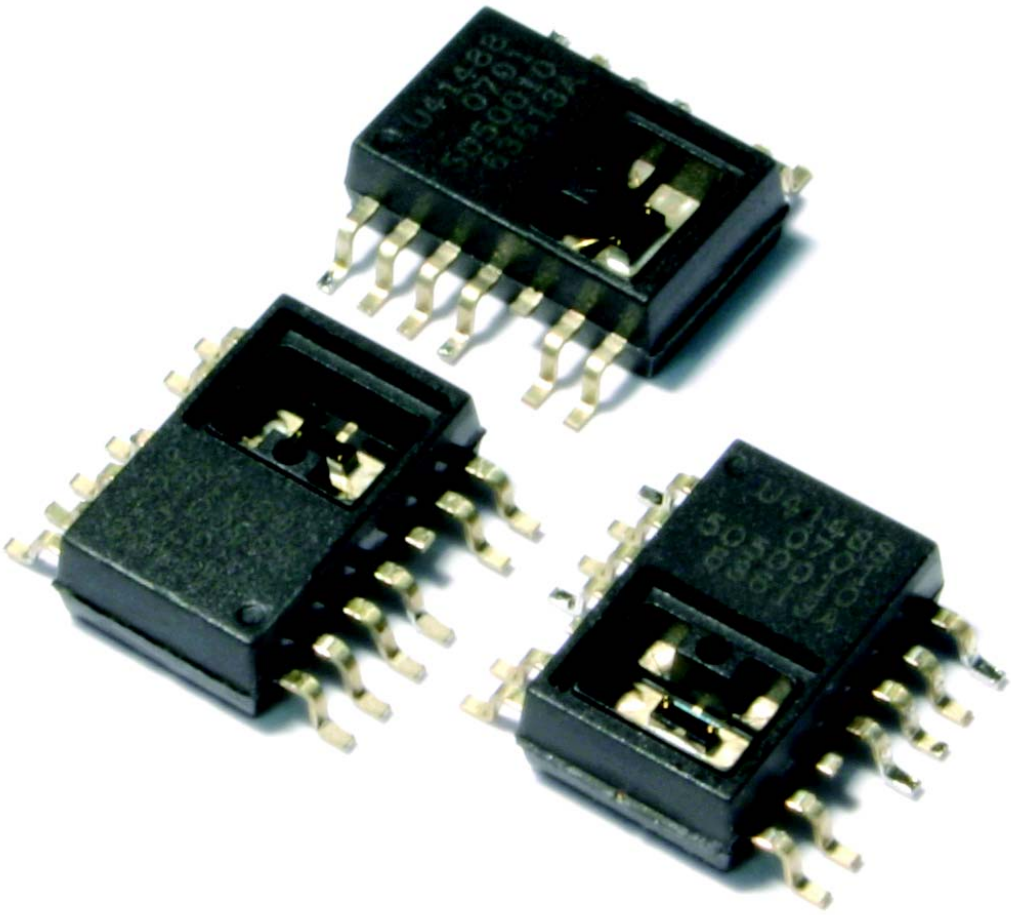
GE

Sensing & Inspection Technologies

Humidity

ChipCap

Application Notes



imagination at work

916-108 Rev. C
April 2009

GE
Sensing & Inspection Technologies

ChipCap

Humidity and Temperature Sensor

Application Notes

916-108 Rev. C
April 2009

GESensingInspection.com

[no content intended for this page - proceed to next page]

1. General Description

The ChipCap series humidity sensor by GE offers a new standard in the field of accurate relative humidity measurement. Based on a capacitive polymer sensing technology, this device offers signal conditioning and temperature compensation for a single SoC (System-on-Chip) solution. The measurement is accurate to ±2% from 20% to 80% RH and ±3% across the entire humidity range at 25°C. The temperature accuracy is ±1°C from 0°C to +70°C. ChipCap provides either analog or digital interfaces on a single, 5 VDC-powered chip. Dual outputs provide humidity and temperature as linear (0 V to 1 V), or ratiometric (10-90% of VDD), or with digital output (the ZACwire one-wire interface).

ChipCap relative humidity sensors change capacitance in direct proportion to ambient relative humidity. An internal solid-state band gap provides temperature measurement.

2. Specifications

2.1 Relative Humidity

RH Sensor: Planar Capacitive Polymer

RH Range: 0 to 100% RH

RH Accuracy @ 25°C: ±2% from 20% to 80%
±3% from 0% to 20% and 80% to 100%

RH Resolution: 0.4% RH

2.2 Temperature

Temperature Sensor: Integral band gap PTAT

Temperature Scale: –50°C to 150°C

Temperature Accuracy: ±0.6°C at 25°C (see Figure 1 on page 2)

Temperature Resolution: 0.2°C

2.3 Environmental

Storage Temperature: –55°C to 150°C

Operating Temperature: –40°C to 85°C

Operating RH Range: 0 to 100% RH, non-condensing

2.3 Environmental (cont.)

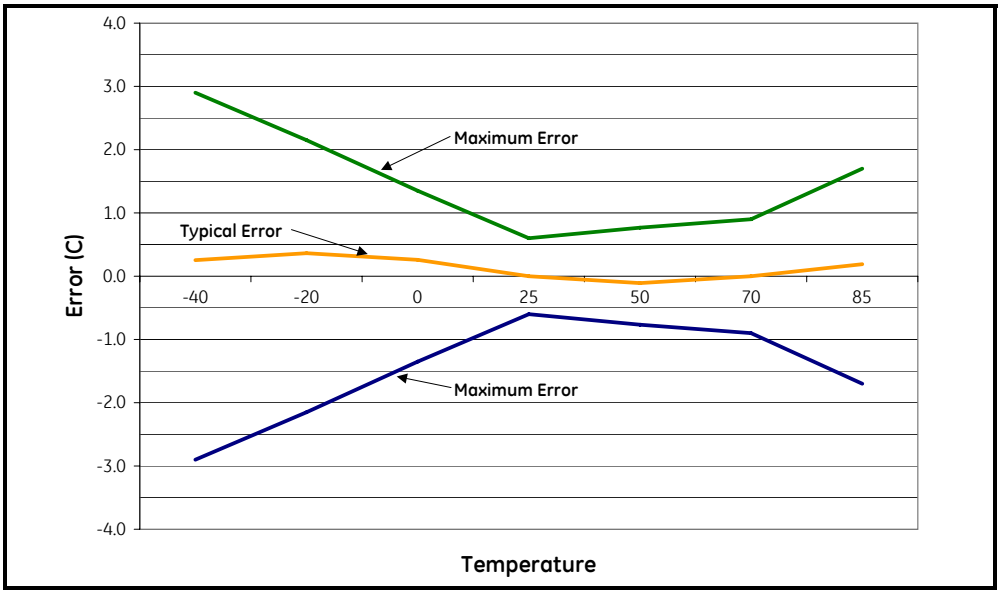


Figure 1: Temperature Accuracy Graph

2.4 Power Supply

Reverse Polarity Protection: 12.5 VDC, 100µA (15 VDC, 60 s)

Voltage Supply: 3 to 5.5 VDC

Current Supply: 500 microamps @ 5 VDC, 25°C

2.5 Outputs

RH Voltage Output (Ratiometric):

$$V_{out} = \frac{V_{supply}}{5} \times [0.5 + (0.04 \times RH)];$$

0.5 to 4.5V ratiometric; 5 VDC nom.

RH Voltage Output (Linear, 0-1V): $V_{out} = 0.01(RH)$

RH Digital Output: Manchester 8-bit encoded

Temperature Voltage Output (Ratiometric):

$$V_{out} = \frac{V_{supply}}{5} \times [1.5 + (0.02 \times T°C)];$$

0.5 to 4.5V ratiometric; 5 VDC nom.

Temperature Voltage Output (Linear): $V_{out} = (0.005T°C) + 0.25$

Temperature Digital Output: Manchester 10-bit encoded

Table 1: Ordering Information

Model	Part Description
CC-L	ChipCap RH and Temperature Sensor; 0-1VDC Linear Output
CC-R	ChipCap RH and Temperature Sensor, 0.5-4.5VDC Ratiometirc Output (5VDC Nominal)
CC-D	ChipCap RH and Temperature Sensor Digital Output

Table 2: Absolute Maximum Ratings

Symbol	Parameter	Min.	Max.	Units	Notes
Vdd	DC Supply Voltage	–0.3	6	V	
Vio	Voltage at all Analog and Digital I/O pins	–0.3	Vdd + 0.3	V	
Ta	Ambient Temperature (operation)	–50 –40	150 85	°C °C	ASIC RH Sensor
Tstrg	Storage Temperature	–55	150	°C	
Tj	Junction Temperature	–55	160	°C	

Table 3: Recommended Operation Conditions

Symbol	Parameter	Min.	Typ.	Max.	Units	Notes
Vdd	Analog DC Supply	3	5	5.5	V	
Idd	Supply Current		550		µA	25°C
Vss	Analog Ground		0.0		V	
Ta	Ambient Temperature	–40	27	85	°C	
Cvdd	External Capacitance between Vdd and Vss	100	220	470	nF	
C_{LD}	Digital Output Load (only capacitive, no resistive)			100	pF	
C_{LA}	Analog Output Load Capacitance			5	nF	
R_{LA}	Analog Output Load Resistance	5			kΩ	

Table 4: Line Driver (Digital/Calibration Mode)

Characteristic/Parameter	Min	Typical	Max	Unit	Notes
V_{oh}	V_{DD} -0.6			V	
V_{ol}			0.6	V	Depends on reverse polarity
Drive Capability Digital			100	pF	Output should see only a capacitive load. No resistive load

2.6 Packaging/Layout

Packaging: SOP-14, SMD

Soldering: IR Solder Reflow, 260°C, 10 sec.

Pin Connection: 4 Pin; +V, Gnd, RH_{out}, T_{out}

PIN Number	Name	Pin Description
1	T_{out}	Temperature output pin - Outputs 0-1 V/0.5 to 4.5 VDD analog temperature or a digital temperature
2	NC	No connection
3	NC	No connection
4	GND	Power/Signal common
5	NC	No connection
6	NC	No connection
7	NC	No connection
8	NC	No connection
9	NC	No connection
10	NC	No connection
11	VDD	ASIC power, +3 to 5.5 VDC
12	NC	No connection
13	NC	No connection
14	RH_{out}	RH output pin - Outputs 0-1 V/0.5 to 4.5 of VDD analog RH or a digital RH

2.6 Packaging/Layout (cont.)

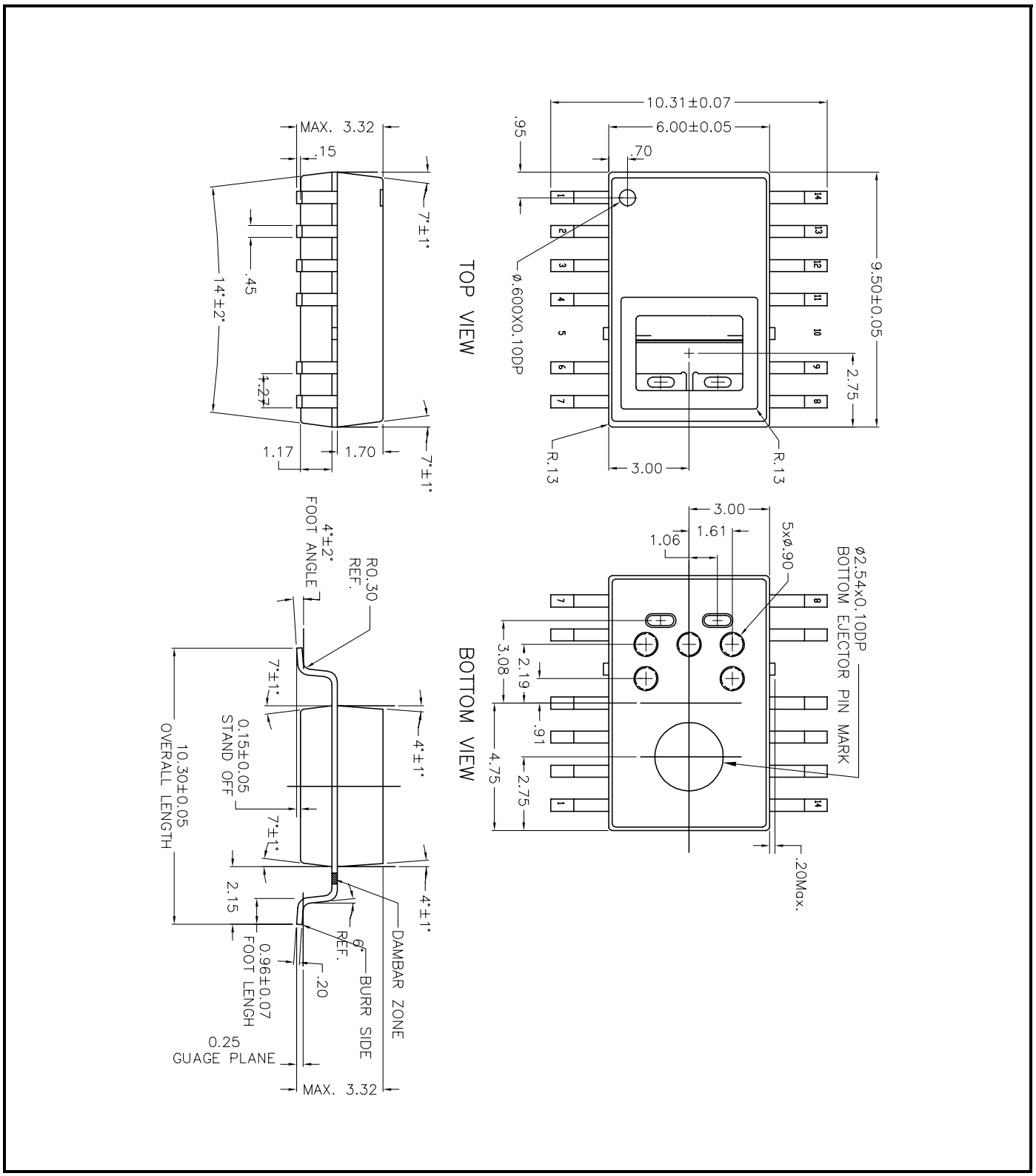


Figure 2: ChipCap Series Humidity Sensor Outline (ref. dwg #17SPP14101A)

2.7 Footprint

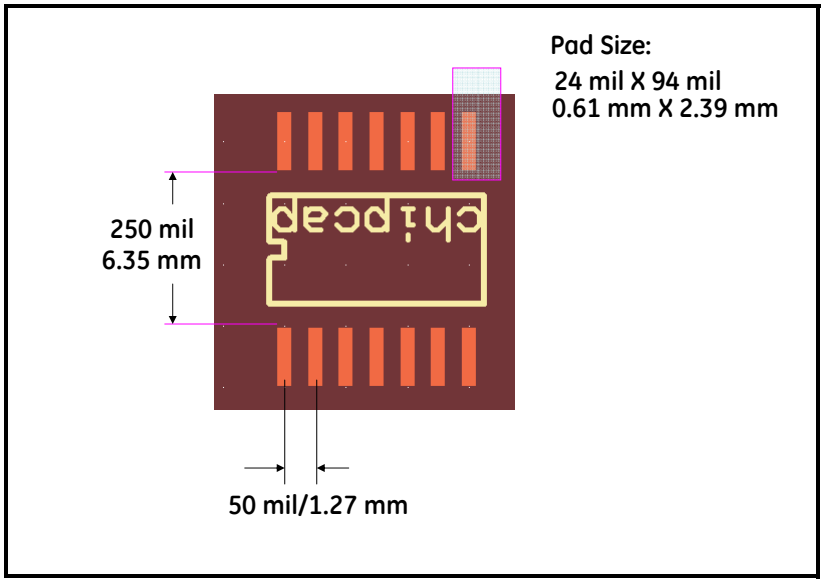


Figure 3: ChipCap Footprint

2.8 Block Diagram

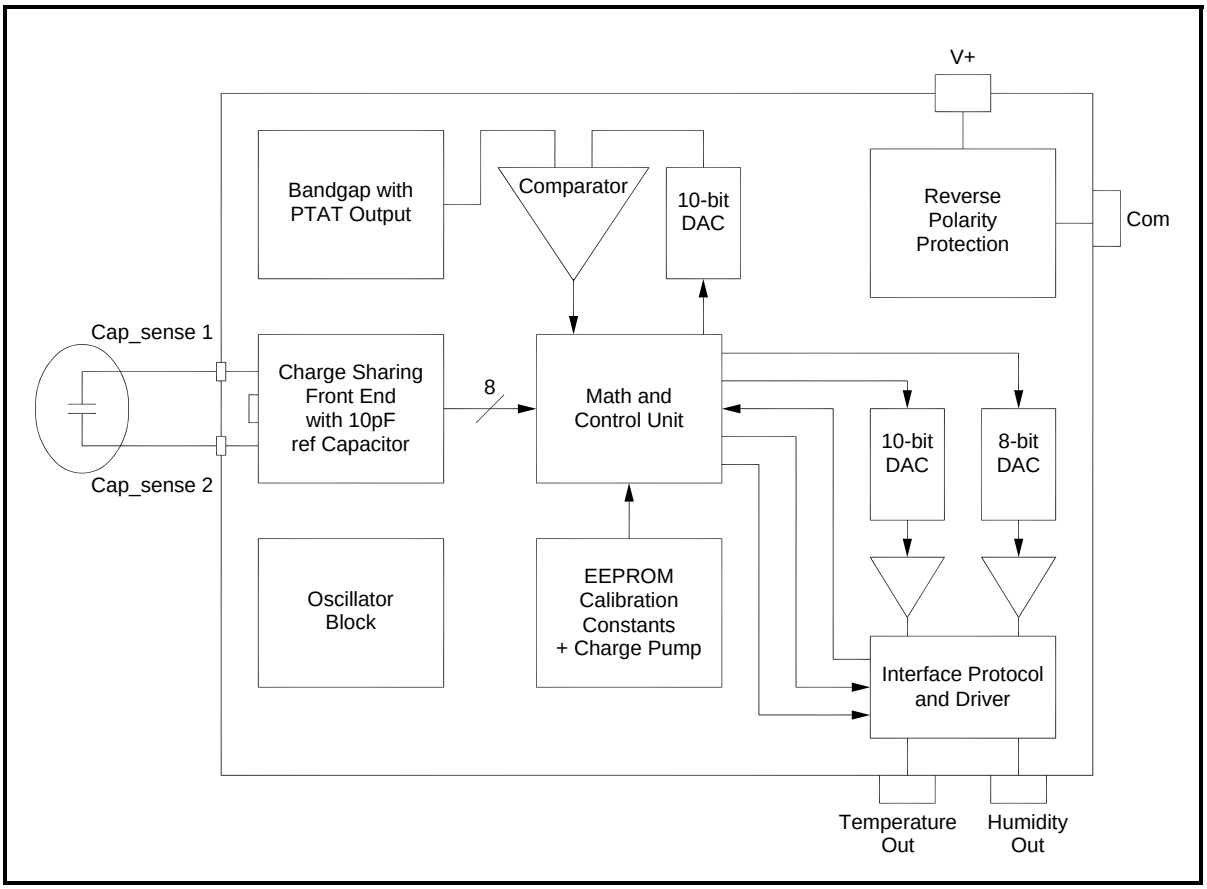


Figure 4: Block Diagram

3. Linear Mode

The linear device provides both relative humidity (RH) and temperature outputs in a linear 0-1 VDC format. The RH channel provides a 0–1 VDC output, which corresponds to 0–100% RH (e.g. 0.5 VDC = 50% RH). The temperature channel provides a 0-1 VDC output, which corresponds to –50° to +150°C (e.g. 0.5 VDC = 50°C).

$\%RH = V_{out} \times 100$

$T_c = V_{out} \times 200 - 50$

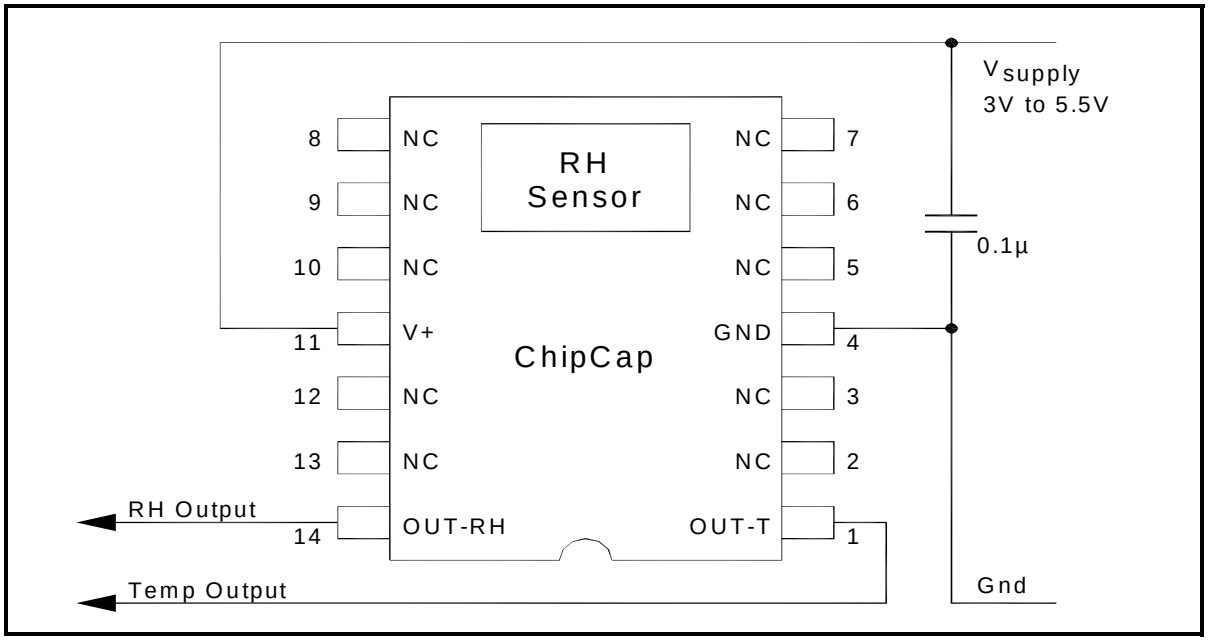


Figure 5: Analog Interface - Linear and Ratiometric Modes

4. Ratiometric Mode

The ratiometric device provides both relative humidity (RH) and temperature outputs in a ratiometric format, corresponding to 10-90% of VDD. For example, using a 5.0 VDC power supply, the RH channel provides a 0.5 to 4.5 VDC output, which corresponds to 0-100% RH (e.g. 2.5 VDC = 50% RH). The temperature channel provides a 0.5 to 4.5 VDC output, which corresponds to –50 to +150°C (e.g. 2.5 VDC = 50°C).

$$\%RH = \frac{V_{out} - (0.1 \times V_{supply})}{0.8 \times V_{supply}} \times 100$$

$$T_C = \frac{V_{out} - (0.1 \times V_{supply})}{0.8 \times V_{supply}} \times 200 - 50$$

5. Digital Mode

5.1 ChipCap ZACwire Communication Protocol

The ZACwire output is a single wire bi-directional communication protocol. The bit encoding is similar to Manchester in that clocking information is embedded into the signal (falling edges of the signal happen at regular periods). This allows the protocol to be largely insensitive to baud rate differences between the two ICs communicating. In end-user applications, the ChipCap sensor will be transmitting humidity and temperature information and another IC in the system (most likely a µController) will be reading the data over the ZACwire output.

Even though the sigRH pin is the bidirectional ZACwire for communication, in digital output mode the RH data is sent to the SigRH (OUT-RH) pin and the Temperature data is sent to the SigT(OUT-T) pin always.In the case that a single wire communication is desired, the two pins can be externally ANDed.
In normal operation:

SigT pin ==> Outputs 2-bytes of temperature info (10-bits worth of info contained in the 2-bytes with the MSByte first)

SigRH pin ==> Outputs 1-byte of humidity info (8-bits)

Temperature is broadcast first, followed immediately by humidity.

5.2 ChipCap Temperature Transmission Packet

The ChipCap sensor transmits 1-byte or 2-byte data packets. 1-byte packets consist of a start bit, 8 data bits, and a parity bit. The nominal baud rate is 8 kHz (125µsec bit window). The signal is normally high. When a transmission occurs, the start bit occurs first followed by data bits (MSB first, LSB last). The packet ends with an even parity bit.

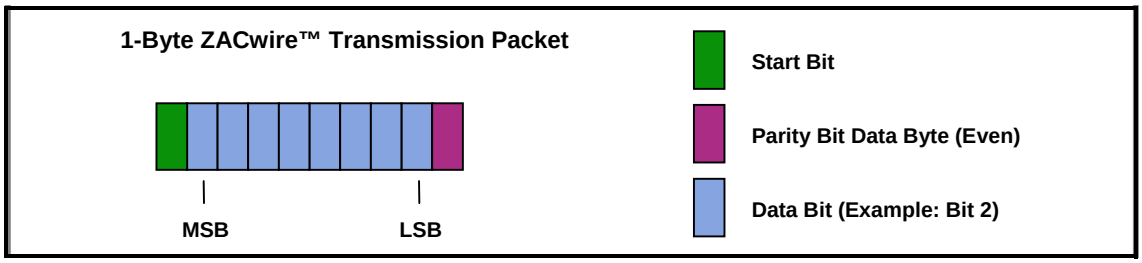


Figure 6: ZACwire Transmission Packet

On the temperature channel, the sensor provides temperature data with 10-bit resolution, which cannot be conveyed in a single byte. A complete temperature transmission from the sensor consists of two bytes. The first byte contains the most significant 2 bits of temperature information, and the second byte contains the least significant 8 bits of temperature information. There is a single bit window of high signal (stop bit) between the two bytes.

Humidity data is transmitted as a 1-byte packet on the humidity channel after the temperature transmission.

5.2 ChipCap Temperature Transmission Packet (cont.)

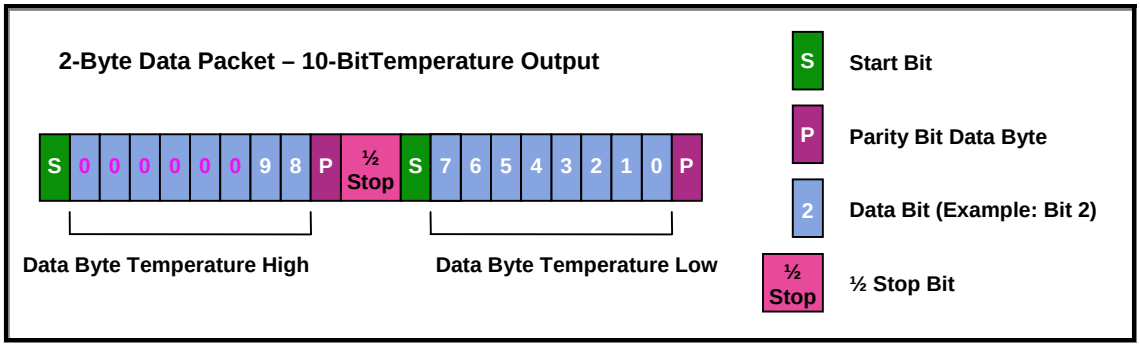


Figure 7: Full ZACwire Temperature Transmission from ChipCap Sensor

5.3 Bit Encoding

The bit format is duty-cycle encoded:
Start bit => 50% duty cycle used to set up strobe time
Logic 1 => 75% duty cycle
Logic 0 => 25% duty cycle

Stop Bit
For the time of a half a bit width, the signal level is high.
There is a half stop bit time between bytes in a packet.

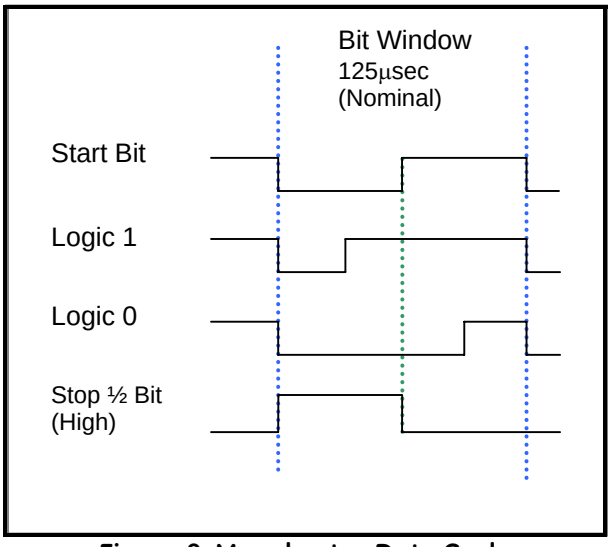


Figure 8: Manchester Duty Cycle

5.3 Bit Encoding (cont.)

An oscilloscope trace of a ZACwire transmission demonstrates the bit encoding. Figure 9 below shows a single packet of 96Hex being transmitted. Because 96Hex is already even parity, the parity bit is 0.

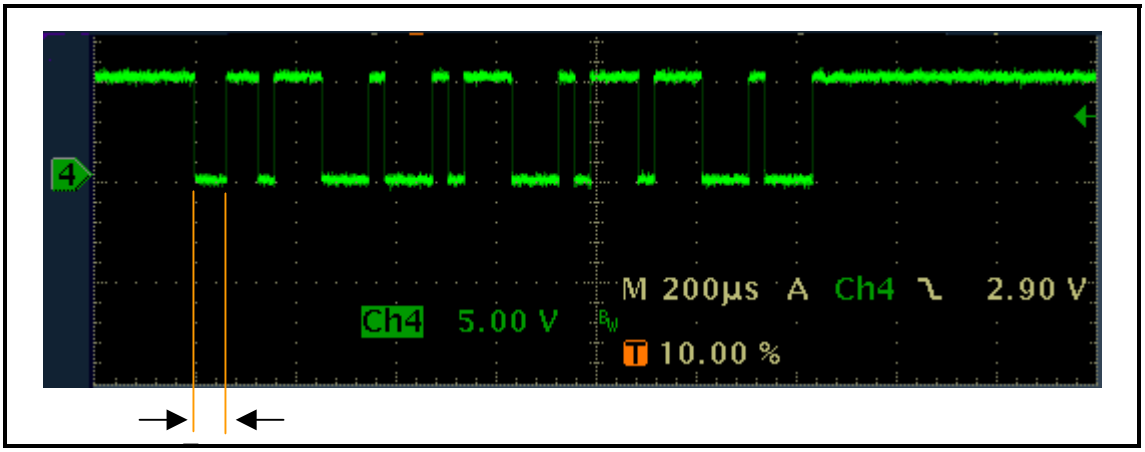


Figure 9: ZACwire Transmission

5.4 How to Read a 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 ZACwire 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, however, that when acquiring the start bit, the sampling rate of the ZACwire signal be at least 16x the nominal baud rate. Because the nominal baud rate is 8 kHz, a minimum 128 kHz sampling rate is recommended when acquiring Tstrobe.

5.5 How to Read a Packet Using a µController

It is best to connect the ZACwire signal to a pin on the µController that is capable of causing an interrupt on a falling edge. When the falling edge of the start bit occurs, it causes the µController to branch to its ISR. The ISR enters a counting loop incrementing a memory location (Tstrobe) until it sees a rise on the ZACwire 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 ZACwire line is driven by a strong CMOS push/pull driver. The parity bit is intended for use when the ZACwire output is driving long (>2m) interconnects to the µController in a noisy environment. For systems in environments without noise interference, the user can choose to have the µController ignore the parity bit.

Example PIC1 Assembly Code on page 13 gives an example of code for reading a ChipCap ZACwire transmission using a PIC16F627 µController.

5.5a How Often Does the ChipCap Sensor Transmit?

If the ChipCap sensor is being read using an ISR, how often is it interrupting the µController with data? The update rate of the sensor is programmed to the setting of 1Hz (1s response time). Servicing a temperature and humidity reading, ISR requires about 4ms, so the µController spends about 0.4% of its time reading the humidity/temperature transmission.

5.5b Solutions for Real Time Systems

Some real time systems cannot tolerate the ChipCap sensor interrupting the µController. The µController must initiate the humidity/temperature reading. This can be accomplished by using another pin of the µController to supply the VDD to the sensor. The ChipCap sensor will transmit its first temperature reading approximately (500-600) ms after power-up. When it is time for the µController to read the humidity/temperature, it first powers the sensor using one of its port pins. It will receive a temperature transmission approximately (500-600) ms later. The humidity byte will immediately follow the temperature transmission on the humidity channel. If during that (500-600) ms window a higher priority interruption occurs, the µController can simply power down the sensor to ensure it will not cause an interruption or be in the middle of a transmission when the higher priority ISR finishes. This method of powering the ChipCap sensor has the additional benefit of acting like a power-down mode and reducing the quiescent current from a nominal 500µA to zero. The ChipCap sensor is a mixed signal IC and provides best performance with a low noise VDD supply. Powering through a µController pin subjects the sensor to the digital noise present on the µController's power supply. Therefore it is best to use a simple RC filter when powering the ChipCap sensor with a µController port pin. See Figure 10 below.

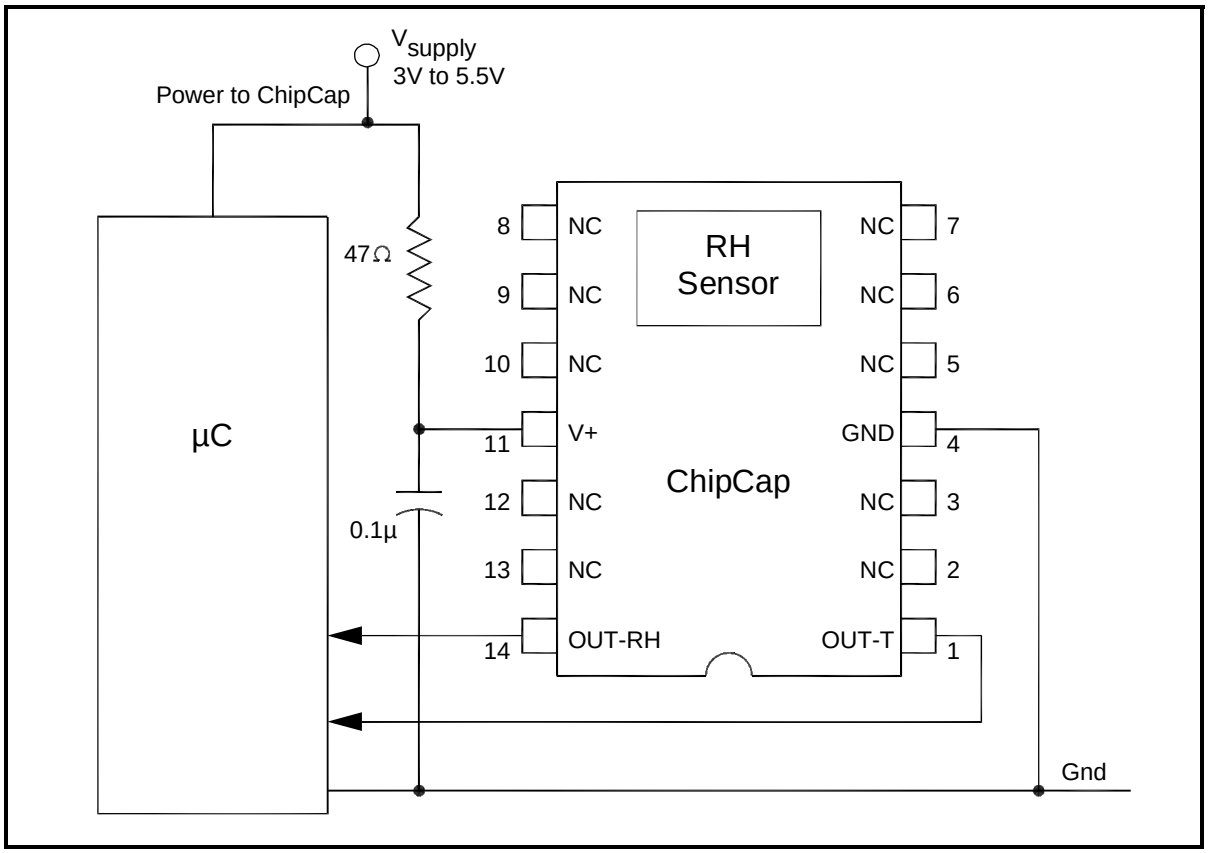


Figure 10: Digital Interface - Digital Mode

5.5b Solutions for Real Time Systems (cont.)

Reading the Digital RH Value:

Read a single byte from the RH communication line.

DigitalRH = (ByteRead* 100) / 255

Reading the Digital Temperature:

Read two bytes from the RH communication line.

DigitalTemp = (UpperByte * 256 + LowerByte) *(200/1023) – 50

6. PIC1 Assembly Code Example

The PIC1 Assembly Code is for reading ZACwire Output. In the following code example, it is assumed that the ZACwire pin was connected to the interrupt pin (PORTB,0) of the PIC and that the interrupt was configured for falling edge interruption. This code would work for a PIC running between 3 and 20 MHz.

```
Temp_high     EQU    0x24   ;; memory location reserved for temp high byte
Temp_low      EQU    0x25   ;; memory location reserved for temp low byte
                            ;; this byte must be consecutive from Temp_high
LAST_LOC      EQU    0x26   ;; this byte must be consecutive from Temp_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 ZACwire transmission then branch to ZAC_TX                           ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ZAC_TX:       movlw  Temp_high     ;; move address of Temp_high (0x24) to W reg
              movwf  FSR           ;; FSR = indirect pointer, now pointing to Temp_high
GET_Tlow:     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 on ZACwire
              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 of ZACwire
              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
```

6. PIC1 Assembly Code Example (cont.)

```
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    ;; ZACwire for data, otherwise keep counting
                                  ;; Length of this loop is 6 states. This must
                                  ;; match the length of the loop that acquired Tstrobe

             bcf    STATUS,C      ;; clear the carry
             btfsc  PORTB,0       ;; sample the ZACwire input
             bsf    STATUS,C      ;; if ZACwire was high then set the carry
             rlf    INDF,1        ;; rotate carry=ZACwire 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
```

6. PIC1 Assembly Code Example (cont.)

```
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_Tlow

              ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
              ;; If here, then done reading the ZACwire and have the data    ;;
              ;; in Temp_high & Temp_low                                     ;;
              ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

WAIT_Tlow:    clrf   time_out
WAIT_TLF:     btfss  PORTB,0      ; wait for fall of PORTB,0 indicating
              goto   GET_Tlow     ; start of Temp Low byte
              incfsz time_out
              goto   WAIT_TLF
              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
```

7. PIC 1 C++ Code Example

This program example reads a single byte of humidity data, and returns a status byte indicating the state of the read.

```
unsigned char dut_zac_hum_rcv (void)
{
  /*******************************************************
  * This routine returns two bytes.  data_byte is a global *
  * unsigned char which will hold the byte read           *
  * from the ZACWireTM of the DUT.  The unsigned char the  *
  * routine returns will be a status indicating the state  *
  * of the read                                           *
  *   0x01 = Good                                         *
  *   0x00 = Time out                                     *
  *   0x05 = parity error                                 *
  * This routine can work for temperature or humidity     *
  * bytes, for this application it was reading humidity   *
  * connected to the PORTB,bit4 port.  It would be called  *
  * when PORTB,bit4 has already fallen                    *
  * Example:                                              *
  *   status = dut_zac_hum_rcv();                         *
  *   if (status==0x01) {                                 *
  *      humidity_byte = data_byte;                       *
  *   }                                                   *
  *******************************************************/
  unsigned char strb_cnt,bit_cnt,Tstrobe,prty_cnt;
  unsigned char time_out;

  /***********************************
  * Time out the start bit low period *
  ***********************************/
  Tstrobe=0x02;  /* account for overhead */
  while (!(PORTBbits.RB4)) {  /* 7 state loop (access RAM mode) */
    if (++Tstrobe==0x00) {
      return(0x00);
    }
  }
  data_byte = 0x00;
  prty_cnt=0x00;

  /*************************************************
  * Now for the 8-bits in a byte wait for falling edge *
  *************************************************/
```

7. PIC 1 C++ Code Example (cont.)

```
for (bit_cnt=0; bit_cnt<8; bit_cnt++) {
  time_out=0x00;
  strb_cnt=0x00;
  data_byte = (data_byte<<1);  /* shift while you have time */
  while (PORTBbits.RB4) {
    if (++time_out==0x00) {
      return(0x00);
    }
  }
    /***********************************
  * Timer loop till strb_cnt = Tstrobe *
  ***********************************/
  while (strb_cnt!=Tstrobe) {  /* 7 state loop when compiled */
    Nop();                     /* in access RAM mode */
    strb_cnt++;
  }
  /*******************************
  * Now sample state of ZACWireTm *
  *******************************/
  if (PORTBbits.RB4) {
    data_byte++;
    prty_cnt++;
  }
  /************************************************
  * Now wait for rise if it hasn't already happened *
  ************************************************/
  time_out =0x00;
  while (!(PORTBbits.RB4)) {
    if (++time_out==0x00) {
      return(0x00);
    }
  }
}
strb_cnt=0x00;
/*********************************
* Now wait for fall of parity bit *
*********************************/
time_out = 0x00;
while (PORTBbits.RB4) {
  if (++time_out==0x00) {
    return(0x00);
  }
}
/***********************************
* Timer loop till strb_cnt = Tstrobe *
***********************************/
while (strb_cnt!=Tstrobe) {
  Nop();                /* 10 state loop */
```

7. PIC 1 C++ Code Example (cont.)

```
  strb_cnt++;
}
/************************
* Now sample parity bit *
************************/
if (PORTBbits.RB4) {
  prty_cnt++;
}
if (prty_cnt % 2) {
  return(0x05);  /* parity error */
}
/***********************************
* Now wait for rise of parity if it *
* hasn't already happened           *
***********************************/
time_out =0x00;
while (!(PORTBbits.RB4)) {
  if (++time_out==0x00) {
    return(0x00);
  }
}
return(0x01);   /* Good data in data_byte */
```

8. 8051 C++ Code Example

In the following code example, assume that the ZACwire pin is connected to the PORT 0 pin (0x80hex) of the µController 8051. This example code is for an 8051 µController running at 24.5 MHz; however, frequencies from 8 to 24.5 MHz can be used. In this case, the number of nops for the "WAIT_60_US" command should be adjusted according to the µController frequency.

This program example reads only the two bytes of temperature data transmitted on the temperature channel.

The program can be easily adapted to measure humidity by redefining the pins for Humidity. Note also that the humidity data is 8 bit.

It does not use interrupts. Contact the factory for additional examples using interrupts.

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

```
/******************************************************************************
* FUNCTION MACROS
******************************************************************************/
#define CHIPCAP_INIT()        {    SFRPAGE    =    CONFIG_PAGE;    PORT_CONFIG    |=
PWR_PIN; PORT &= ~PWR_PIN; /* power */\
                  PORT_CONFIG &= ~SIG_PIN; PORT |= SIG_PIN; /* signal */ }
#define CHIPCAP_ON()          SFRPAGE = CONFIG_PAGE; PORT |= PWR_PIN;
#define CHIPCAP_OFF()               SFRPAGE = CONFIG_PAGE; PORT &= ~PWR_PIN;
#define CHIPCAP_SIGNAL()      (PORT & SIG_PIN)
/******************************************************************************
* FUNCTION MACROS
******************************************************************************/
// assuming MCU runs at 24.5MHz then 1 nop =1/(24.5MHz ÷ 8)=~0.33µs
// used as blocking wait function
#define WAIT_60_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_();_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_();_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_(); \
```

8. 8051 C++ Code Example (cont.)

```
_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_();_nop_();_nop_();_nop_();_nop_();_nop_();_nop_();_nop_();_nop_();_nop_();_nop_();_nop_();
/*****************************************************************************
* Function    : getCHIPCAPTemp
* Description : reads from the CHIPCAP its output value
* Parameters  : pointer for return value
* Returns     : read value
* Notes       : blocking function, assuming MCU runs at (24.5 ÷ 8) MHz
*****************************************************************************/
UINT16 getCHIPCAPTemp (UINT16 *temp_value16)
{
        UINT16 temp_value1 = 0;
        UINT16 temp_value2 = 0;
        UINT8 i;
        UINT16 Temperature;
        UINT8 parity;

        CHIPCAP_ON();
        WAIT_60_US();                   // wait for stabilization
        WAIT_60_US();

        SFRPAGE = CONFIG_PAGE;
        while (CHIPCAP_SIGNAL());     // wait until start bit starts

        // wait, TStrobe
        while (CHIPCAP_SIGNAL() == 0x00);

        // first data byte
        // read 8 data bits and 1 parity bit
        for (i = 0; i < 9; i++)
        {
                while (CHIPCAP_SIGNAL());                       // wait for falling edge
                WAIT_60_US();
                if (CHIPCAP_SIGNAL())
```

8. 8051 C++ Code Example (cont.)

```
                temp_value1 |= 1 << (8-i);              // get the bit
        else
                while (CHIPCAP_SIGNAL() == 0x00);       // wait until line comes high again
}

// second byte
while (CHIPCAP_SIGNAL());

// wait, TStrobe
while (CHIPCAP_SIGNAL() == 0x00);

// read 8 data bits and 1 parity bit
for (i = 0; i < 9; i++)
{
        while (CHIPCAP_SIGNAL());                       // wait for falling edge
        WAIT_60_US();
        if (CHIPCAP_SIGNAL())
                temp_value2 |= 1 << (8-i);              // get the bit
        else
                while (CHIPCAP_SIGNAL() == 0x00);       // wait until line comes high again
}

CHIPCAP_OFF();                                  // switch CHIPCAP off

// check parity for byte 1
parity = 0;
for (i = 0; i < 9; i++)
        if (temp_value1 & (1 << i))
                parity++;
if (parity % 2)
        return FALSE;

// check parity for byte 2
parity = 0;
for (i = 0; i < 9; i++)
        if (temp_value2 & (1 << i))
                parity++;
```

8. 8051 C++ Code Example (cont.)

```
        if (parity % 2)
                return FALSE;

        temp_value1 >>= 1;              // delete parity bit
        temp_value2 >>= 1;              // delete parity bit
        Temperature = (temp_value1 << 8) | temp_value2;
        *temp_value16 = Temperature;
        return TRUE;                    // parity is OK
}
/*******************************************************************************
* Function    : cmdGetChipCapValue
* Description : debug function
* Parameters  : none
* Returns     : none
* Notes       : none
*******************************************************************************/
void cmdGetCHIPCAPValue (void)
{
        UINT16 temp_value;
        float Temp_float;

        printf("cmdGetCHIPCAPValue\n");
        CHIPCAP_INIT();                 // init the I/O pins used for the CHIPCAP
        CHIPCAP_OFF();                          // switch the CHIPCAP off until use
        if (getCHIPCAPTemp(&temp_value))
        {
                Temp_float = ((float)temp_value / 1023 * 200) - 50;     // decimal conversion equation
                SFRPAGE_UART();
                printf("temp %u, %2.1f\n", temp_value, Temp_float);
        }
}
```

9. Sensor Rehydration/Reconditioning

The following are recommendations for treating ChipCap sensors which have been exposed to various extreme environments. Their effectiveness will depend on how extreme and how long the exposure was, but these recommendations will at least provide a good start in bringing a sensor back to normal operation. In each case, the rate of recovery will depend on the environment, but these pre-conditioning steps should help it come back more quickly.

Note: *All of these processes can be done with the sensor unpowered.*

1. After being exposed to a condensing environment, the ChipCap sensor needs to be dry baked for one hour at 125°C (~0% RH), then exposed to 70% RH for 2-3 hours to rehydrate.

2. After being exposed to a high-temperature dry environment just rehydrate at 70% RH for 2-3 hours.

3. After being exposed to 85% RH at 85°C for 500 hours, the ChipCap sensor will need the same treatment as in #1.

4. After being exposed to a high-temperature dry environment (dry packed storage or reflow soldering), the ChipCap sensor needs only to be rehydrated at 70% RH for 2-3 hours. Alternately, storage at 30 to 50% RH for one week will rehydrate the ChipCap humidity sensor.

Note: *The result of exposure to these environments will not permanently damage the sensor. The sensor will typically return to its calibration value after reconditioning.*

10. ChipCap Packaging/Final Assembly

CAUTION! **Be careful not to expose the ChipCap to chemical vapors which can adversely affect the performance or function of the device. Avoid spray coating the ChipCap with any chemical, such as conformal coating. Be sure to prevent the coating or covering of the sensor element with any type of chemical during assembly. Care should also be taken not to break the wire bond t hat holds the sensor element.**

10.1 Chemical Resistivity

Table 5: Long Term Exposure Results

	∆% RH Change Over Exposure Time							
Chemical	**89.0 hr**		**231.5 hr**		**400.0 hr**		**899.0 hr**	
	0%	**100%**	**0%**	**100%**	**0%**	**100%**	**0%**	**100%**
Ammonia Hydroxide	F	F	F	F	F	F	F	F
Acetone	F*	F	F	F	F	F	F	F
Ethanol	F	F	F	F	F	F	F	F
Methanol	-1.9	25.1	-1.9	29.4	-9.7	35.0	-5.4	39.8
50% Ethanol + 50% Methanol	14.5	-17.4			7.9	-31.8	4.2	-22.0
Formaldehyde neutral soin.	0.9	0.0	1.5	-0.3	1.5	-.14	1.9	-3.5
Formaldehyde norm & buffd	0.4	0.9	1.2	-0.4	1.1	-1.9	1.5	-3.2
Benzene	-2.0	1.5	-1.1	-1.7	-0.3	-8.1	-1.1	-24.7

Table 5: Long Term Exposure Results (cont.)

	Δ% RH Change Over Exposure Time							
Chemical	**89.0 hr**		**231.5 hr**		**400.0 hr**		**899.0 hr**	
	0%	**100%**	**0%**	**100%**	**0%**	**100%**	**0%**	**100%**
Toluene	-1.7	1.4	-0.8	0.4	0.4	0.0	-0.9	-4.9
Xylene	-1.7	1.5	-0.8	-0.2	-0.6	-0.7	-0.9**	0.0
30% Benzene + 30% Toluene + 40% Xylene	-0.2	-1.2			-0.1	-6.0	-0.6	-16.1

*Sensors are resistant to acetone over shorter exposures.

Table 6: Saturation and Recovery Results

	Post Saturation		Post Recovery	
Chemical	**Δ% at 0% RH**	**Δ% at 75.3% RH**	**Δ% at 0% RH**	**Δ% at 75.3% RH**
Alcohol Isopropyl, 66%	+0.1	+1.13	+0.0	+1.83
Endo-Spor Hydrogen Peroxide	+0.46	-0.16	+0.4	-0.43
Glutaraldehyde Cydex Plus	+0.56	-2.13	+0.63	-1.63
Idophore Solution Westoodyne	+0.23	+0.16	+0.36	+0.93
Kleenaseptic	+3.13	+4.5	+2.96	+4.66
Quaternary Ammonium Virex 0.2%	+0.43	+0.2	+0.3	+0.8
Sodium Hypochlorite	+0.36	+0.6	+0.43	+1.53

11. Soldering Procedure

The ChipCap can be soldered using standard reflow ovens (including lead free soldering) at a maximum of 260°C for 30 seconds. The ChipCap requires careful handling during and after the soldering process to prevent damage to the sensing elements.

11.1 Solder Paste

Use “No-Clean” solder paste only.

Table 7: Recommended Soldering Profile

	Convection or IR/Convection
Average ramp-up rate (183°C to Peak)	4°C/second max.
Preheat temperature 160 (±10)°C	90 seconds max.
Temperature maintained above 183°C	To 150 seconds
Time within 5°C of actual peak temperature	30 seconds
Peak temperature	260°C (lead free soldering)
Ramp-down rate	6°C/second max.

11.2 PCB Cleaning

Do not clean or wash the Printed Circuit Board (PCB) after soldering.

11.3 Manual Soldering

Contact time must be limited to 5 seconds at up to 350°C.

12. ESD & Latch-Up Test

Table 8: ESD & Latch-UP Test

ESD	HBM 1.5kΩ/100pF, 3 x ±2.0 kV	PASS
ESD	CDM, 3 x ±250V, 3 x ±500V, 3 x 500V, and corner pins ±750V	PASS
Latch-Up	V_{DD} = 5V → 8V, I_{TR} = ±100mA, T_{amb} = +85°C	PASS

CAUTION! Standard ESD procedures should be followed when handling ChipCaps.

13. Contact Information

GE Sensing & Inspection Technologies
Worldwide Headquarters:
The Boston Center
1100 Technology Park Drive
Billerica, MA 01821-4111
USA
e-mail: sensing@ge.com
Tel: 978-437-1000
800-833-9438

[no content intended for this page - proceed to next page]

Warranty

Each instrument manufactured by GE Sensing is warranted to be free from defects in material and workmanship. Liability under this warranty is limited to restoring the instrument to normal operation or replacing the instrument, at the sole discretion of GE Sensing. Fuses and batteries are specifically excluded from any liability. This warranty is effective from the date of delivery to the original purchaser. If GE Sensing determines that the equipment was defective, the warranty period is:

- one year from delivery for electronic or mechanical failures
- one year from delivery for sensor shelf life

If GE Sensing determines that the equipment was damaged by misuse, improper installation, the use of unauthorized replacement parts, or operating conditions outside the guidelines specified by GE Sensing, the repairs are not covered under this warranty.

The warranties set forth herein are exclusive and are in lieu of all other warranties whether statutory, express or implied (including warranties or merchantability and fitness for a particular purpose, and warranties arising from course of dealing or usage or trade).

Return Policy

If a GE Sensing instrument malfunctions within the warranty period, the following procedure must be completed:

1. Notify GE Sensing, giving full details of the problem, and provide the model number and serial number of the instrument. If the nature of the problem indicates the need for factory service, GE Sensing will issue a RETURN AUTHORIZATION NUMBER (RAN), and shipping instructions for the return of the instrument to a service center will be provided.
2. If GE Sensing instructs you to send your instrument to a service center, it must be shipped prepaid to the authorized repair station indicated in the shipping instructions.
3. Upon receipt, GE Sensing will evaluate the instrument to determine the cause of the malfunction.

Then, one of the following courses of action will then be taken:

- If the damage is covered under the terms of the warranty, the instrument will be repaired at no cost to the owner and returned.
- If GE Sensing determines that the damage is not covered under the terms of the warranty, or if the warranty has expired, an estimate for the cost of the repairs at standard rates will be provided. Upon receipt of the owner's approval to proceed, the instrument will be repaired and returned.

[no content intended for this page - proceed to next page]

www.gesensinginspection.com

910-108 Rev. C