HYGROSENS
INSTRUMENTS

TSic™ Temperature Sensor IC
Application Notes – ZACwire™
Digital Output

RoHS conform

CE

CONTENTS

1 TSic™ ZACwire™ Communication Protocol

ZACwire™ 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 TSic™ will be transmitting temperature information, and another IC in the system (most likely a ⌈Controller) will be reading the temperature data over the ZACwire™.

1.1 Temperature Transmission Packet from a TSic™

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

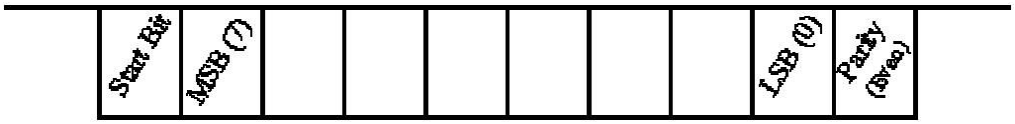


Figure 1.1 – ZACwire™ Transmission Packet

The TSic™ provides temperature data with 11-bit resolution, and obviously these 11-bits of information cannot be conveyed in a single packet. A complete temperature transmission from the TSic™ consists of two packets. The first packet contains the most significant 3-bits of temperature information, and the second packet contains the least significant 8-bits of temperature information.

There is a single bit window of high signal (stop bit) between the end of the first transmission and the start of the second transmission.

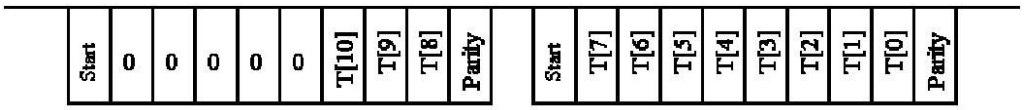


Figure 1.2 – Full ZACwire™ Temperature Transmission from TSic™

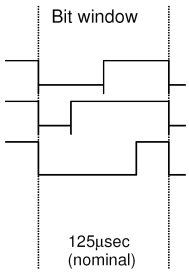
1.2 Bit Encoding

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



Perhaps the best way to show the bit encoding is with an oscilloscope trace of a ZACwire™ transmission. The following shows a single packet of 96Hex being transmitted. Because 96Hex is already even parity, the parity bit is zero.

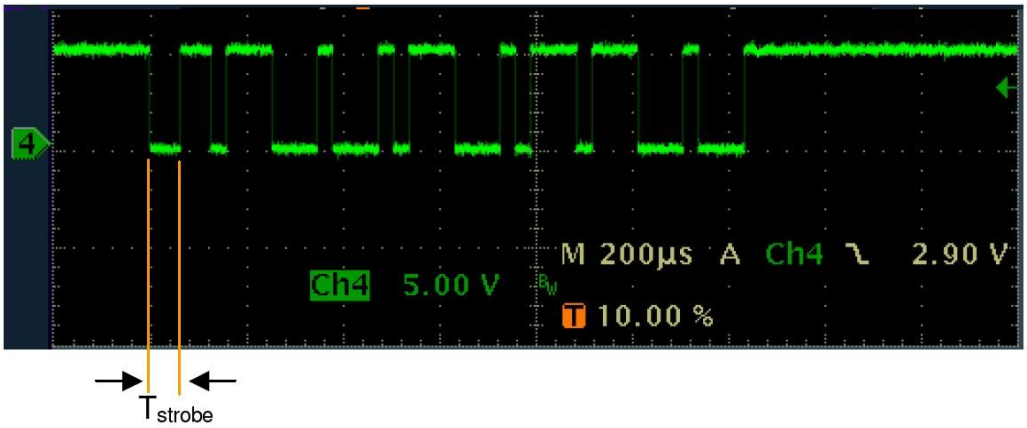


Figure 1.3 – ZACwire™ Transmission

1.3 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 a time period equal to Tstrobe, and then sample the ZACwire™signal. The data present on the signal at this time is 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 such as RS232. It is recommended, however, that the sampling rate of the ZACwire™ signal when acquiring the start bit be at least 16x the nominal baud rate. Because the nominal baud rate is 8kHz, a 128kHz sampling rate is recommended when acquiring Tstrobe.

1.4 How to Read a Packet using a µController

It is best to connect the ZACwire™ signal to a pin of 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 9 falling edges (8-data, 1-parity). After each falling edge, it waits 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™ is driving long (>2m) interconnects to the µController in a noisy environment.

For systems in which the "noise environment is more friendly," the user can choose to have the µController ignore the parity bit.

In the appendix of this document is sample code for reading a TSic™ ZACwire™ transmission using a PIC16F627 µController.

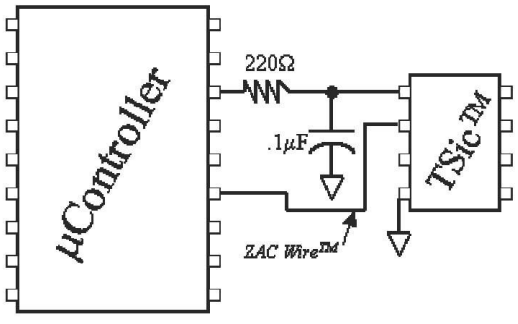
1.4.1 How Often Does the TSic™ Transmit?

If the TSic™ is being read via an ISR, how often is it interrupting the µController with data? The update rate of the TSic™ can be programmed to one of 4 different settings: 250Hz, 10Hz, 1Hz, and 0.1Hz. Servicing a temperature-read ISR requires about 2.7ms. If the update rate of the TSic™ is programmed to 250Hz, then the µController spends about 66% of its time reading the temperature transmissions. If, however, the update rate is programmed to something more reasonable like 1Hz, then the µController spends about 0.27% of its time reading the temperature transmissions.

1.4.2 Solutions if Real Time System Cannot Tolerate the TSic™ Interrupting the µController

Some real time systems cannot tolerate the TSic™ interrupting the µController. The µController must initiate the temperature read. This can be accomplished by using another pin of the µController to supply VDD to the TSic™. The TSic™ will transmit its first temperature reading approximately 65- 85ms after power up. When the µController wants to read the temperature, it first powers the TSic™ using one of its port pins. It will receive a temperature transmission approximately 65 to 85ms later. If during that 85ms, a higher priority interrupt occurs, the µController can simply power down the TSic™ to ensure it will not cause an interrupt or be in the middle of a transmission when the high priority ISR finishes. This method of powering the TSic™ has the additional benefit of acting like a power down mode and reducing the quiescent current from a nominal 150µA to zero. The TSic™ is a mixed signal IC and provides best performance with a clean VDD supply. Powering through a µController pin does subject it to the digital noise present on the µController's power supply. Therefore it is best to use a simple RC filter when powering the TSic™ with a µController port pin. See the diagram below.

µController powers TSic™ with a port pin through a simple RC filter.



2 Appendix A: An Example of PIC1 Assembly Code for Reading the ZACwire™

In the following code example, it is assumed that the ZACwire™ pin is connected to the interrupt pin (PORTB, 0) of the PIC and that the interrupt is configured for falling edge interruption. This code should work for a PIC running between 2-12MHz.

```
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 NEEDED STATE AND TO DETERMINE THE SOURCE OF
THE ISR GOES HERE. ONCE YOU HAVE DETERMINED THE SOURCE IF THE
INTERRUPT WAS A ZAC WIRE TRANSMISSION THEN YOU 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 WRONG AND RETURN
                                  FROM INTERRUPT
             BTFSS  PORTB,0       ;; LOOK FOR RISE ON ZAC WIRE
             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 ZAC WIRE
             GOTO    PAUSE_STRB    ;; NEXT FALLING EDGE OCCURRED
             INCFS   TIME_OUT,1    ;; CHECK IF EDGE TIME OUT
             Z                     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    ;; ZAC WIRE FOR DATA,
                                   OTHERWISE KEEP COUNTING
```

LENGTH OF THIS LOOP IS 6-STATES. THIS HAS TO MATCH THE LENGTH OF THE LOOP THAT ACQUIRED TSTROBE

```
             BCF     STATUS,C      ;; CLEAR THE CARRY
             BTFSC   PORTB,0       ;; SAMPLE THE ZAC WIRE INPUT
             BSF     STATUS,C      ;; IF ZAC WIRE WAS HIGH THEN
                                   SET THE CARRY
             RLF     INDF,1        ;; ROTATE CARRY=ZAC WIRE 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
                                   WE ARE DONE
             GOTO    NEXT_BIT
             INCFS   TIME_OUT,1    ;; INCREMENT THE EDGE TIME OUT
             Z                     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_TLOW
IF HERE YOU ARE DONE READING THE ZAC WIRE AND HAVE THE DATA IN TEMP_HIGH & TEMP_LOW
WAIT_TLOW:      CLRF    TIME_OUT
WAIT_TLF:       TFSS    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:                                 TORE ANY STATE SAVED OFF AT
                                     BEGINNING OF ISR
                BCF     INTCON,INTF  ;; CLEAR INTERRUPT FLAG
                BSF     INTCON,INTE  ;; ENSURE INTERRUPT RE-ENABLED
                RETFIE               ;; RETURN FROM INTERRUPT
```

HYGROSENS INSTRUMENTS

The technical information in this document has been checked with adequate care at our end and is intended to inform about the product and its applications. The descriptions are not to be understood as assurance of the defined characteristics of the product and should be checked by the user for the intended application. Any possible industrial third party patent rights are to be considered.

Issued October 2008 - This documentation supersedes all previous editions.