TinyFSK
FSK Implementation in Writelog and N1MM+

RTTY ops fall into two camps for transmit: those who use AFSK, audio frequency shift keying, and those who use FSK, frequency shift keying. AFSK does not require an expensive interface, and is easy to set up, but is prone to audio overdrive and signal splatter if not adjusted properly. Windows system sounds can inadvertently be sent over the air, as well. FSK, which involves on/off commands to the transmitter, is more foolproof, but requires some sort of interface between the computer and transmitter. Historically, with FSK, the 5-bit Baudot code was generated by a computer and sent via RS-232 serial port, which puts out voltages to its pins. Since a transmitter with a built-in FSK generator requires a pin to be grounded to generate RTTY, an interface is required. This has been achieved with an open collector NPN transistor circuit: voltage is applied to the base, bringing the collector to ground. This, however, has some disadvantages:

1. Modern computers do not have RS-232 serial ports. On a desktop computer, serial boards can be installed. Not on a laptop. USB-to-serial converter cables are an invitation to bad things: they do not carry the 5-bit code properly. The Edgeport USB is an exception, but is pricey.
2. Without a high quality serial UART, signals must be generated by "bit banging," performed by intermediate software such as EXTFSK. This is done at the mercy of Windows timing, and on an underpowered computer this leads to jitter and possibly terrible signals.

The answer has until now been very expensive commercial interfaces that perform the computing functions internally, taking ASCII data from a USB port. They are expensive and sometimes a pain to configure.

K0SM recently designed an interface using an Arduino chip. It accepts ASCII data via a USB port (which will appear as a COM port in Device Manager), and generates an FSK signal via "bit banging" with precise timing that is independent of the computer: the TinyFSK. This has many advantages:

1. Regardless of the computer used, timing is perfect and signal jitter is not an issue.
2. No hardware serial RS-232 computer port is needed.
3. The interface is very simple, and very inexpensive. If you are willing to wait for a few parts from eBay, the whole thing costs less than $10.
4. Programming is provided by K0SM, and it is elegant and works. Copy and paste.
5. The interface is small and mechanically simple.

N1MM+ and Writelog are programmed to use TinyFSK: they send ASCII data from the computer over a USB line to Tiny, which generates FSK signals.

Following are the parts needed, some construction details, and a few photos.

Parts:

1. Arduino Nano or clone with header pins for socketing. $2-3 on eBay.
2. Terminal socket for a Nano. $1.50-3.00 on eBay. The terminal socket makes assembly easier and adjustable.

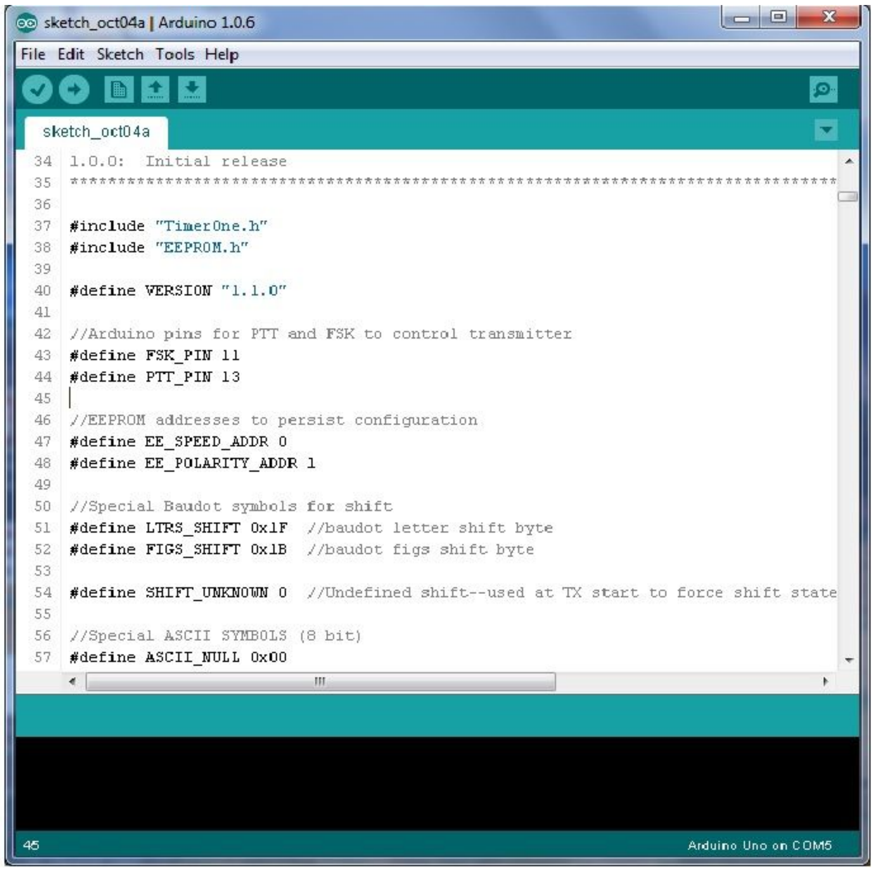
3. Two 2n2222 or equivalent NPN transistors. As low as $0.01. Yup.
4. Two 1k ohm resistors. Free from the junk box.
5. Two bypass capacitors--.01-.1 microfarad. Probably not necessary, but makes me feel better!
6. Two RCA jacks for the case.
7. A shell and connector for you radio accessory port. For the K3, this is a VGA plug.
8. A case, if you want to make it pretty.

Programming and construction:

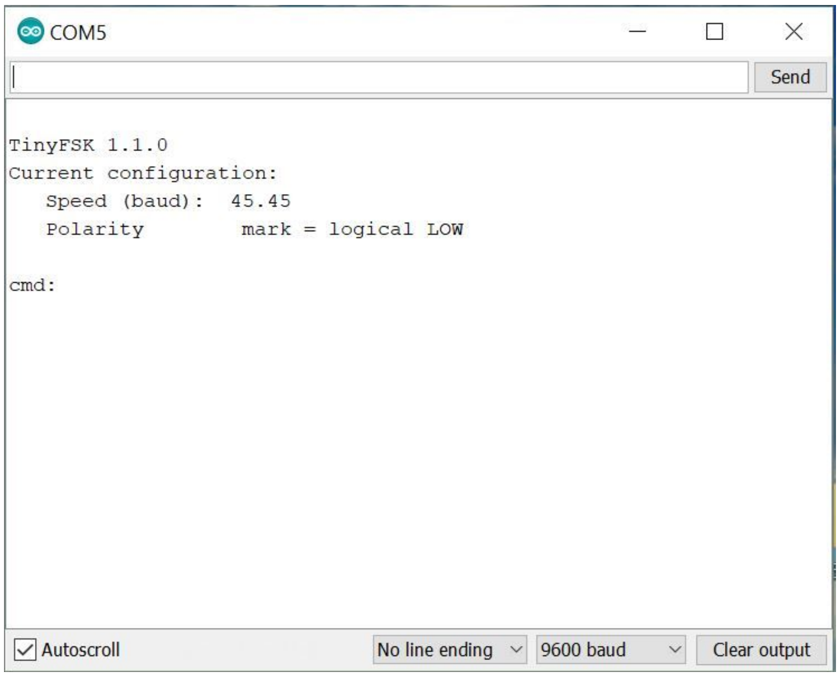
Download the Arduino IDE, which is free, from https://www.arduino.cc/en/Main/Software.

Mount the Nano in the terminal socket, and plug in a USB cable (supplied with many of the Nano clones) to your computer. Open a new Arduino sketch and copy and paste the software from K0SM. Flash to your Nano. http://www.frontiernet.net/~aflowers/tinyfsk/

If you have successfully programmed the Nano, you can open a serial monitor and interact directly with the TinyFSK. The serial monitor in the Arduino IDE works fine. Typing "~?" will report back the current configuration. RTTY speed and Mark/Shift can be set, if needed. Settings persist when power is off. RTTY can be sent directly from the serial window to the rig, which is also helpful for testing and debugging.

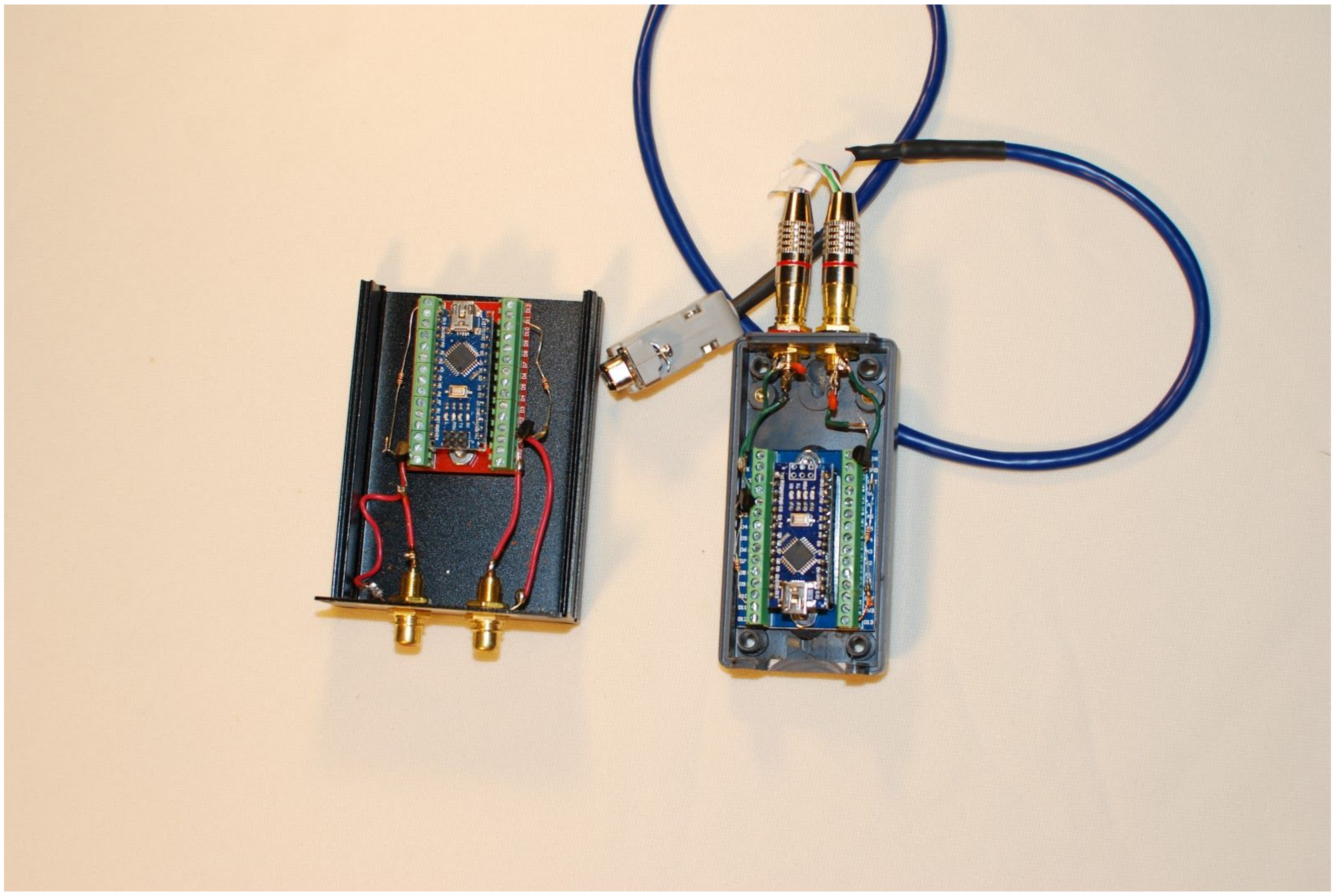


```
1.0.0:  Initial release
********************************************************************************

#include "TimerOne.h"
#include "EEPROM.h"

#define VERSION "1.1.0"

//Arduino pins for PTT and FSK to control transmitter
#define FSK_PIN 11
#define PTT_PIN 13

//EEPROM addresses to persist configuration
#define EE_SPEED_ADDR 0
#define EE_POLARITY_ADDR 1

//Special Baudot symbols for shift
#define LTRS_SHIFT 0x1F  //baudot letter shift byte
#define FIGS_SHIFT 0x1B  //baudot figs shift byte

#define SHIFT_UNKNOWN 0  //Undefined shift--used at TX start to force shift state

//Special ASCII SYMBOLS (8 bit)
#define ASCII_NULL 0x00
```



```
TinyFSK 1.1.0
Current configuration:
   Speed (baud):  45.45
   Polarity         mark = logical LOW

cmd:
```

Fasten the terminal socket to the bottom of a case, put two RCA connectors on the end of the case, and wire in the two transistors and resistors. A pictorial is on the K0SM website noted above. Shown are two Tinys. See photo.



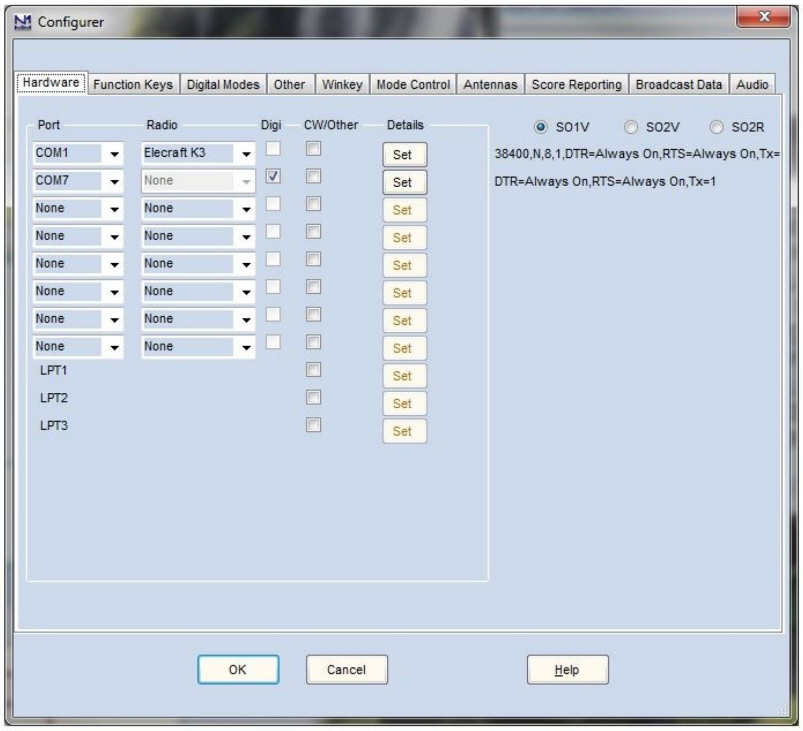
A cable needs to be made for your radio: two male RCA plugs to the TinyFSK, and an appropriate end for your radio acc. port. If you are unfortunate enough to have a radio that has a many-pinned DIN port, consider using a pre-wired DIN plug: Icom provides one with their radios. Also consider using twisted pair wires: a length of Ethernet patch cable will work. This provides the best possible RFI protection.

Plug the Acc. Port connector to the radio, and a USB cable from the Tiny to your computer: note in Device Manager the new COM port—you will need to know the number for your software.

Download the latest version of 2Tone from www.rttycontesting.com, and remember where you put it!

For N1MM+, have TinyFSK plugged in, and go to "configure"....ports, mode control, audio, other.

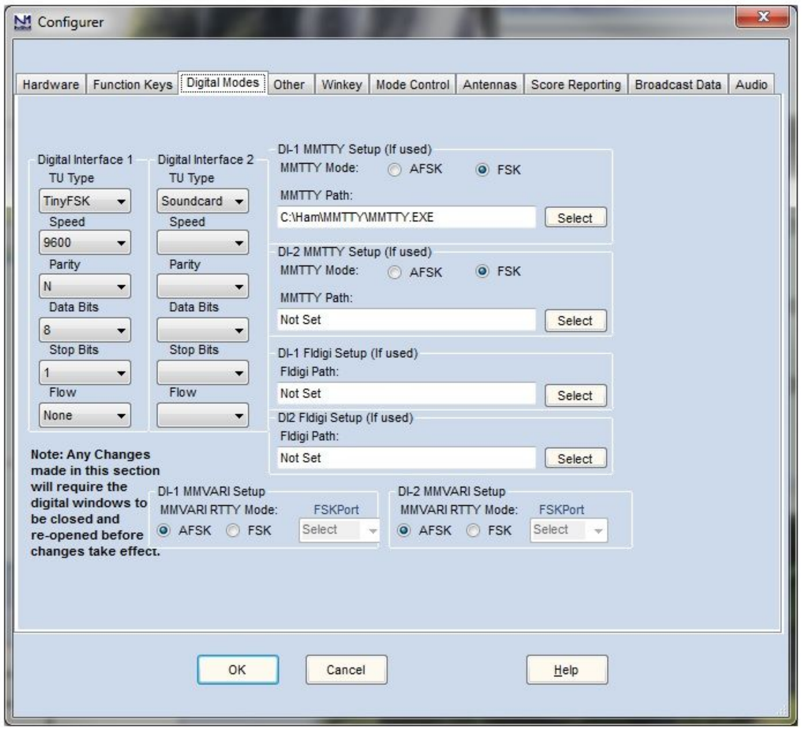
In the Hardware Tab select the COM port that shows up in Device Manager when Tiny is plugged in. Then in the Digital Modes Tab choose TinyFSK as the Digital Interface TU Type, set the speed to 9600, parity N, data 8, stop bit 1, and flow none.



(● SO1V) SO2V SO2R

Port	Radio	Digi	CW/Other	Details
COM1	Elecraft K3	☐	☐	38400,N,8,1,DTR=Always On,RTS=Always On,Tx=
COM7	None	✓	☐	DTR=Always On,RTS=Always On,Tx=1
None	None	☐	☐	
None	None	☐	☐	
None	None	☐	☐	
None	None	☐	☐	
None	None	☐	☐	
None	None	☐	☐	
LPT1			☐	
LPT2			☐	
LPT3			☐	

In the “Digital Modes” tab set the path to MMTTY, which will bring up MMTTY as the primary decoder. Note that the MMTTY panel Transmit command will not work: MMTTY is working as a receive window only.



Digital Interface 1
TU Type: TinyFSK
Speed: 9600
Parity: N
Data Bits: 8
Stop Bits: 1
Flow: None

Digital Interface 2
TU Type: Soundcard
Speed:
Parity:
Data Bits:
Stop Bits:
Flow:

DI-1 MMTTY Setup (If used)
MMTTY Mode: AFSK [✓ FSK]
MMTTY Path: C:\Ham\MMTTY\MMTTY.EXE

DI-2 MMTTY Setup (If used)
MMTTY Mode: AFSK [✓ FSK]
MMTTY Path: Not Set

DI-1 Fldigi Setup (If used)
Fldigi Path: Not Set

DI2 Fldigi Setup (If used)
Fldigi Path: Not Set

Note: Any Changes made in this section will require the digital windows to be closed and re-opened before changes take effect.

DI-1 MMVARI Setup
MMVARI RTTY Mode: [✓ AFSK] FSK
FSKPort: Select

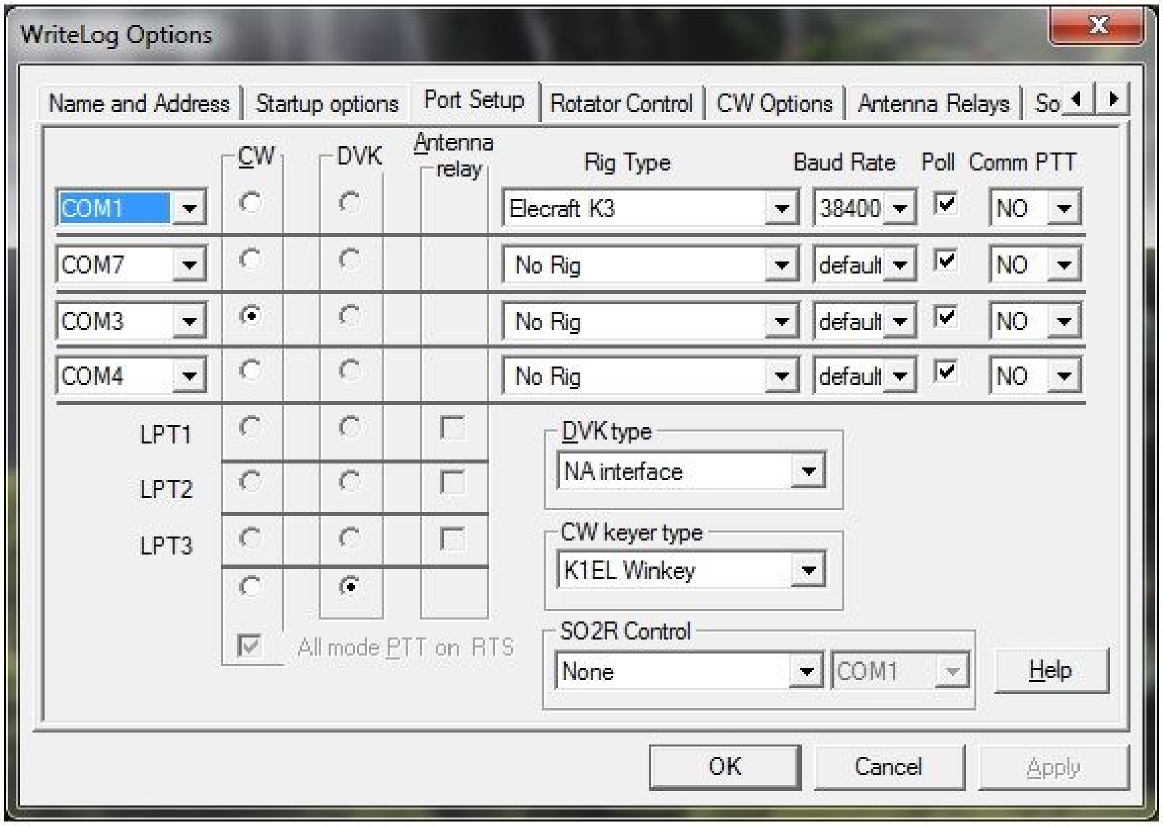
DI-2 MMVARI Setup
MMVARI RTTY Mode: [✓ AFSK] FSK
FSKPort: Select

All of this will set TinyFSK as the TU type, and will open the Digital Interface window with the TNC commands at the bottom of the Rx window, loads MMTTY and shows the RX text in the top ⅔ of the RX window with the TinyFSK commands in the lower ⅓.

It is helpful to add Rx screens, one of which should be 2Tone, so the different receive variants can be set---flutter, flat, selectiv, etc. in 2Tone.   It is important to note that no transmit parameters in the 2Tone set up need be changed. The Tx operating mode, in fact, must be left to an AFSK mode, not TinyFSK, because that requires a COM port selection, which has already been made in the Configurer.

FSK polarity should be checked to avoid frustration: this can be checked in the Arduino IDE serial monitor, or with any terminal program. With a K3 configured to default POL=1, the logical FSK Mark should be = low. The instructions on K0SM’s web page are clear.

Setting up Writelog is even easier. V. 12.25 (and up) is needed. In the main logging window put in the COM port generated in Device Manager by Tiny, in this case 7:


WriteLog Options

Name and Address | Startup options | Port Setup | Rotator Control | CW Options | Antenna Relays | So

	CW	DVK	Antenna relay	Rig Type	Baud Rate	Poll	Comm PTT
COM1	○	○		Elecraft K3	38400	✓	NO
COM7	○	○		No Rig	default	✓	NO
COM3	●	○		No Rig	default	✓	NO
COM4	○	○		No Rig	default	✓	NO
LPT1	○	○	☐				
LPT2	○	○	☐				
LPT3	○	○	☐				
	○	●					

✓ All mode PTT on RTS

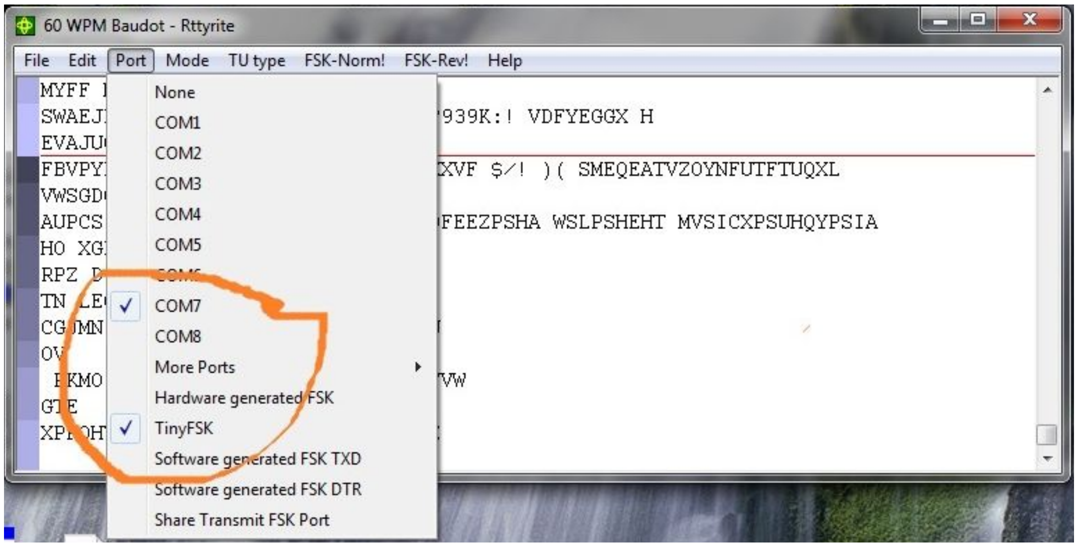
DVK type: NA interface

CW keyer type: K1EL Winkey

SO2R Control: None COM1

Help OK Cancel Apply

In the RTTYWRITE window set the "TU type" as XMMT.ocx, and Port as (whatever COM port TinyFSK generated in Device Manager) and TinyFSK. Note that the G3YYD 2Tone control panel comes up, and is set to "AFSK" so as not to have the same COM port in more than one place.


60 WPM Baudot - Rttyrite

```
MYFF I
SWAEJ         939K:! VDFYEGGX H
EVAJU
FBVPY         XVF $/! )( SMEQEATVZOYNFUTFTUQXL
VWSGD
AUPCS         FEEZPSHA WSLPSHEHT MVSICXPSUHQYPSIA
HO XG
RPZ D
TN LE
CG MN
OV
 EKMO         VW
GT E
XP OH
```

Some notes:

1. Some ops, notably W7AY and G3YYD (the author of 2Tone) strongly prefer AFSK to FSK. They cite absence of timing jitter and the ability to filter down the transmit signal to avoid “RTTY key clicks”. TinyFSK takes care of the first issues. Certain radios, notably the K3, take care of the second by generating FSK via digital signal processing, DSP, and filtering that signal.
2. Some radio manufacturers claim the ability to send FSK over a USB cable directly to the radio. This is usually a dubious claim: EXTFSK is required to do this, and unless the computer is very capable, poor quality RTTY may be generated. Certain hams in the annual RTTY Round Up have earned “special” awards for gruesome signals.
3. Note that K0SM's protocol has two programs: Timer-1 takes care of timing the signal at 45.45 or 75 baud. Windows timing is not involved at all.
4. The alert observer will note that all RTTY FSK interfaces end with a transistor, which keys the FSK generator in the radio. This relates all the way back to the giant mechanical RTTY machines of yesteryear: to key them a pin at their input was shorted to ground. So even if you elect to spend several hundred dollars on a commercial USB RTTY interface, those 1 cent NPNs will be there!
5. Bit banging is a technique for serial communications using software instead of dedicated hardware. Software directly sets and samples the state of pins on the microcontroller, and is responsible for all parameters of the signal: timing, levels, synchronization, etc. If this is not done properly, inaccurate timing and synchronization of the Baudot code will occur, and this is what may happen with EXTFSK and various USB to serial conversion cables. TinyFSK completely controls the process.
6. There is nothing sacred about construction. I use RCA connectors because of past problems with wires pulling out of the case when just “wired through”. The Nano socket is an option: you can buy very cheap Nano clones with the headers not attached. Your choice as to whether solder them in for a socket, or just use the bare Nano board. If you are wiring an accessory connector for your rig, I strongly suggest Cat-5 patch cable for wire: it is stranded so will not break easily, and if the ends are trimmed very short soldering--even to a DIN plug--is not so difficult.

Some credits:

1. G3YYD, David, author of 2Tone. Early versions were receive only; current versions do both receive and transmit.
2. JE3HHT, Makoto Mori, author of MMTTY. A legend among keyboarders.
3. K0SM/2, Andy Flowers, designer of TinyFSK and programmer extraordinaire.
4. WA2TMC, Bruce, whose endless supply of small parts and components populate RTTY interfaces and radio projects in many a shack!

Chuck, N8CL
Ben, KD9IWX