|  |
| --- |
| GNB002 V2 |
| Generic Network Interconnect Board |
| This manual describes GNB02, the generic network interconnect board. The board is a printed circuit board for use as a generic embedded interconnect hub for a Half Duplex RS-485 network. The board provides RS-232, RS-485 and TTL interfaces while interconnecting to an XBee radio. The board supports an XBee embedded radio with currents up to 500 Ma. |

6/22/2014

Cranwell & Associates, Inc.

GNB002

Reference Guide Manual

Version 2.0

© Copyright 2012.

This document and related design, is property of Peter C Cranwell. The document and design may be used for non-commercial purposes.

The Board Design (Eagle Files) may be obtained by sending an email to [pete@pcranwell.com](mailto:pete@pcranwell.com).

Cranwell & Associates, Inc..

**GNB002 V2**

**Reference Manual**

© 2012, Cranwell & Associates, Inc.

Phone (540) 864-6517

Contents

[Overview. 4](#_Toc298851380)

[Board Layout 6](#_Toc298851381)

[Jumper Settings 7](#_Toc298851382)

[Schematic 8](#_Toc298851383)

[Power Requirements 9](#_Toc298851384)

[Supply Voltage 9](#_Toc298851385)

[Current Requirements 9](#_Toc298851386)

[Source Code for PIC12F683 10](#_Toc298851387)

# Overview.

The GNB002 is a generic printed circuit board with interfaces to RS-232, RS-485, TTL and an XBee or XBee Pro radio.

The board accepts +6 - +35 volts input to power the circuitry and radios. In receive, the modem draws 70 Ma, in Transmit it draws 240 Ma (at the highest power level).

Interfaces to RS-232, RS-485 and TTL are provided through male headers and board function is selected with jumpers connected to two function selection Male headers.

The board operates in one of the following four modes:

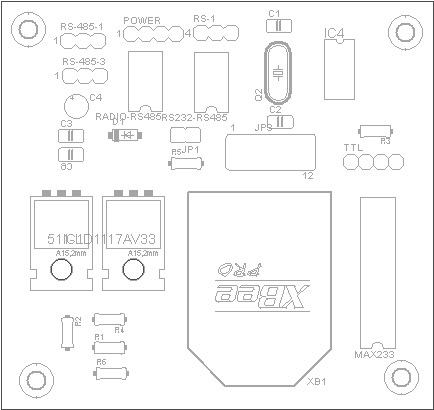
1. Network mode where RS-232, RS-485 and Radio networks interoperate in Half Duplex mode.
2. Radio Program mode where the RS-232 interface is used to program the radio.
3. TTL – Radio mode where the TTL interface controls the radio.
4. RS-232-RS-485 where the two interfaces operate in Half Duplex (2 wire) mode.

The GRB002 develops the necessary RS-485 transmit enable signal from the RS-232 data lines. This is done with a Microchip PIC12F683 microprocessor. At the current implementation, the minimum speed of operation is 9600 BPS. Slower speeds are possible by loading different software into the PIC microprocessor.

The board incorporates fail safe RS-485 biasing. The board is terminated with a 100 Ohm resistor and expects the network to have a single 100 ohm termination resistor at the end of the network chain. These values were selected to operate with a CAT-5 twisted pair network with a characteristic Impedance of 100 Ohms. When properly terminated, the RS-485 A and B lines operate with a fail-safe 200 Millivolt difference.

## 

## Board Layout



## Jumper Settings

JP1 Header jumper used to connect the RS-485 transceiver to the RS-232 interface.

JP3 Mode configuration Header jumper.

The following diagram depicts JP3 looking at the board from Top down with the, TTL input positioned towards the viewer.

JP3 Positions (Top view)

(2) (1)

(4) (3)

(6) (5)

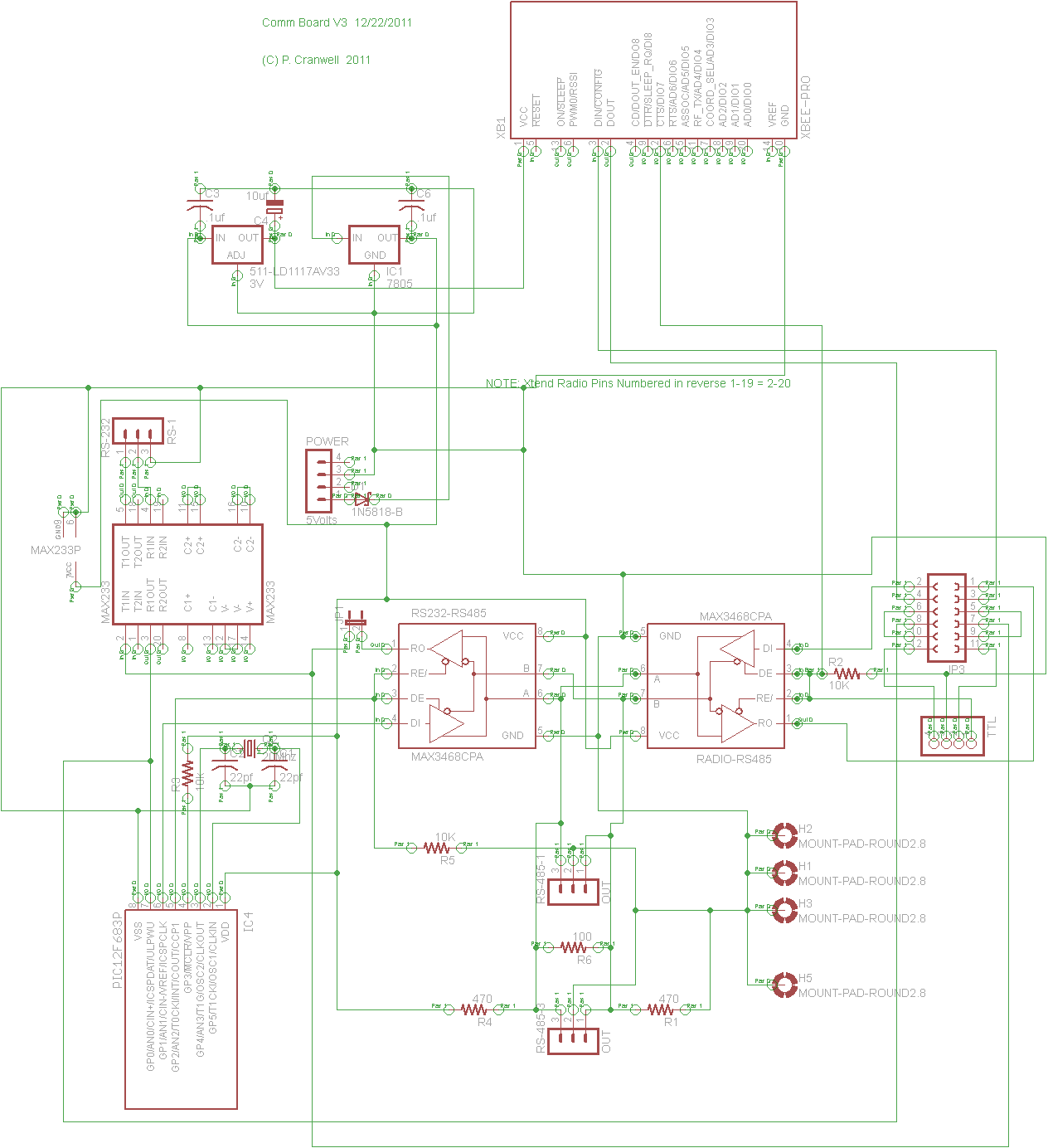
(8) (7)

(10) (9)

(12) (11)

|  |  |
| --- | --- |
| **Mode** | **Jumper Settings** |
| Network Mode  RS-485 & RS-232 - Radio | (JP1) Closed, (JP3) 1-3, 2-4 |
| Program Radio Mode  RS-232 - Radio | (JP1) Open, (JP3) 3-6, 4-5, 7-9, 8-10 |
| TTL – Radio Mode  TTL - Radio | (JP1) Open, (JP3) 3-6, 4-5, 10-12, 11-19 |
| RS-232 – RS-485 | (JP1) Closed (JP3) Open |

## Schematic

****

## Power Requirements

### Supply Voltage

Power requirements depend upon the mode of operation and the source voltage.

The board will operate at between 6 and 35 volts. At least 6 volts is required to supply the 5 Volt regulator and the maximum input voltage is determined by the maximum value supported by the 7805 voltage regulator. The board should preferable be powered from a 12 volt supply. Higher input voltages cause the 5 volt regulator to dissipate more power.

If a 6 volt sully is used, then the “Idiot” diode in the board supply should be bypassed so that the voltage regulators can be supplied with a full 6 volts input.

### Current Requirements

The current requirements depend upon the mode that the board operates in and the radios used.

For non-radio applications (RS-232 – RS-485) the board draws 20 Ma.

For XBee Pro radios the receive mode current is 65 Ma, the Transmit current is 235 Ma.

## Source Code for PIC12F683

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; File Name : RTS RS-485 Transmit Enable

; Version : 2.0

; Description : Generates RTS signal for RS-485 from RS-232 Data line.

; Author : Peter Cranwell

; Target : Microchip PIC 12F683 Microcontroller

; Compiler : Microchip Assembler (MPASM)

; IDE : Microchip MPLAB IDE v8.00

; Programmer : PICKIT2

; Last Updated : 6/22/2014

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;

; RS/232 - RS-485 Transmit Data Enable

; (C) 2011 P. Cranwell

;

; Port Usage

;

; GP0 Date Input Pin 7

; GP1 Data Output Pin 6

; GP2 Data Enable Pin 5

;

; NOTE: TTL levels are inverted from RS-232 levels. A TTL level Start Pulse is a SPACE = 0 Level.

; A TTL Level STOP bit ia a Mark = + Level.

;

;

; The device repeats the signal level on the input pin & propagates it to the output pin.

;

; Upon receipt of a 0 level (Start bit or data bit) the Transmit Enable output is raised and the timer is turned on.

; The transmit enable is raised 8 microseconds before the data is repeated (at a clock speed of 20MHZ). This is 1/2 bit time at 56KBPS.

;

;

; Timer1 ioperates at FOSC/4 = @5MHZ or .2 microseconds per timer count.

;

; 1 Byte time @ 38400 BPS = 260 Microseconds

;

; 1 Byte time @ 9600 BPS = 1041 Microseconds

; Time to drop DE = 1 Byte time Timer Count. = X'1455' (FFFF - 1455 = E6AA)

;

;

;

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Bank0 macro

bcf STATUS,RP0

endm

Bank1 macro

bsf STATUS,RP0

endm

#include <p12F683.inc>

; Compiler configuration bits set to override the config setting.

; The hardware is using a 20 MHZ crystal.

; Data definitions follow

cblock 0x20

TimerLow ; Define two 8-bit timer values

TimerHigh

endc

org 0 ; Main Loop

goto Start

; Interrupts servied here

org 4 ; Interrupt Vector

Bank1

; check to see what type of interrupe. Either Port change or Timer

btfss INTCON,GPIF ; skip if IO Interrupt

goto TimerPop

; I/O interrupt logic

; get GP0 latch value into W Reg

Bank0

btfsc GPIO,GP0

goto Mark

; SPACE Logic is entered on Space and Start Bit = 0 Level.

; Bank0 already selected

bsf GPIO,GP2 ; Enable RS-485 Transmit & RS-232 RTS

; initialize timer Count for 9600 BPS.

; This will work at higher speeds if peer uses slight delay before responding.

;

movlw 0xE6

movwf TMR1H

movlw 0xAA

movwf TMR1L

; Enable Timer

movlw 0x05

movwf T1CON

goto MarkAndSpace

Mark:

; Entered on Space and Stop Bit = + Level

; Keep Mark and Space pathlengths the same

nop

nop

MarkAndSpace:

; Echo Input Latch data to Output Latch at GP0

Bank0

btfss GPIO,GP0

goto SetLow

bsf GPIO,GP1

goto Done

SetLow

bcf GPIO,GP1

Done:

Bank1

bcf INTCON,GPIF ; Clear interrupt

retfie

TimerPop:

; bank 1 already selected

bcf INTCON,GPIF

Bank0

bcf GPIO,GP2 ; Drop RTS

bcf T1CON,TMR1ON ; Turn Off Timer

bcf PIR1,TMR1IF

retfie

org 0x100

Start:

;Setup chip options, enable timer and IOC interrupts and then sleep until interrupted.

Bank0

movlw 0x07

movwf CMCON0 ; Turn Comparators off

movlw 0xC8 ; GIE=1, PEIE=1, GPIE=1, GPIF=0

movwf INTCON

Bank1

movlw 0x00 ; GP0 is input, all other are output

movwf TRISIO

clrf ANSEL ; Make all ports as digital I/O

Bank0

movlw 0x03 ; Initialize Input & Output to +, Transmit Enable to 0

movwf GPIO

Bank1

movlw 0x01 ; GP0 is input, all other are output

movwf TRISIO

bsf IOC,IOC0 ; GP0 Interrupt on Change

bcf T1CON,TMR1ON ; Turn Off Timer

bsf PIE1,TMR1IE ; Enable Timer interrupts

Loopforever:

; the following is just a test

movwf 0x02

goto Loopforever

end; EOF: RTS