QEX 7 August 1982

The ARRL Experimenters' Exchange

Computers and Amateur Radio

Over the past few years, I have received numerous letters asking how to connect a computer to ham radio gear. Of course, the answer varies with the radio/computer equipment involved, also with what the individual wants to do. There is no hope of coming up with a comprehensive answer in one page or less, but I'll try to give those interested a place to start.

First, what is the intended application? Sending and receiving cw often comes to mind. Basically that can be done by doing the ASCII/Morse code conversion in software. Some hardware is needed to filter the receiver's audio output signal and convert it to digital form with the correct levels for a computer input port. Also, there is usually need to convert the levels from the computer output port to a level suitable for keying a transmitter. There have been articles dealing with this subject in various magazines over the past few years. A search of cumulative indexes will reveal quite a few.

Another popular application these days is to send and receive RTTY. By "RTTY," many people mean 45-baud Baudot, though the term includes ASCII and other teleprinter or computer codes sent via radio. Basically what's needed is an ASCII/Baudot code-conversion program, an RTTY modem (afsk modulator and demodulator or TU), and, depending on the modem and computer designs, possibly a circuit to change voltage levels. These conversions are also in the amateur literature. Also, there are manufacturers who offer complete converters; most are regularly advertised in the amateur magazines. Many of the Baudot RTTY stations heard on the hf bands are using popular computers with commercially manufactured ASCII/Baudot converters or commercial stand-alone "glass TTY" systems, rather than the aging mechanical teletypewriters. If fact, there is more Baudot RTTY than ever. It is ironic that Baudot RTTY is being popularized by computers whose native language is ASCII.

This brings up an interesting question. When all the hf RTTYers discover that everyone's using computers, why won't they all decide one day to go to ASCII. One problem is that there is no such thing as 45-baud ASCII. So a speed change is necessary to accommodate the slowest common ASCII speed of 110 baud, which is the speed used by the older ASCII teleprinters. Speeds accepted by the newer printers typically range from 300 to 1200 baud, while computer terminals can handle speeds from 300 to 19,200 baud. The FCC presently limits ASCII speeds to 300 baud below 21.25 MHz, 1200 baud from 28 to 225 MHz, and 19.6 kilobauds above 420 MHz. One problem is that many of the older RTTY modems were designed for 45- to 50-baud speeds and have filters designed accordingly. Scaling up these filters to handle 75 to 110 baud is fairly straightforward, and manufacturers will usually supply the conversion information. Some will convert to 300 baud. Some manufacturers offer modems which include 300 baud. Comparisons of 45-baud Baudot RTTY and 110/300-baud ASCII RTTY with the same modem (even adjusted for the speed change) would reveal that the ASCII transmissions are degraded by an amount greater than the ratio of the speed change. As speed is changed, the noise bandwidth increases with no increase in the signal. Also, speeds higher than 50 baud or so are subject to intersymbol distortion (a bit is smeared into the next one) due to multipath propagation. Multipath doesn't occur at the maximum usable frequency (muf) but does at frequencies below the muf. Those interested in reading something on this subject can start with the ITT "Reference Data for Radio Engineers," 6th ed., p.28-9. On hf, speeds from 300 to 1200 baud are practical if there is a band opening near the muf. Amateur experimentation with speeds above 300 baud on hf presupposes an STA or FCC rules change.

The key to reliable higher-speed ASCII on hf does not lie in simply widening the filters in existing modems. Instead, we must look to newer technology and the ingenuity of experimenters for the answers.

Automatic repeat request (ARQ) is a technique in which the sender transmits a block of data then listens for an acknowledgement (ACK) before sending the next block. The receiving staion is able to use redundancy in the code to decide whether the characters are correct. Baudot has no such redundancy. The code used in AMTOR (See ARRL petition RM-4122 to the FCC, May 26, 1982) uses the same character set as Baudot but encodes them as 7 bits, 2 of which are redundant. AMTOR sends 3 characters per block between ACKs. In packet radio, error detection is accomplished by using a frame check sequence (FCS), also called cyclic redundancy code (CRC) at the end of the frame. Packet frame lengths vary, but a length of 128 characters is common. Again, ARQ is used.

Forward error correction (FEC) is another technique to be used with, or in lieu of, ARQ. More redundancy is added to permit the receiving station to correct some errors without having to ask for a repeat.

New modems with wider shifts which process mark and space channels independently and employ diversity techniques are also needed. We also need to experiment with modems which help detect possible errors and which are capable of adaptively changing the error-recovery strategy, signaling rate, and possibly the band used as conditions vary.

In between my editing and other activities, I'm home-brewing a new multi-speed (up to 1200 baud) modem with some of the above features. Following that, the plan is to work on a companion data-link controller (similar to a vhf/uhf packet terminal node controller [TNC]) specifically designed for hf use.

I'd welcome articles for QEX on this type of technology, also articles on experimental vhf/uhf modems using QPSK or MSK modulation. - W4RI.

Correspondence

Comments on WB2SZK 50-MHz Solid-State Amplifier

A few notes regarding solid-state amplifiers and the QEX 50-MHz Linear Amplifier, by WB2SZK, appear appropriate. I have been operating 100-watt class solid-state amplifiers for 50 MHz for more than six years and have had some excellent results, and some results that were not quite as good (still good enough for a 50-MHz WAS, though). These amplifiers have been based upon the TRW PT9780 and derrivations of the TRW test circuit. To keep a degree of clarity I shall try to categorize the comments as follows:

A. Bias Circuits:

1. The passive power diode regulator circuit shown by TRW and QEX/WB2SZK works well and is based on the KISS principle. It has, however, a dreadful amount of power dissipation that doesn't help the problem of collector power dissipation. (More on this later.)
2. The active regulator circuit promoted by H. Granberg, et. al. (Motorola AN-758) works quite well over the dynamic range of transistor requirements without large steady-state power dissipations. While this bias circuit uses a few more components, it has been modularized and used in several applications from 1.8 MHz to 435 MHz.

B. Impedance Matching Networks:

1. My own attempts to create a reasonably broad-banded network have been very unsuccessful. These efforts were attempts to circumvent some of the limitations of the "L" networks used by TRW and QEX/WB2SZK. All failures were so classed by low coupling efficiencies.
2. Selection of L and C should not be random. If L is too large on the input, for instance, the circuit Q is high, and the input SWR is thermally quite variable. This is a nuisance in a content to keep retuning the input. WB2SZK and QEX brought me back to my senses with much lower-Q coils, and no active retuning is needed within a range of at least 500 kHz.
3. Do not underestimate the voltage requirements of the junction between the series L and C on the output. I have destroyed 500-V mico capacitors and arced over compression trimmers. Use of 1000-V ratings is probably not high enough for these caps.

C. Feedback:

TRW uses a linearizing R-L-C series negative feedback network from collector to base. The QEX/WB2SZK circuit is free of these components, and my tests (based on WB2SZK) show that they are not missed.

D. Power and Efficiency:

My version of the QEX/WB2SZK amplifier shows the best efficiency yet. With 10 - 11 watts drive I'm achieving a 175-W collector input (25 V, 7.0 A) and a 115-W rf output for a very satisfying 66% efficiency.

E. Heat Sink:

Thermal design (my profession) and heat transfer of/from solid-state power amplifiers has been treated in a rather sorry manner in the amateur literature. The QEX/WB2SZK amplifier with a 60-W ICAS dissipation (ssb contest service) needs at least 1200 cm^2 (186 in.^2) of finned heat transfer area in a free vertical airflow path (free convection). Furthermore, if the heat sink fins are any closer than 10 mm the effectiveness diminishes quite rapidly. I'm using a heat sink approximately 100 mm x 150 mm (4 in. x 6 in.) with ten 38-mm high fins (across the 100-mm dimension). This amplifier is placed on the rear edge of a shelf with the entire heat sink overhanging the edge with the 150-mm dimension (and fins) in a vertical orientation to promote air flow. With a ssb ICAS 60-W dissipation (est. duty ratio of 0.33 to 0.50) the heat sink gets hot (Florida shack air @30 degrees C) up to 60 to 73 degrees C. Transistor junction temperatures are estimated to range from 80 to 120 degrees C. These values are safe, but don't expect to operate RTTY CCS service with that heat sink!

Please keep up the great work in QEX - it is a pleasure to look forward to each month. - Dick Jansson, WD4FAB, 1130 Willowbrook Trail, Maitland, FL 32751.

More on Calibrating Frequency Counters

W4ANN's suggested technique for calibrating frequency counters against the 3.579545-MHz color oscillator in a television set is a good one, but one comment is in order. He warns that one should be sure that the TV station is carrying a direct network broadcast so that the color lock will be derived from the network standard. That was valid in the "old days," but is increasingly less so for the simple reason that many stations - probably most in all major market areas - no longer broadcast any network programming directly, but feed it through a digital box called a frame store synchronizer. This box digitally samples the incoming video (usually at a 14-MHz rate), stores enough numbers in random-access memory to make up 1/30 second of video (525 scan lines, plus a few more as a buffer) and marches the video back out exactly in time with the station's synchronizing generator. The advantage of this is that it makes the remote or network programming exactly timed to all local sources such as cameras and videotape machines, rather than having to time all local sources to the network. Coincident timing is necessary for any special effects such as dissolves or superimposing captions on screen. Even if a particular station does not use a synchronizer, there may others on the line between the original network source and the station transmitter - everyone has seen by now "panic freezes" which occur when video is momentarily lost and the synchronizer keeps spitting out its last frame of correct video until its input is restored. Nevertheless, all stations are required by the FCC to keep their 3.579545-MHz standard to within 10 Hz, and we try to keep outs to within 1 Hz. But for our viewers that standard will be the same whether they watch our locally originated programming or that coming from NBC. - James Brodsky, N6JQ, Assistant Chief Engineer, KSBY-TV, P.O. Box 1368, San Luis Obispo, CA 93406.

Avoid PEEKs and POKEs like the Plague

When computer programs are published, please try to omit POKE and PEEK statements, and, when other really esoteric commands used only by a particular computer are included, attempt to furnish translations for a couple of the other biggies — Radio Shack's TRS-80 microcomputer, model I/III, and the Apple. "Computer RTTY in BASIC" in the June issue of QEX is helpful, I'm sure, for OSI users, but it is of little value to anybody else. POKE statements are nearly impossible to translate to other machines. Avoid them like the plague unless you have a huge computer user base, such as the TRS-80 or Apple, and even then they should be avoided (or at the least, clearly explain in the text what the POKEs are accomplishing). - David D. Holtz, WB2HTH, 91 Valley St., Rochester, NY 14612.

Quest for SITOR/AMTOR Program for Apple II

My request is for info on receiving the CCIR 476 SITOR (an ARQ type RTTY). I would be willing to pay for a program to receive SITOR/AMTOR, ARQ and FEC modes on my Apple II computer. This program would not have to perform any error indicating or correcting features; just decode the received characters (as best it can) and output to screen. As you might guess, I am not much interested in receiving merchant marine radiograms as I am in observing the timing, decoding, etc., of SITOR/AMTOR. From recent issues of Wireless World I have the impression AMTOR (very similar to the CCIR 476 SITOR) is being used by some radio amateurs in Japan, South Africa, England, and Ghana. - E. Adams, K2YEF, 718 Graisbury Ave., Haddonfield, NJ 08033.

1750-Meter Transverter

By Charles L. Faulkner,* W6FPV

Here is a 1750-meter transverter that is very easy to build with a minimum of coil winding. It makes use of standard 455-kHz i-f transformers. Only two of the transformers need to be rewound - T2 and T3. This is accomplished by removing all the existing windings and rewinding with 45 turns of small enamel wire, #34 or so, on the primary and 8 turns on the secondary. On the transformers that I used, there was a 250-pF capacitor built in. By padding with a 0.001-uF disc capacitor, T1, T4, T5 and T6 will tune down to 180 kHz and lower.

The final tank coil uses a ferrite rod from a pocket broadcast radio. Any similar type would do. I cut 5 discs of cardboard to slip over the rod and then wound the primary of 30 turns of #22 wire in the second slot. The 375 turns of #30 cotton-covered enamel wire is wound in the three other slots.

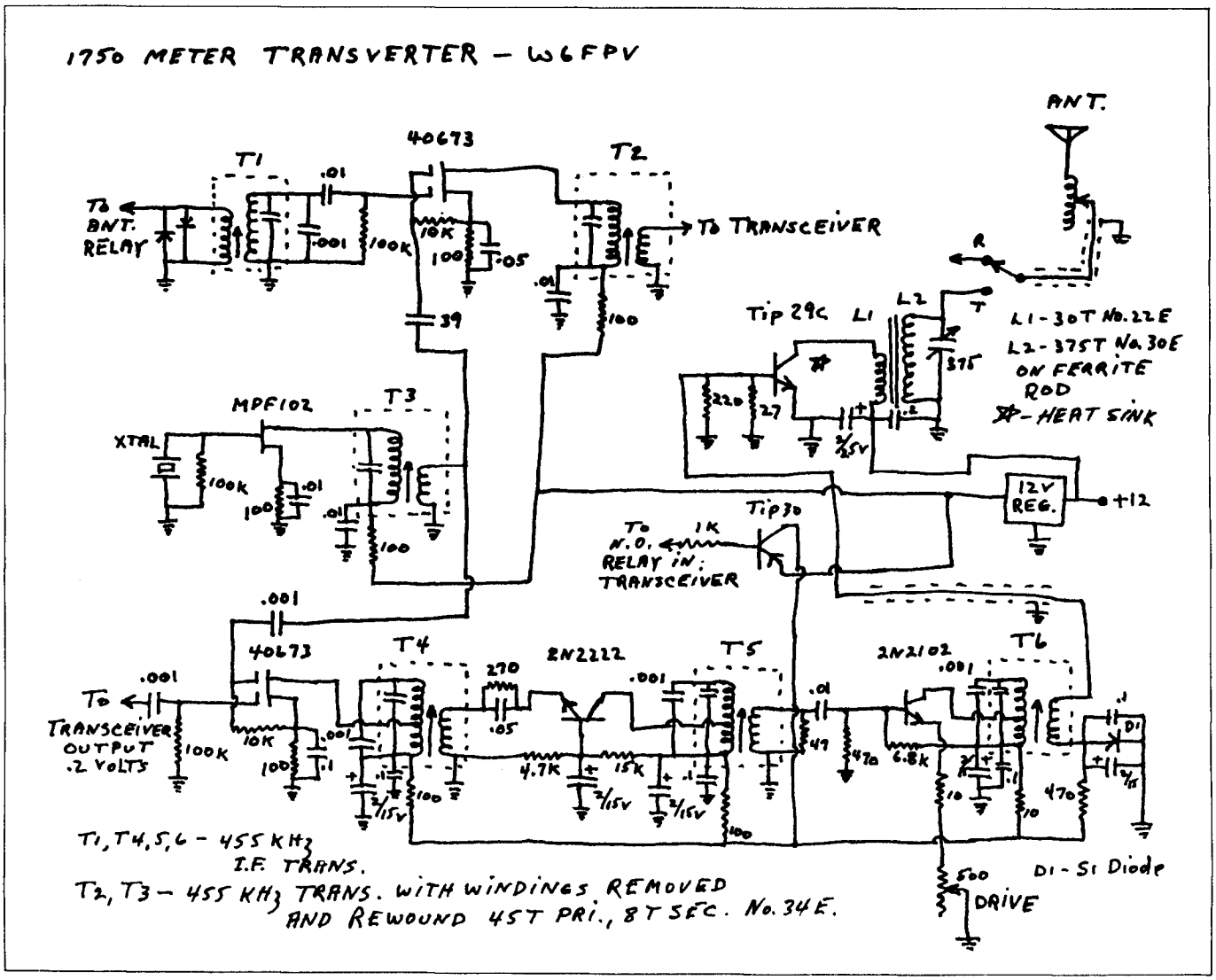
*15844 McKeever St, Granada Hills, CA 91344.

The 375-pF variable tank capacitor is a two-section capacitor from a broadcast set. The crystal for the oscillator can be in any of the ham bands. I used 3400 kHz. By using a crystal on an even 100 kHz, the digital display on the transceiver will read the correct frequency on the last two digits ahead of the decimal. My Kenwood 820 reads 3560 to 3590 to cover the 1750-meter band. Of course if you use a different frequency, you will have to wind T2 and T3 to match your crystal frequency.

On my Kenwood 820, the transverter output is only on 10 meters. So to get output on 3580 kHz, I tapped a 2- to 3-pF capacitor to the stator of the driver stage and mounted a phono jack near the handle of the 820. Only 0.2 volt is needed to drive the transverter.

I built everything but the power supply and the final on a perf board. Keep the transmitter section in a straight line and you should have no

(continued on page 5)



Using the Xicor NOVRAM

By Lyle Johnson,* WA7GXD

The "Components" column of QEX Number 4 (May, 1982) mentioned the 5-volt-only Xicor NOVRAM, a unique form of nonvolatile memory. Since the Tucson Amateur Packet Radio (TAPR) Terminal Node Controller (TNC) uses this device for setting terminal parameters, and since TAPR believes that this is the first use of a NOVRAM device in Amateur Radio equipment, it was decided to share the experience gained in implementing this part with the readership of QEX.

(In the following text, note the use of ! as a prefix denoting "not" or "negative-true" logic.)

What is NOVRAM?

NOVRAM (tm of Xicor) is a device that contains both a volatile RAM and a "shadow" EEPROM. Like any RAM device, when power is lost; so is the data. However, the NOVRAM has a control line (!STORE) that will "burn" a copy of the RAM data into the EEPROM. The control line (!AR or !ARRAY RECALL) is used to copy the EEPROM into the RAM area. Note that the RAM functions like any other RAM device, with !CS and !WE lines, and that the user has direct access only to the RAM area — the EEPROM data must be "recalled" into the RAM area for user access. The RAM <==> EEPROM communication is on the basis of the entire array, not selective bits or bytes.

Why Use NOVRAM?

NOVRAM has many applications in Amateur Radio systems. In the specific case of the TAPR TNC, it is used to store the station call sign, the terminal characteristics (baud rate, parity options, screen width, page length, echo parameters, etc.), packet i-d and so forth. It could also be used in a digital frequency synthesizer to store memory channels or in a spread-spectrum rig to update the frequency-hoping algorithm.

In all of these cases the parameters are user-variable, yet remain when power is removed. No backup batteries are required.

A Closer Look...

The specific requirement for communicating with the chip is summarized below.

Function	!CS	!AR	!WE	!STORE
RAM READ	0	1	1	1
RAM WRITE	0	1	0	1
EEPROM WRITE	X	1	1	0
EEPROM "READ"	X	0	1	1

Note that !AR has priority and that, when Vcc is below about 3 volts dc, !STORE is locked out to prevent spurious data modification during power-up procedures.

The !STORE function may be asserted up to 1000 times in the older parts, and up to 5000 times in the newest parts, before the device degrades to unreliability. The other functions are essentially unlimited during the life of the device.

When !STORE is asserted, an internal mechanism tri-states the data (I/O) lines for about 10 ms, after which the user may again access the NOVRAM. For this reason, and because the NOVRAM is organized as a by-1 or by-4 part (meaning the "data bus" is 1 bit or 4 bits wide), TAPR elected to access the part via a parallel port. This relaxes timing specifications considerably, and assures complete control of the NOVRAM.

*c/o TAPR, P.O. Box 22888, Tucson, AZ 85734

Finally, to allow access to the NOVRAM data, a copy of the NOVRAM is maintained in an array in system RAM. This speeds access of parameters on a by-8 basis (the data bus width of the 6502 microprocessor used in the TAPR TNC) and allows data access during the 10-ms STORE cycle.

The Hardware

A hardware implementation of an XD2210 NOVRAM interface is shown in the schematic, Fig. 1. The 6522 has, among other features, two 8-bit ports that can be configured as inputs or outputs on a bit-by-bit basis via the Data Direction Registers (DDRA for port A and DDRB for port B). A logical "1" in a bit in the DDR causes the corresponding port bit to be an output; a logical "0" causes the port bit to be an input.

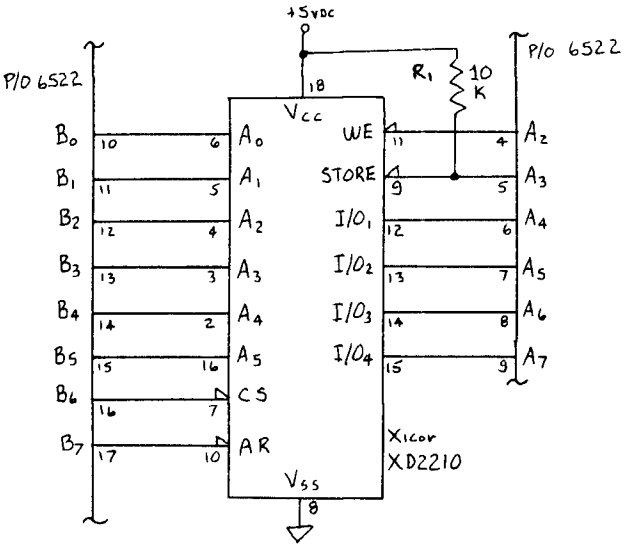


Fig. 1 - NOVRAM Interface

Upon power-up, port A is specified as an input, and R1 ensures that the !STORE line is negated (to avoid a STORE cycle). Port B is specified as an output. (Note that this configuration is not exactly the one used in the TAPR TNC, but was used during development of the hardware and software for the TNC -- it is a working, debugged interface.)

The Software

The software used to debug the NOVRAM interface and use the NOVRAM is presented below (see Listing 1). The program is written in STC FORTH, a language developed by TAPR Packeteer Dave McClain, N7AIG. For those readers unfamiliar with FORTH, the following program explanation is offered. This program was developed on a Rockwell AIM65 microcomputer.

FORTH is a "bottom-up" structured language, with no forward references allowed. This means that all procedures invoked must have been defined previously (there are exceptions to this rule). It also means that the small routines that make up a program get debugged as they are written.

(continued on next page)

NOVRAM (continued from previous page)

In order of their appearance, the program modules do the following:

INIT-VIA places all 1's (hex [$] FF) in port-A and -B data registers. This eliminates glitches in the NOVRAM control lines when the ports are allowed to become outputs. Port B is then set as output only, while port A is set for input on the upper half byte (nibble) and output on the lower nibble.

GET-NIBBLE fetches a 4-bit value from the upper half of port A, which is connected to the 4-bit data bus of the XD2210 NOVRAM, shifts it to the lower half byte, and stores the value in the variable NIBL. It then increments port B, which is connected to the address lines of the NOVRAM and fetches the next 4-bit nibble. This upper half byte is masked off and ORed with the previous (lower) half byte. The resultant byte is stored in variable NIBL.

NOVRAM@ (NOVRAM fetch) takes the entire 64-by-4 NOVRAM RAM data and loads it into a 32-byte array called NO-V-RAM. It works as follows:

INIT-VIA is called to put port A and port B in know states. The !AK line is negated and the NOVRAM address bus set to 0. A DO-LOOP of length $40 (64) is then invoked which accomplishes the following:

The loop index (I) is placed on the stack and added to $80 (to keep the !AR line negated). The resultant value is placed on the NOVRAM address bus and control lines of port B. GET-NIBBLE is called to read a byte from the NOVRAM (see above). The byte in variable NIBL is then placed on the stack. The indix counter (I) value is then halved and added to the base address of the array NO-V-RAM. The value from NIBL, already on the stack, is now loaded into the proper location within the array NO-V-RAM. Finally, the loop index is incremented by 2 (to account for the fact that GET-NIBBLE fetches two consecutive locations of NOVRAM), and control is passed back to DO.

This loop is executed until all 64 nibbles are read from the NOVRAM and sequentially stored into the 32-byte array NO-V-RAM. The NOVRAM is then deselected by negating !CS (as well as !AR).

To fetch the entire NOVRAM contents and load them into memory, the operator has only to type the word "NOVRAM@" on his terminal.

NIBBLE! loads the upper half byte of the 6502 A register into NOVRAM. The nibble is placed in the A-port data register with !WE asserted. The 6502 A register is temporarily pushed on the FORTH return stack (dynamic storage) and the A port reconfigured as output only. This places the data on the NOVRAM data bus where it is fetched and loaded into the NOVRAM RAM area pointed to by the address on the B-port lines. The data is now pulled from the return stack and !WE negated. This updated value is placed in the A-port data register, and the fact that the A-port upper half byte corresponds exactly to the data in the addressed NOVRAM location prevents data line contention between the NOVRAM and the A-port lines. The A port is then reconfigured to be an input port on the upper half byte and output on the lower half byte.

PUT-NIBBLES fetches the lower half-byte value in NIBL and shifts it to the upper half byte in the 6502 A register. This nibble is then loaded into the NOVRAM by invoking NIBBLE! (above). The NOVRAM address is incremented and the upper half byte fetched from NIBL. NIBBLE! is again invoked and the nibble stored in the NOVRAM.

NOVRAM! writes the entire 32-byte NO-V-RAM array into the NOVRAM RAM area. After negating the !AR control line, a DO-LOOP is executed for $40 (64) repetitions. The DO-LOOP works as follows:

The value of the loop index (I) is halved and added to the base address of the array NO-V-RAM. This value is used as an address to fetch a byte from the array, which is then stored in the temporary variable NIBL. I is added to $80 (to assure that !AR is negated) and this value put on the B port as a NOVRAM address. Since PUT-NIBBLES uses two locations in NOVRAM, the loop index counter is incremented by two and control returned to DO.

After the DO-LOOP is completed, both !CS and !AR are negated.

To fetch the entire NO-V-RAM array and load it into NOVRAM RAM, the operator has only to type the word "NOVRAM!" on his terminal.

/P! is used to load the NO-V-RAM array into the NOVRAM EEPROM. The execution address of NOVRAM! (above) is placed on the stack and the routine called. Upon completion, the array NO-V-RAM will be in NOVRAM volatile storage. The state of the A port is pushed on the return stack, and !STORE is asserted via A-port line 3. The previous state of A port is then pulled from the return stack and A port restored. This will negate !STORE, and the NOVRAM then continues its internal EEPROM-burning function for about 10 ms. During this time, of course, the array NO-V-RAM is accessible by the microprocessor for systems use.

GET-NOVRAM is a routine that strobes the EEPROM into the NOVRAM RAM area for access by the system. !AR is asserted on the NOVRAM by the A port, after which INIT-VIA is invoked to return the status of the NOVRAM control lines to a "safe" state.

Wrap-Up

As can be seen from the above, NOVRAM is very versatile, and using it in a microprocessor-based system is straightforward. It, or its variants and derivatives, undoubtedly will be found in many advanced Amateur Radio products in the future, of which the TAPR TNC is a precursor. The XD2210 256-bit NOVRAM referenced in this article costs less than $9.00 in single quantities; so get a data sheet, some NOVRAMs, and experiment! (You even may want to try FORTH! It is a very powerful, interactive hardware/engineering language.)

Listing 1>>

1750-m Transverter (continued from page 3)

trouble with instability. I mounted the circuit board below a 5- by 9 1/2-inch aluminum chassis with the power supply and final on the top side. If you try to put the final below the chassis you will probably have instability problems.

Bias on the final is critical. With power on but without carrier inserted, adjust idling current on the Tip 29C to about 10 mA. Try various resistors from base to ground on the Tip 29C. I ended up with 27- and 220-ohm resistors in parallel. With carrier inserted, adjust input to one watt with the drive control. I use a NE2 neon bulb with one wire soldered to the stator of the final capacitor and inserted in a rubber grommet in the panel as the visual tuning indicator.

Be sure to switch off the screen voltage to the 6146s in the 820 when on the 1750-meter band.

There are five of us on 1750 meters here in the San Fernando Valley, and we have over 100 hours of round-table QSOs. A couple of hams in the Valley here are building the transverter. QAY will use his with a Kenwood 820, but ZHJ will use his with a CB sideband rig. Of course, he will use a different oscillator frequency.

In the daytime when the noise is low, the range is out to about 10 miles. At night the atmospheric and light-dimmer noises limit the range to about four miles. On cw, a range of 20 to 30 miles is possibile. With a low-noise receiving location, 200 miles is possible.

Hope to hear you on 180-kHz sideband. By the way, you cannot use your ham call sign on 1750 meters, so I drop the W6 and just sign FPV.

LISTING 1

```
( TAPR TNC FORTH+
( NOVRAM INTERFACING
( 16 MAY 1982
( by LYLE JOHNSON, WA7GXD




( DEFINE AIM-65 CONSTANTS FOR 6522

$A000 CONSTANT VIAB  ( B-PORT DATA
$A001 CONSTANT VIAA  ( A-PORT DATA
$A002 CONSTANT DDRB  ( B-PORT DATA DIR REG
$A003 CONSTANT DDRA  ( A-PORT DATA DIR REG

( BYTE/NIBBLE CONVERSION STORAGE LOACTION

0 VARIABLE NIBL

( 32-BYTE ARRAY FOR NOVRAM DATA

0 VARIABLE NO-V-RAM ( 16-BIT VARIABLE
#30 ALLOT           ( INC DICTIONARY PTR

( EXCEPT FOR PB6 & PB7, THIS CONFIGURATION
( CORRESPONDS TO THE TNC 6522 HARDWARE

: INIT-VIA          ( SETUP 6522 REGISTERS
-1 VIAB !           ( A & B DATA = $FF
$0FFF DDRB !        ( 1/2 A INPUT, REST OUT
;                   ( END DEFINITION

( FETCH 2 4-BIT NIBBLES AND BUILD A BYTE
( FROM NOVRAM DATA

CODE GET-NIBBLE
VIAA    LDA     ( NOVRAM => UPPER HALF-BYTE
.A      LSR     ( SHIFT TO LOWER HALF-BYTE
.A      LSR
.A      LSR
.A      LSR
NIBL    STA     ( SAVE LOWER HALF-BYTE
VIAB    INC     ( ADDR NEXT NOVRAM NIBBLE
VIAA    LDA     ( NOVRAM => UPPER HALF-BYTE
$F0  #  AND     ( CLR LOWER HALF-BYTE OF A
NIBL    ORA     ( CONVERT NIBBLES TO BYTE
NIBL    STA     ( SAVE BYTE IN NIBL
RTS             ( END DEFINITION


( FETCH ENTIRE NOVRAM DATA AND LOAD INTO
( ARRAY NO-V-RAM

: NOVRAM@       ( "NOVRAM FETCH"
INIT-VIA        ( SETUP A- & B-PORTS
$80 VIAB C!     ( NEGATE !AR
$40 0 DO        ( LOOP 64 TIMES
  I $80 +       ( GET INDEX & SET BIT 7
  VIAB C!       ( ADR NOVRAM & NEGATE !AR
  GET-NIBBLE    ( 2-NIBBLESS ==> 1 BYTE
  NIBL @        ( PUSH ON P-STACK
  I 2 /         ( HALVE INDEX AND
  NO-V-RAM +    ( AND OFFSET INTO ARRAY
  C! 2 +LOOP    ( SAVE BYTE,INC INDEX,LOOP
$C0 VIAB C!     ( NEGATE !AR & !CS
;               ( END OF DEFINITION

( STORE UPPER NIBBLE OF A-REG IN NOVRAM &
( CONTROL !WE TO AVOID CONTENTION ON Q-BUS

CODE NIBBLE!    ( "NIBBLE STORE"
$0F  #  ORA     ( SET LOWER HALF-BYTE = $F
VIAA    STA     ( NEGATE !STORE & !WE
$FB  #  AND     ( ASSERT !WE
VIAA    STA
        PHA     ( SAVE ON R-STACK
$FF  #  LDA     ( A-PORT = OUTPUT
DDRA    STA
        PLA     ( POP A-PORT DATA
$04  #  ORA     ( NEGATE !WE
VIAA    STA
$0F  #  LDA     ( NEGATE !WE & !STORE
DDRA    STA
        RTS     ( END OF DEFINITION

( FETCH BYTE IN NIBL AND LOAD AS 2 4-BIT
( NIBBLES TO NOVRAM
( EVEN NIBBLE = LOWER HALF-BYTE
(  ODD NIBBLE = UPPER HALF-BYTE

CODE PUT-NIBBLES
NIBL    LDA     ( FETCH BYTE-VALUE
.A      ASL     ( SHIFT LOWER HALF-BYTE
.A      ASL     ( TO UPPER HALF-BYTE
.A      ASL
.A      ASL
'E NIBBLE! JSR  ( UPPER HALF-BYTE => NOVRAM
VIAB    INC     ( INCREMENT NOVRAM ADDRESS
NIBL    LDA     ( FETCH BYTE-VALUE
$F0  #  AND     ( MASK UPPER HALF-BYTE
'E NIBBLE! JMP  ( HALF-BYTE=> NOVRAM & EXIT

( 32-BYTE ARRAY NO-V-RAM => NOVRAM RAM

: NOVRAM!       ( "NOVRAM STORE"
$80 VIAB C!     ( NEGATE !AR
$40 0 DO        ( LOOP 64 TIMES
  I DUP 2 /     ( GET LOOP INDEX AND HALVE
  NO-V-RAM + C@ ( INDEX IN ARRAY & GET BYTE
  NIBL C!       ( SAVE BYTE IN NIBL
  $80 + VIAB C! ( SET I BIT 7 & ADDR NOVRAM
  PUT-NIBBLES   ( BYTE ==> NOVRAM
2 +LOOP         ( INC INDEX & LOOP
$C0 VIAB C!     ( NEGATE !AR & !CS
;               ( END DEFINITION

( LOAD NO-V-RAM ARRAY INTO NOVRAM EEPROM

CODE /P!        ( "PERMANENT STORE"
'E NOVRAM! JSR  ( NO-V-RAM => NOVRAM RAM
VIAA    LDA     ( GET A-PORT STATUS AND
        PHA     ( SAVE ON FORTH R-STACK
$F7  #  AND     ( ASSERT !STORE
VIAA    STA     ( NOVRAM RAM => EEPROM
        PLA     ( RESTORE A-PORT STATUS
VIAA    STA     ( NEGATE !STORE
        RTS     ( END DEFINITION

( AIM-SPECIFIC ROUTINE TO RECALL
( EEPROM DATA INTO NOVRAM RAM

CODE GET-NOVRAM
$3F  #  LDA     ( ASSERT !AR
VIAB    STA
'E INIT-VIA JMP ( NOVRAM CNTRL TO "SAFE"

;S              ( END COMPILATION
```

Two Experimental Keyers

By Fritz Baur,* KCØF

After several years of working code, my copying speed finally and gradually pulled ahead of the transmitting speed achieved with a straight key. Some experiments with conventional single- and squeeze-paddle keys in conjunction with an iambic, electronic keyer were disappointing. The required side-to-side motions struck me as unnatural and tiresome. I started to look for a better way.

I remembered seeing an alternate approach in an ARRL publication [1] and accordingly built a keyer with two side-by-side buttons (or keys) with up-down motion and intended to be operated by two fingers of the same hand. I incorporated several novel mechanical features in an attempt to make operation as comfortable as possible and have worked cw exclusively with it for over a year now. I definitely prefer it to any conventional paddle. This two-finger keyer is described in the second part of this article.

In building up my sending speed with the new keyer, I tried to keep track of the errors (all of them human) that occur with "automatic" keyers and found that most have their root in the physiological difficulty of sending just one short "dit" without involuntarily running out a row of them. For some unexplained reason this occurs mostly, not when the dit is by itself, but when followed by more letter components, such as in A, L, J, etc. I noticed that other operators seem to have the same difficulty.

Some thinking on this and related keyer problems led to the idea of the five-finger keyer that is described in the following. It has **five** keys, one for each finger of one hand. When depressed, each key generates its own letter **element**, which, when strung together, form complete letters, numbers and punctuation marks. The five chosen elements are:

Dit - dit - pause (hereafter called "I")
Dit - pause - pause - pause (E)
Dah - pause (T)
Dit - dah - pause (A)
Dah - dit - pause (N)

The function is largely self-explanatory: letter P, for example is formed by stringing A and N together, X is N-A, U is I-T, V is I-A, and so on. The E key is special in that **three** pause bits always follow the single dit. This makes E **four** bits long, just as long as T, or in other words, the dit becomes just as noncritical as the dah and is unlikely to be involuntarily repeated. One must, of course, remember that such an E can be used only by itself or at the **end** of a letter. To form letter S, I-E will be correct, E---I is not!

The circuit is built with unconditionally self-completing letter elements and included a one-element storage feature comparable to most present electronic keyers. This permits a letter element to be entered while the previous one is still in process. This guarantees "seamless" stringing of elements into well-formed letters.

The circuit also repeats the same element as long as its key is depressed: to form a J requires only a quick hit on the A followed by a longer one on T for A-T-T. A period mark is just one long stroke on A for A-A-A.

The circuit further has a five-key version of iambic (alternating) operation. Any of the keys can be made to alternate with any other key simply by holding both down: parenthesis (-.--.-) is produced by holding down N and T for N-T-N-T. The iambic function permits finding a second, and perhaps more convenient, way to form some letters that also can be done otherwise: X is either T-I-T (iambic) or N-A (non-iambic); Y is T-A-T or N-T-T.

*P.O. Box 1242, Canon City, CO 81212.

With the new keyer assembled, debugged and working, I proceeded to learn its use and, much to my pleasure, found it to be easier to use. It was fraught with fewer "goofs" than any code devices that I taught myself to use earlier including the straight key, paddle key, two-finger keyer and typewriter keyboard. Speculation says that this may not be just a subjective observation tinged with parental pride but an inherent property of the new keyer because it operates in a more suitable domain than the widely used paddle keyer.

The latter is operated mostly in the time domain: the difference between (say) E, I, S, H and 5 is solely the point **in time** that we let go of the dit control. Letters F and L are differentiated by the **time** that the dash is inserted in the dit stream, and so on.

The five-finger keyer, by contrast, works mostly in the spatial domain. Most letters are differentiated by the location **in space** where a finger comes down: to make an F on keys I and N, for L on keys A and I, etc. The human mind is not well equipped to judge elapsed time which it is expected to do with the paddle keyer (and my own two-finger keyer, for that matter). The mind is much better atuned to associate a site (a key) with a given cue. The buffered full keyboard works entirely in the preferable spatial context, but the large number of keys (four to eight assigned to each finger of both hands) may be too much of a good thing for quick and error-free learning.

The five-finger keyer can be regarded as taking the middle ground between the popular iambic paddle and the full keyboard. If my experience is any indication, it may be easier to learn than either, it is less error prone than the former, and, unlike the latter, is compact and can be run with one hand. I hope that other experimenters will build their versions of a multi-finger keyer, try them thoroughly and either confirm or dispute the above assertions.

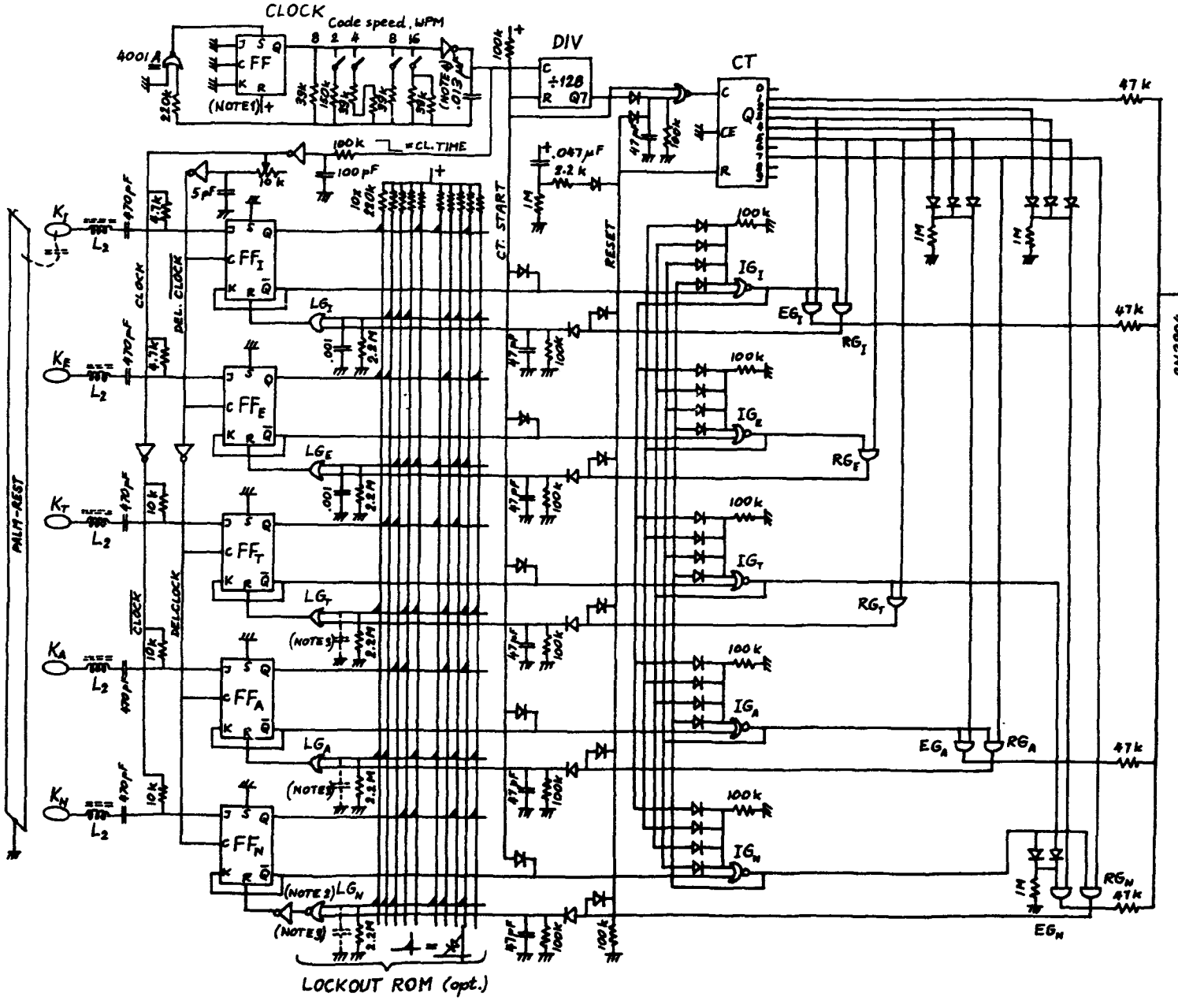


FIG. 1

FIVE-FINGER KEYER

Fritz Baur, KCØF
Cañon City, Colorado, 81212
1-11-82

COMPONENTS:

1	CT: 4017 B	Total of 11 I.C. packages
1	DIV: 4024 B	
3	FF: 4027 B (½)	
2	NOR: 4001 A (¼)	
1	OR: 4071 B (¼)	
2	AND: 4081 B (¼)	
1	INV: 4049 B (⅙)	
96	DIODE: 1N914	
1	L_1: 2x60 T on Amidon FT-50-43	
5	L_2: 50 T on Amidon FT-37-43	

SUPPLY:

9 V Trans. battery, decoupled with 1 x 470 µF, 16 V and 4 x .1 µF cer.

NOTES:

(1) FF used as non-inv. buffer
(2) NOR + INV used as OR
(3) Add discr. cap. if stray-capacitance < 10 pF
(4) Use .0082 + .0047 µF

CIRCUIT OF FIVE-FINGER KEYER

Basic Circuit: (See Fig. 1) Touching one of the keys sets its associated flip-flop FF, which stays set even after the key is released due to the connection from $\overline{Q}$ to K. Output $\overline{Q}$ connects the divided-down clock signal from divider DIV to counter CT, which starts to step at a rate set at the clock to the desired code speed. CT can be compared to a 10-position rotary switch: it normally rests on its "0" output; each cycle at its C input advances it by one output. At each output, in turn, the supply voltage appears for the duration of one input cycle while all other outputs remain at gound potential.

The counter outputs are decoded five different ways according to which key is actuated to form the five letter elements. Dots are made by connecting one output, dashes by three consecutive outputs, and pauses between them or at the end by unconnected outputs. The decoding is done with element gates EGi, EGa, EGn and some diodes. The output following the end of an element goes to the appropriate reset gate (RGi thru RGn), then resets flip-flop, counter and divider and disconnects the counter input. The circuit is now ready for the next element.

Decoding Table:

Counter Output	Letter Element I	E	T	A	N	
0						
1	X	X	X	X	X	
2	-	-	X	-	X	
3	X	-	X	X	X	X=connected for element
4	-	-	-	X	-	R=connected for reset
5	R	R	R	X	X	-=unconnected for pause
6				-	-	
7				R	R	

Since all five elements use output 1, it need not be decoded and is connected directly to the keyer output stage. Element T "borrows" the dash from element N but resets before the dot comes along. These simplifications make element gates for E and T unnecessary and reduce the IC count.

Clock: The clock output is used directly to supply a square wave to the touch-key circuit and divided down by 2^7=128 to step the counter. The clock runs continuously, I.e., there will be a waiting time of up to one clock cycle after a key is touched, but since this is at most 1/128th of one dot length it is imperceptible. The clock circuit is a conventional astable multivibrator with two exceptions:

1) I found that the clock generated harmonics that could be heard as a hiss or ignition-type noise in a nearby receiver. I further found that using a 4001 A (unbuffered, low gain) NOR gate in place of the originally used 4001 B (buffered, high gain) eliminated the harmonics. This concerns the gate connected with its input to the capacitor-resistor node of the multivibrator; the other gate or inverter in the circuit is noncritical. This substitution, however, made the clock susceptible to rf radiation from a nearby transmitter, probably because the "snap action" of the circuit is less pronounced with the lower gain, and the gate input lingers longer in the noise-susceptible zone around the trigger voltage. The cure was to reintroduce more gain in a separate buffer stage. Since only a flip-flop was still available, it was connected as a noninverting buffer (R input high). Admittedly, the whole thing is a kludge, but works so well that the keyer doesn't need a shielded case - it is in a wooden enclosure! [2]

2) The code speed is set with a 4-pole DIP switch rather than with the usual potentiometer. This gives speed selection in equal "quasi-digital" steps, avoids the end-crowded characteristic of most potentiometers and permits repeating settings exactly. The values of the circuit components R and C that determine the clock frequency can be calclulated exactly enough to need no later trimming, as follows:

The frequency at the counter input and the code speed in wpm are related by: fCT = wpm/1.2. [3]

At the clock output the frequency is: fCL = 128 x wpm/1.2

The frequency of a multivibrator, as used here is approximately: fCL = 1/ 2.3 RC.

Combining the last two equations gives: RC = 0.0041/wpm.

I chose C = 0.013 uF (standard values 0.0082 and 0.0047 uF in parallel) and obtained the following values for R=150 kohm for 2 wpm, 78 kohm (two 39-kohm in series) for 4 wpm, 39 kohm for 8 wpm, and 19.5 (two 39 kohm in parallel) for 16 wpm. Paralleling any of the resistors with the DIP switch gives a sped equal to the sum of the speeds associated with each resistor. One further 39-kohm resistor is permanently connected, giving a minimum speed of 8 wpm with all switch poles OFF. Maximum speed is 8+2+4+8+16=38 wpm. Any speed between these limits can be set in 2-wpm increments. Other limits or increments easily can be produced from the above example.

Touch Keys: The square-wave output from the clock is applied to the J inputs of the flip-flops through resistors and to the clock inputs C through a delay circuit (10-kohm pot and 5 pF) and an inverter (total delay including effects of stray capacitance and inverter propagation time on the order of 100 ns). Thus, the J input goes low before the clock input goes high, and the flip-flop remains at rest. When a key is touched, the finger capacitance (on the order of 30 to 50 pF for light touch) is added, causing a delay in the path to the J input, such that J will now be high when C goes high, and the flip-flop will change state. The 10-kohm pot adjuts the touch sensitivity of the keys.[4]

This process is independent of frequency, therefore the same variable-frequency clock used for the counter can be used.

Two more inverters reverse the signals to J, as well as C, of three of the five flip-flops. This does not affect the set process but staggers the clocking times (rise at C) to avoid the possibility of the keyboard jamming if all keys were accidentally touched simultaneously.

The toroidal coils L2 in the leads to the key pads act as rf chokes to further increase the resistance to stray rf from a nearby transmitter. In spite of their relatively high inductance of approximately 1 mH (which makes them quite effective as chokes) they affect the clock signal with its sharp rises and falls very little. (See Fig. 2, also [5].)

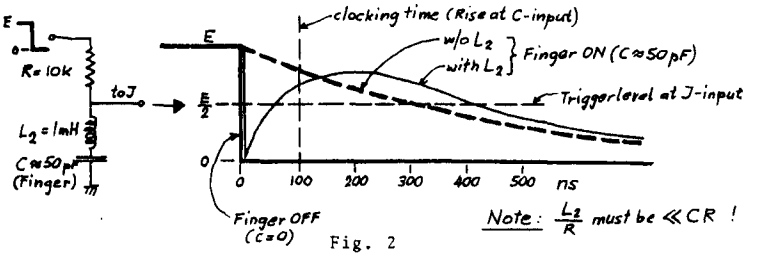


Fig. 2

At first, there was some occasional interaction between keys. This was eliminated by a grounded, conductive palm-rest plate (see photo). The palm can be considered a circuit node with five capacitors (the fingers) attached and evidently should not be left floating. The 470-pF capacitors in the leads to the key pads serve to "isolate" the resistive component of the fingers, which tended to reduce the voltage at J by being part of a voltage divider. All components of the touch-key circuit are very tolerant to changes of up to 2:1 and more -- the entire circuit is reliable and not fussy.

Iambic Operation: Placed in the path from each flip-flop to its associated element and reset gates is NOR gate IG. The output of each IG is cross-connected to an input of the other found IG gates (instead of using 5-input gates, diodes are used to expand one of the inputs of a 2-input gate). When just one key is touched, gate IG does nothing except to invert. However, when a second key is touched in short succession, the two IG gates will form a basic NOR-gate set-reset (RS) flip-flop that will alternate as long as the two keys are activated. This will make the two letter elements belonging to the two keys alternate in iambic fashion.

Lockout Feature: The lockout feature provides that then two keys are in use, the keyboard will accept no further input; i.e., the remaining three keys are locked out until at least one of the two activated keys is released. The purpose is to reduce errors from accidentally touching a key. There are several ways that this piece of logic can be implemented. I chose a homemade diode matrix (or ROM) that combines the functions of ten 2-input AND gates and five 6-input OR gates. The ROM requires 50 diodes. By arranging them vertically on the 0.1- by 0.1-inch grid of the circuit board, the entire ROM occupies only slightly over 1 square inch of board space - about what two IC packages would take up.

Timing Sequence: The iambic gates IGi through IGn have one of their inputs expanded with diodes. This reduces the IC count but slows the switching OFF of the gates because charged-up stray capacitance must discharge through the 100-kohm gate resistor. The turn-off time is about 1 us. This made it necessary to "stretch" the reset pulse from the reset gate to the flip-flop to give the "slow" IG gates enough time to properly do their alternating job. The stretching is done with a diode, a 47-pF capacitor and a 100-kohm resistor in the reset path. The reset pulse is now about 5 us long. This, in turn, made it necessary to delay the "end-of-inhibit" signal going from the lockout ROM to the reset inputs of the flip-flops so as to inhibit the locked-out flip-flops a little longer to give the two activated flip-flops priority for at least the next following clocking time. This delay is done with 0.001-uF capacitors for FFi and FFe (at input of LGi and LGe), while for the other three flip-flops the circuit's own stray capacitance provides enough delay (the flip-flops are clocked at different times, as mentioned above under "Touch Keys").

A welcome byproduct of these delay circuits is that the propagation times of the integrated circuits are completely "swamped out" by them. Any variations in the propagation time due to changes in supply voltage, temperature or manufacturing variances become irrelevant. Furthermore, the circuit layout also becomes quite noncritical and insensitive to varying amounts of stray capacitance. This, together with the inherent noise resistance of the CMOS ICs, should make it easy to build the five-finger keyer and have it work well.

MECHANICAL DESIGN OF THE TWO-FINGER KEYER

The two-finger keyer (Fig. 3) uses an adaptation of the excellent keyer circuit designed by Al Lorona, W6WQC [6] which provides for self-completing symbols, one-bit storage and iambic operation. My modifications include: clock circuit as in five-finger keyer schematic, omission of weight control, output stage, output relay (because my rig is very easy to key electronically), side-tone oscillator and line power supply (battery instead). The novel features are all on the mechanical side. I further limited the development work to mechanical key switches in order to retain the tactile feedback (feel) provided by them. At the time, I did not include capacitive (nonmoving, nonmechanical) keys but chose them for the later five-finger design to explore a second alternative. Also, they are easier to build in quantity.

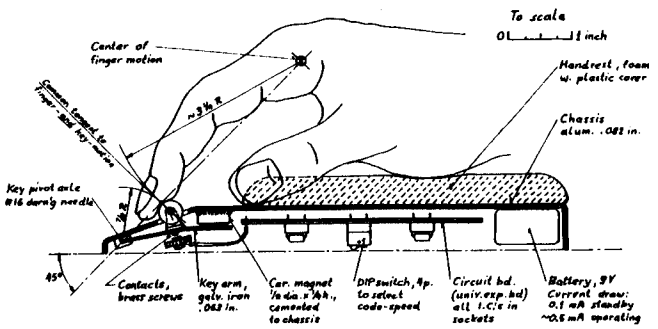


Fig. 3 - *TWO-FINGER KEYER* Nov. 29, 1981 Fritz Baur, KC0F

Key Return Force: Commercial keys come with gradually increasing key force (as key is pushed down) provided by a spring or elastic pad, with virtually constant force over the entire stroke provided by a very long spring compared with the travel (e.g., as in Bencher paddle) or with gradually decreasing force. Trying all three, I found the last one to give the most pleasant, most positive feel, a sort of "over-center" sensation. This characteristic can be attained easiest with a magnetic return force, but springs are also suitable when used in the buckling mode (also called "oil-canning" and used in top-of-the-line pocket computers).

The magnitude of the force is also open to investigation. I finally settled on 50 - 70 gram initial force, dropping off by about 20 g toward the end of the stroke. By comparison, the lightest gun trigger requires no more than 1000 g.

Key Travel: When the hand is held in a natural, relaxed pose above a keyboard, the point where the fingers join the hand (joint between phalanx and metacarpus) is on a line angling up at 45 degrees from where the fingertip contacts the key. This is the joint that does all the motion. The arc described by the fingertip thus has a tangent (at the contact point) also inclined by 45 degrees to the vertical. It would seem logical to set the key's line of travel at the same angle to give the least anount of sliding of fingertip on keytop as both travel down. I tried the usual vertical travel (used on computer or typewriter keys) as well as the inclined travel. I really believe that one can feel a difference, with the slanted travel more comfortable after lengthy use of the key. I therefore used slant travel - wondering all the while if all other keyboards could be "wrong." Different lengths of travel were tried too. I settled on a small travel of about 0.01-0.015 in.

Keytops: Convex, flat and concave tops, set at various angles were tried. The most satisfactory was a small, freely rotating roller (wooden macrame bead, 3/8 in dia.) because it allows an occasional wiping motion over the keytop without much friction.

Overall Configuration: I found that a palm rest elevated about 1/2 to 1 inch above the keytops reduces fatigue. This led to the idea of placing all components, electronic as well as mechanical, in a low-profile box under the palm rest, rather than having them in the conventional little box in front of the hand. This solved several other problems in one stroke. The entire keyer fits under the hand, so does not use any space of its own on the bench. The keyer does not walk around, even if very lightweight. The keyer works fine, as is, on the car seat next to the operator and in other improvised situations. As a stunt, I can even pick up the entire keyer with thumb and ringfinger and work it with index and middle finger while walking around. If the walk-around mode should become a genuinely desirable feature, the keyer's case could be designed along somewhat slimmer lines to fit better into the hand. This and other modifications are quite possible and can be expected to result in a still better keyer.

Notes

[1] Johler, W9UZS, "Finger Keying Consolidates," QST, August, 1963, p.33.

[2] It is sometimes not easy to identify A and B series of ICs since the various manufactures mark them differently. B series will usually have a "B" immediately after the number, but A eries may have an "A," a "C" or nothing. Radio Shack states that all their CMOS ICs are of the B series, but in the case of the 4001, this is incorrect; theirs are of the A series. The ultimate test is to measure the gain by applying a variable voltage to the inputs with a potentiometer and to connect a voltmeter to the output.

[3] "The Radio Amateur's Handbook," ARRL, 1980, p.11-6.

[4] Lancaster, "CMOS Cookbook," H. W. Sams Inc., 1980, p.281.

[5] ITT, "Reference Data for Radio Engineers," H.W. Sams Inc., 1975, p.6-11.

[6] Lorona, "The W6WQC IC Keyer," CQ, December, 1980, p.48.

Components

Conducted by Mark Forbes,* KC9C

If you have any information on components that I haven't covered, please send it along, and I'll try to get it in an upcoming QEX. With QEX now monthly, I'll be able to cover a wider range of products, but my supply of information may dwindle if I don't have some input from you.

Anzac Hf, Vhf, Uhf and Microwave Components

I recently received data sheets on several products from a new company to me, that is Adams & Russell, Anzac Division. They make a variety of hf through microwave active components. A few of these components are described here.

AM-153 LNA: The AM-153 is a 4-terminal low-noise amplifier. Gain is 12 dB, and the AM-153 operates over a frequency range of 300 to 1800 MHz. Power supply requirements are 15 volts at 15 mA maximum. Delivery is from stock, and the price is $135.00.

MD-148 Double-Balanced Mixer: A double-balanced mixer is available in an 8-pin flat pack. The DBM operates over a frequency range of 10 to 1500 MHz. The i-f port operates from dc to 1500 MHz. Typical mid-band conversion loss is 6 dB, and the compression point is +5 dBm (for 1 dB compression). Two-tone intermoduation ratio is 50 dB. These parts are available for $50.00.

MD-167 High I-f DBM: The MD-167 is a DBM with a high i-f and is used for microwave mixing. The frequency response is 0.5 to 18 GHz, and the i-f port operates from 0.5 to 8 GHz. Typical mid-band conversion loss on this part is 8 dB. This unit is also available with N connectors as the MDC-167. These parts aren't cheap, but nothing is at this frequency range. The MD-167 is $565, and the MDC-167 is $640.

FM-104 Broadband Frequency Doubler: Input frequencies of 75 to 1500 MHz can be doubled by using the FM-104. Conversion loss is 12 dB up to 1 GHz and 14 dB from 1 to 1.5 GHz. Fundamental and third-harmonic spurs are down at least 17 dB. The hermetically sealed doubler is available for $90.

JH-136 Quadrature Hybrid: The JH-136 is a hermetically sealed quadrature hybrid circuit. The frequency range is 175 to 350 MHz. Insertion loss is 0.5 dB maximum, and VSWR is less than 1.3:1. Up to 4 watts of rf can be coupled to the JH-136. Its price is $85.

*1000 Shenandoah Dr, Lafayette, IN 47905, 317-447-4272, 2300-0230 UTC weekdays, until 0230 weekends.

All parts can be ordered, and additional data requested, from: Adams & Russell, Anzac Division, 80 Cambridge St., Burlington, MA 01803, 617-273-3333.

Sprague Modem Filters

The series 207C300 transmit module and 207C400 receive module active hybrid filters for low-speed modem use are available from Sprague. These modules provide all the necessary filtering for originate-only, answer-only, or answer/originate type modems. They are directly compatible with Motorola's MC6860 and MC14412 modem chips.

A very informative publication describing these modules, "Engineering Bulletin 22113A," is available for the asking from Sprague. Also recommended are Motorola's application notes AN731, "Low Speed Modem Fundamentals," and AN747, "Low Speed Modem System Design."

Sprague's address is: Sprague Electric Co., 481 Marshall St, N. Adams, MA 01247, 413-664-4411.

For Motorola, try: Motorola Semiconductor Products, Box 20912, Phoenix, AZ 85036, 602-244-6900.

National Semiconductor MM82PC12 I/O Port

To complement the NSC800 family, National has made available the MM82PC12 8-bit input/output port. The device is constructed using the new double-poly CMOS ("PC") fabrication technique, as used in the NSC800. The 82PC12 is functionally and pin compatible with the Intel 8212 NSMOS 8-bit I/O port. Supply requirements for this part are 3 to 12 volts at 300 uA.

The 82PC12, of course, can be be used in other CMOS systems such as the RCA 1802-family processors. For information on this and other NSC800-family devices, contact: National Semiconductor Corp., 2900 Semiconductor Drive, Santa Clara, CA 95051, 408-737-5000.

NE572 Compandor Kit

A kit is available from Signetics which will allow the experimenter to evaluate the NE572 compandor. The kit containing a pc board, ICs and all components necessary to build a complete audio compandor is available for $65 and is part number AAS572. The kit can be configured for either 2:1 or 4:1 companding. If you would like one, remit $65 to: Advanced Analog Systems, Inc., 790 Lucerne Dr., Sunnyvale, CA 94086, 408-730-9786.

QEX: The ARRL Experimenter's Exchange is published by the

American Radio Relay League
225 Main Street
Newington, CT 06111 USA
telephone 203-666-1541

Victor C. Clark, W4KFC
President

David Sumner, K1ZZ
General Manager

Paul L. Rinaldo, W4RI
Editor

The purposes of QEX are to:

1) provide a medium for the exchange of ideas and information between Amateur Radio experimenters,

2) document advanced technical work in the Amateur Radio field, and

3) support efforts to advance the state of the Amateur Radio art.

Subscriptions are available to ARRL members at the special rate of $6 for 12 issues. For nonmembers, the rate is $12 for 12 issues. The foregoing rates apply only to subscribers with mailing addresses in the U.S. and possessions; Canadian and Mexican subscribers must add $1.74, and will be serviced by First Class mail. Overseas subscribers should add $6.78 for air mail delivery or $2.34 for surface mail. Because of the uncertainty of postal rates, prices are subject to change without notice.

Applications for subscriptions, changes of address, and reports of missing or damaged copies should be sent to the American Radio Relay League, Newington, CT 06111. Members are asked to include their membership control number or a label from their QST wrapper when applying.

QEX is edited in, and mailed from, McLean, Virginia. Authors are invited to submit articles to Editor, QEX, 1524 Springvale Avenue, McLean, VA 22101. Both theoretical and practical technical articles are welcomed. Manuscripts should be typed and double spaced. Please use the standard ARRL abbreviations found on page 67 of the December 1981 issue of QST. Authors should supply their own artwork using black ink on white paper. When essential to the article, photographs may be included. Photos should be glossy, black-and-white positive prints of good definition and contrast.

Any opinions expressed in QEX are those of the authors, not necessarily those of the editor or the League. While we attempt to ensure that all articles are technically valid, authors are expected to defend their own material. Material may be excerpted from QEX without prior permission provided that the original contributor is given credit, and QEX is identified as the source.

QEX: The ARRL
Experimenters' Exchange
1524 Springvale Avenue
McLean, VA USA 22101

Nonprofit Organization
U.S. Postage
PAID
McLean, Virginia 22101
Permit No. 235

PRINTED MATTER