# SCI Interrupts & GPS

## NDSU ECE 376

### Lecture #25

### Inst: Jake Glower

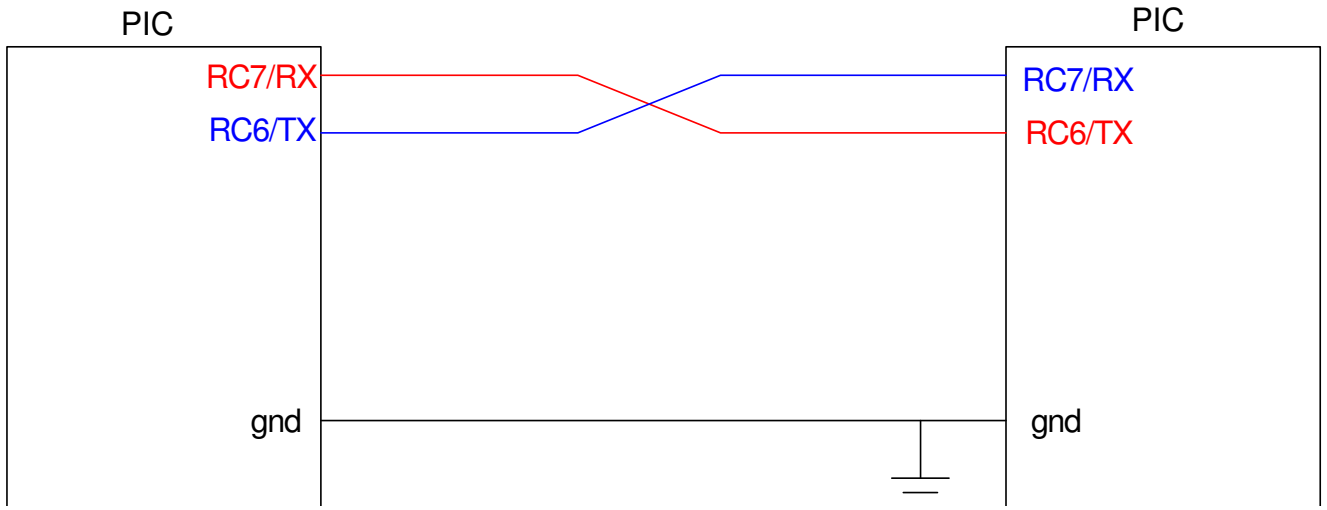
Please visit Bison Academy for corresponding lecture notes, homework sets, and solutions

# SCI Communications

- Send and receive serial data using SCI protocol.
- Send data to a PC via the RS232 serial port.

Why:

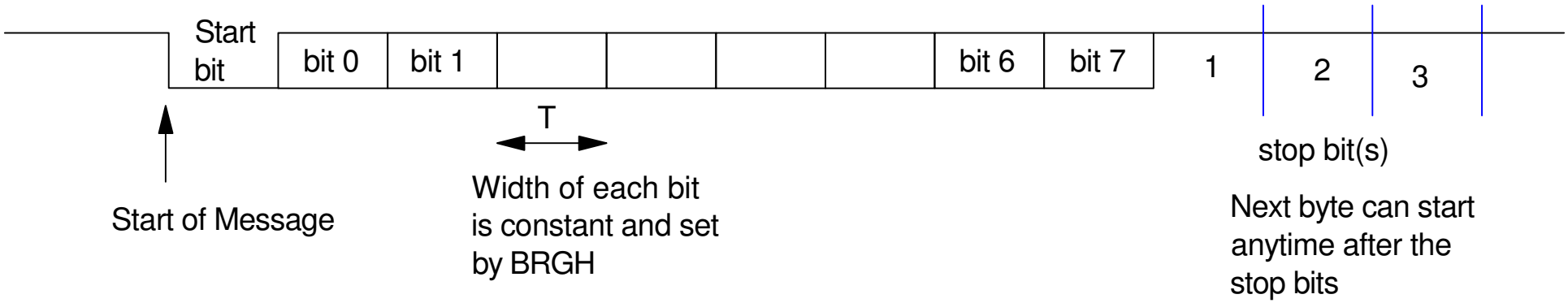
- Debugging code.
  - Sending data to a PC is a useful way to debug code, similar to the LCD display, or collect data.
  - Hyperterm lets you see and save data that comes in on the COM1 or COM2 port.
- Have two PIC processors talk to each other



# Timing:

A generic SCI message has

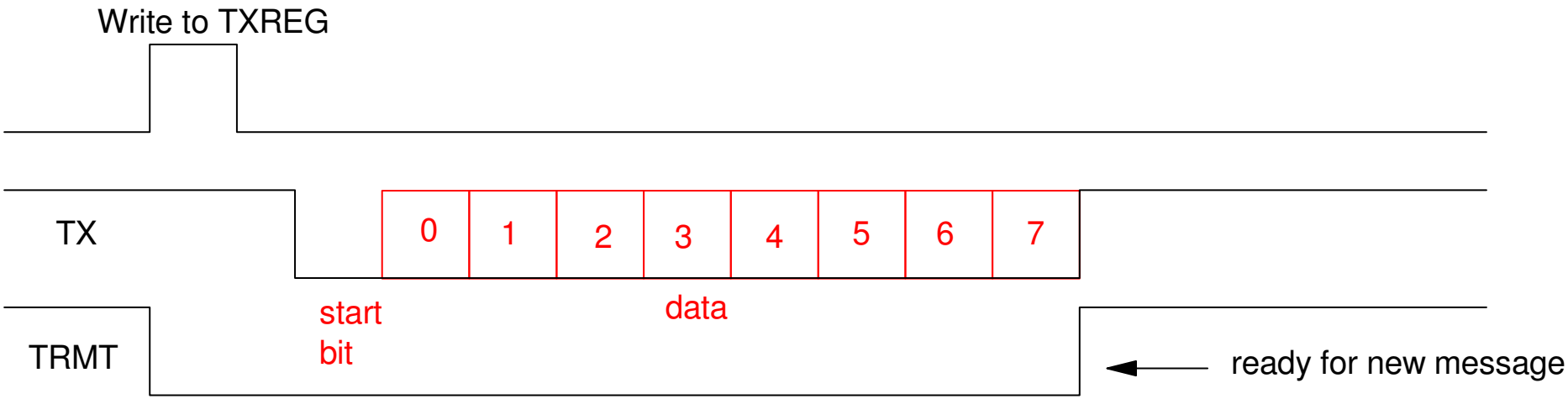
- A start bit (high to low transition)
- 8 data bits - least sgnificant bit first
- 0, 1, or 2 stop bits



# SCI Transmit:

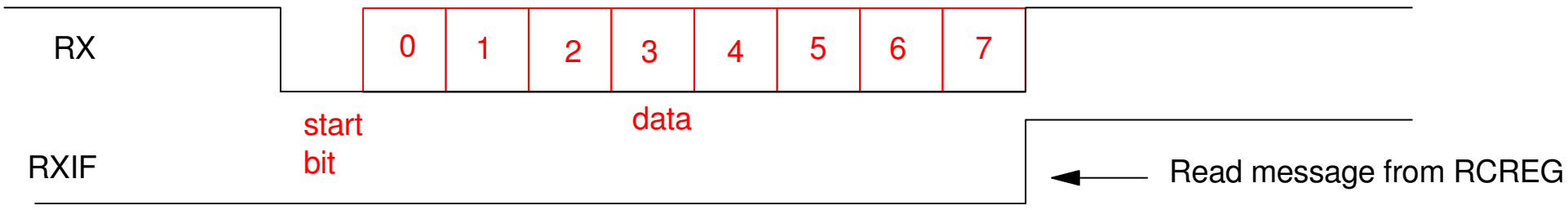
Built into the PIC processor

- Wait for buffer to become free (TRMT = 1)
- Write to TXREG



# SCI Receive

- The hardware detects the start bit automatically
- The hardware sets up the timing for reading in each bit
- The hardware actually reads each bit three times and does best-of-three voting to reduce errors
- The hardware then signals the software when 8-bits have beed read in by setting RCIF
- Once a byte has been recieved, the data can be read from RCREG.



# How: Software:

1. Set up PORTC as follows:

**TRISC (address 0x__ - Bank __)**

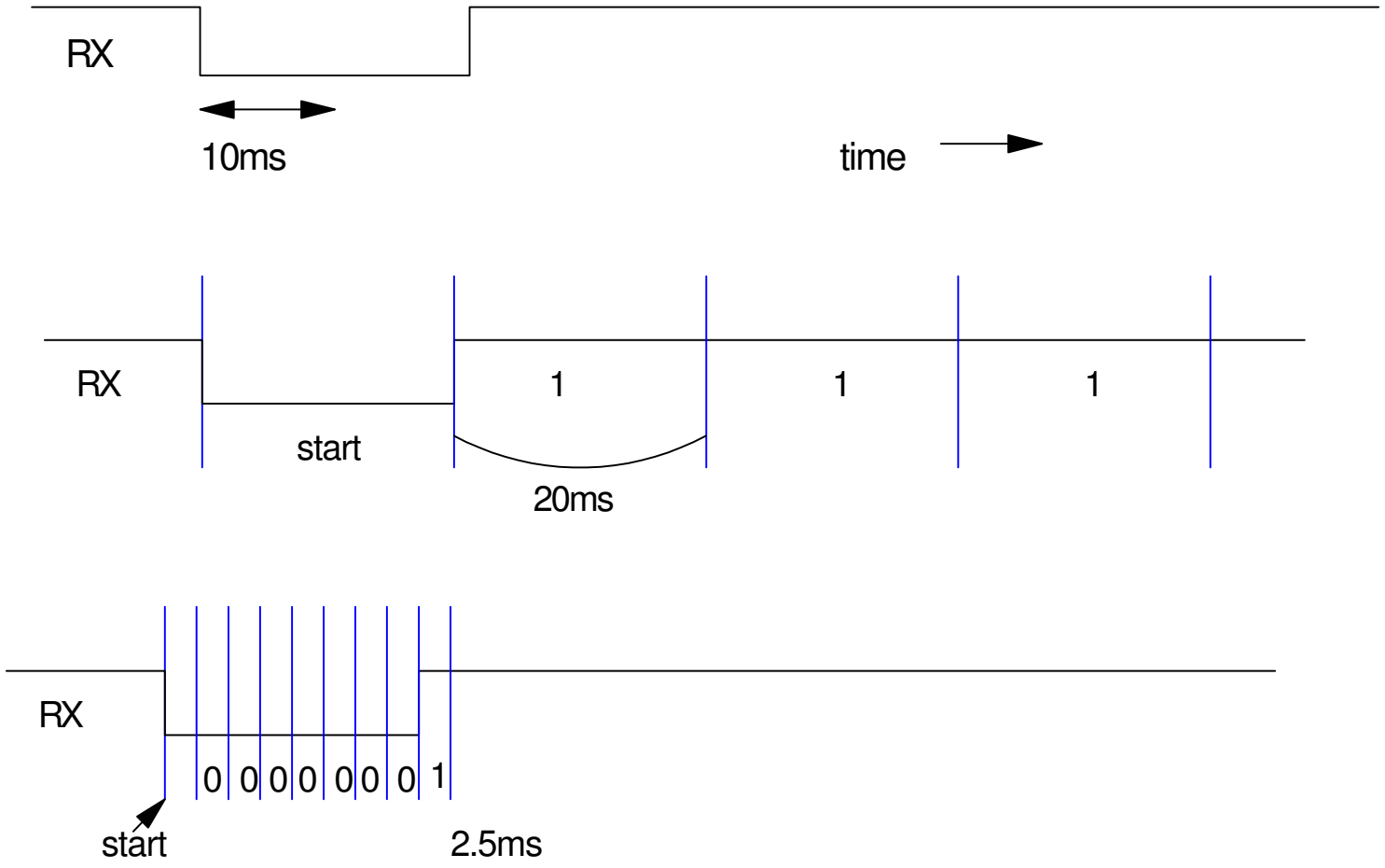
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| name | RX | TX | - | - | - | - | - | - |
| value | 1 | 1 | x | x | x | x | x | x |

note: Both transmit and receive are set up as input.

- When TXEN=1 (transmit enable), you override TRISC and make RC6 an output.
- When TXEN=0 (transmit disabled), RC6 returns to high-impedance. This allows someone else to drive the data line.

2. Set the baud rate.

- There is no clock, so the two devices **must** know how long each bit is.



# Baud Rates with 20MHz Crystals

| Baud Rate | SPBRG | BRGH | BRG16 | SYNC | Error (%) |
|---|---|---|---|---|---|
| 2400 | 255 | 0 | 1 | 0 | -1.70% |
| 4800 | 129 | 0 | 1 | 0 | -0.16% |
| 9600 | 255 | 1 | 1 | 0 | -1.70% |
| 19,200 | 129 | 1 | 1 | 0 | -0.16% |
| 38,400 | 64 | 1 | 1 | 0 | -0.16% |
| 57,600 | 42 | 1 | 1 | 0 | -0.95% |
| 115,200 | 21 | 1 | 1 | 0 | +1.44% |

# Initialization:

Initialize the SCI port to send and recieve data at 9600 baud:

```
void SCI_Init(void)
{
   TRISC = TRISC | 0xC0;
   TXIE = 0;
   RCIE = 0;
   BRGH = 1;
   BRG16 = 1;
   SYNC = 0;
   SPBRG = 255;
   TXSTA = 0x22;
   RCSTA = 0x90;
   }
```

# Display raw serial port data

SCI Interrupt:

- Reads in data from the PC
- Echos back each character as you type it in
- Saves the message in a buffer, and
- Looks for a carriage return <13> to terminate the message.

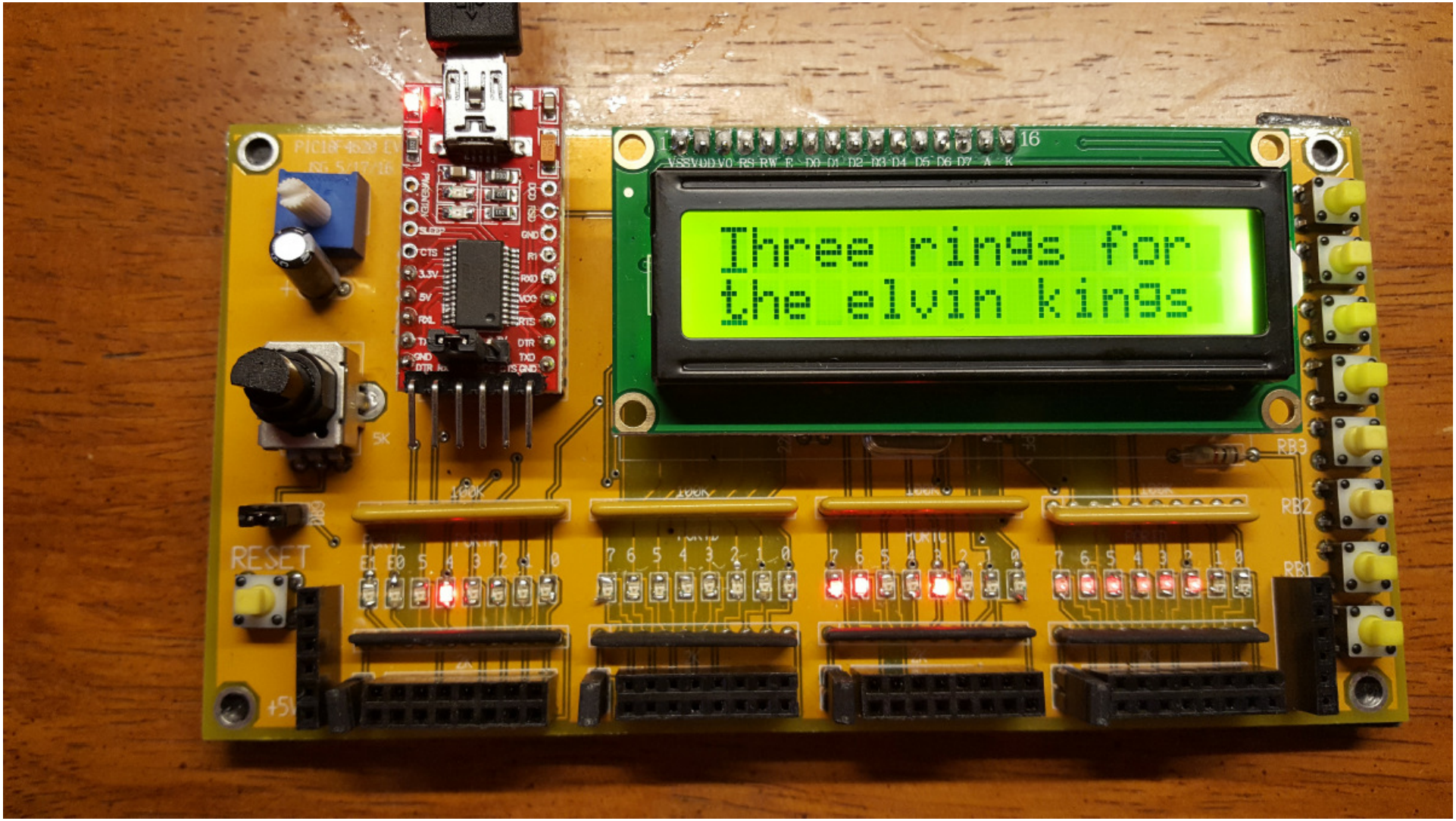
Since the data could arrive at any time

- Use interrupts to save the data.
- Use a stack to save the data as it comes in.
- Echo back what you read on the SCI port. If you connect to a PC, you should see Display the data on an LCD display.

# Interrupt Service Routine

Main routine displays MSG0[] and MSG1[]

```
void interrupt IntServe(void)
{
   if (RCIF) {
      TEMP = RCREG & 0x7F;
      TXREG = TEMP;
      if (TEMP > 20) MSG0[M++] = TEMP;
      if (M > 21) M = 21;
      if (TEMP == 13){
         for (i=M+1; i<21; i++) MSG1[i] = ' ';
         for (i=0; i<20; i++) {
            MSG1[i] = MSG0[i];
            MSG0[i] = ' ';
            }
         M = 0;
         }
      RCIF = 0;
      }
   }
```


three rings for
the elvin kings
RESET
RB3
RB2
RB1

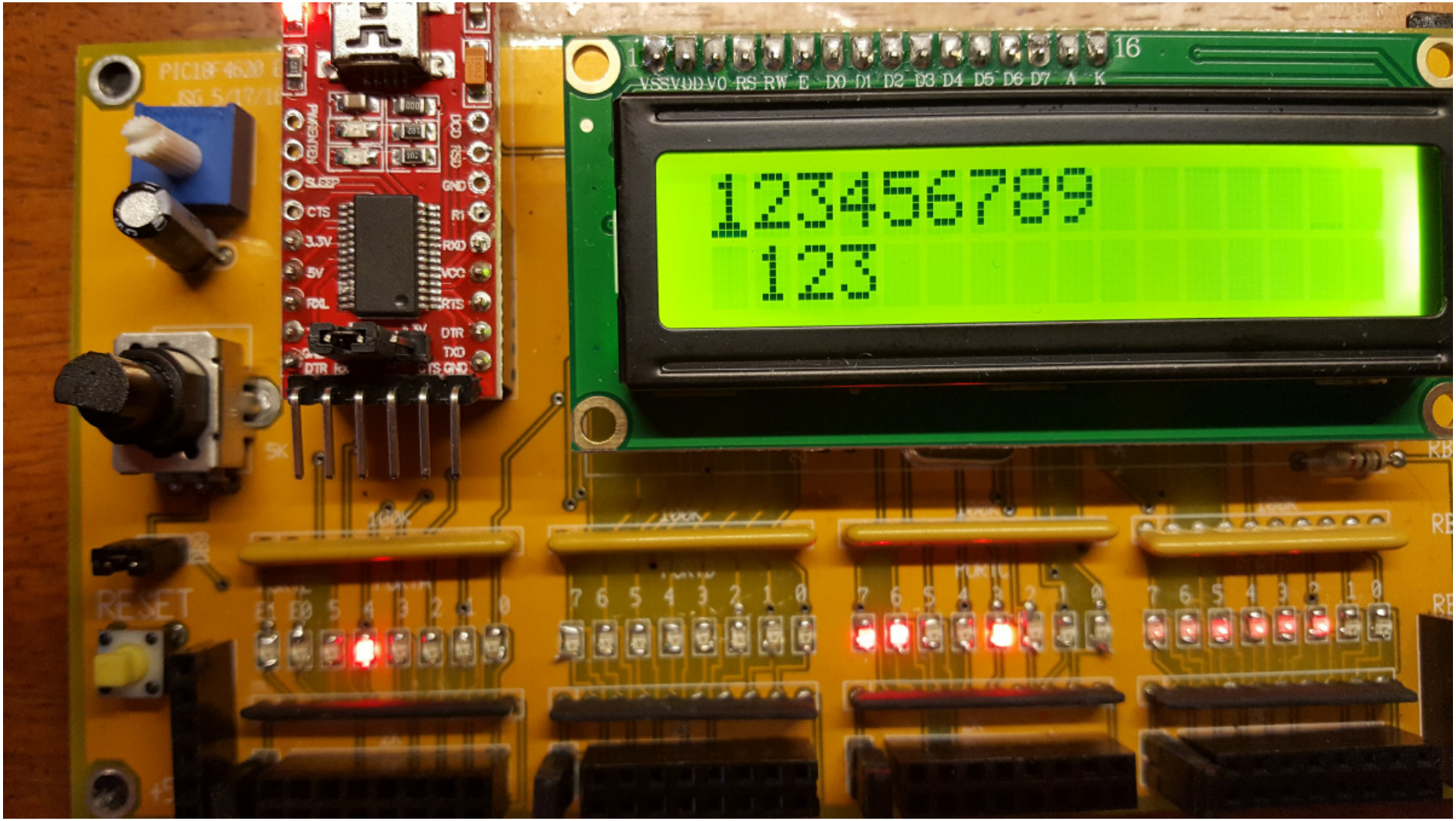
# Read 0000 to 9999 from the keyboard

FLAG tells the main routine that data is ready

```
void interrupt IntServe(void)
{
   if (RCIF) {
      TEMP = RCREG & 0x7F;
      TXREG = TEMP;
      if (TEMP > 20) MSG0[M++] = TEMP;
      if (M > 21) M = 21;
      if (TEMP == 13){
         FLAG = 1;
         for (i=M+1; i<21; i++) MSG1[i] = ' ';
         for (i=0; i<20; i++) {
            MSG1[i] = MSG0[i];
            MSG0[i] = ' ';
            }
         M = 0;
         }
      RCIF = 0;
      }
   }
```

# Main Routine

```
while(1) {
   LCD_Move(0,0);
   for(i=0; i<16; i++) LCD_Write(MSG1[i]);

   if(FLAG) {
      DATA = (MSG1[0] - 48) * 100
           + (MSG1[1] - 48) * 10
           + (MSG1[2] - 48);
      LCD_Move(1,0);
      LCD_Out(DATA, 3, 0);
      FLAG = 0;
      }
   }
```


VSS VDD VO RS RW E D0 D1 D2 D3 D4 D5 D6 D7 A K
1
16
123456789
123
RESET
SLEEP
CTS
3.3V
5V
GND
RI
RXD
VCC
RTS
DTR
TXD
GND

# GPS Data

GPS data is often transmitted using SCI protocol at 9600 baud

# Problems:

- How to read in the GPS data into a buffer, and
- How to parse the data, and
- How to convert to meters.

```
$GPGGA,152410.979,4731.42559,N,09233.10091,W,1,10,0.8,436.16,M,-30.59,M,
$GPGSA,A,3,15,05,08,29,27,18,21,26,06,22,,,1.4,0.8,1.1*30
$GPGSV,3,1,12,21,74,292,42,15,68,119,47,18,56,265,45,26,38,053,46*76
$GPGSV,3,2,12,48,25,230,23,29,25,190,39,06,24,310,39,27,18,120,42*7F
$GPGSV,3,3,12,03,15,319,,22,13,258,33,05,11,074,40,08,08,026,33*7D
$GPRMC,152410.979,A,4731.42559,N,09233.10091,W,0002.15,172.05,140312,,
$GPVTG,172.05,T,,M,0002.15,N,00003.98,K,A*08
$GPZDA,152410.979,14,03,2012,00,00*55
```

# Reading GPS data into a buffer:

Already done. Just use 80-bits per message.

# Parse the Data

```
$GPRMC,152410.979,A,4731.42559,N,09233.10091,W,0002.15,172.05,140312,,
```

| Field | Data | Meaning |
|---|---|---|
| 1 | GPRMC | Recommended minimum GPS data |
| 2 | 152410.979 | Time: 15:23:10.979 UTC |
| 3 | A | A = OK, V = warning |
| 4,5 | 4731.42559,N | Latitude: 47d 21.42559' |
| 6,7 | 09233.10091,W | Longitude: 092d 33.10091' |
| 8 | 0002.15 | Speed (knots) |
| 9 | 172.05 | Direction of motion (degrees) |
| 10 | 140312 | Date: 14:03:12<br>March 14, 2012 |

# Code:

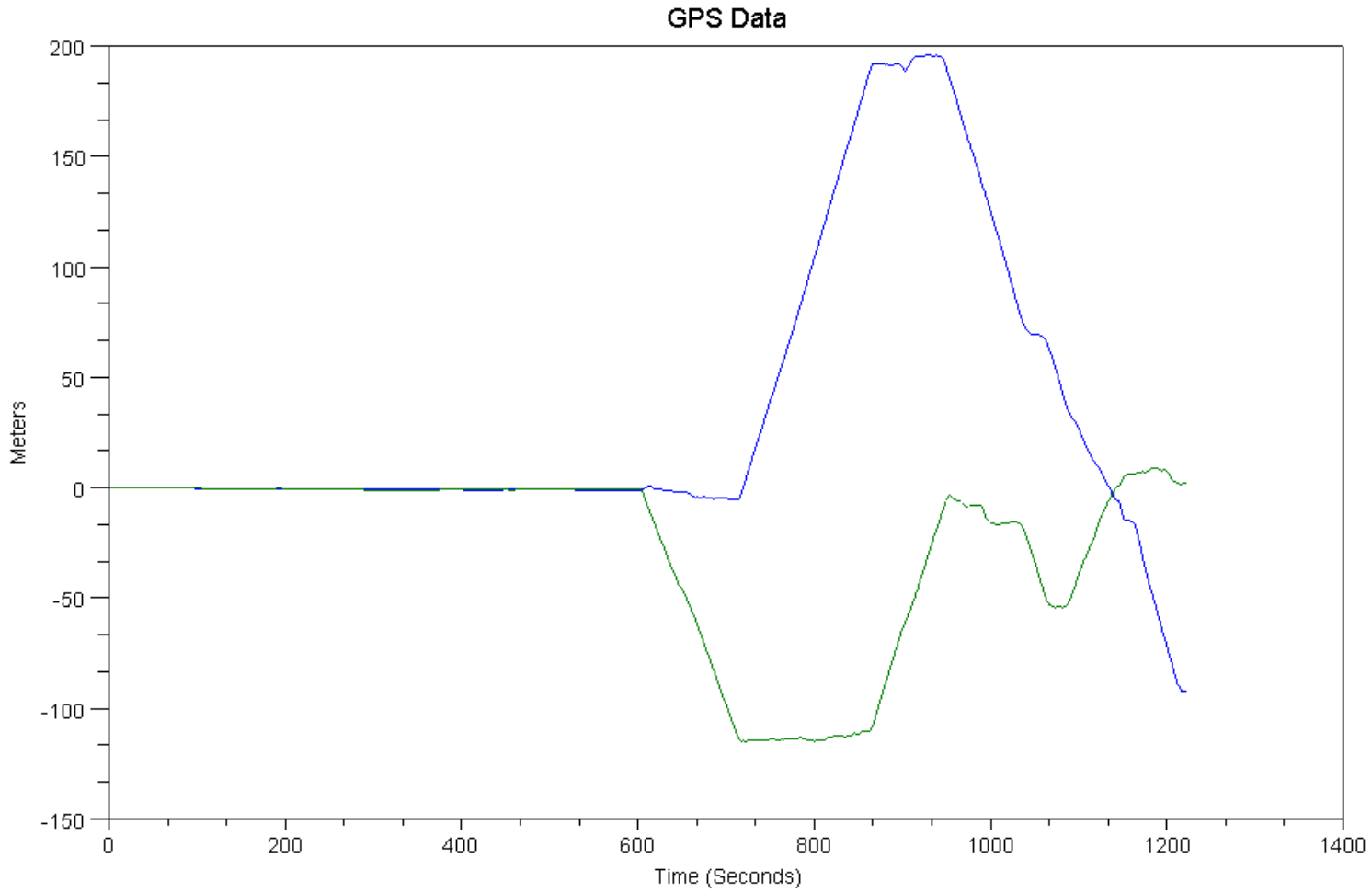
```
// Latitude in minutes

      LATITUDE = (GPS[20] - 48)*600 +
                 (GPS[21] - 48)*60  +
                 (GPS[23] - 48)*10 +
                 (GPS[24] - 48)*1 +
                 (GPS[25] - 48)*0.1 +
                 (GPS[26] - 48)*0.01 +
                 (GPS[27] - 48)*0.001;

// Longitude in minutes

      LONGITUDE = (GPS[33] - 48)*6000 +
                  (GPS[34] - 48)*600  +
                  (GPS[35] - 48)*60  +
                  (GPS[37] - 48)*10 +
                  (GPS[38] - 48)*1 +
                  (GPS[39] - 48)*0.1 +
                  (GPS[40] - 48)*0.01 +
                  (GPS[41] - 48)*0.001;
```

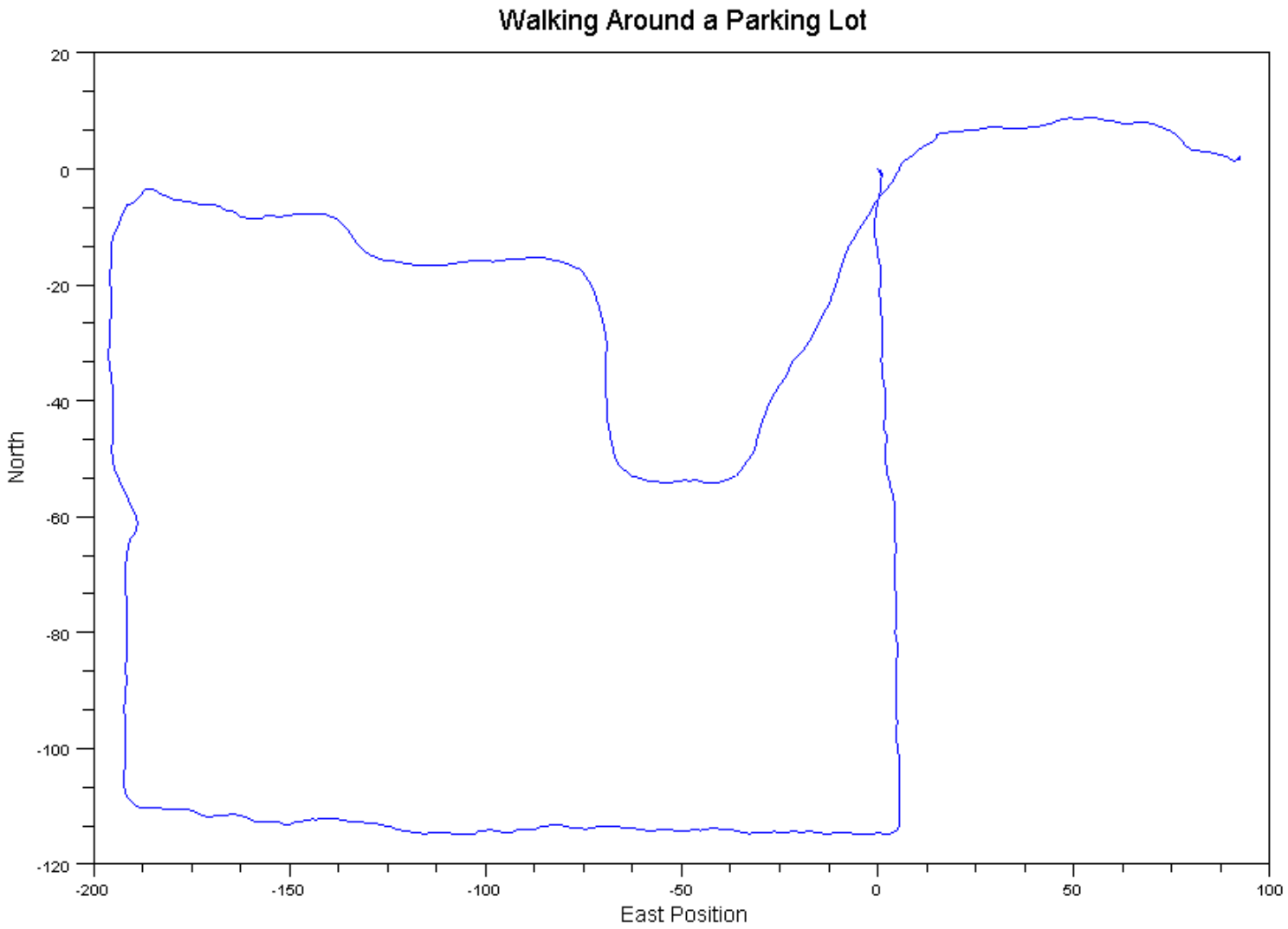
# Conversion to meters

If you want to convert to normal units, at Fargo, ND, this is

- Polar Radius = 6,356.8 km
  - 1 minute = 1849.12 meters
- Equatorial Radius = 6,378.1 km
  - 1 minute = 1,855.31 meters at the equator
  - 1 minute = 1,334.60 meters at 46 degrees north (Fargo)
- 1 Knot = 0.5144 meters/second

# Example:


GPS Data
200
150
100
50
0
-50
-100
-150
Meters
0
200
400
600
800
1000
1200
1400
Time (Seconds)


Walking Around a Parking Lot
20
0
-20
-40
-60
-80
-100
-120
North
-200
-150
-100
-50
0
50
100
East Position

# Data Encryption

Encrypt data as you transmit it

Caeser

- Send your message in Latin
- Shift each letter by 3
  - Easy to crack

```
void interrupt IntServe(void)

   if (RCIF) {

      TEMP = RCREG + 3;
      while (!TRMT);  TXREG = TEMP;

      // etc.

      RCIF = 0;
      }
   }
```

Three rings for the elvin

Wkuhh#ulqjv#iru#wkh#hoyl

# Data Encryption (take 2)

1-to-1 mapping of each letter

- Encryption key is also the decription key
  - Frequency of letters allows you to decrypt

```
void interrupt IntServe(void)

   if (RCIF) {

      TEMP = RCREG ^ 0x0F;
      while (!TRMT);  TXREG = TEMP;

      // etc.

      RCIF = 0;
      }
   }
```

Three rings for the elvin

[g}jj/}fah|/i`}/{gj/jcyfa/dfah|

# Data Encryption (take 3)

Use a different encryption key for each letter

- Scooby Doo:  "This book is the key!"
- "Three rings for the elvin kings under the sky..."

```
void interrupt IntServe(void)

   if (RCIF) {

      TEMP = RCREG ^ LOTR[i++];
      while (!TRMT);  TXREG = TEMP;

      // etc.

      RCIF = 0;
      }
   }
```

One ring to find them,

|gxfxj}oot}aag}eu}nfiamdcjm  f

# Data Encruption

Use a random number generator for the key

- Random number generators generate the same sequence given the same seed
- If you don't know the seed, you can't decipher the message
- Basis for Ultra in WWII

# Modern Data Encryption

Come up with an encryption scheme that even the programmer can't break