RF Datalink using a PICAXE UART

Written by Administrator
Sunday, 11 January 2009 08:30

Living on a farm presents some interesting opportunities for electronic projects. My current work in progress is wireless monitoring of the water levels in the large 5000 Gallons tanks we have on the property.

Many years back I ordered some simple 433.920 mhz transmitter and receiver modules (similar to the RF Link Transmitter and Receiver from SparkFun Electronics) for the parts bin. I didn't have a specific project in mind although at the time did some basic experimentation with limited success..... partly becaue I didn't really have the time to coding a robust manchester encoding scheme to ensure the receiver DC balance was correct for the bit slicer.

The water tank monitoring project got the parts out of the bin and back on the workbench.... but how to build a reliable RF link with the least amount of effort?

Firstly a statement of requirements:

- Single direction data link
- Small amount of data to transfer (4 bytes)
- Multiple transmitters sharing the same frequency
- The receiver is centrally located on a high location but the transmission paths can be up to 300 meters (clear line of sight).

The solution was to use low cost/easy to program PICAXE 08M parts.

Each tank has a PICAXE 08M which reads the tank level and periodically builds a data packet and transmits it on 433.920 mhz.

There is a central 433.920 mhz receiver which has a PICAXE 18X (only because it was in the junkbox... other parts could equally be used) dedicated to decoding the received packets and for valid packets outputting a serial data message to be recorded in the database.

To make this work

- Data must be sent from transmitters to the receiver in packets.
- Each packet must have an ID so the receiver knows which transmitter sent it.
- The receiver must have some way of validating that the packet received is valid and complete.
- The packet assembly and disassembly must be performed by a part with 256 bytes of code space and a limited programming language.

The message format I used is as follows:

PreAmble	$55,$55,$55,$55,$55,$55
Packet Header	$FF,$00,$01,$7F
Transmitter ID	STATIONID,INVSTATIONID
Transmitter Data	TANKLEVELH,INVTANKLEVELH
	TANKLEVELL,INVTANKLEVELL
	PKTCOUNT,INVPKTCOUNT
Packet Checksum	CHECKSUM,INVCHECKSUM
Finalization	$AA,$AA,$AA,$AA

The PreAmble is sent before the data packet to set the receiver DC balance for the packet that follows.

The Packet Header is used by the receiver to identify the start of what might be a valid packet. Having received a header the receiver reads the next 8 bytes which comprise the packet of data. Each byte that is sent is immediately followed by it's inverse. I have found that up to 2400 baud this works fine to maintain the receiver DC balance and ensure reliable decoding (assuming the received signal level is sufficient).

Following the 8 bytes of data is the packet checksum. The packet checksum is the low byte of the sum of STATIONID, TANKLEVELH, TANKLEVELL and PKTCOUNT.

If the packet checksum is valid for the received data packet then the receiver PICAXE accepts this as a valid packet.

This is by no means a scheme that for high performance data transfer but for simple single direction links using low cost parts I have found it to work well.

The great thing is that PICAXE 08M Software UART does all the sending and receiving.

I use the PICAXE SERIN command to receive the data. SERIN handles fixed length packets very well including the search that identifies the packet header but the downside is that it cannot be interrupted while waiting for a new packet. Unfortunately this means the receiving PICAXE is dedicated to decoding incoming packets and cannot do other things. To do that would require coding closer to the hardware (not PICAXE basic), writing your own packet management code and probably a hardware UART that generates interrupts as new bytes are received.

My objective was to get something workable and reliable. That I have achieved. Use fewer parts and you get to write more code.

Here is the code for those who want to try something similar. I can't see why this technique would not work with any microcontroller.

```
' PICAXE 08M Tank Level Transmitter
'
' Pin Assignments:
' 0 = (Out) Serial
' 1 = (In) Tank Level ADC
' 2 = (Out) Tramsitter Serial Data
' 3 = (In)
' 4 = (Out) Power Up Control for Transmitter and LM324

#picaxe 08m

' Important - Change this for the device being installed
'
' Current assignments as follows"
'
' $00 = Big Shed 25000 litre tank
' $01 = House 25000 litre tank
' $02 = Bore tank


symbol STATIONID = $00

' Constants - Pin Assignments

symbol TANKLEVEL = 1
symbol TXDATA = 2
symbol POWER = 4

' Constants - Other

symbol SENDDELAY_H = $300
symbol SENDDELAY_L = $5
symbol FASTTXLIMIT = 48
symbol POWERSETTLE = 150

' Variable Usage

symbol TANKLEVELW = w0
symbol TANKLEVELL = b0
symbol TANKLEVELH = b1
symbol INVTANKLEVELL = b2
symbol INVTANKLEVELH = b3
symbol PKTCOUNT = b4
symbol INVPKTCOUNT = b5
symbol FASTTXCOUNT = b6
symbol INVSTATIONID = b8
symbol CHECKSUM = b9
symbol INVCHECKSUM = b10

PKTCOUNT = 0
FASTTXCOUNT = 0

main:

' Power Up the Transmitter and LM324

low POWER
pause POWERSETTLE


' Read ADC for tank level

readadc10 TANKLEVEL,TANKLEVELW
sertxd("Tank Level : ",#TANKLEVELW,CR,LF)
```

```
' Transmit to base

gosub transmit

' Shutdown the Transmitter and LM324

high POWER

' Short delay is sent on startup until it has been sent STARTUPCOUNT times and then reverts to the long delay

if FASTTXCOUNT > FASTTXLIMIT then maindelay_h
sleep SENDDELAY_L
FASTTXCOUNT = FASTTXCOUNT + 1
goto main2
maindelay_h:
sleep SENDDELAY_H
main2:

' Add extra random delay to keep stations out of sync

nap STATIONID
goto main

transmit:

' Transmits lead $55 bytes followed by station id and tank level high/low bytes
' All data transmitted is followed by it's inverse to keen DC level correct for receiver

PKTCOUNT = PKTCOUNT + 1
INVPKTCOUNT = $FF-PKTCOUNT
INVTANKLEVELH = $FF-TANKLEVELH
INVTANKLEVELL = $FF-TANKLEVELL
INVSTATIONID = $FF-STATIONID

CHECKSUM = STATIONID + TANKLEVELH + TANKLEVELL + PKTCOUNT
INVCHECKSUM = $FF-CHECKSUM

' Send data

serout TXDATA,N600,($55, $55, $55, $55, $55, $55, $FF, $00, $01, $7F, STATIONID, INVSTATIONID, TANKLEVELH, INVTANKLEVELH,
TANKLEVELL, INVTANKLEVELL, PKTCOUNT, INVPKTCOUNT, CHECKSUM, INVCHECKSUM, $AA, $AA, $AA, $AA)

' Make sure transmitter is off at the end

low TXDATA
return
```

```
' PICAXE 18 Tank Receiver
'
'
'
' Pin Assignments:
' In0 =
' In1 = Transmitter Serial In
' In2 =
' In6 =
' In7 =
' Out0 =
' Out1 =
' Out2 = Decoded Packet Data Out
' Out3 =
' Out4 = Decode OK LED
' Out5 =
' Out6 =
' Out7 =
```

```
#picaxe 18x

' Constants - Other

symbol RAMBASE = $50
symbol TXMAX = $07
symbol TXBUFFER = $03
symbol VALIDRX = $04
symbol DECODEOUT = $02

' Variables

symbol TANKLEVELW = w0
symbol TANKLEVELL = b0
symbol TANKLEVELH = b1
symbol INVTANKLEVELL = b2
symbol INVTANKLEVELH = b3
symbol PKTCOUNT = b4
symbol INVPKTCOUNT = b5
symbol CHECKSUM = b8
symbol INVCHECKSUM = b9
symbol STATIONID = b10
symbol INVSTATIONID = b11
symbol TEMP1 = b12
symbol TEMP2 = b13

init:
' Initialize the RAM so there is not data from any station - 2 bytes per station

FOR TEMP1 = 0 TO TXMAX
TEMP2 = TEMP1 * TXBUFFER
TEMP2 = TEMP2 + RAMBASE
poke TEMP2,0
TEMP2 = TEMP2 + 1
poke TEMP2,0
TEMP2 = TEMP2 + 1
poke TEMP2,0
NEXT

rxloop:
serin 1,N600,($FF, $00, $01, $7F),STATIONID, INVSTATIONID, TANKLEVELH, INVTANKLEVELH, TANKLEVELL, INVTANKLEVELL, PKTCOUNT,
INVPKTCOUNT, CHECKSUM, INVCHECKSUM

' Check if the Packet Received is valid

TEMP1 = STATIONID + TANKLEVELH + TANKLEVELL + PKTCOUNT
IF TEMP1 = CHECKSUM THEN rxok

rxinvalid:
sertxd("Invalid Packet - Checksum Invalid",13,10)
goto rxloop

rxok:
' Debugging Information

toggle VALIDRX
sertxd("Valid Packet Received",CR,LF)
sertxd("Station ID : ",#STATIONID,CR,LF)
sertxd("Tank Level : ",#TANKLEVELW,CR,LF)
sertxd("CheckSum : ",#CHECKSUM,CR,LF)

' Save the new data into RAM

TEMP1 = STATIONID
TEMP2 = TEMP1 * TXBUFFER
```

```
TEMP2 = TEMP2 + RAMBASE
poke TEMP2,TANKLEVELH
TEMP2 =TEMP2 + 1
poke TEMP2,TANKLEVELL
TEMP2 = TEMP2 + 1
poke TEMP2,PKTCOUNT

' Write all data in RAM to the serial Port

serout DECODEOUT,T4800,($2B)

' Remote Station 0
FOR TEMP1 = 0 TO TXMAX
TEMP2 = TEMP1 * TXBUFFER
TEMP2 = TEMP2 + RAMBASE
peek TEMP2,TANKLEVELH,TANKLEVELL,PKTCOUNT
serout DECODEOUT,T4800,(TANKLEVELH,TANKLEVELL,PKTCOUNT)
NEXT
serout DECODEOUT,T4800,(CR,LF)

goto rxloop
```