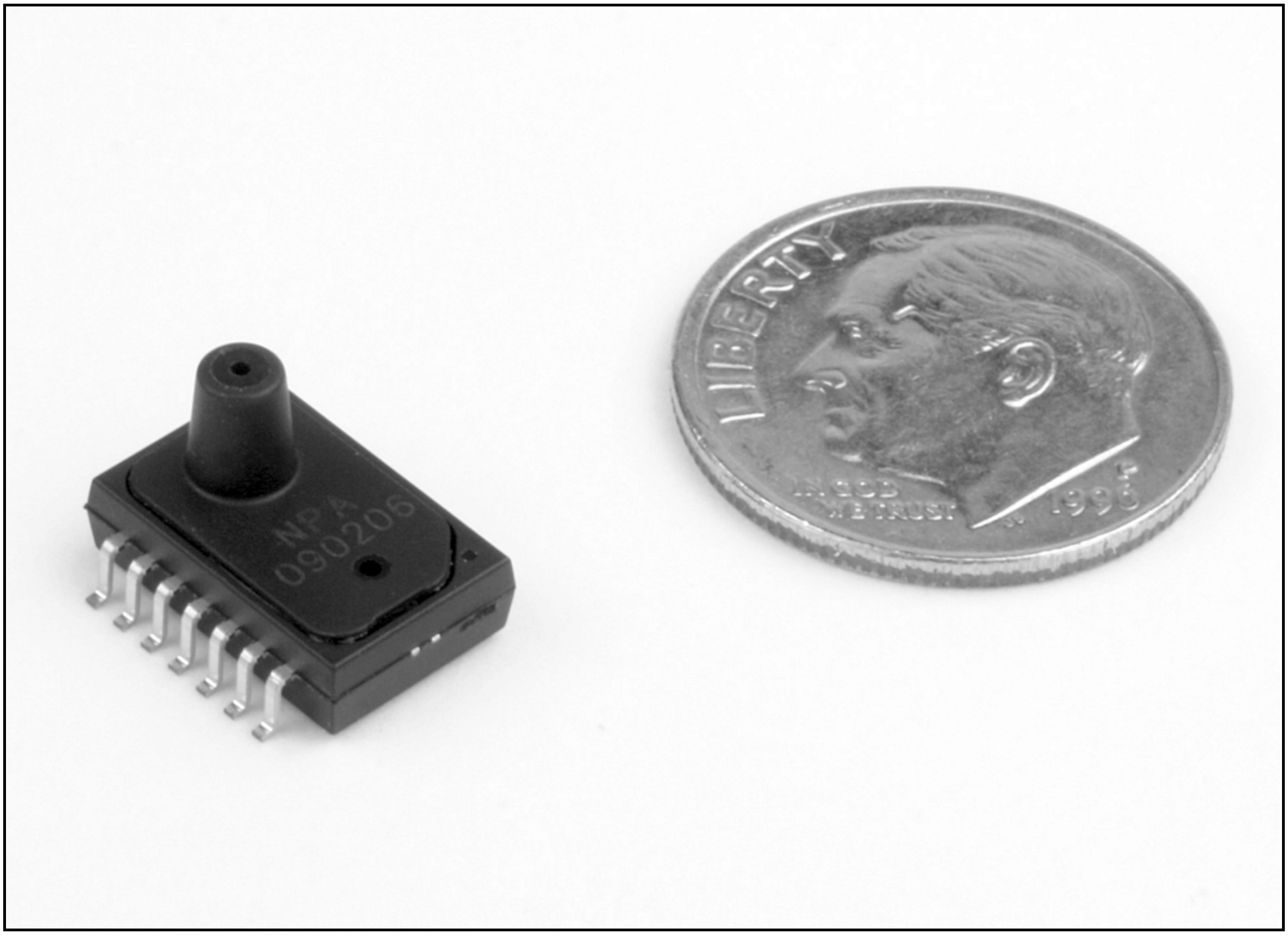
GE
Measurement & Control Solutions

Advanced Sensors

Pressure Sensor Type NPA

Application Guide



imagination at work

910-289 Rev. B
August 2010

1. Typical Application Circuits

1.1 General Description

The NPA series of pressure sensors combines GE's SenStable® silicon fusion bonded pressure die technology with packaged electronics to provide highly stable, amplified and calibrated pressure measurement in a cost effective surface mount package.

The NPA is offered in a range of pressure ratings. Various port configurations are available to measure absolute, gauge and differential pressures. Versions are offered with either analogue or digital outputs.

This application guide should be read in conjunction with product data sheet 920-477.

1.2 Block Diagram

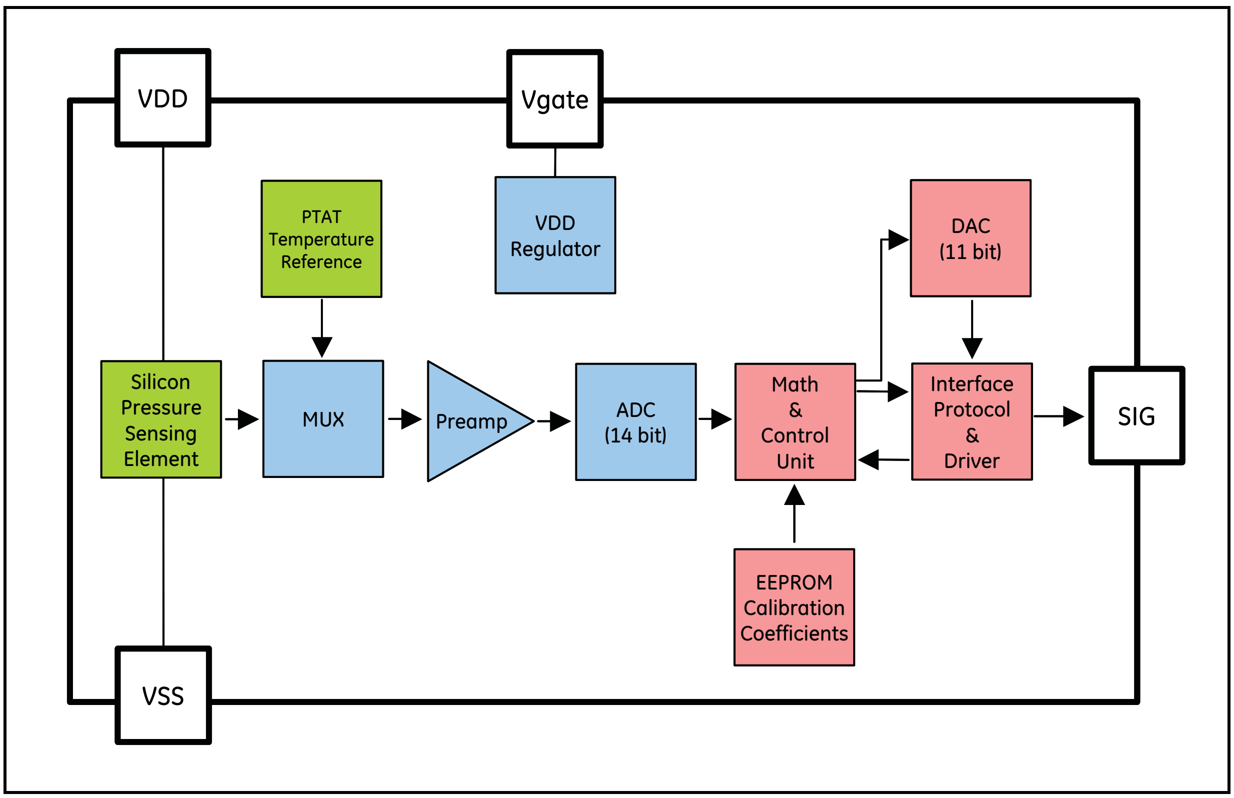


Figure 1: NPA Pressure Sensor Block Diagram

1.3 Analog Circuits

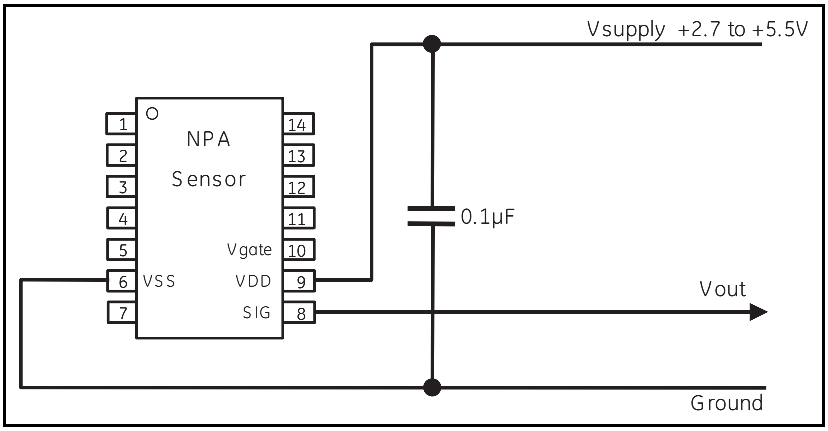


Figure 2: Ratiometric Voltage Output

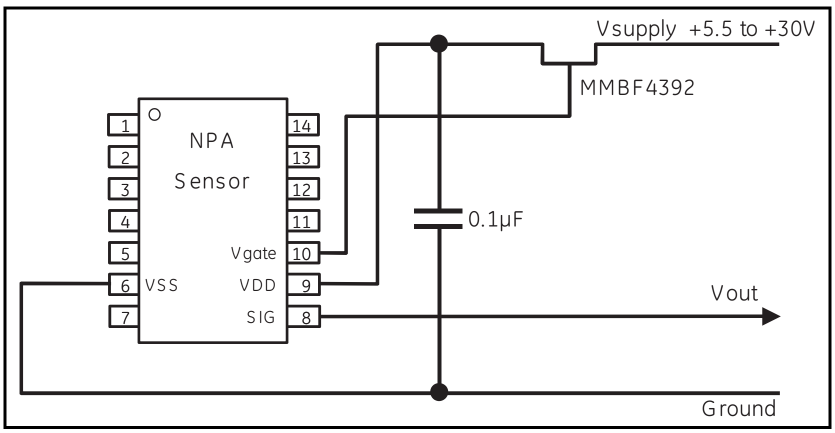


Figure 3: Absolute Voltage Output

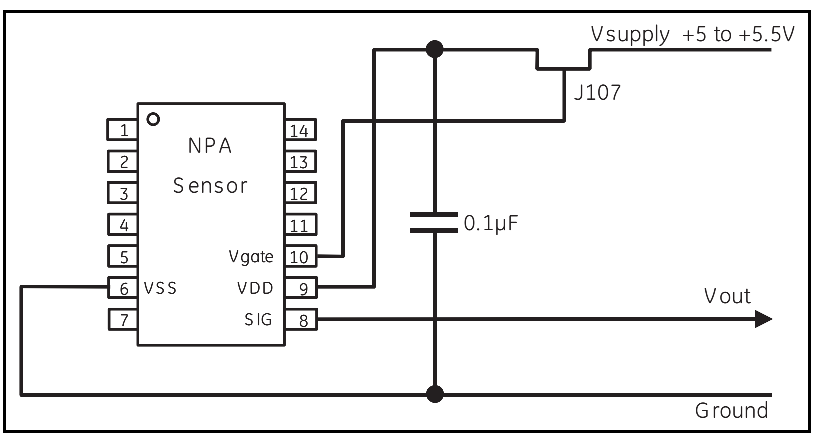


Figure 4: Ratiometric Voltage Output with Overvoltage Protection

Note: Standard analogue versions of NPA are configured for ratiometric operation as shown in Figure 2. For the circuits in Figures 3 and 4, a special version of the NPA with pin 10 (Vgate) activated should be used.
Typical output load resistor R_L (to V_{supply} or Ground) = 10 kΩ (minimum 2.5 kΩ)
Typical output load capacitance C_L = 10 nF (maximum 15 nF)

1.4 Digital Circuits

The circuits in *Figures 2,3 & 4 on Page 2* can also be used for digital outputs with the following conditions:

- The load resistor or capacitance is not required
- No pull-down resistor is allowed
- If a line resistor or pull-up resistor is used, the requirement for rise time (≤5 µs) must be met

1.5 Output Impedance

The source impedance defined by voltage drop due to load current is not specified as such for the NPA. The sensor incorporates a unity gain buffer at the output to control the output voltage within a range of load current / resistances. An error compensation circuit tracks and reduces the amplifier offset voltage to <1 mV.

1.6 Output Short Protection

The NPA does not incorporate short protection. A resistor R_{SP}, as specified in *Table 1* below, should therefore be connected in series with the output.

Table 1: Short Protection Resistor

Ambient Temp. (°C)	Resistor R_{SP} (ohm)
Up to 85	51
Up to 125	100

To minimize the effect on measurement accuracy caused by this resistance in analog mode, the load impedance R_L should be chosen to ensure that $R_L >> R_{SP}$.

2. Pin Connections

Table 2: Pin Connections

Pin No.	Pin Name	Pin Function
1	NC	Not connected
2	NC	Not connected
3	VP	Sensor interface positive output
4	VSSP	Sensor interface ground
5	NC	Not connected
6	VSS	Ground supply
7	NC	Not connected
8	SIG	Analog out, Digital out
9	VDD	Supply voltage
10	Vgate	Gate control for external JFET regulation/overvoltage protection
11	VN	Sensor interface negative output
12	VDDP	Sensor interface positive excitation
13	NC	Not connected
14	NC	Not connected

3. Digital Output

3.1 General

The NPA sensor is available in 2 versions with digital output interface:

- Pressure data only: NPA-600 series
- Pressure & temperature data: NPA-601 series

The bit encoding is similar to Manchester in that clocking information is embedded into the signal (falling edges of the signal occur at regular periods). This allows the protocol to be largely insensitive to baud rate differences between the two ICs communicating. In user applications, the NPA sensor will be transmitting information and another IC in the system (typically a microcontroller) will read the data.

3.2 Bit Encoding

Data bits are transmitted as a Manchester duty-cycle encoded signal:

Start bit	50% duty cycle used to set up strobe time
Logic 1	75% duty cycle
Logic 0	25% duty cycle
Stop Bit	High signal for half a bit width

There is a half stop bit time between bytes in a packet.

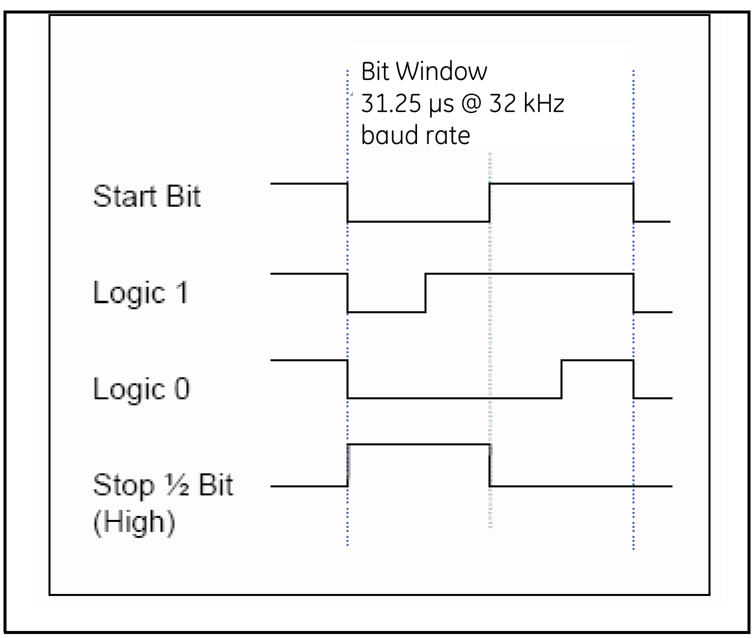


Figure 5: Manchester Duty Cycle

3.2 Bit Encoding (cont.)

An oscilloscope trace of the data transmission demonstrates the bit encoding. Figure 6 below shows a single packet of value 96 hex (= 10010110 bin = 150 dec) being transmitted. Because 96Hex is already even parity, the parity bit P is set to 0.

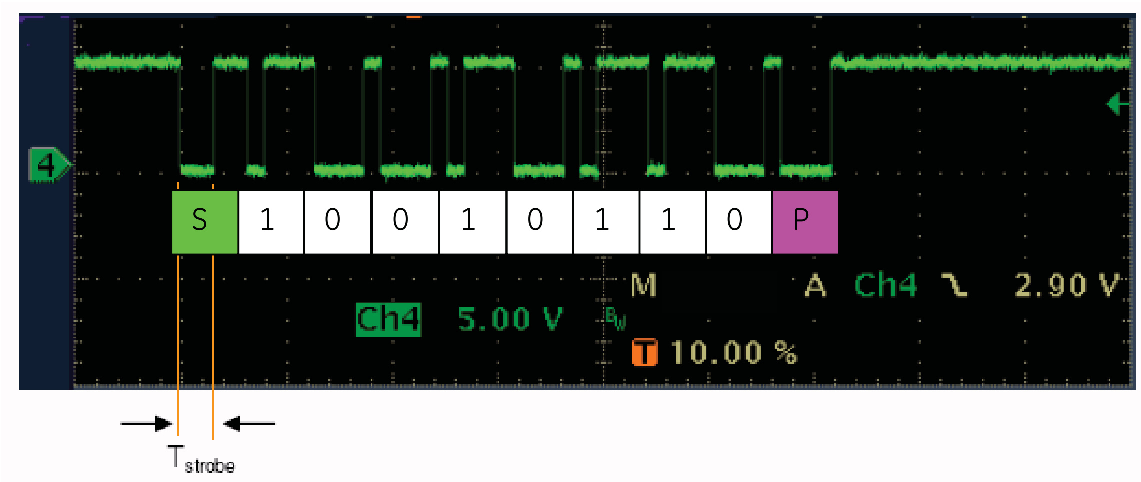


Figure 6: Oscilloscope trace of data packet transmission

3.3 Mode 1 - Pressure Only

The sensor first transmits the high byte of *pressure data*, followed by the low byte. The data resolution is 14-bits, so the upper two bits of the high byte are always zero padded. There is a half stop bit time between bytes in a packet. That means that, for the time of half a bit width, the signal level is high. Combining the high and low data bytes provides a 14-bit number corresponding to the pressure reading (see *Figure 7* below).

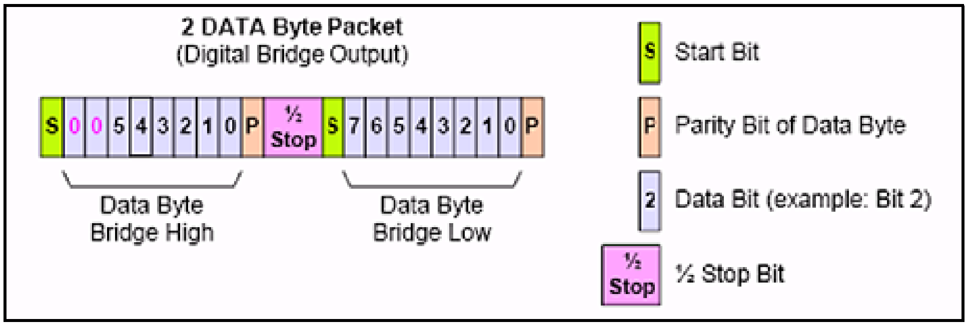


Figure 7: Digital Output Bridge Readings

The actual pressure reading is dependent on the calibration set points for zero and full-scale pressure. This is illustrated in the following example, for a sensor that has been calibrated with the following parameters:

- *Excitation:* 5 Volts
- *Full-scale Pressure:* 10” H_2O
- *Full-scale Pressure:* set to 90% of rail
- *Zero Pressure:* set to 10% of rail.

In this example, the sensor would deliver the analog/digital outputs shown in *Table 4* below.

Table 3: Analog/Digital Outputs for Example

Pressure (in. H_2O)	Analog V	% of Rail Voltage	Digital Value*
-1.25	0.0	0	0
0	0.5	10	1638
5	2.5	50	8192
10	4.5	90	14745
11.25	5.0	100	16383
*this is the 14-bit number referenced in *Figure 7* above, in decimal format			

3.4 Mode 2 - Pressure With Temperature

The second digital output mode is a *digital pressure reading* with a *temperature reading*, which is transmitted as a 3-data-byte packet (see *Figure 8* and *Table 4* below). The temperature byte represents an 8-bit value ranging from -50°C to +150°C.

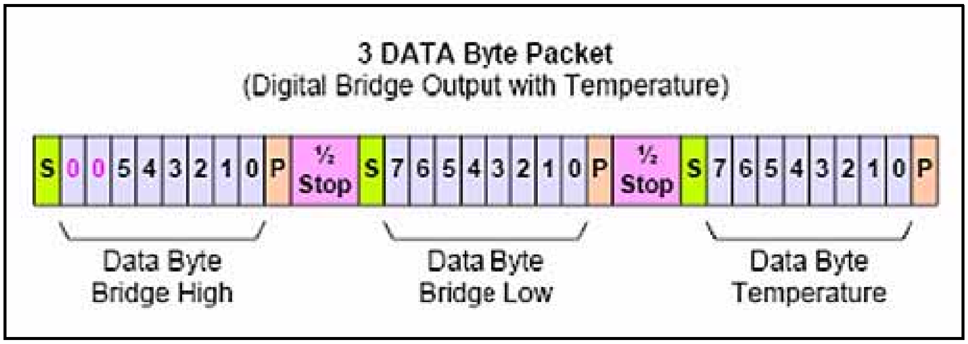


Figure 8: Digital Output Bridge Readings with Temperature

Table 4: Analog/Digital Outputs for Example

T_{actual} (°C)	T_O = 8-Bit Output Temperature
-50	0
-40	13
-30	26
-20	38
-10	51
0	64
10	77
20	89
25	96
30	102
40	115
50	128
60	140
70	153
80	166
90	179
100	191
110	204
120	217
130	230
140	242
150	255

3.5 Data Transmission

Data is transmitted in packets as shown in Figure 9 below.

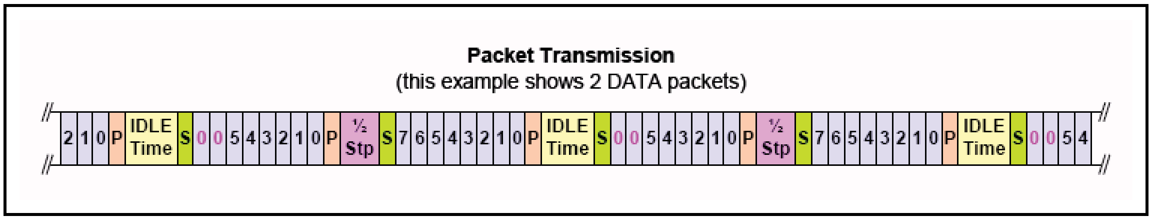


Figure 9: Transmission of data packets (NPA-600 pressure only mode)

The total transmission time for both of digital output modes is shown in *Table 5* below:

Table 5: Data Transmission Times

Baud Rate	Bit Length	Idle Time*	Transmission Time (Pressure Only)		Transmission Time (Pressure & Temperature)	
			# Bits	Total Time (inc. Idle)	# Bits	Total Time (inc. Idle)
32 kHz	31.25 µs	1.0 ms	20.5	1.64 ms	31.0	1.97 ms
*The idle time between packets can vary by a nominal ±15% between parts, and over a temperature range of -50 to +150°C.						

3.6 How to Read a Data Packet

When the falling edge of the start bit occurs, measure the time until the rising edge of the start bit. This time (Tstrobe) is the strobe time. When the next falling edge occurs, wait for time period equal to Tstrobe, and then sample the signal. The data present on the signal at this time will be the bit being transmitted. Because every bit starts with a falling edge, the sampling window is reset with every bit transmission. This means errors will not accrue for bits downstream from the start bit, as it would with a protocol like RS232. It is recommended that when acquiring the start bit, the sampling rate should be at least 16x the nominal baud rate i.e. 16 x 32 kHz = 512 kHz.

3.7 How to Read a Data Packet Using a Microcontroller

It is best to connect the digital signal to a pin on the microcontroller that is capable of causing an interrupt on a falling edge. When the falling edge of the start bit occurs, it causes the microcontroller to branch to its ISR. The ISR enters a counting loop incrementing a memory location (Tstrobe) until it sees a rise on the digital signal. When Tstrobe has been acquired, the ISR can simply wait for the next nine falling edges (8-data, 1-parity). After each falling edge, it will wait for Tstrobe to expire and then sample the next bit.

The digital line is driven by a strong CMOS push/pull driver. The parity bit is intended for error checking when the digital signal is driving long (>2m) interconnects to the microcontroller in a noisy environment. For systems in environments without noise interference, the user can choose to have the microcontroller ignore the parity bit.

3.8 PIC1 Assembly Code Example

The following code reads the digital transmission of pressure data (2 bytes). It is assumed that the NPA digital output is connected to the interrupt pin (PORTB,0) of the PIC and that the interrupt is configured for falling edge interruption. This code would work for a PIC running between 3 and 20 MHz.

```
Pressure_high EQU 0x24          ;; memory location reserved for pressure high byte
Pressure_low EQU 0x25           ;; memory location reserved for pressure low byte
                                ;; this byte must be consecutive from Pressure_high
LAST_LOC EQU 0x26               ;; this byte must be consecutive from Pressure_low
Tstrobe EQU 0x26                ;; location to store start bit strobe time.
ORG 0x004                       ;; ISR location

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Code to save any required state and to determine the source of the ISR goes here.   ;;
;; After the source is determined, if the interrupt was a transmission from NPA then    ;;
;; branch to NPA_TX.                                                                    ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

NPA_TX:      movlw Pressure_high      ; move address of Pressure_high (0x24) to W reg
             movwf FSR                ; FSR = indirect pointer, pointing to Pressure_high

GET_Plow:    movlw 0x02               ; Start Tstrobe counter at 02 to account for
             movwf Tstrobe            ; overhead in getting to this point of ISR
             clrf INDF                ; clear the memory location pointed to by FSR

STRB:        incf Tstrobe,1           ; Increment Tstrobe
             btfsc STATUS,Z           ; if Tstrobe overflowed to zero then
             goto RTI                 ; something is wrong and return from interrupt
             btfss PORTB,0            ; look for rise in NPA signal
             goto STRB                ; if rise has not yet happened increment Tstrobe
             clrf bit_cnt             ; memory location used as bit counter

BIT_LOOP:    clrf strb_cnt            ; memory location used as strobe counter
             clrf time_out            ; memory location used for edge time out

WAIT_FALL:   btfss PORTB,0            ; wait for fall in NPA signal
             goto PAUSE_STRB          ; next falling edge occurred
             incfsz time_out,1        ; check if edge time out counter overflowed
             goto RTI                 ; edge time out occurred.
             goto WAIT_FALL

PAUSE_STRB:  incf strb_cnt,1          ;; increment the strobe counter
             movf Tstrobe,0           ;; move Tstrobe to W reg
             subwf strb_cnt,0         ;; compare strb_cnt to Tstrobe
             btfss STATUS,Z           ;; If equal then it is time to strobe
             goto PAUSE_STRB          ;; NPA signal for data, otherwise keep counting
                                      ;; Length of this loop is 6 states. This must
                                      ;; match length of the loop that acquired Tstrobe
```

```
            bcf STATUS,C            ;; clear the carry
            btfsc PORTB,0           ;; sample the NPA signal
            bsf STATUS,C            ;; if signal was high then set the carry
            rlf INDF,1              ;; rotate carry=NPA into LSB of register
                                    ;; that FSR currently points to
            clrf time_out           ;; clear the edge timeout counter

WAIT_RISE:  btfsc PORTB,0           ;; if rise has occurred then done
            goto NEXT_BIT
            incfsz time_out,1       ;; increment the edge time out counter
            goto WAIT_RISE
            goto RTI                ;; edge time out occurred.

NEXT_BIT:   incf bit_cnt,1          ;; increment bit counter
            movlw 0x08              ;; there are 8 bits of data
            subwf bit_cnt,0         ;; test if bit counter at limit
            btfss STATUS,Z          ;; if not zero then get next bit
            goto BIT_LOOP
            clrf time_out           ;; clear the edge time out counter

WAIT_PF:    btfss PORTB,0           ;; wait for fall of parity
            goto P_RISE
            incfsz time_out,1       ;; increment time_out counter
            goto WAIT_PF
            goto RTI                ;; edge timeout occurred

P_RISE:     clrf time_out           ;; clear the edge time out counter

WAIT_PR:    btfsc PORTB,0           ;; wait for rise of parity
            goto NEXT_BYTE
            incfsz time_out,1       ;; increment edge time out counter
            goto WAIT_PR
            goto RTI                ;; Edge time out occurred

NEXT_BYTE:  incf FSR,1              ;; increment the INDF pointer
            movlw LAST_LOC
            subwf FSR,0             ;; compare FSR to LAST_LOC
            btfss STATUS,Z          ;; if equal then done
            goto WAIT_Plow

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; If here, then done reading the NPA signal and have the data ;;
;; in Pressure_high & Pressure_low
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

WAIT_Plow:  clrf time_out
```

```
WAIT_PLF:     btfss PORTB,0                  ; wait for fall of PORTB,0 indicating
              goto GET_Plow                  ; start of pressure low byte
              incfsz time_out
              goto WAIT_PLF
              goto RTI                       ; edge timeout occurred

RTI:          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
              ;; Restore any state saved at beginning of ISR ;;
              ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
              bcf INTCON,INTF                ;; clear interrupt flag
              bsf INTCON,INTE                ;; ensure interrupt re-enabled
              retfie                         ;; return from interrupt
```

3.9 8051 C++ Code Example

The following code reads the digital transmission of pressure data (2 bytes). The code also includes functionality to switch the sensor power to reduce consumption in battery-powered applications.

It is assumed that the NPA digital output is connected to the PORT 0 (0x80hex) of the 8051 microcontroller. This code is for a microcontroller running at 24.5 MHz. However, frequencies from 8 to 24.5 MHz can also be used, in which case the number of nop (No Operation) commands in the wait routine should be adjusted accordingly.

```
Hi#define PWR_PIN 0x40
#define SIG_PIN 0x80
#define PORT P2

/****************************************************************************
* FUNCTION MACROS
****************************************************************************/

#define NPA_INIT()

{

        SFRPAGE = CONFIG_PAGE;
        PORT_CONFIG |= PWR_PIN;
        PORT &= ~PWR_PIN; /* power */\
        PORT_CONFIG &= ~SIG_PIN;
        PORT |= SIG_PIN; /* signal */

}

#define NPA_ON()     SFRPAGE = CONFIG_PAGE;     PORT |= PWR_PIN;

#define NPA_OFF()     SFRPAGE = CONFIG_PAGE;     PORT &= ~PWR_PIN;

#define NPA_SIGNAL()     (PORT & SIG_PIN)

/****************************************************************************
* Blocking wait function
* Assuming MCU runs at 24.5MHz, 1 nop = 1/(24.5MHz ÷ 8) µs = ~0.33µs
* Number of nops for 15 µs = 15 ÷ 0.33 = 45
****************************************************************************/

#define WAIT_15_US()
_nop_(); _nop_(); _nop_(); _nop_(); _nop_(); _nop_(); _nop_(); _nop_() _nop_();
_nop_(); _nop_(); _nop_(); _nop_(); _nop_(); _nop_(); _nop_(); _nop_() _nop_();
_nop_(); _nop_(); _nop_(); _nop_(); _nop_(); _nop_(); _nop_(); _nop_() _nop_();
_nop_(); _nop_(); _nop_(); _nop_(); _nop_(); _nop_(); _nop_(); _nop_() _nop_();
_nop_(); _nop_(); _nop_(); _nop_(); _nop_(); _nop_(); _nop_(); _nop_() _nop_();
```

```
/*****************************************************************************
* Function : getNPAPressure
* Description : reads from the NPA its output value
* Parameters : pointer for return value
* Returns : read value
*****************************************************************************/

UINT16 getNPAPressure (UINT16 *Pressure_value16)
{
        UINT16 Pressure_value1 = 0;
        UINT16 Pressure_value2 = 0;
        UINT8 i;
        UINT16 Pressure;
        UINT8 parity;
        NPA_ON();
        sleep(200); // wait for stabilization
        SFRPAGE = CONFIG_PAGE;
        while (NPA_SIGNAL()); // wait until start bit starts
        // wait, TStrobe
        while (NPA_SIGNAL() == 0x00);
        // first data byte
        // read 8 data bits and 1 parity bit
        for (i = 0; i < 9; i++)
        {
            while (NPA_SIGNAL()); // wait for falling edge
            WAIT_15_US();
        if (NPA_SIGNAL())
        Pressure_value1 |= 1 << (8-i); // get the bit
        else
        while (NPA_SIGNAL() == 0x00); // wait until line comes high again

}
// second byte
while (NPA_SIGNAL());
// wait, TStrobe
while (NPA_SIGNAL() == 0x00);
// read 8 data bits and 1 parity bit
for (i = 0; i < 9; i++)
{
        while (NPA_SIGNAL()); // wait for falling edge
        WAIT_15_US();
        if (NPA_SIGNAL())
        Pressure_value2 |= 1 << (8-i); // get the bit
        else
        while (NPA_SIGNAL() == 0x00); // wait until line comes high again
}\
NPA_OFF(); // switch NPA off
// check parity for byte 1
parity = 0;
```

```
for (i = 0; i < 9; i++)
if (Pressure_value1 & (1 << i))
parity++;
if (parity % 2)
return FALSE;
// check parity for byte 2
parity = 0;
for (i = 0; i < 9; i++)
if (Pressure_value2 & (1 << i))
parity++;
if (parity % 2)
return FALSE;
Pressure_value1 >>= 1; // delete parity bit
Pressure_value2 >>= 1; // delete parity bit
Pressure = (Pressure_value1 << 8) | Pressure_value2;
*Pressure_value16 = Pressure;
return TRUE; // parity is OK
}

/******************************************************************************
* Function : cmdGetNPAValue
* Description : converts digital pressure value to pressure in real units
* Parameters : none
* Returns : none
* Notes : none
******************************************************************************/
void cmdGetNPAValue (void)

{
        UINT16 Pressure_value;
        float Pressure_float;
        float pressure_min = -1.25; // pressure corresponding to output value 0 dec
        float pressure_max = 11.25; // pressure corresponding to output value 16383 dec
        // values dependant on specific sensor rating
        // values shown are for sensor with full range of -1.25 to +11.25 inH2O
        printf("cmdGetNPAValue\n");
        NPA_INIT(); // init the I/O pins used for the NPA
        NPA_OFF(); // switch the NPA off until use
        if (getNPAPressure(&Pressure_value))
        {
            Pressure_float = ((float)Pressure_value * (pressure_max - pressure_min) / 16383
        + pressure_min; // conversion to real pressure units
            SFRPAGE_UART();
            printf("Pressure %u, %2.1f\n", Pressure_value, Pressure_float);
        }
}
```

4. Package Dimensions

4.1 Manifold Fitting

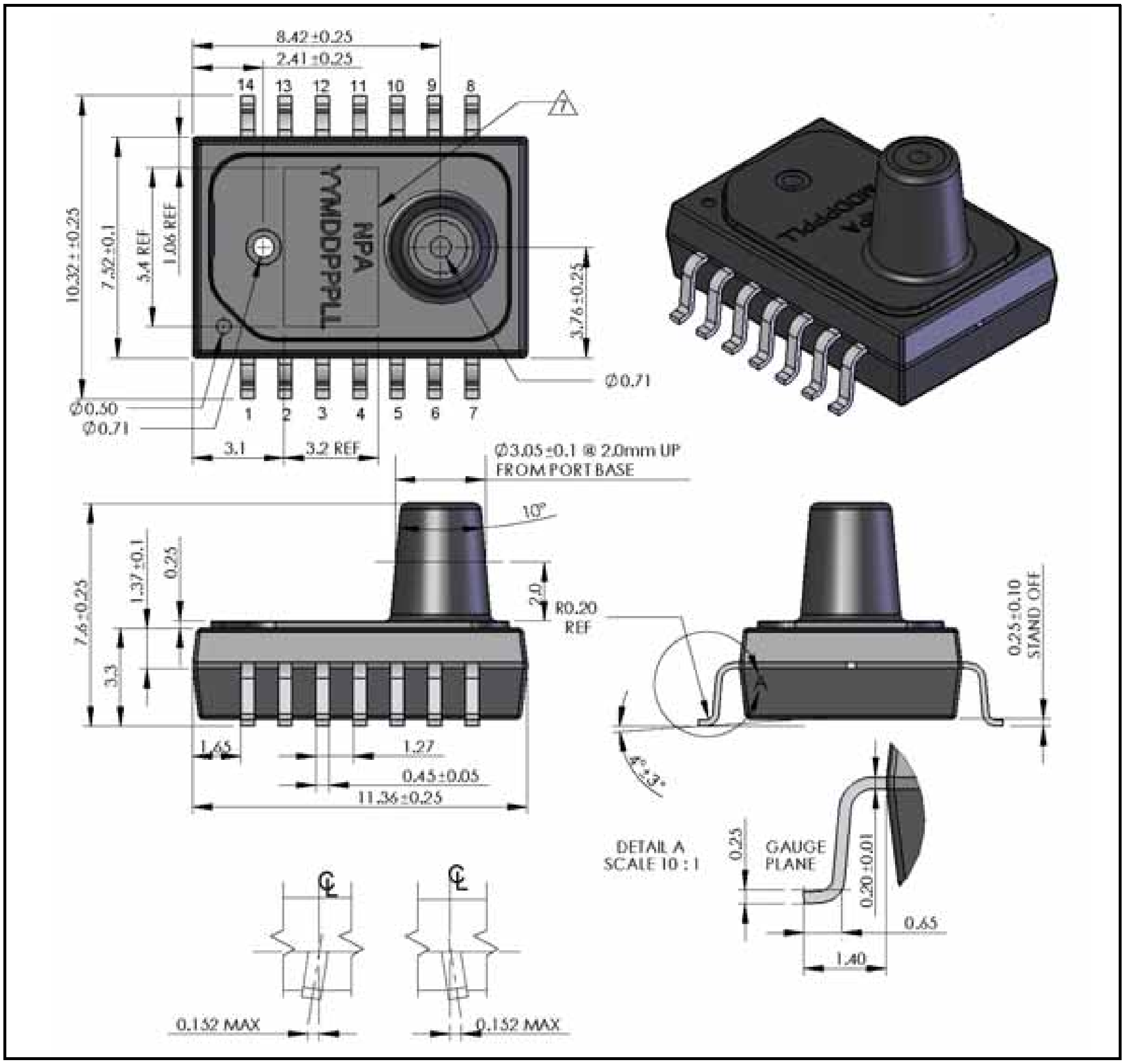


Figure 10: Package Dimensions (mm) - Manifold Fitting

4.2 Barbed Fitting

The recommended tube size for the barbed fitting package shown in *Figure 11* below is:

- 3/32” I.D. x 7/32” O.D. (1/16” Wall Thickness)

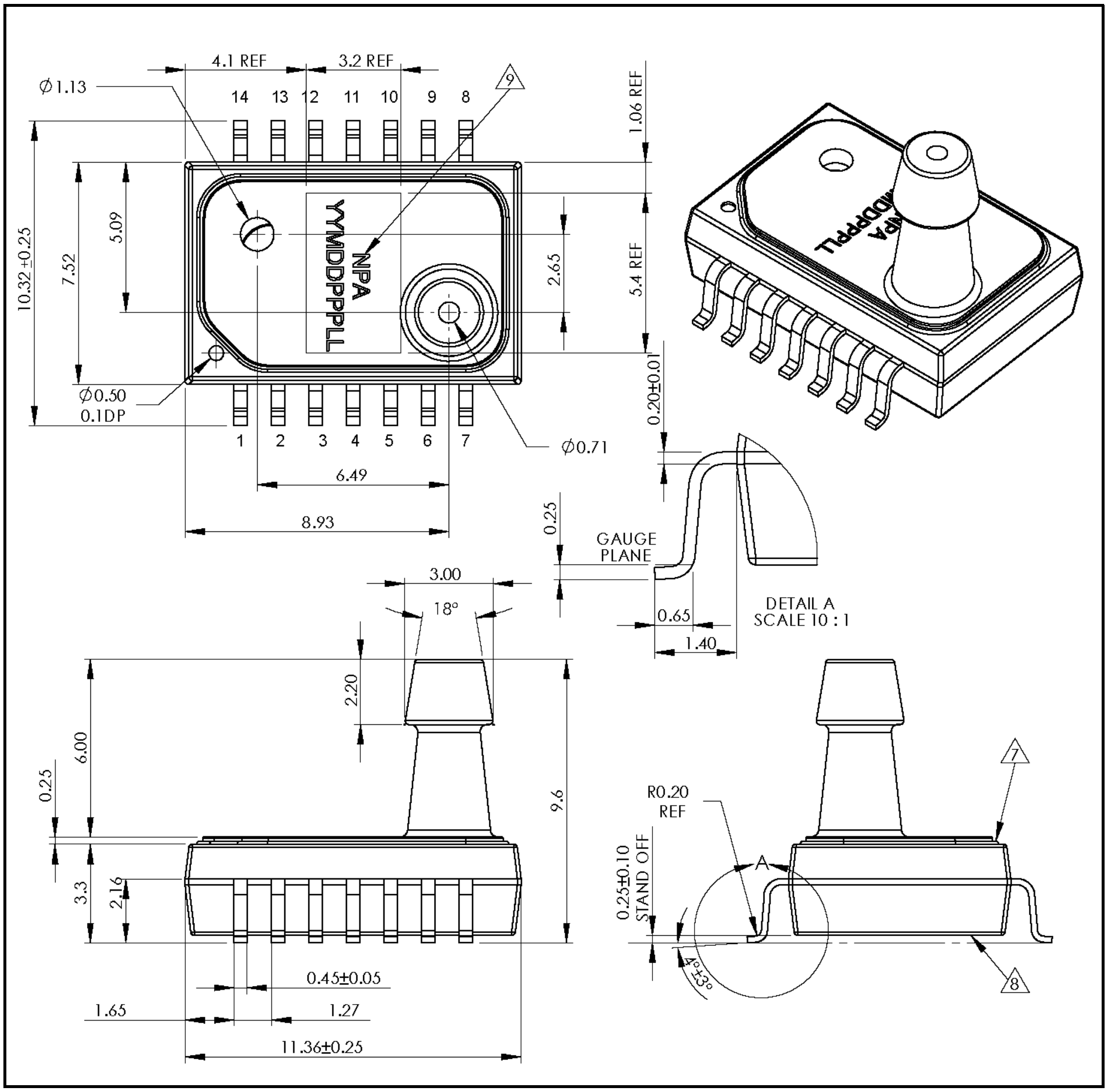


Figure 11: Package Dimensions (mm) - Barbed Fitting, 1 Port

4.2 Barbed Fitting (cont.)

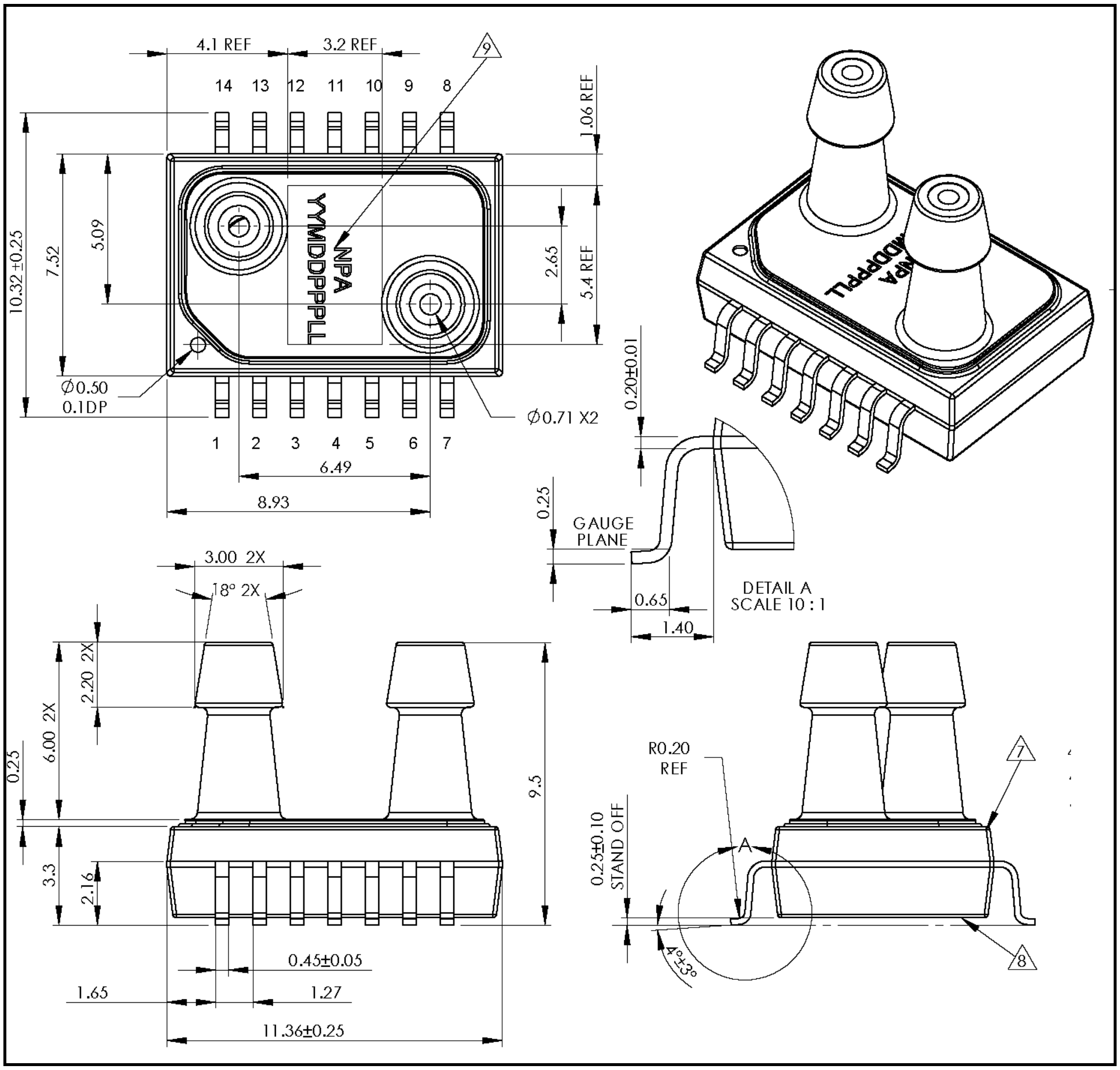


Figure 12: Package Dimensions (mm) - Barbed Fitting, 2 Port

4.3 No Port

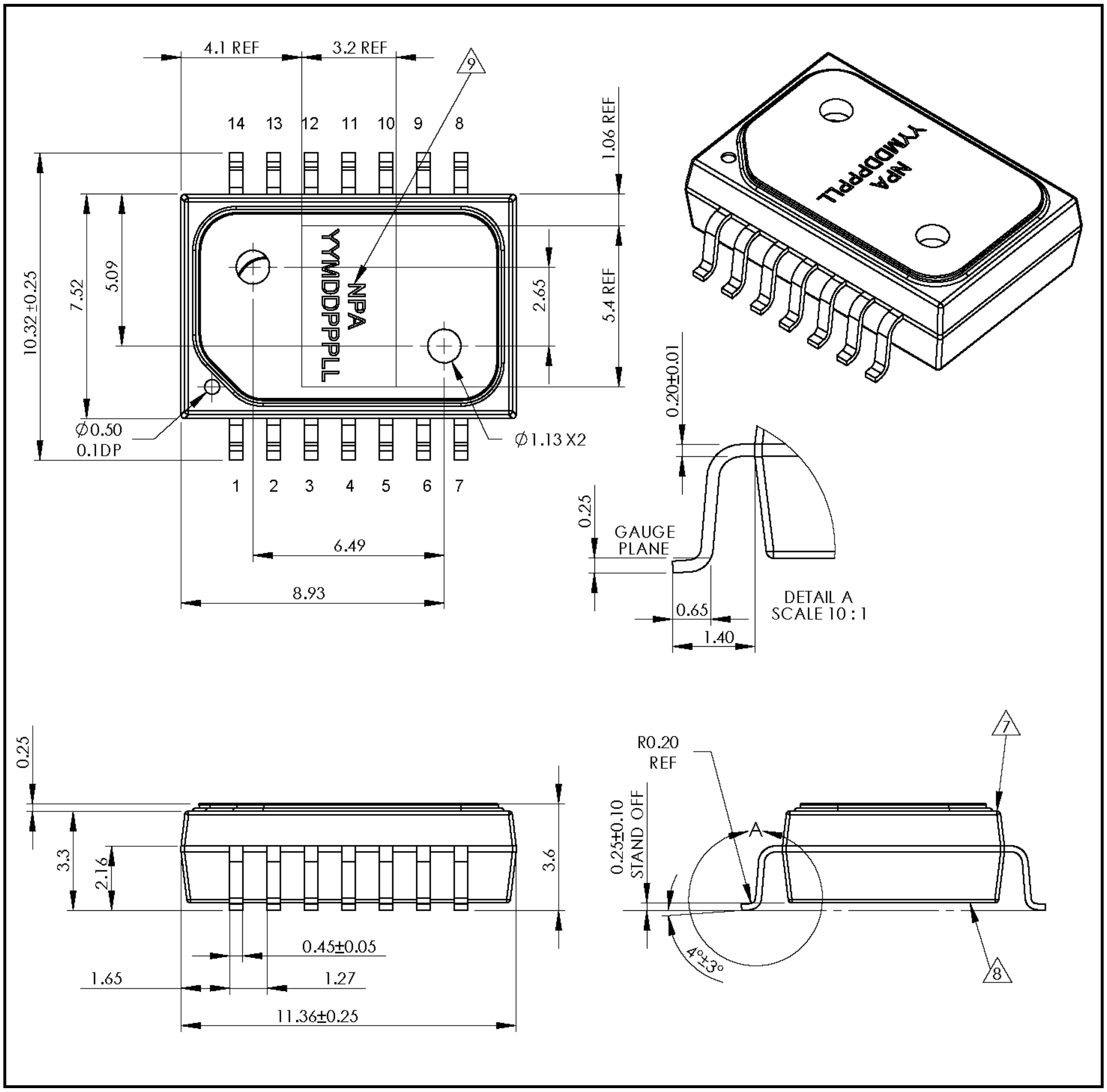


Figure 13: Package Dimensions (mm) - No Port

5. Suggested PCB Pad Layout

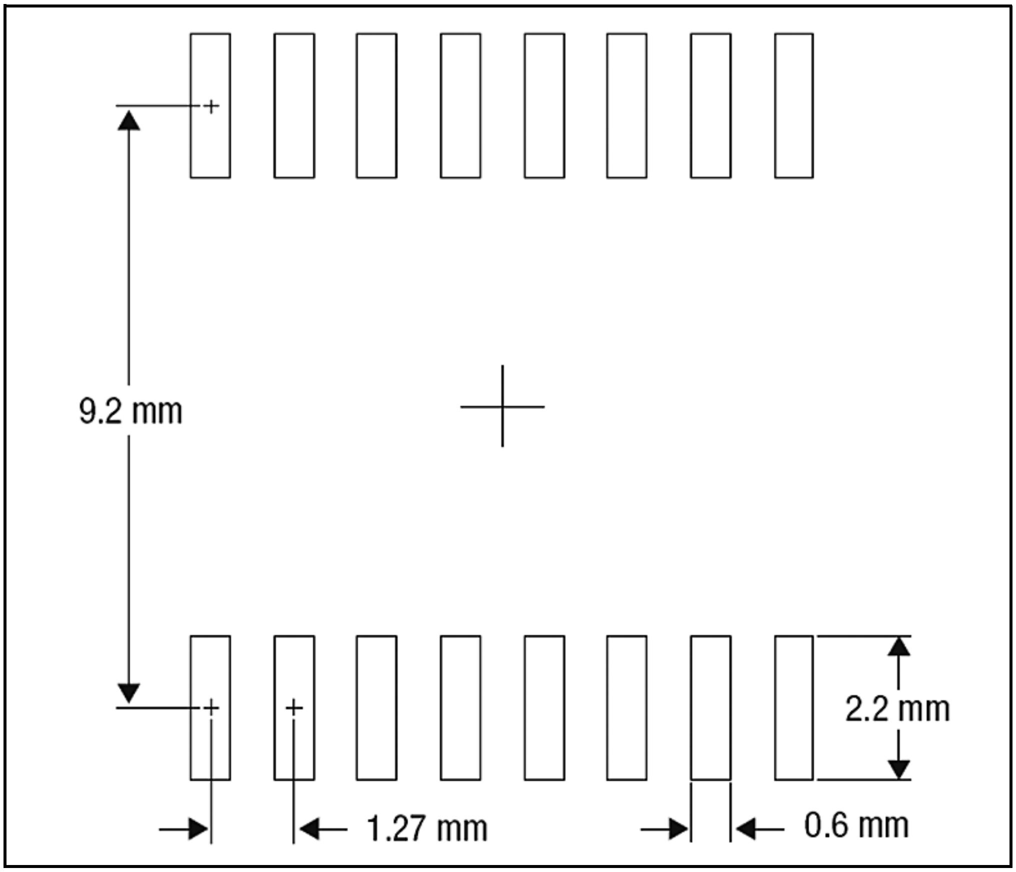


Figure 14: Suggested PCB Pad Layout (SOIC14 Wide Package)

6. Soldering

6.1 Standard Reflow Soldering

The NPA sensor can be soldered using standard reflow ovens (including lead-free soldering) with the following conditions:

- *Maximum temperature:* 250°C for 30 seconds
- *Solder paste:* Use “No-Clean” solder paste only
- *PCB cleaning:* Do not clean or wash circuit boards after soldering

The NPA is classified as *moisture sensitivity level (MSL) 6*, as defined in *Jedec standard J-STD-20*. Product is supplied on carrier tape/reels sealed in moisture-proof bags. Bags are labelled with guidelines on thermal conditioning prior to reflow soldering. Users should follow these instructions in conjunction with *Jedec specification J-STD-033*. Under normal factory conditions parts must be mounted within 8 hours of opening the bag. If parts are not mounted within this time, the taped parts should be removed from the reel and baked at 100°C for 24 hours.

6.2 Manual Soldering

- *Contact time:* Limited to 5 seconds at up to 350°C

Customer Support Centers

U.S.A.

GE Sensing & Inspection Technologies
1055 Mission Court
Fremont, CA 94539
U.S.A.
Tel: 510-661-6000
E-mail: sensing@ge.com

United Kingdom

GE Sensing & Inspection Technologies
Crown Industrial Estates
Priorswood Road
Taunton, Somerset TA2 8QY
United Kingdom
Tel: 44 (0) 1823 348311

China

GE Sensing & Inspection Technologies
5F, Building 1, No.1 Huatuo Road,
Zhangjiang High-Tech Park,
Shanghai 201203
China
Tel.: +86 800 915 9966

Japan

GE Sensing Japan Ltd,.
Shuwatsukishima Bld.
4-6-13 Tsukishima Chuo-ku, Tokyo,
Japan 104-0052
Tel.: +81-3-3551-8711
Fax: +81-3-3551-8721

Korea

GE Sensing & Inspection Technologies
837-2 Hansan-Ri
Cheongbuk-Myeon
PyongtaekShi
Kyunggi-Do
Korea 451-833
Tel: 82-31-680-0610

Singapore

GE Sensing & Inspection Technologies
240 Tanjong Pagar Road
GE Tower, #06-00
Postal code 88540
Singapore.
Tel: +65 6326 3513

www.gesensinginspection.com