uAPP224 (v1.0) July 12,2005

Converting numbers in the CR1000 Data Logger to Pseudo Binary Format for transmitting using the MICROCOM DESIGN GTX Satellite Transmitter

Author: Richard Schwarz

PRELIMINARY

SUMMARY

The Microcom Model GTX Satellite Transmitter and Data Collector works on GOES, GMS, ARGOS, SCD & METEOSAT systems. The GTX has some data logger functions built into it, including an SDI-12 and counter input. The GTX can interface to external data acquisition systems like the Campbell Scientific Programmable Data Logger via its RS-232 port.

Microcom GTX Satellite Transmitter

Campbell Scientific Programmable Data Logger

For systems like the GOES DCS, a Pseudo Binary Data format has been specified. The Pseudo Binary format called out is a modified ASCII format utilizing 6 bits of an 8 bit character to represent part of each binary number. For data requiring 12 bit precision, two consecutive modified ASCII characters are needed (6 bits in each byte). Bits 7 and 8 of each character are a '1' and odd parity bit respectively. Data is always expressed by N characters, each representing N*6 bits of information. Data within a character is transmitted least significant bit first.

OddParity	1	2^5	2^4	2^3	2^2	2^1	2^0

Pseudo Binary Character Format

P	1	2^{11}	2^{10}	2^9	2^8	2^7	2^6	P	1	2^5	2^4	2^3	2^2	2^1	2^0

Pseudo Binary Character Format (12 bit format)

For signed parameters the value is expressed in two's complement form having a precision of (plus/minus) (Nx6-1). For example a value of 17 degrees below zero (-17) with 6 bits of precision is represented as 101111. A positive 17 would be 0100001.

INTRODUCTION

This Application note covers the conversion of numbers in the CR1000 data logger into the Pseudo Binary format.

A subroutine is introduced which can be used in CRBASIC programs to do the conversions into Pseudo Binary format.

The basic setup and connections for the CR1000 and GTX is referenced in the Microcom Application Note uAPP222.

User's of this application note should be familiar with, or have access to basic coding techniques, user's manuals and software associated with using the GTX and CR1000, including:

1) Microcom GTX User's Manual
2) Microcom GTX GUI Software
3) Campbel Scientific CR1000 User's Manual
4) Campbell Scientific LoggerNet Software

GOES DCS Pseudo-Binary Format Subroutine Code:

For GOES DCS, header and sensor data in Pseudo Binary has been specified. The data is transmitted in a modified ASCII format using 6 bits of an 8 bit char to represent part of each binary number. For data requiring 12 bit precision two consecutive modified ASCII characters are needed as shown in the example below:

3rD data character- **[P] [1] [X] [X] [X] [X] [2^{13}][2^{12}]**
2nd data character- **[P] [1] [2^{11}] [2^{10}] [2^{9}] [2^{8}] [2^{7}] [2^{6}]**
1st data character**-** **[P] [1] [2^{5}] [2^{4}] [2^{3}] [2^{2}] [2^{1}] [2^{0}]**

For 18 bit precision, three characters are required.

Note in each byte that bits 7 is a one and bit 8 is an odd parity bit.

For signed values the format is two's compliment.

So a conversion subroutine should be written in CRBASIC which will convert the incoming number into a PBN based on a certain precision and on the format (signed, unsigned) and an offset and multiplier.

Sub PBN(n, pb_num, format, Do_Parity)

Where:

n is the number to be converted whose max value is 14 bits.

Pb_num is the Pseudo Binary Number

Format: 0- positive value 1- signed twos complement

Do_Parity: -If set to 1 then include then compute and include the parity bit

MyArray – is a three byte string which stores the ASCII characters

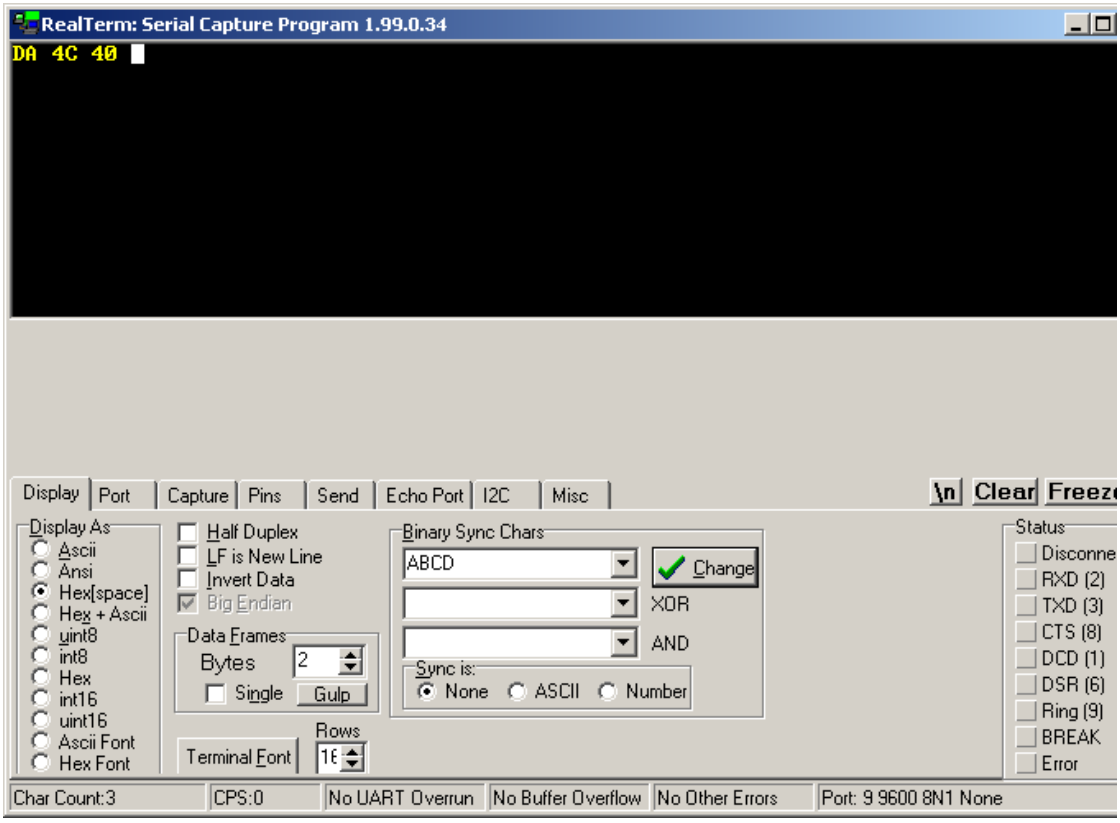
An example of sending out **MyArray** to a serial port is done by:

SerialOut(Com1, MyArray,"",0,0)

For example, lets say we read back a value of 794 in format number 0.

We assign n=794 and then call the routine as follows:

PBN(n, pb_num, 0, 1)



Our DEBUG terminal show that the conversion is :

404cDA hex

First we can see that each byte has an odd parity which is what is required.

First lets zero out the two most significant bits on each byte.
000C1A

Then lets combine the bits to get

31A which equals 794 decimal

Two's Complement Signed Values

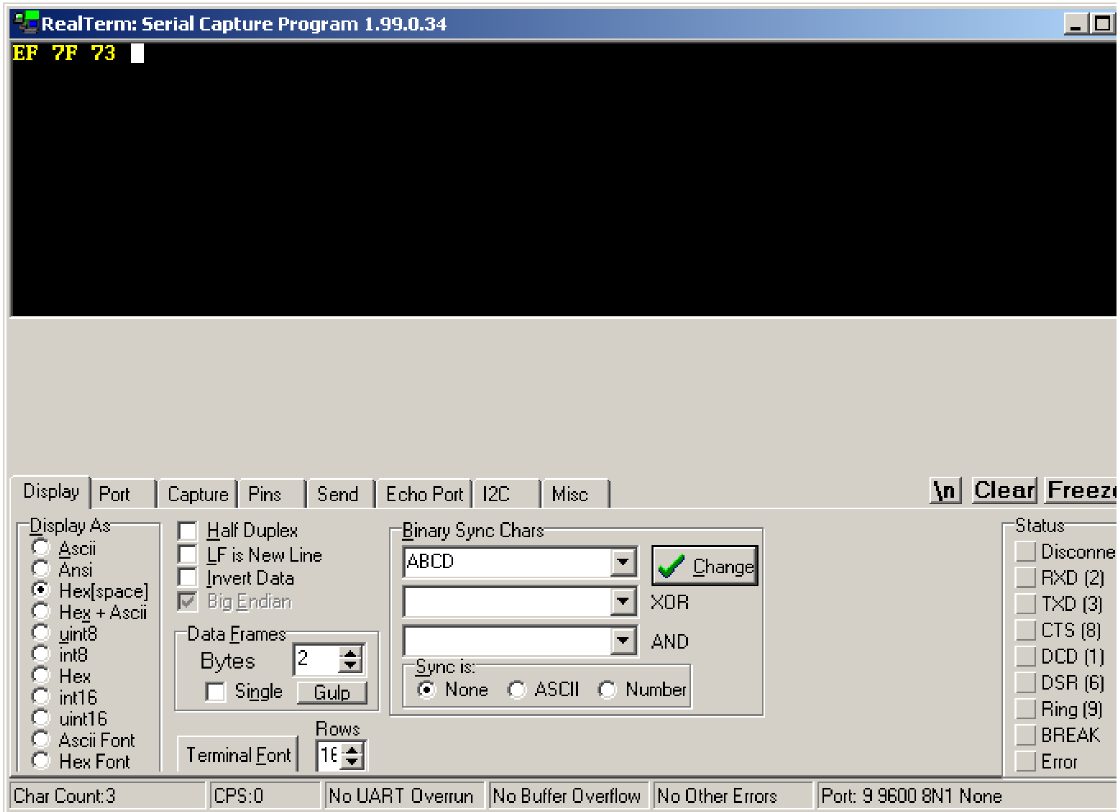
Now if we need to send a negative value, we use a two's complement value. For example if we wish to send a –17 we do the following:

We know that a 17 is represented as 010001. To convert this value to a 2's compliment –17 we simply invert all the bits, and then add one to get 101111. Since we have a maximum of 14 bits when we convert, the actual value is 11111111101111. After doing our Pseudo Binary Transformation we get

xxxxxx11 –111111 —101111
xxxxxx11-1111111-1101111
xxxxxx11011111111111101111
xxxxxx11 01111111 11101111 = 37FEF

filling out the remaining upper byte we get

using our program we see that we get:



73 7f EF

since the upper 6 bits are don't cares this matches our expectations.

```
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'S u b r o u t i n e''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Pseudo Binary Number: Takes n in (14 bits max) and converts the number into pseudo
'                binary format.
'
'       n= is the incoming number
'       pb_num is the converted number
'       format - 0- unsigned  1- signed (two's complement)
'       My_Array - 3 byte Pseudo Binary (14 bits valid)
'
Sub PBN(n, pb_num, format, Do_Parity)

if n=NAN then n=0

if format = 1 then 'signed twos comp
 'if n is negative
         if Debug =1 then SerialOut(Com1,"Number into PBN routine: ","",0,0)
                  if Debug =1 then   SerialOut(Com1,n,"",0,0)
         if Debug =1 then   SerialOut (Com1, CRLF,"",0,100)
         if (n < 0) then 'invert all bits and add one
            'invert all bits
            n= ABS(n)
```

```
                '  n= (n XOR 16777215) '(0xffffff)
                n= (n XOR 65535) '(0xffffff)
                        if Debug =1 then   SerialOut(Com1,"Number into PBN routine: ","",0,0)
            if Debug =1 then   SerialOut(Com1,n,"",0,0)
            if Debug =1 then   SerialOut (Com1, CRLF,"",0,100)
                'add one
                n=n+1
            endif
endif

                        if Debug =1 then   SerialOut(Com1,"Number into PBN routine: ","",0,0)
            if Debug =1 then   SerialOut(Com1,n,"",0,0)
            if Debug =1 then   SerialOut (Com1, CRLF,"",0,100)


  x=1
  y=1
  pb_num =0

  Do Until n < 1
      '     if Debug =1 then  SerialOut(Com1,x,"",0,0)
      '     if Debug =1 then  SerialOut(Com1," ","",0,0)
      '     if Debug =1 then  SerialOut(Com1,y,"",0,0)
      '     if Debug =1 then  SerialOut(Com1," ","",0,0)
      '     if Debug =1 then  SerialOut(Com1,n,"",0,0)

       ' check lsbit and accumulate resulting value
                        if (n Mod 2)>0 Then pb_num = (pb_num + y)
                                ' insure number is even
                                n = n AND 16777214
                                ' shift bits right
                                n = n / 2
                                'if next bit weight is a prohibited bit then shift y left 3 times (2 extra
spaces)
                                if ((x=32)or(x=64)or(x=16384)or(x=8192)or(x=4194304)or(x=2097152))
then
                                                                                                y =
x * 8
                                else 'else shift y value left one time
                                                                                                y =
x * 2
                                endif
                                'if in second byte or third byte then shift y left 2 additional
                                if x > 64 then y=y*4
                                'if in third byte then shift y left 2 more
                                if x > 1024 then y=y*4


                                x=x*2
                        '       if Debug =1 then SerialOut(Com1," ","",0,0)
                        '       if Debug =1 then SerialOut(Com1,pb_num,"",0,0)
                        '       if Debug =1 then SerialOut (Com1, CRLF,"",0,100)


            Loop
```

```
        ' Set bits 7, 14, 22 all high

        pb_num = pb_num OR 4210752 '(404040 hex)

                byte0 = pb_num AND 255
                byte1 = (pb_num AND 65280)/256
                byte2 = (pb_num AND 16711680)/65536


if (Do_Parity) then

        '  odd parity

        ' byte 0
        byte0 = pb_num AND 255
        temp= byte0
        parity = 0
        do until temp <1
                if (temp Mod 2)>0 Then parity = parity +1
                        ' insure number is even
                        temp = temp AND 16777214
                        ' shift bits right
                        temp = temp / 2
  loop

  '       if Debug =1 then SerialOut(Com1,"Parity 0: ","",0,0)
  '       if Debug =1 then SerialOut(Com1,parity,"",0,0)
   '      if Debug =1 then    SerialOut(Com1," ","",0,0)
   '      if Debug =1 then    SerialOut (Com1, CRLF,"",0,100)

  if ((parity AND 1) = 0 )  then
                        pb_num = pb_num OR 128 '(0x80 hex)
                        byte0 = byte0 OR 128
  endif

' byte 1
        byte1 = (pb_num AND 65280)/256
        temp = byte1
        parity = 0
        do until temp <1
                if (temp Mod 2)>0 Then parity = parity +1
                        ' insure number is even
                        temp = temp AND 16777214
                        ' shift bits right
                        temp = temp / 2
  loop

  if ((parity AND 1) = 0 )  then
                pb_num = pb_num OR 32768 '(0x8000 hex)
                byte1 = byte1 OR 128
  endif

' byte 2
        byte2 = (pb_num AND 16711680)/65536
        temp = byte2
        parity = 0
```

```
            do until temp <1
                    if (temp Mod 2)>0 Then parity = parity +1
                            ' insure number is even
                            temp = temp AND 16777214
                            ' shift bits right
                            temp = temp / 2



    loop

      if ((parity AND 1) = 0 )  then
                                    pb_num = pb_num OR 32768 '(0x8000 hex)
                    byte2 = byte2 OR 128
            endif
endif
            MyArray= CHR(byte0)+CHR(byte1)+CHR(byte2)

                    if Debug =1 then SerialOut(Com1, " Pseudo Binary Out: "+MyArray,"",0,0)
            if Debug =1 then SerialOut (Com1, CRLF,"",0,100)



endsub
```

GOES DCS Pseudo-Binary Format Example Application

```
' Program: PBNExample
' Description: Pseudo Binary Subroutine Example Code Snippet
'
'program author: R. Schwarz  Microcom Design Incorporated
'program date: Original Code May 1, 2005
'
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'Declare Public Variables''''''''''''''''''''''''''''''''''''''''''''''''''''''''
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

Public Mess1 as string * 100
Public Mess2 as string * 100
Public n as long
Public y,x,pb_num,format, offset,multiplier,error
Public temp,parity,byte0,byte1,byte2
Public CR as string * 1
Public LF as string * 1
Public CRLF as string * 2
Public MyArray as string * 4
Public Debug

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'Define Subroutines''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

' Pseudo Binary Number: Takes n in (14 bits max) and converts the number into pseudo
' binary format.
Sub PBN(n, pb_num, format, Do_Parity)

if format = 1 then 'signed twos comp
  'if n is negative
 ' if Debug =1 then SerialOut(Com1," N in: ","",0,0)
```

```
 'if Debug =1 then   SerialOut(Com1,n,"",0,0)
 'if Debug =1 then   SerialOut (Com1, CRLF,"",0,100)
  if (n < 0) then 'invert all bits and add one
     'invert all bits
       n= ABS(n)
     '  n= (n XOR 16777215) '(0xffffff)
      n= (n XOR 65535) '(0xffffff)
     'if Debug =1 then   SerialOut(Com1,"N out:  ","",0,0)
      'if Debug =1 then   SerialOut(Com1,n,"",0,0)
      'if Debug =1 then   SerialOut (Com1, CRLF,"",0,100)
      'add one
          n=n+1
    endif
endif

   x=1
   y=1
   pb_num =0

   Do Until n < 1
       '                              if Debug =1 then  SerialOut(Com1,x,"",0,0)
       '                             if Debug =1 then  SerialOut(Com1," ","",0,0)
       '                              if Debug =1 then  SerialOut(Com1,y,"",0,0)
       '                             if Debug =1 then  SerialOut(Com1," ","",0,0)
       '                              if Debug =1 then  SerialOut(Com1,n,"",0,0)

        ' check lsbit and accumulate resulting value
    if (n Mod 2)>0 Then pb_num = (pb_num + y)

'if Debug =1 then SerialOut(Com1," ","",0,0)
'                              if Debug =1 then SerialOut(Com1,pb_num,"",0,0)
'                              if Debug =1 then SerialOut (Com1, CRLF,"",0,100)
   Loop

   ' Set bits 7, 14, 22 all high
    pb_num = pb_num OR 4210752 '(404040 hex)

   '  odd parity
    ' byte 0
   byte0 = pb_num AND 255
   temp= byte0
   parity = 0
   do until temp <1
   if (temp Mod 2)>0 Then parity = parity +1
    ' insure number is even
     temp = temp AND 16777214
   ' shift bits right
    temp = temp / 2
    loop

    '                             if Debug =1 then SerialOut(Com1,"Parity 0: ","",0,0)
    '                               if Debug =1 then SerialOut(Com1,parity,"",0,0)
     '                               if Debug =1 then     SerialOut(Com1," ","",0,0)
     '                            if Debug =1 then     SerialOut (Com1, CRLF,"",0,100)
   if ((parity AND 1) = 0 )  then
         byte0 = byte0 OR 128
```

```
  endif

' byte 1
 byte1 = (pb_num AND 65280)/256
 temp = byte1
 parity = 0
 do until temp <1
 if (temp Mod 2)>0 Then parity = parity +1
   ' insure number is even
   temp = temp AND 16777214
  ' shift bits right
   temp = temp / 2
  loop

  if ((parity AND 1) = 0 )  then
                                            pb_num = pb_num OR 32768 '(0x8000 h
                                                      byte1 = byte1 OR
  endif

' byte 2
                           byte2 = (pb_num AND 16711680)/65536
                                 temp = byte2
                                  parity = 0
                                do until temp <1
                                        if (temp Mod 2)>0 Then parity = parity



  loop

  if ((parity AND 1) = 0 )  then

                                                      byte2 = byte2 OR
                                    endif
                          MyArray= CHR(byte0)+CHR(byte1)+CHR(byte2)

                                      if Debug =1 then SerialOut(Com1, MyArray,""




endsub



''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'Main Program''''''''''''''''''''''''''''''''''''''''''''''''''''''''
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
BeginProg


' Use Com1 as an ECHO output port to viwe on Com1
 SerialOpen (Com1,9600,0,0,1000)
' Setup Com4 serial port to GTX
```

```
  SerialOpen (Com4,9600,0,0,1000)
' Setup Com4 serial port to GTX
'  SerialOpen (ComRS232,9600,0,0,1000)
   CR=CHR(13)
   LF=CHR(10)
   CRLF=CHR(13)+CHR(10)
   Debug =1

'Execution loop
Scan(2,sec,1,0)
n= -17
PBN(n, pb_num, 1,1)
'if Debug =1 then SerialOut(Com1,pb_num,"",0,0)
'if Debug =1 then SerialOut (Com1, CRLF,"",0,100)
NextScan


EndProg
```

Conclusion

For systems like the GOES DCS, a Pseudo Binary Data format is called out. The Pseudo Binary format called out is a modified ASCII format utilizing 6 bits of an 8 bit character to represent part of each binary number. This Application Note presented a subroutine which the CR1000 could use to translate numbers into the Pseudo Binary format.

Revision History

Date	Version	Revision
July 12, 2005	1.0	Initial Microcom Preliminary Release