) and the first two bytes are the compensated temperature output in unsigned format (16 bit). The master must acknowledge the receipt of each byte and can terminate the communication by sending a Not Acknowledge bit followed by a stop bit after receiving the required bytes of data.

Operating modes

The ABP3 Series sensors can be configured at the factory to function in three different operating modes single shot, periodic or continuous mode. ABP3 standard listings are configured to operate in single-shot mode with **a default update rate of 1 KHz (ADC sample rate 4 KHz, average sampling 4x).** Custom listings can be configured to operate in single-shot, periodic or continuous mode with a wide selection of sample rates and filter options, for more information please contact Honeywell.

- Single shot (default mode): Sensor will perform a single measurement and update its output register whenever an external trigger is provided. The device remains active only during data acquisition and then stays in idle mode to minimize power consumption. **The maximum update rate in single shot mode is 4 KHz (ADC sample rate 16 KHz, average sampling 4x).**
- Periodic: Sensor performs periodic measurements based on the sample rate preloaded at the factory. **The maximum update rate in periodic mode is 4 KHz (ADC sample rate 16 KHz, average sampling 4x).**
- Continuous: Sensor performs continuous measurements, based on the sample rate preloaded at the factory, never leaving the power state; maximum update rate is 50 kHz



Honeywell Technologies

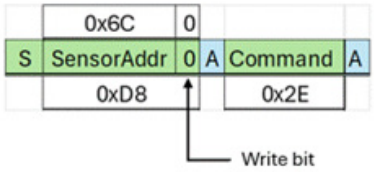
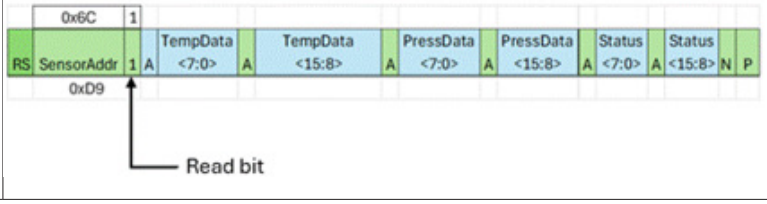
I²C communications | I²C command format

I²C Single-shot model - (standard digital listing) The ABP3 sensor will update its output register whenever the external trigger is provided and with the current setting the max achievable update is 1k samples per second. The device will be on only during the data acquisition time and remains in idle state in rest of the time for saving the power consumption. Note: Reach out to Honeywell for custom update rate from 0.5 K to 50 K along with Periodic Mode and Continuous Mode It is only required to read the necessary bytes of data from the memory addresses to communicate with the ABP3 I²C output sensor in the continuous mode. Please refer to the below diagrams for the stepwise description of the operations required to fetch the required bytes of sensor data.

I²C Output Measurement Command

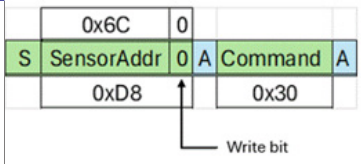
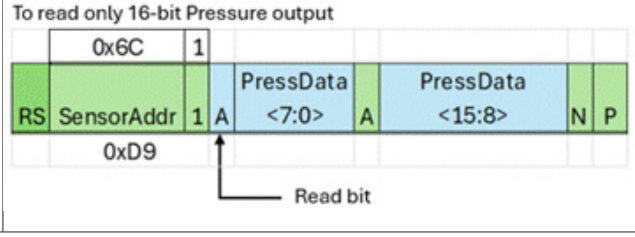
a. Steps to read temperature, pressure, and status output

1. To communicate with ABP3 series I²C output read 16-bit temperature, pressure and status contents the master should request the slave to read 6 bytes of the data from the register address 0×2E.
2. Once the data is received it should be converted to a 16-bit signed format using Equation 1

Step	Action	Notes
1		Master to Sensor Sensor to Master S Start condition P Stop condition A Acknowledge N Not acknowledge RS Repeated start condition
2	To read 16-bit pressure output and 16-bit temperature output along with 16-bit status bytes 	

b. Steps to read only pressure output

1. To read 16-bit pressure output, the master should request the slave to read 2 bytes of the data from the register address 0×30.
2. Once the data is received it should be converted to a 16-bit signed format using Equation 1.

Step	Action	Notes
1		Master to Sensor Sensor to Master S Start condition P Stop condition A Acknowledge N Not acknowledge RS Repeated start condition
2	To read only 16-bit Pressure output 	

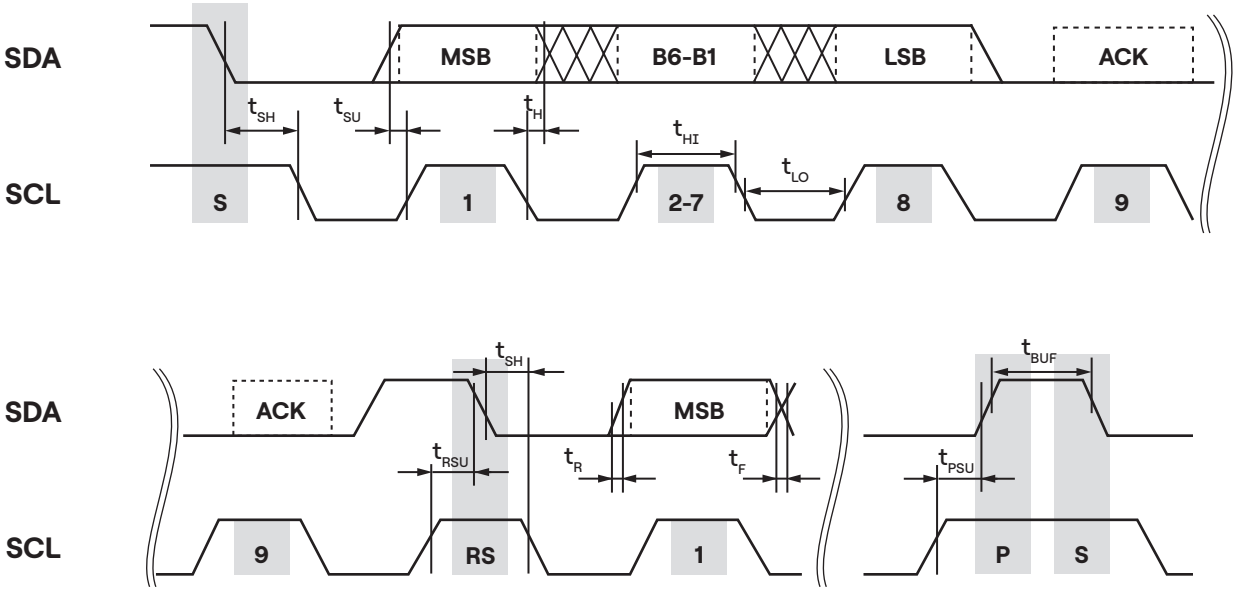
Data acquisition in single shot mode and periodic/ continuous mode

In periodic and continuous modes, the sensor automatically performs measurements at the configured interval without any external trigger. For every update cycle, the sensor acquires raw data, converts it to digital form, and updates the corrected temperature and pressure registers, such as 0×2E and 0×30. In these modes, the host can directly read the required registers using the standard random read method, since the data is always refreshed internally.

In contrast, in single-shot mode the sensor remains in idle state to minimize power consumption. A measurement is performed only when an external trigger is issued by the host. This trigger is generated by writing the START command (0×8B93) into the command register 0×22. Once this command is written, the sensor performs a single measurement, after which the host can wait for the required conversion time and then read the corrected temperature and pressure data from registers 0×2E and 0×30 using the same read mechanism.

	0x28				CMD Reg		Start CMD		
S	Sensor Addr	0	0	ack	0x22	ack	0x93	0x8B	ack

I²C timing parameters



Description and Condition

No.	Description	Condition	Symbol	Min.	Typ.	Max.	Unit
1	I²C clock frequency	CFG_PADS2.i2c_filt_off=1	I2C_F_SCK	–	–	400	kHz
2	I²C clock low time (t_{LO})	CFG_PADS2.i2c_filt_off=1	I2C_T_LO	1300	–	–	ns
3	I²C clock high time (t_{HI})	CFG_PADS2.i2c_filt_off=1	I2C_T_HI	800	–	–	ns
4	I²C (repeated) start condition hold time (t_{SH})	CFG_PADS2.i2c_filt_off=1	I2C_T_SH	600	–	–	ns
5	I²C data setup time (t_{SU})	CFG_PADS2.i2c_filt_off=1	I2C_T_SU	100	–	–	ns
6	I²C data hold time (t_{H})	—	I2C_T_H	0	–	–	ns
7	I²C repeated start setup time (T_{RSU})	CFG_PADS2.i2c_filt_off=1	I2C_T_RSU	600	–	–	ns
8	I²C stop condition setup time (T_{PSU})	CFG_PADS2.i2c_filt_off=1	I2C_T_PSU	600	–	–	ns
9	I²C rise time (t_{R}) [1)]	CFG_PADS2.i2c_filt_off=1	I2C_T_R	–	–	300	ns
10	I²C fall time (t_{F})	—	I2C_T_F	–	–	300	ns
11	I²C bus free time between STOP and START conditions (t_{BUF})	CFG_PADS2.i2c_filt_off=1	I2C_T_BUF	600	–	–	ns

1) Dependent on external bus components

NOTE: The external pullups on the I²C lines can typically be between 1k Ohm and 10k Ohm (To Be Tested and Confirmed).

Status registers

When 6 bytes of data are read after writing the 0×2E register, the last 2 bytes will be the STATUS_SYNC register. These two bytes are copied from the STATUS register.

0×32 (STATUS SYNC)

STATUS_SYNC BYTES EXPLANATION

Bit (Meaning)	Status
15 (dsp_t_missed)	STATUS.dsp_t_missed
14 (dsp_s_missed)	STATUS.dsp_s_missed
13	always 0
12	always 0
11 (com_crc_error)	STATUS.com_crc_error
10 (dsp_sat)	STATUS.dsp_sat
9	always 0
8 (bc_fail)	STATUS.bc_fail
7 (bs_fail)	STATUS.bs_fail
6	always 0
5	always 0
4 (dsp_t_up)	dsp_t_up (Copy of STATUS.dsp_t_up when DSP_T is read.)
3 (dsp_s_up)	dsp_s_up(Copy of STATUS.dsp_s_up when DSP_S is read.)
2 (adc_t_up)	adc_t_up(Copy of STATUS.adc_t_up when ADC_T is read.)
1 (adc_s_up)	adc_s_up(Copy of STATUS.adc_s_up when ADC_S is read.)
0	STATUS.idle

0×36(STATUS SYNC)
STATUS_SYNC BYTES EXPLANATION

Bit (Meaning)	Status
15 (dsp_t_missed)	Will be set to 1 when dsp_t_up was 1 and DSP_T is updated again.
14 (dsp_s_missed)	Will be set to 1 when dsp_s_up was 1 and DSP_S is updated again.
13	always 0
12	always 0
11 (com_crc_error)	Will be set to 1 when communication CRC error occurs.
10 (dsp_sat)	Will be set to 1 when one of the DSP computations leading to the current DSP_T/S values was saturated to prevent overflow
9	always 0
8 (bc_fail)	will be set to 1 when bridge check failure occurs
7 (bs_fail)	will be set to 1 when bridge supply failure occurs
6	always 0
5	always 0
4 (dsp_t_up)	will be set to 1 when DSP_T register is updated with new value. Can be cleared by reading DSP_T.
3 (dsp_s_up)	will be set to 1 when DSP_S register is updated with new value. Can be cleared by reading DSP_S.
2 (adc_t_up)	Will be set to 1 when ADC_T and ADC_TC are updated.
1 (adc_s_up)	Will be set to 1 when ADC_S value is updated.
0 (idle)	1 = chip is in busy state 0 = chip is in idle state

HW_DIAG (0×34)

Bit	Status
15	Confirms validated bridge supply fault
14	Indicates non-volatile memory access failure
[13:0]	Always 0

The bits STATUS_SYNC [15:5,0] are identical to the bits STATUS [15:5,0].

The bits STATUS_SYNC [4:3] are copied from the STATUS register when the corresponding DSP registers are read. First reading the DSP registers and then STATUS_SYNC ensures that both values are consistent to each other.

Data conversion

The above-mentioned pressure and temperature counts are in unsigned 16-bit format. After the two bytes are combined, the resulting word shall be interpreted as a singed 16-bit two's compliment value.

The signed 16-bit numerical range is:

-32768 to +32767

The value **2^{15}=32768** represents the scaling magnitude of the signed Q0.15 output format. It shall not be interpreted as a **2^{15}**-bit ADC resolution.

The user needs to convert the counts to a 16-bit signed format for proper interpretation of the output as per datasheet specifications.

Equation 1: Conversion of the unsigned counts
((Unsigned Counts + 2^{15}) % 2^{16}) - 2^{15}

Example: Calculate the pressure for an -1 psi to 1 psi differential sensor with a 10 % to 90 %FSS calibration when the recorded pressure count from the sensor:

I) 65500 (in decimal): when the unsigned counts > 32767

Signed Counts = ((65500 + 2^{15}) % 2^{16}) - 2^{15}

Signed Counts = -36

Output(%FSS) = (-36+32000)/640 *//Pressure Sensor Transfer function Eqn2*

Output(%FSS) = 49.94

II) 12750 (decimal): When the unsigned counts < 32767

Signed Counts = ((12750 + 2^{15}) % 2^{16}) - 2^{15}

Signed Counts = 12750

Output(%FSS) = (12750+32000)/640 *//Pressure Sensor Transfer function Eqn2*

Output(%FSS) = 69.9

Note: The above conversion of the unsigned counts to signed counts applies for temperature counts as well.

Equation 2: Pressure Sensor Transfer Function
Output(%FSS) = (Signed Counts + 32000)/640

Equation 3: Temperature Output Transfer Function
Output(%FSS) = (Signed Counts + 16485)/387.88

ABP3 Calculations | Pressure output

Pressure output

The ABP3 Series sensor pressure output may be expressed by the transfer function of the device as shown in Equation 2:

Equation 2: Pressure output function

$$Pressure = \frac{(Output - Output_{min}) * (P_{max} - P_{min})}{Output_{max} - Output_{min}} + P_{min}$$

Where,

$Output_{min}$ = output at minimum pressure [counts]
$Output_{max}$ = output at maximum pressure [counts]
Output = digital pressure reading [counts]
P_{max} = Maximum pressure range
P_{min} = Minimum pressure range
Pressure = pressure reading [mBar, Bar, psi, kPa, etc.]

Example
Calculate the pressure for a -2.5 mBar to 2.5 mBar differential sensor with a 10 % to 90 % calibration, and a pressure output of 12750 counts.

The maximum counts +32000(2^{15}) and minimum counts is = -32000(-2^{15}) and the span to be considered (90-10) % = 80 %

$Output_{max}$ = +25600 counts (90 % of 2^{15} counts or 0×6400)

$Output_{min}$ =-25600 counts (10 % of 2^{15} counts or 0×9C00)

Output = 12750 counts

Pressure = pressure in mBar

$$Pressure = \frac{(12750 - (-25600)) \times (2.5 - (-2.5))}{25600 - (-25600)} + (-2.5)$$

$$Pressure = \frac{191750}{51200} - 2.5$$

$$Pressure = 1.24 \text{ mBar}$$

Temperature output

The ABP3 Series temperature output may be expressed by the transfer function as shown in Equation 3:

Equation 3: Temperature output transfer function

$$Temperature = \frac{T_{out} + 16485}{387.88}$$

Where,

Temperature = calculated temperature output in °C

T_{out} = digital temperature output in counts (decimal)

Example

Calculate the temperature in °C for a temperature output of -6783 (decimal) counts:

$$Temperature = \frac{(-6783+16485)}{387.88}$$

$$Temperature = \frac{9702}{387.88}$$

$$Temperature = 25.01°C$$

Demo Arduino code

Below is the reference demo code to communicate and read the data from the sensor using a microcontroller

```
#include <Wire.h>
#define SENSOR_ADDR 0×6C
#define Press_REGISTER_ADDR 0×30
#define SYNC_REGISTER_ADDR 0×32
#define HWD_REGISTER_ADDR 0×34
#define STATUS_REGISTER_ADDR 0×36
#define CMD_REGISTER_ADDR 0×22 #define CMD_START_SIN-
GLESHOT  0×8B93
static void write16(uint8_t reg, uint16_t value)
{
  Wire.beginTransmission(SENSOR_ADDR);
  Wire.write(reg);
  Wire.write((uint8_t)(value & 0xFF));       // LSB first
  Wire.write((uint8_t)((value >> 8) & 0xFF)); // MSB
  Wire.endTransmission(true);
}
void setup() {   Serial.begin(115200);
  Wire.begin();
  Wire.setClock(400000);   delay(1000);
  Serial.println("Starting data acquisition...");
}
void loop() {
    write16(CMD_REGISTER_ADDR, CMD_START_SINGLESHOT);
    int16_t press_value = 0;
    Wire.beginTransmission(SENSOR_ADDR);
    Wire.write(Press_REGISTER_ADDR);
    Wire.endTransmission(false);     Wire.requestFrom(SENSOR_
ADDR, 2);
    if (Wire.available() == 2) {      uint8_t lsb = Wire.read();      uint8_t
msb = Wire.read();
      press_value = (msb ← 8) | lsb;
    } else {      press_value = 0;
    }
    int16_t sync_value = 0;
    Wire.beginTransmission(SENSOR_ADDR);
    Wire.write(SYNC_REGISTER_ADDR);
    Wire.endTransmission(false);     Wire.requestFrom(SENSOR_
ADDR, 2);     if (Wire.available() == 2) {      uint8_t lsb = Wire.read();
uint8_t msb = Wire.read();
      sync_value = (msb ← 8) | lsb;
    } else {      sync_value = 0;
    }
    int16_t hwd_value = 0;

    Wire.beginTransmission(SENSOR_ADDR);
    Wire.write(HWD_REGISTER_ADDR);
    Wire.endTransmission(false);     Wire.requestFrom(SENSOR_
ADDR, 2);
    if (Wire.available() == 2) {      uint8_t lsb = Wire.read();      uint8_t
msb = Wire.read();
      hwd_value = (msb ← 8) | lsb;
    } else {
      hwd_value = 0;
    }
    int16_t status_value = 0;
    Wire.beginTransmission(SENSOR_ADDR);
    Wire.write(STATUS_REGISTER_ADDR);
    Wire.endTransmission(true);     Wire.requestFrom(SENSOR_
ADDR, 2);
    if (Wire.available() == 2) {      uint8_t lsb = Wire.read();      uint8_t
msb = Wire.read();
      status_value = (msb ← 8) | lsb;
    } else {      status_value = 0;
    }
  Serial.print("Pressure Value = ");   Serial.print(press_value);
  Serial.print(" sync_value = 0x");
  Serial.print((uint16_t)sync_value, HEX);
  Serial.print(" hwd_value = 0x");
  Serial.print((uint16_t)hwd_value, HEX);
  Serial.print(" status_value = 0x");
  Serial.print((uint16_t)status_value, HEX);
  Serial.println(" ");   delay(1000);
```

⚠ WARNING
PERSONAL INJURY

DO NOT USE these products as safety or emergency stop devices or in any other application where failure of the product could result in personal injury.

Failure to comply with these instructions could result in death or serious injury.

⚠ WARNING
MISUSE OF DOCUMENTATION

- The information presented in this datasheet is for reference only. Do not use this document as a product installation guide.
- Complete installation, operation, and maintenance information is provided in the instructions supplied with each product.

Failure to comply with these instructions could result in death or serious injury.

Warranty/Remedy

Honeywell Technologies warrants goods of its manufacture as being free of defective materials and faulty workmanship. Honeywell Technologies standard product warranty applies unless agreed to otherwise by Honeywell Technologies in writing; please refer to your order acknowledgement or consult your local sales office for specific warranty details. If warranted goods are returned to Honeywell Technologies during the period of coverage, Honeywell Technologies will repair or replace, at its option, without charge those items it finds defective. **The foregoing is buyer's sole remedy and is in lieu of all other warranties, expressed or implied, including those of merchantability and fitness for a particular purpose. In no event shall Honeywell Technologies be liable for consequential, special, or indirect damages.**

While Honeywell Technologies may provide information or engineering support for its products through Honeywell personnel, literature and website, it is the buyer's sole responsibility to determine the suitability of the Honeywell product(s) for the buyer's requirements.

Specifications may change without notice. The information we supply is believed to be accurate as of this writing. However, Honeywell assumes no responsibility for its use.

For more information

Honeywell Technologies services its customers through a worldwide network of sales offices and distributors. For application assistance, current specifications, pricing, or the nearest Authorized Distributor, visit our website or call:

USA/Canada	+1 302 613 4491
Latin America	+1 305 805 8188
Europe	+44 1344 238258
Japan	+81 (0) 3-6730-7152
Singapore	+65 6355 2828
Greater China	+86 4006396841

Visit our webite
https://automation.honeywell.com/

Honeywell Technologies
830 East Arapaho Road
Richardson, TX 75081
www.honeywell.com

008370-1-EN | 1 | 09/26
© 2026 Honeywell International Inc.