PIC16 Microcontroller-based Ideal Observer Circuit

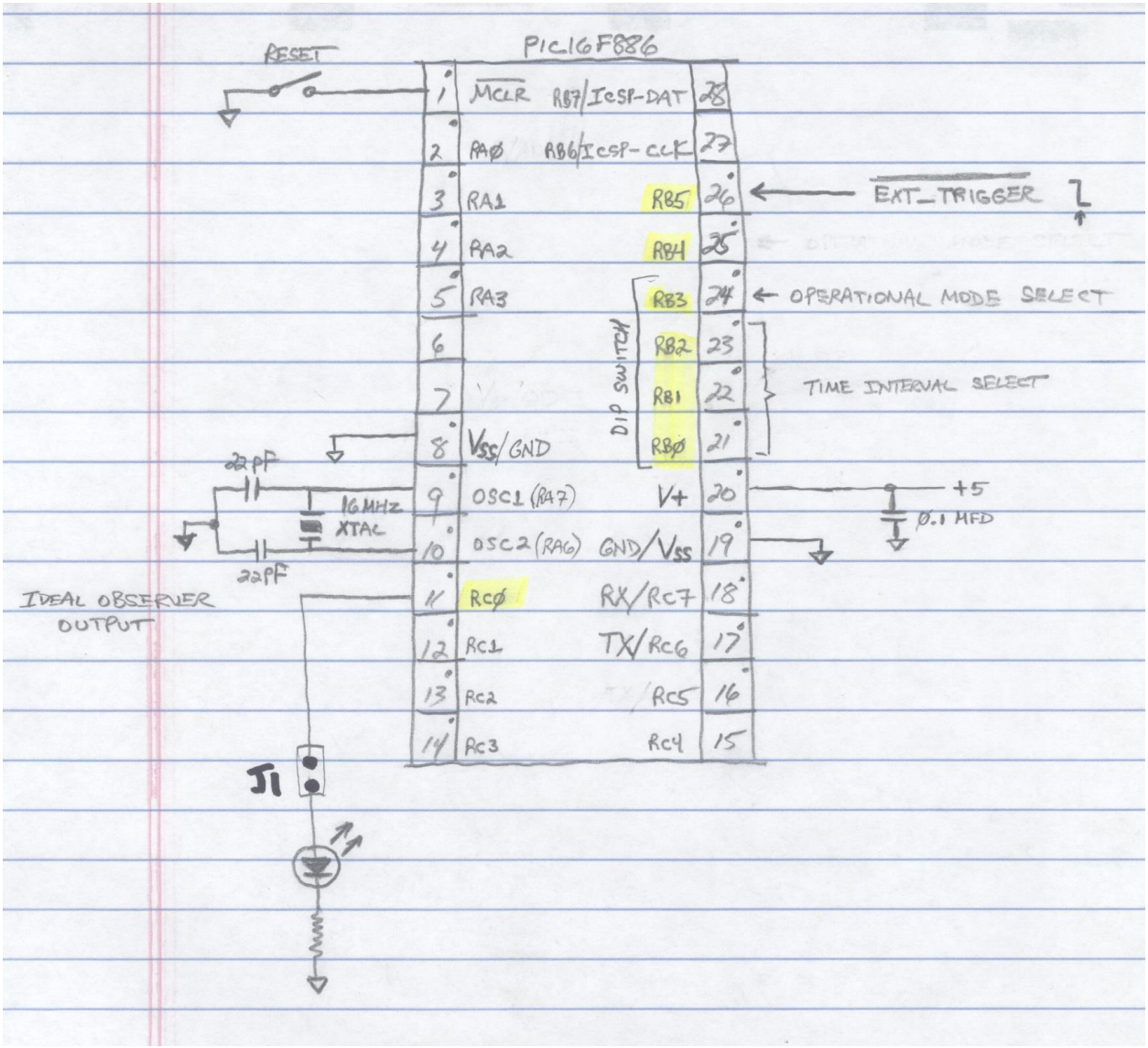

Figure 1.

Simple 16-MHz PIC16 Microcontroller Circuit

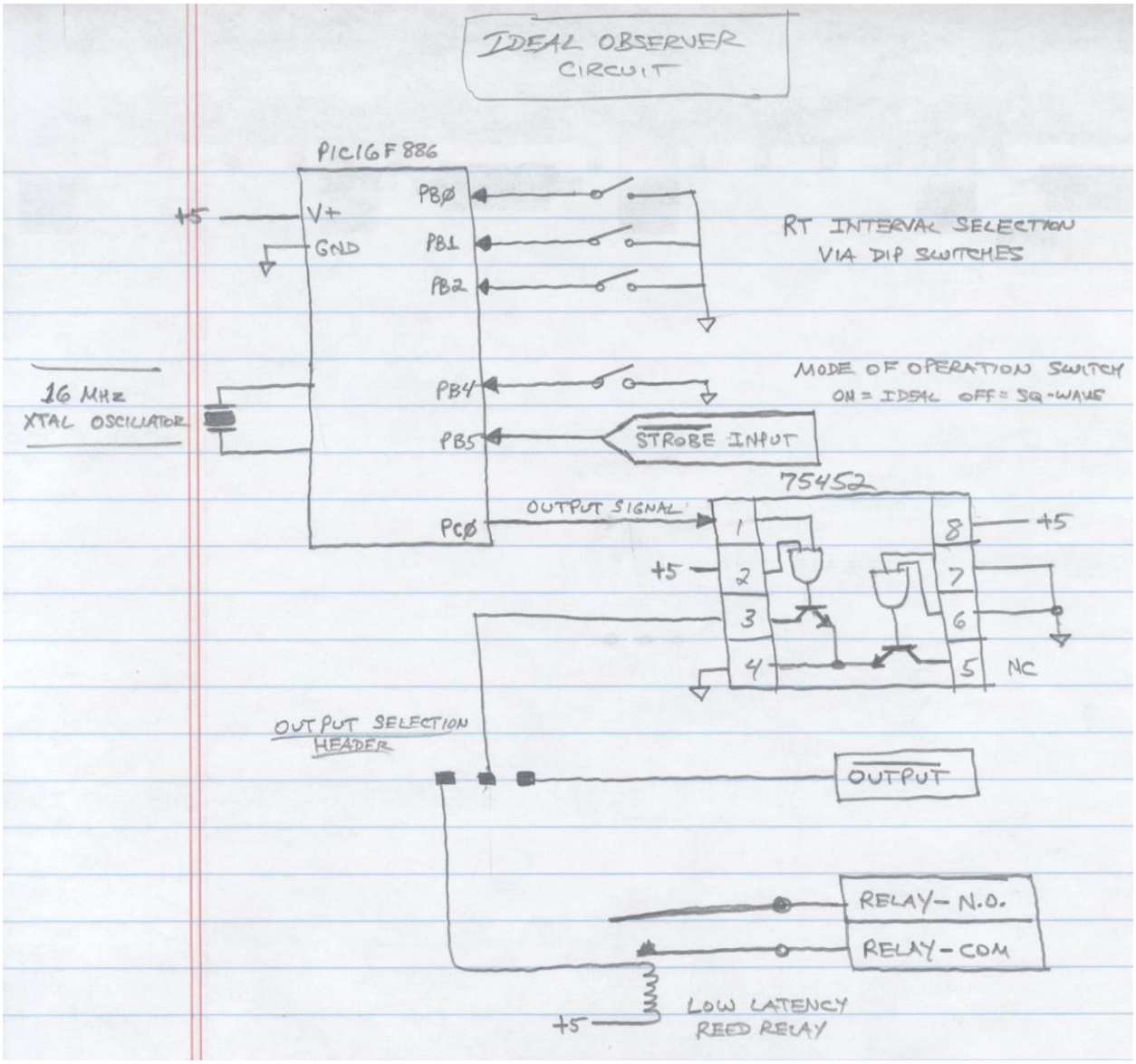

Figure 2.

Dual-Peripheral Driver for PIC16 Circuit

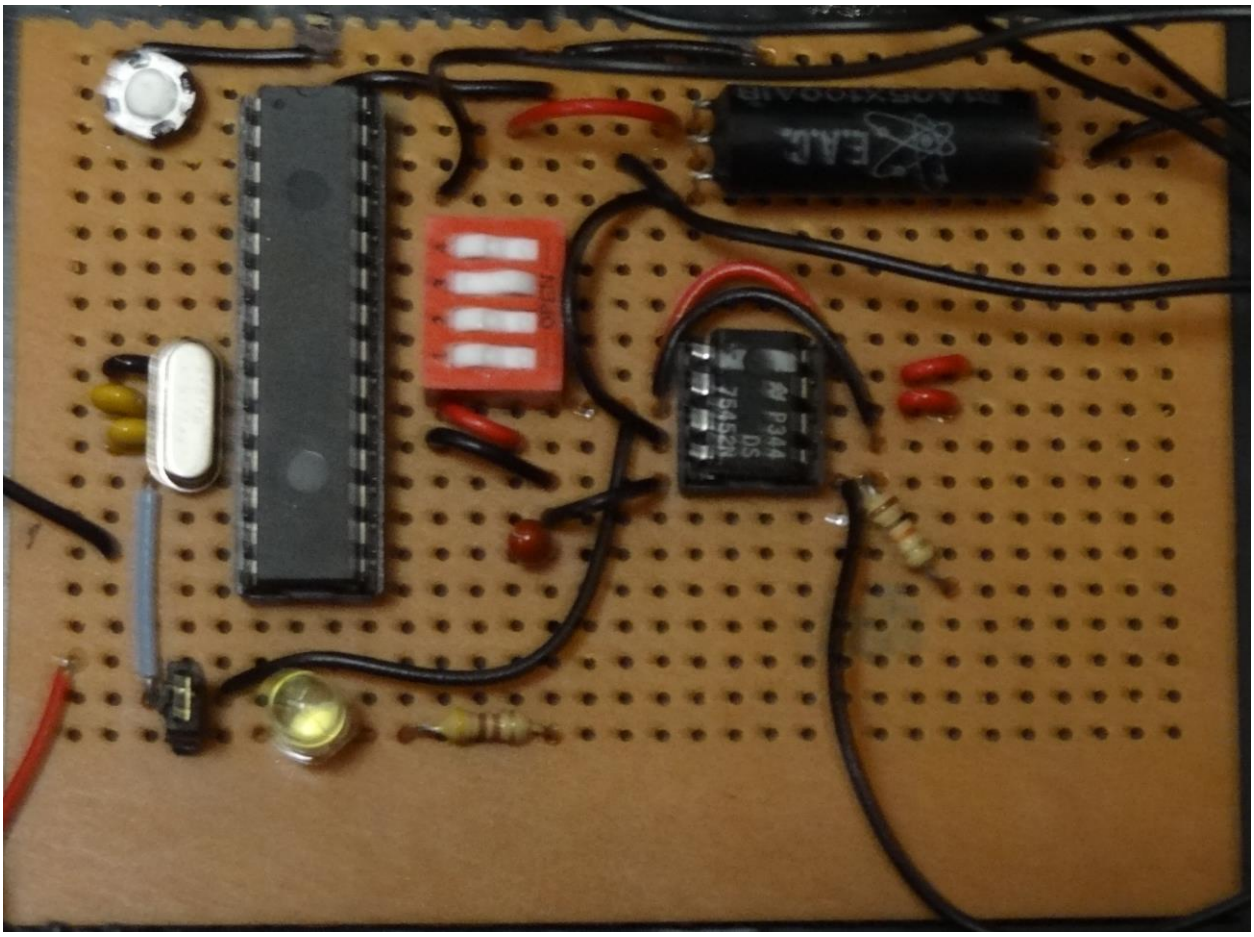

Fig. 3

Photograph of Prototype PIC16-based Ideal Observer Circuit

PIC16 C Code for Ideal Observer

```
////////////////////////////////////////////////////////////////////////////
//
//   Simple Ideal Observer Program for Chipino 16 MHz 16F886 MPU           //
//                                                                         //
//   Implements IDEAL OBSERVER or SQUAREWAVE function generator functions: //
//                                                                         //
//   MODE selected via DIP switch connected to RB3                         //
//      0 = SQUAREWAVE;   1 = IDEAL                                        //
//                                                                         //
//   INTERVAL selected via DIP switches connected to RB0-RB2               //
//      000 = 100   011 = 250   110 = 400 msec                             //
//      001 = 150   100 = 300   111 = 500 msec                             //
//      010 = 200   101 = 350                                              //
//                                                                         //
//   Input strobe:  RB5 (triggers on high-to-low transition)               //
//   Output on RC0 line (active = HIGH)                                    //
//                                                                         //
////////////////////////////////////////////////////////////////////////////


#define _LEGACY_HEADERS //Added for compiler versions 9.81+
#include <htc.h>
#include <stdio.h>


__CONFIG (HS & WDTDIS & MCLREN & UNPROTECT & LVPDIS & IESODIS & FCMDIS);
//HS = Fosc uses 16 MHz xtal at RA4:RA5
//WDTDIS = Watchdog Timer disabled
//MCLREN = MCLR Reset on pin-1 enabled
//         (must manually force high to start from PICKit2)
//         (i.e., Programmer|Release from Reset)
//UNPROTECT = program code is unprotected
//LVPDIS = Low-Voltage program mode disabled (frees-up RB3 pin)
//IESODIS = Internal/External clock switchover disabled
//FCMDIS = Failsafe Clock disabled
```

```
//function prototypes
void pause( unsigned short milliseconds);
void msecbase(void);

//local variables
unsigned short interval;
unsigned short mode;
unsigned char port;
#define IDEAL 1
#define SQUAREWAVE 2

void main() {

    //generic MCU configuration setup
    ANSEL   = 0;    //ADC pins set to digital; ADC disabled
    ANSELH =  0;
    CM1CON0 = 0;    // Comparator-1 disabled
    CM2CON0 = 0;    // Comparator-2 disabled
    INTCON =  0;    // all interrupts disabled

    //setup output line
    PORTC = 0;      // state of all PortC bits set to zero
    TRISC = 0;      // all PortC bits configured as digital OUTPUTS

    //setup input lines
    TRISB0 = 1;      // RB0-RB5 = digital INPUTs
    TRISB1 = 1;
    TRISB2 = 1;
    TRISB3 = 1;
    TRISB4 = 1;
    TRISB5 = 1;
    RABPU = 0;       // enable Port B pull-up capabilities (OPTIONS reg)
    WPUB0 = 1;       // connect pull-up on RB0-RB5 inputs
    WPUB1 = 1;
    WPUB2 = 1;
    WPUB3 = 1;
```

```
WPUB4 = 1;
WPUB5 = 1;

//read RB0-RB2 DIP switches and interpret timing interval
port = (~PORTB & 0x07);   //mask-off irrelevant bits
switch(port) {
    case 0:
        interval = 100;
        break;
    case 1:
        interval = 150;
        break;
    case 2:
        interval = 200;
        break;
    case 3:
        interval = 250;
        break;
    case 4:
        interval = 300;
        break;
    case 5:
        interval = 350;
        break;
    case 6:
        interval = 400;
        break;
    case 7:
        interval = 500;
        break;
}   //end switch(port)
```

```
//read RB3 DIP switch and interpret mode of operation
    port = (PORTB & 0x08);   //mask MODE bit
    if(port == 0x08) {
        mode = SQUAREWAVE;
    }
    else
    {
        mode = IDEAL;
    }

    ////////////////////////////////////
    // Execute IDEAL OBSERVER function //
    // (if selected)   Infinite loop   //
    ////////////////////////////////////
    while(mode == IDEAL) {
        //wait for RB5 trigger to ho HIGH
        while(RB5 == 0) {}
        //wait for RB5 trigger to go LOW (High-to-Low Edge Triggered)
        while(RB5 == 1) {}
        //
        //RC0=1;
        //pause(10);
        //RC0=0;
        //
        //delay for specified number of milliseconds
        pause(interval);
        //activate RC0 output line for 50 msec pulse
        RC0 = 1;
        //delay 50 msec
        pause(50);
        //de-activate RC0 output line
        RC0 = 0;
    }
```

```
    ////////////////////////////////////////
    // Execute SQUAREWAVE function generator //
    // (if selected)  Infinite loop          //
    ////////////////////////////////////////
    while(mode == SQUAREWAVE ) {
        //activate RC0 output line
        RC0 = 1;
        //delay for specified interval
        pause(interval);
        //de-activate RC0 output line
        RC0 = 0;
        //delay for specified interval
        pause(interval);
    }


} //end main()


////////////////////////////////////////
//          millisecond timer         //
////////////////////////////////////////
void pause( unsigned short millisecs ) {
    unsigned short x;
    for (x=0; x<=millisecs; x++) {
        msecbase();    // call millisec delay routine
    }
}
```

```
/*******************************************************
* msecbase - 1 msec pause routine                      *
* The Internal oscillator is set to 16Mhz and the      *
* internal instruction clock is 1/4 of the oscillator. *
* This makes the internal instruction clock 4Mhz or    *
* 0.25 usec per clock pulse.                           *
* Using the 1:16 prescaler on the clock input to Timer0 *
* slows the Timer0 count increment to 1 count/4 usec.  *
* Therefore 250 counts of the Timer0 would make a one  *
* millisecond delay (250 * 4 usec). But there are other *
* instructions in the delay loop so using the an oscope *
* we find that we need Timer0 to overflow at 246 clock *
* ticks. Preset Timer0 to 10 (0A hex) to make Timer0   *
* overflow at 246 clock ticks (256-10 = 246).          *
* This results in a 1.00 millisecond delay.            *
*******************************************************/
void msecbase(void)
{
    OPTION = 0b00000011;    //Set prescaler to TMRO 1:16
    TMR0 = 0xA;             //Preset TMRO to overflow on 250 counts
    while(!T0IF);           //Stay until TMRO overflow flag equals 1
    T0IF = 0;               //Clear the TMR0 overflow flag
}
```