Embedded Control Design Contest – Entry MT2267

NUE-PSK

A digital modem for PSK31 field operation… without using a PC!

ABSTRACT

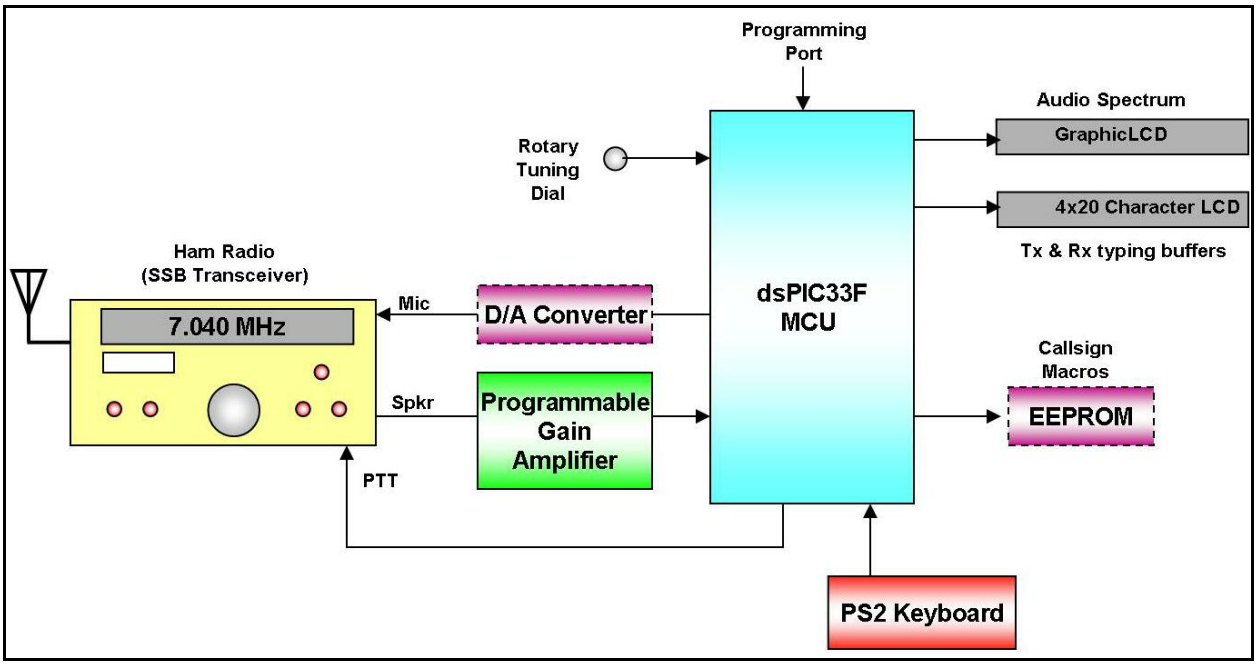
PSK31 is one of the latest communications modes to capture the interest of hams worldwide. Its inherent ability to dig out low, near-inaudible signals is ideally suited for low power QRP enthusiasts. The PSK31 digital modem engine, however, requires intense DSP processing that is only commonly available in PC sound card. Thus the PSK operator desiring portability for field operation is locked into using a laptop computer as a controller, which results in a cumbersome station. But there's hope!

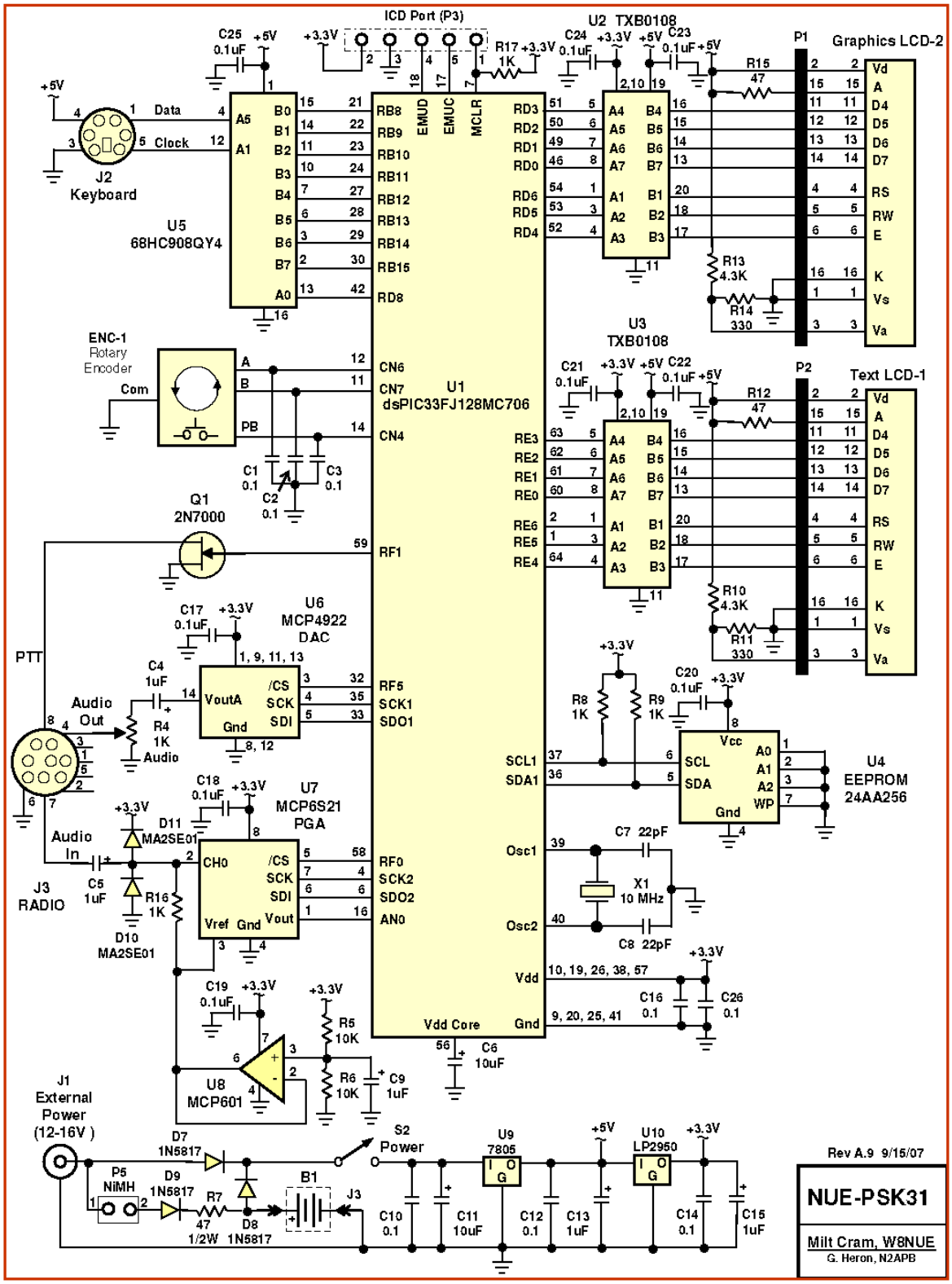
This paper presents the design and construction of a standalone, battery-operated digital modem using a Microchip dsPIC microcontroller. The project includes a character display for transmit and receive text data, and a graphic display showing band spectrum and tuning indicator. Using GPL open source software, the modem can be homebrewed for less than $50 parts cost. When coupled with an SSB-capable transceiver or with a popular PSK-xx transceiver board from Small Wonder Labs, you too can have an effective portable PSK31 station.

ELIGIBLE PARTS

- dsPIC33F128MC706
- MCP4922
- MCP6S21
- MCP601

BLOCK DIAGRAM




ICD Port (P3)
U2 TXB0108
P1
Graphics LCD-2
J2
Keyboard
Data
Clock
U5
68HC908QY4
U1
dsPIC33FJ128MC706
U3
TXB0108
P2
Text LCD-1
ENC-1
Rotary
Encoder
Com
Q1
2N7000
PTT
Audio
Out
Audio
In
J3
RADIO
U6
MCP4922
DAC
U7
MCP6S21
PGA
U8
MCP601
U4
EEPROM
24AA256
J1
External
Power
(12-16V)
P5
NiMH
B1
S2
Power
U9
7805
U10
LP2950
Rev A.9 9/16/07
NUE-PSK31
Milt Cram, W8NUE
G. Heron, N2APB

Embedded Control Design Contest – Entry MT2267

PARTS LIST

Designator	Part	Qty	Description
U1	dsPIC33FJ128MC706	1	Microchip DSC, 64-pin QFP
U5	MC68HC908QY4	1	Freescale microcontroller, DIP-16
U7	MCP6S21	1	Programmable Gain Amplifier
U8	MCP601	1	Op Amp
U6	MCP4922	1	Dual-DAC MCP4922
U4	24AA256	1	Microchip EEPROM
Q1	2N7000	1	NFET
U2, U3	TXB0108	2	Octal Level Shifting Buffer
U10	LP2950	1	3.3V regulator (TO-92)
U9	78L05	1	5V regulator (TO-92)
LCD-1	4x20 character LCD	1	Primary character display
LCD-2	144 X 32 graphics LCD	1	Crystalfontz CFAG14432A-YYH-TT
P1, P2	16 pin pinheader	2	Pinheader, 2x8 (cut 4 from 2x36 strip)
P5	2 pin header	1	Pinheader, 2x1 (cut 36 from 2x36 strip)
P4	8-pin Mini-DIN plug	1	8-pin Mini-DIN plug
J2	6-pin Mini-DIN	1	6-pin Mini-DIN
J3	8-pin Mini-DIN	1	8-pin Mini-DIN
J1	DC power connector	1	DC power connector
X1	crystal	1	crystal
R8, R9, R16	1k ohm resistors	3	resistor
R5, R6	10k ohm resistors	2	resistor
R4	1k ohm potentiometer	1	resistor
R7, R12, R15	47, 1/2W	3	resistor
R11, R14	330 ohm	2	resistor
R10, R13	4.3K ohm	2	resistor
C6	10uF Tantalum	1	capacitor
C9, C13, C15,	1uF ceramic	3	capacitor
C1, C2, C3, C10, C12, C14, C16	0.1uF ceramic	11	capacitor
C7, C8	22pF	2	capacitor
	IC socket	1	16-pin DIP IC socket
D7, D8, D9	schottky diode	3	Diode, Schottky, 1N5817, DO-41
D10, D11	schottky diode	2	Diode, Schottky, 1N5711, SMT
ENC	Rotary Encoder /w PB	1	rotary encoder
	PCB	1	
	Overlay Label	1	
	Enclosure	1	

SOURCE CODE SAMPLE (main.c)

```
/*==========================================================================

               dsPIC33FJ128MC706_SDR_PSK main

               Milton E. Cram W8NUE
               Began On:      09/24/2006
               Current Date:  04/01/2007

==========================================================================*/

#include       "dsPIC33FJ128MC706_SDR_PSK_Header.h"

/*==========================================================================*/
/* Macros for setting device configuration registers                        */
/*==========================================================================*/

       _FBS           ( RBS_NO_RAM & BSS_NO_FLASH & BWRP_WRPROTECT_OFF )
       _FSS           ( RSS_NO_RAM & SSS_NO_FLASH & SWRP_WRPROTECT_OFF )
       _FGS           ( GSS_OFF & GWRP_OFF )
       _FOSCSEL       ( FNOSC_PRIPLL & IESO_OFF & TEMP_OFF )
       _FOSC          ( FCKSM_CSDCMD & OSCIOFNC_OFF & POSCMD_XT )
       _FWDT          ( FWDTEN_OFF & WINDIS_OFF & WDTPRE_PR128 & WDTPOST_PS32768 )
       _FPOR          ( PWMPIN_ON & HPOL_ON & LPOL_ON & FPWRT_PWR32 )

       _FUID0         ( 0x01 )
       _FUID1         ( 0x23 )
       _FUID2         ( 0x45 )
       _FUID3         ( 0x67 )

/*     unsigned int SFR_ARRAY[7] __attribute__ ((noload)) = {0}; */

/*==========================================================================*/

int main ( void ) {

       extern int count;
       extern Flag_type Flag;
       extern char Gain_Select, CharBfr;
       extern int m_TXState;
       int code;
       extern double Old_Freq;
       INTCON1bits.NSTDIS = 0;
       init_clock ();
       init_SPI ();
       init_ADC ();
       init_CN ();
       init_I2C1 ();
/*     init_open_drains (); */
       Init_LCD ();
       init_Key_isr ();
       Set_PGA1_Gain (x16);
       Gain_Select = x16;
       Set_PGA2_Gain (x16);
       Init_Filter_Keyboard_Pointers ( );
       Init_FIR ();
       Flag.ProcPSK = 0;
       Flag.ProcFFT = 0;
       CPSKInitDet ();
       InitPSK (8000);
       PSKTXSetup ();
       InitPSKMod ();
       init_TIM1 ();
       count = 0;
       m_TXState = TX_OFF_STATE;
       Reset_LCD_buffer ();
       Set_DD_Address_LCD1_Display (LCD_Addr[3]);
       Clear_FIFO ();

/*==========================================================================*/
```

```
while (1)      {

               if (m_TXState == TX_OFF_STATE) {
                       if (Flag.ProcPSK == 1) {
                              ProcPSKDet ();
                              Flag.ProcPSK = 0;
                       }
               }
               if (m_TXState == TX_PAUSED_STATE) {
                       if (Flag.Setup == 1) {
                              Flag.Setup == 0;
                              Setup_Menu ();
                       }
               }
               if (Flag.TglRnT == TRUE) {
                       Toggle_RnT ();
                       Flag.TglRnT = FALSE;
               }
               if (Flag.TglTune == TRUE) {
                       Toggle_Tune ();
                       Flag.TglTune = FALSE;
               }
               if (Flag.DAV == TRUE) {
                       Flag.DAV = FALSE;
                       Buffered_LCD_Write (CharBfr);
               }
               if (Flag.DispFreq == 1) {
                       Flag.DispFreq = 0;
                       Display_Freq();
               }
               if (Flag.RXTune ==1) {
                       if (RXTuneCnt > 0x2000) {
                              Flag.RXTune = 0;
                              Acquire();
                       }
               }
               if (Flag.ProcFFT == 1) {
                       Flag.ProcFFT = 0;
                       Proc_FFT();
               }
               if (abs(NCO_Frequency - Old_Freq) > 5.) Write_NCO_Cursor_LCD2 ();
               Old_Freq = NCO_Frequency;

       Process_Key_FIFO();

       }
}
/*=====================================================================*/
```