QEX

$1.75

ARRL Experimenter's Exchange

April 1995

Measure High-Dynamic-Range Receivers

QEX: The ARRL
Experimenter's Exchange
American Radio Relay League
225 Main Street
Newington, CT USA 06111

QEX (ISSN: 0886-8093 USPS 011-424) is published monthly by the American Radio Relay League, Newington, CT USA.

Second-class postage paid at Hartford, Connecticut and additional mailing offices.

Offices

225 Main St, Newington, CT 06111-1494 USA
Telephone: 203-666-1541
Telex: 650215-5052 MCI
FAX: 203-665-7531 (24 hour direct line)
Electronic Mail: MCIMAILID: 215-5052
Internet:qex@arrl.org

Subscription rate for 12 issues:

In the US: ARRL Member $12, nonmember $24;

US, Canada and Mexico by First Class Mail: ARRL Member $25, nonmember $37;

Elsewhere by Surface Mail (4-8 week delivery): ARRL Member $20, nonmember $32;

Elsewhere by Airmail: ARRL Member $48, nonmember $60.

QEX subscription orders, changes of address, and reports of missing or damaged copies may be marked: *QEX* Circulation. Postmaster: Form 3579 requested. Send change of address to: American Radio Relay League, 225 Main St, Newington, CT 06111-1494.

Members are asked to include their membership control number or a label from their *QST* wrapper when applying.

About the Cover

Using crystal oscillators and filters, KI6QP's test set allows two-tone measurement of high-dynamic-range HF receivers.

ISSUE NO. 158

Features

Columns

April 1995 QEX Advertising Index

Purpose of *QEX*:

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

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

All correspondence concerning *QEX* should be addressed to the American Radio Relay League, 225 Main Street, Newington, CT 06111 USA. Envelopes containing manuscripts and correspondence for publication in *QEX* should be marked: Editor, *QEX*.

Both theoretical and practical technical articles are welcomed. Manuscripts should be typed and doubled spaced. Please use the standard ARRL abbreviations found in recent editions of *The ARRL Handbook*. Photos should be glossy, black and white positive prints of good definition and contrast, and should be the same size or larger than the size that is to appear in *QEX*.

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. Products mentioned in the text are included for your information; no endorsement is implied. The information is believed to be correct, but readers are cautioned to verify availability of the product before sending money to the vendor.

Empirically Speaking

Linux Rules

In this month's "Digital Communications" column, Harold Price turns his soapbox over to Bruce Perens, AB6YM, to tell you about *Linux*, a unix-clone operating system. At present, *Linux* runs on 386/486/ Pentium systems, although it is being ported to systems based on 68000, MIPS and DEC Alpha processors.

With the proliferation of operating systems for the 386 architecture these days (*Windows 95*, *Windows NT*, *OS/2 Warp*, et al), why should you spend the time to learn a new one, particularly when it's not mainstream (meaning you probably won't by seeing much *Linux* application software at Egghead any time soon)? There are several answers to that question. One is that *Linux* is indistinguishable from *Unix*, and *Unix* is very much mainstream. There are literally thousands of applications written for *Unix*, many of which have been compiled and run under *Linux*. These include X windows applications. (X windows is a graphical-user-interface popular on *Unix* systems.) Another indication that *Linux* is becoming mainstream is the appearance of published books about installing, maintaining and using *Linux* systems.

A second reason to be interested in *Linux*—one that's close to the hearts of most hams—is its price: zero. Is it worth what you pay for it? Hardly. It's a *real* operating system, not a demonstration or toy. How real? Well, for the last couple of years, we've been running a *Linux* system as our Internet gateway here at ARRL HQ. It's connected to the Internet via a 14.4-kbit/s modem line and to our desktop computers via a 10baseT Ethernet LAN. On an average weekday, the gateway handles over 1000 email messages, comprising between 5 and 10 Mbytes of data, plus the daily traffic of a couple dozen Usenet newsgroups. HQ staff can—and do—log onto the *Linux* system to read news and perform FTP and Telnet transactions with remote machines on the Internet. And it acts as a proxy server to let staffers browse the World Wide Web (WWW) from their desktop machines. The *Linux* machine? An old 386 running at 33 MHz, with a 320-MByte disk drive. (Before we ran *Linux*, the same machine ran DOS and handled only email. It bogged down badly, which is why we changed to *Linux*.) The total cost of the multiplicity of software we run to perform all these tasks is, again, zero. And the system just runs and runs...

Of course, the more interesting applications to *QEX* readers are those having to do with Amateur Radio. Packet is supported directly by *Linux*, as AX.25 protocol support is in the kernel (the main part of the operating system). That means that the same network applications, such as FTP and Telnet, that work across a LAN or the Internet also work across the local TCP/IP packet network. (An intriguing possibility is to provide WWW capability—a LAN Wide Web?—on the local packet network. Yes, WWW browsers and servers are available for Linux—for free, of course.) Non-TCP/IP AX.25 applications are supported, too. And other useful application programs for amateurs have been written and are being written now. A list, the *Linux* "HAMS-HOWTO," is at the end of Harold's column.

For more detail, read "Digital Communications" to see just what Bruce and Harold are jazzed about.

This Month in *QEX*

Continuing the tradition of applying low-cost equipment to amateur needs, Paul Wade, N1BWT, and Don Twombly, WB1FKF, put their ham ingenuity to work in "Modification of TVRO LNBs for 10 GHz."

In "A Precision Two-Tone RF Generator for IMD Measurements," Stuart Rumley, KI6QP, describes his circuit that allows you to measure the performance of high-dynamic-range receivers.

Connecting a single antenna to multiple receivers isn't as simple as you might first think, but Michael Lass, DJ3VY, shows a circuit that does a good job of letting you use "Up to Four Receivers on a Single Antenna."

Finally, Harold Price, NK6K, turns his "Digital Communications" column for the month over to Bruce Perens, AB6YM, who introduces us to the *Linux* operating system and its application to Amateur Radio—*KE3Z, email: jbloom@arrl.org (Internet)*

Modification of TVRO LNBs for 10 GHz

Applying consumer electronic equipment to Amateur Radio has a long tradition—but at 10 GHz? Yes!

By Paul Wade, N1BWT, and Don Twombly, WB1FKF

Direct-broadcast satellite TV service using small dish antennas has recently become available in the US. The frequencies used for these broadcasts are in the 11 to 12-GHz range, so it seems natural to wonder if any of the equipment is usable for the 10-GHz amateur band. The small parabolic dishes are obvious candidates that work well.[1] And we found that the LNB (low-noise block downconverter) units used in these systems are easily converted to excellent 10-GHz preamplifiers.

During the past year, several types of LNBs have appeared at hamfests. We were able to acquire several at quite reasonable prices, allowing us to attempt modifications knowing that even if we were unsuccessful, we could probably salvage enough components to justify our investment.

An LNB contains a number of small surface-mount microwave components in a small space—Figs 1 (cover and shielding removed) and 2 (partially modified) are photographs of the interior of two typical units. All the units we have looked at have the same basic block diagram, shown in Fig 3. A waveguide fitting, which bolts to the feedhorn, is part of the metal casting that forms the LNB housing. A probe in the waveguide makes the transition to the microstrip circuitry. Since waveguide has no dc connection, no blocking capacitor is needed, and a direct connection is made to the input of a three-stage, low-noise amplifier. The first stage, typically marked with a red dot, is probably a very low-noise transistor such as an HEMT. The amplifier is followed by a printed filter, then a diode mixer. A dielectric-resonator oscillator (DRO) operating at 10.75 GHz provides the LO for the mixer. The rest of the circuitry is on the reverse side of the units, so it is not visible in the photographs. It consists of voltage regulators and bias circuits for the GaAsFETs and a 900 to 1300-MHz postamplifier following the mixer. (We haven't yet tried one of these at 1296 MHz, but expect good performance.) The type-F output connector also supplies the power to the LNB.

Since it is very difficult to retune a printed filter, our first modification involved bypassing the printed filter with a length of semirigid coax, as shown in Fig 4, to isolate the low-noise

[1] Wade, P., N1BWT, "Practical Microwave Antennas, Part 2", *QEX*, October 1994, pp 13-22.

161 Center Road
Shirley, MA 01464

23 Maura Drive
Woburn, MA 01801

amplifier. Since the gaps in the printed filter provided the original dc isolation, a chip capacitor is now required as a dc block for the output coax. We also removed power from the oscillator and the postamplifier by disconnecting jumper wires. We then measured the low-noise amplifier with a waveguide-to-coax adapter at the input. It had greater than 25 dB of gain from roughly 11 to 12.5 GHz but rolled off about 10 dB at 10 GHz.

Since it appeared that the 10-GHz gain was reduced by high input SWR, we took a hacksaw to the waveguide flange and replaced the probe with an SMA connector, as shown in Fig 5. The Teflon dielectric of typical SMA connectors is the same diameter as the dielectric surrounding the probe, so fitting the connector was straightforward. Since there is now a dc connection, a blocking capacitor is again necessary—a 1-pF microwave chip capacitor worked better than a larger value.

With coax connectors at both ends, the low-noise amplifiers have high gain from roughly 9 to 13 GHz, with typically 30 dB of gain at 10.3 GHz. The noise figure is also very good, typically 3 dB or lower with no tuning. Some types of LNB need additional tuning, performed by trimming the microstrip lines and adding metal flakes, which you can do if you have the facilities for measuring noise figure. After a bit of tuning, the typical noise figure is around 2 dB. Don's best unit measured 1.6 dB at the 1994 Eastern VHF/UHF Conference, with no tuning.

A preamp with this kind of performance is more than adequate for almost all 10-GHz work. A noise figure under 3 dB is fine for most terrestrial contacts, and 30 dB of gain is plenty to mask the noise of any decent mixer. In fact, during testing, the output cable on one unit developed a short circuit, but the noise figure only increased by 1 dB!

We won't attempt to give specific modification instructions since there are so many types available, and the modifications are mainly a matter of mechanical ingenuity. For instance, the LNB shown in Fig 2 has a convenient place near the printed filter to

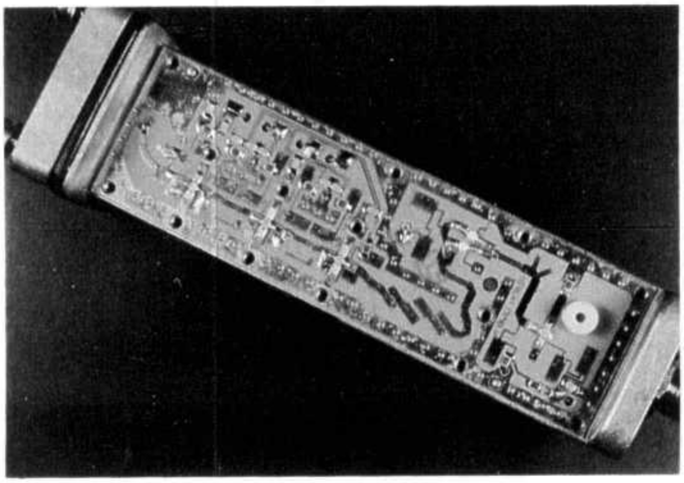


Fig 1

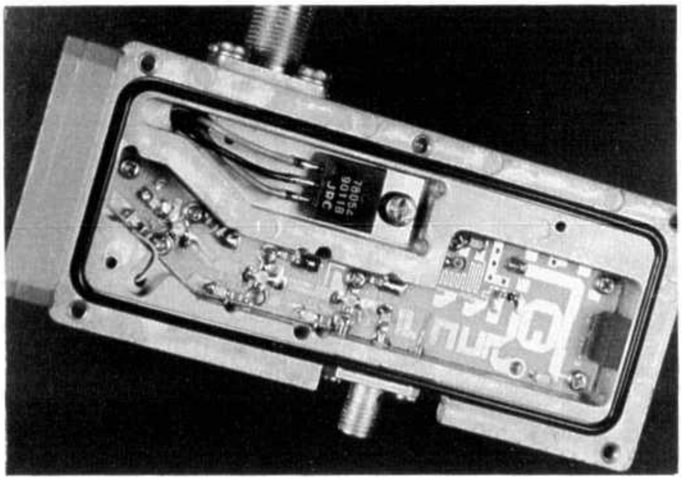


Fig 2

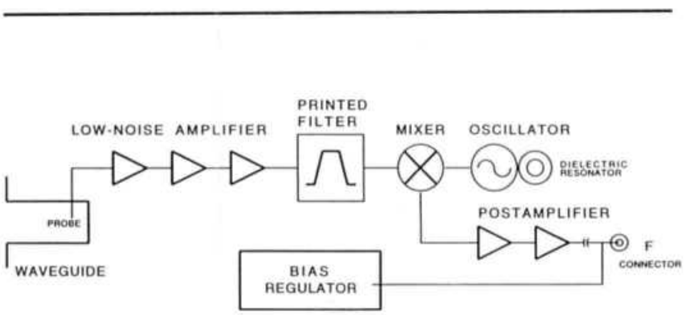


Fig 3—Block diagram of a typical TVRO LNB.

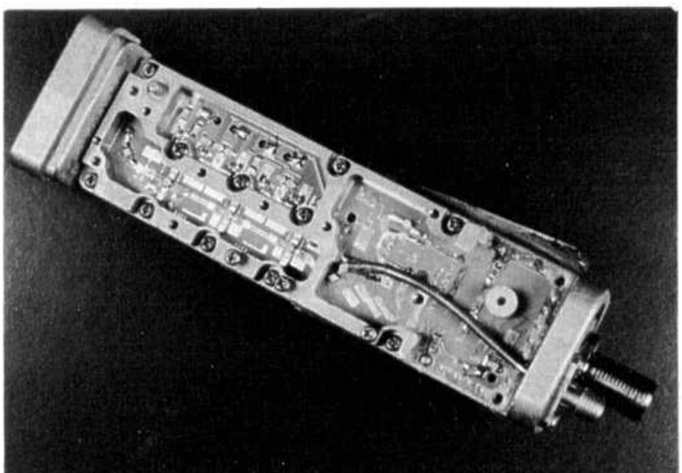


Fig 4

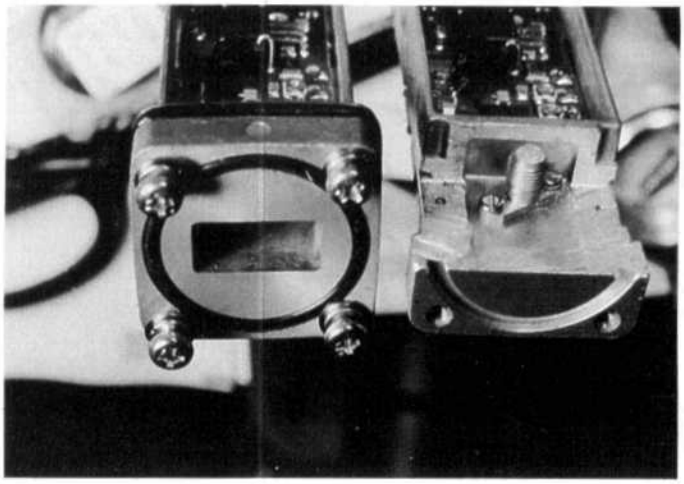


Fig 5

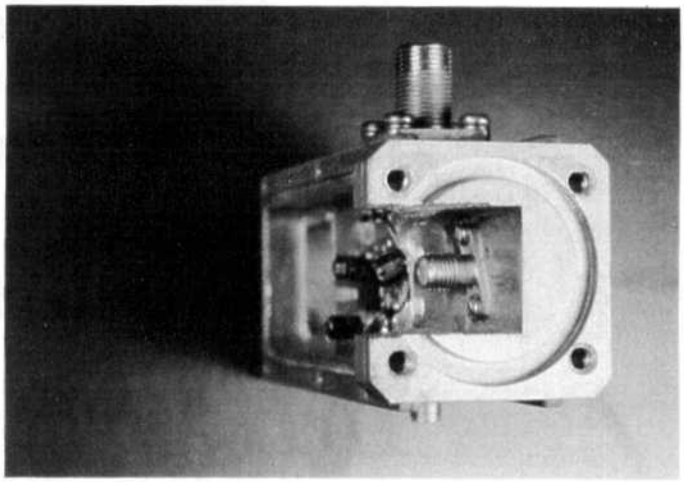


Fig 6

mount an output connector, after a bit of work with hacksaw and file, eliminating the need for semirigid coax. Modification of the input waveguide, shown in Fig 6, is not quite the same as for the unit shown in Fig 5, but is also accomplished with a hacksaw.

The holes for the SMA connector mounting screws are easily tapped in the cast aluminum housing, but the location should be chosen carefully to avoid drilling through components or microstrip lines. Drilling is best accomplished with a drill press, and judicious use of masking tape can help keep metal chips from landing in unwanted locations.

Direct-broadcast satellite TV systems are predicted to become inexpensive consumer products, so components like the LNB and dish will probably be readily available. Already, a brand new LNB with extremely low-noise figure sells for less than $100.

The modifications we have described only use the most valuable part of the LNB, the low-noise amplifier. The other parts can be removed for other uses.

It might even be possible to phase-lock the internal oscillator to a stable reference to make a complete 10-GHz converter. □□

A Precision Two-Tone RF Generator for IMD Measurements

Crystal oscillators plus crystal filters yield extremely low phase noise in a high-IP_3 IMD test generator for measuring the dynamic range of HF receivers.

By Stuart Rumley, KI6QP

There have been numerous articles in amateur literature on the significance of high-dynamic range performance in HF receivers.[1] Without sufficient dynamic range, a receiver's other important virtues, mainly selectivity and sensitivity, soon become ineffective. Dynamic range can be specified and measured as either blocking dynamic range or intermodulation distortion (IMD) dynamic range. Blocking dynamic range refers to a receiver's ability to not be desensitized when strong signals are present. IMD dynamic range, on the other hand, refers to the receiver's ability to not generate false signals whenever there are two or more strong signals present. The IMD measurement leads directly to a convenient figure of merit for dynamic range known as the third-order intercept or IP_3. The purpose of this article is to show how to construct an inexpensive source for making the IMD measurements and how to make the measurements and calculate IP_3, as well as to compare some typical receivers.

Making the IMD measurements requires two high-stability, low-phase-noise RF signal sources at the frequencies of interest. The typical setup for measuring IMD usually looks something like Fig 1.[2] Unfortunately, most enthusiastic radio amateurs do not have one, let alone two, low-phase-noise synthesized signal generators such as the HP8640A shown. Two synthesizers are required because the two-tone IMD measurement requires two frequency stable sources set only a few kilohertz apart.

Attempting to make this measurement with unstable sources proves quite frustrating. If you are prepared to trade frequency agility for cost, two crystal oscillators operating at the desired test frequencies can be just as useful and, to some extent, more convenient than a pair of synthesizers. Fig 2 shows a block diagram of the essential functions of such a test system. There are two crystal oscillators, one for generating each tone.

Each crystal oscillator is followed by a crystal filter network tuned to the same frequency as the oscillator. The output of each crystal filter network is attenuated, then summed together in a hybrid combiner. The combined output from the hybrid is further filtered to remove harmonics. I selected 14.20 and 14.22 MHz as the optimum frequencies for the two crystal oscillators. This is approximately mid-range for most continuous-coverage HF

[1]Notes appear on page 12.

308 Nevada St
Redwood City, CA 94062
E-mail: stuart@itron-ca.com

receivers (0.5 to 30 MHz) and is also in the middle of the 20-meter band so it may be used with ham-band-only receivers.

The two-tone IMD generator concept is rather simple, but a number of subtleties must be taken into consideration. In order to accurately measure receiver performance, the two-tone source must possess the following attributes:

1. Low equivalent IMD to ensure that the distortion products measured are only from the receiver or system under test, not from the test generator.

2. Low phase noise. The phase noise of both tones, at the frequency offset of the expected IMD products, must be much less than these IMD products.

3. Low harmonic energy. Because third-order IMD products do contain energy at the second and third harmonics of F1 and F2, it is essential that the harmonic energy from the two-tone generator be extremely low.

4. Careful filtering and shielding to ensure that external signals and noise do not interfere with the measurements.

Circuit Description

The detailed circuit schematic is shown in Fig 3. Q1 and Q2 are fundamental-mode Colpitts crystal oscillators operating on 14.200 and 14.220 MHz, respectively. The selection of the actual frequencies is determined by two bounding requirements. One, the frequencies cannot be so close together that the test receivers cannot resolve the IMD products because of either selectivity limitations or phase noise from the receiver's local oscillator. Two, the frequencies cannot be so far apart that the input band-pass filters will cause unequal attenuation of either tone. A good choice is 20 to 25 kHz. The output from each oscillator is taken from a tuned collector circuit in order to minimize harmonic energy and provide a low-impedance source to the crystal filter network. The oscillators are operating at a rather high power level, with over 10 mA of collector current in each transistor. The high power is required in order to provide −10 dBm at each tone to the output. This power level was selected so that at least 10 dB of attenuation could always be left in the step attenuators and still allow adequate signal energy to create measurable IMD products in high-dynamic-range receivers. The reason it is desirable to leave some attenuation between the IMD generator and receiver under test is because the receiver cannot be trusted to provide a good 50-Ω match to the IMD generator's output filter. The 10 dB of attenuation guarantees at least 20 dB of return loss to both the IMD output filter and the receiver's input band-pass filters.

Crystal oscillators generally have quite low phase noise, but operation above 1 mA can degrade their performance to some degree. In order to maintain extremely low phase noise, I took a couple of additional measures. The emitter degeneration resistors, R3 and R7, provide negative feedback, which reduces the oscillator phase noise. Following each oscillator is a narrow-band crystal filter network C1, C5, Y2 and C10, C15, Y3. The crystal filters provide an additional 30 dB of phase-noise attenuation at a carrier offset of ±10 kHz. The addition of the crystal filters might seem a bit excessive but is necessary in order to be absolutely sure that the phase-noise performance of the two-tone generator does not preclude the ability to measure low-level IMD products in high-performance receivers. An additional benefit of having two very clean sources is the ability to evaluate the phase noise and reciprocal mixing of the receiver under test.

The outputs from each of the crystal filter networks are passed through 6-dB attenuators comprised of R12, R13, R14 and R15, R16, R17 and summed in the hybrid combiner network made up of T1, R9, R10 and R11. The attenuators are required in order to isolate the combiner from the crystal filters and to provide a constant impedance for both. The combiner sums the two frequencies with an insertion loss of 6 dB and approximately 40 dB of isolation. This combination of filters, attenuators and combiner provides a great deal of isolation between the collector circuits of the two oscillators (more than 90 dB). Together with good physical isolation, this topology prevents the generation of any significant internal IMD products.

The output harmonic filter is a seven-section, all-pole design using the same inductors used for the oscillator. This filter adds approximately 1 dB of insertion loss at the operating frequency, more than 50 dB of loss at the second harmonic and more than 70 dB at the third harmonic. From the output of the crystal filter, the second harmonic is down more than 40 dB and the third harmonic is down more than 55 dB. The resulting output has a second harmonic of less than −100 dBm and a third harmonic less than −135 dBm. These levels are so low that they will not contribute in any measurable way to the IMD values of the receiver under test. Because the second harmonic is less than −100 dBm, it's practical to search for second-order products at 28.400 MHz, 28.420 MHz and 28.440 MHz.

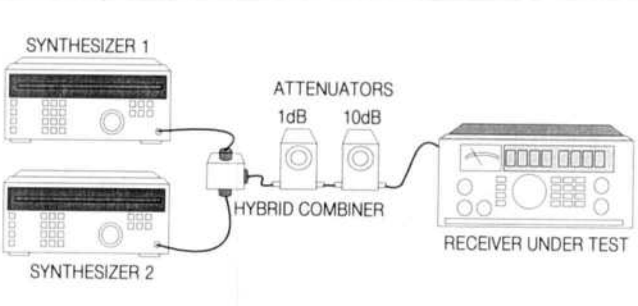


Fig 1—IMD measurement setup.

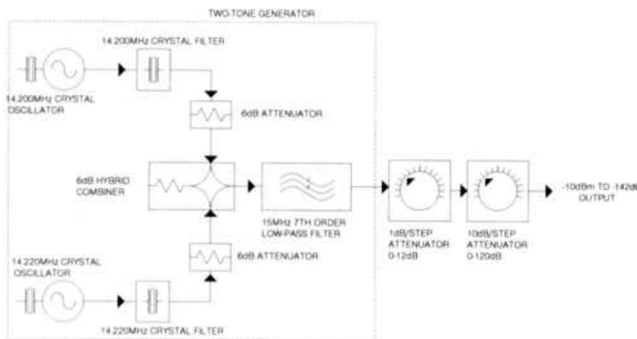


Fig 2—Block diagram of the two-tone source.

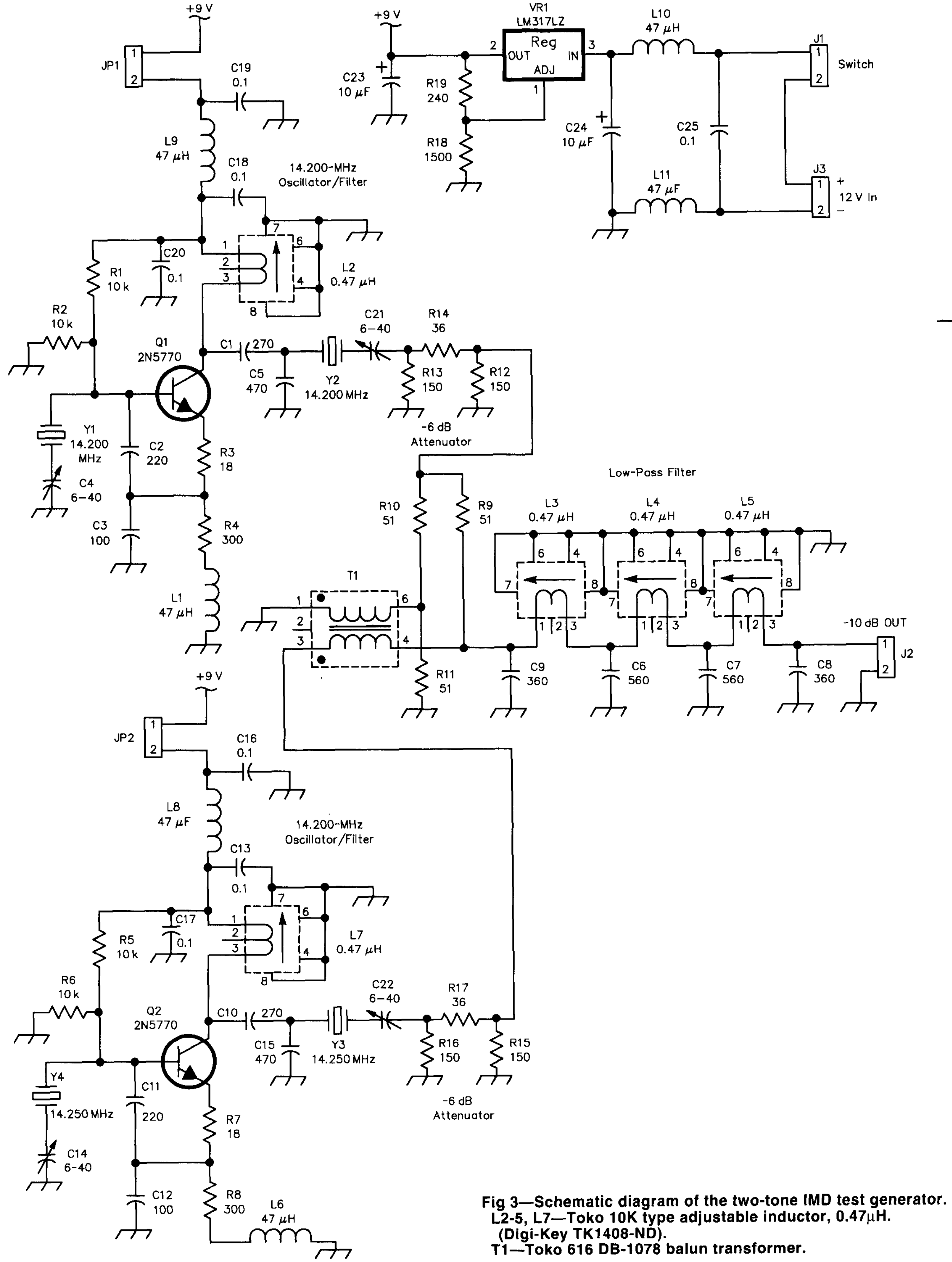
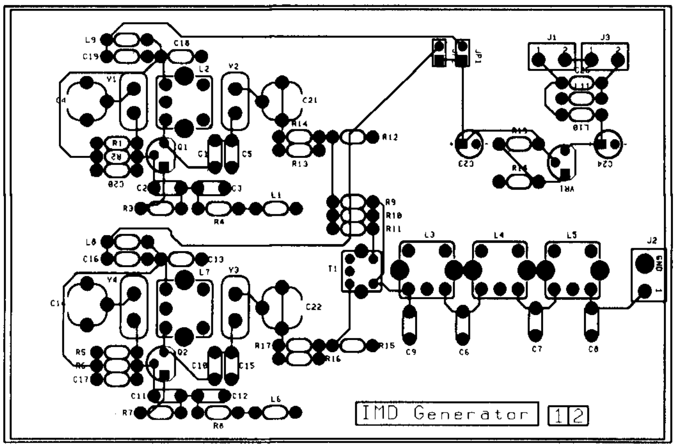


Fig 3—Schematic diagram of the two-tone IMD test generator.
L2-5, L7—Toko 10K type adjustable inductor, 0.47µH. (Digi-Key TK1408-ND).
T1—Toko 616 DB-1078 balun transformer.



Use good RF prototyping techniques in constructing your unit: short lead lengths, good grounding and shielding. Be particularly careful to provide some physical separation between the oscillators, crystal filters, combiner and harmonic filter. If inadequate shielding or isolation is provided, IMD products will form in the oscillators from cross coupling. Similarly, isolation is required around the crystal and low-pass filters in order to maintain their good cutoff characteristics.

Because there are a number of coils used in this project, I choose not to use hand-wound toroid inductors. I think they would be too tedious to wind and lack adjustability. The inductors used, however, are a quality product, inexpensive and readily available.

Alignment

Alignment of the generator is straightforward: only an oscilloscope, frequency counter and a 50-Ω feedthrough termination is required. The feedthrough termination is used with the oscilloscope to provide the proper load impedance to the generator.

First, each oscillator and filter combination is tuned up independently. Begin by disabling one of the oscillators by removing its supply jumper (JP1 or JP2). Optimize the output of the other oscillator by adjusting its collector inductor (L2 or L7) for maximum output on the oscilloscope. The oscillator is then set on frequency with the frequency counter by adjusting the appropriate trimmer, either C4 or C14. After the oscillator frequency is set, remove the counter and connect the oscilloscope again. Now carefully adjust the corresponding filter-tuning trimmer (C21 or C22) for maximum output. Repeat this process for the other oscillator and filter combination.

Adjustment of the output filter is not critical. Alternately adjust each inductor, L3, L4 and L5, for maximum output from either oscillator. Adjustment by this method should give second-harmonic attenuation values within a dB or so of what you might achieve using a network analyzer to set this filter.

Finally, with the oscilloscope and termination still connected, (by the way, the termination should be at the scope end of the cable), disable one of the oscillators at a time and again trim L2 or L7 for −10 dBm of output at each frequency. Minus 10 dBm corresponds to 200 mV pk-pk into 50 Ω.

Intermodulation Distortion (IMD) Measurements

By definition, any completely linear circuit element would produce no IMD products (see **Appendix A**). But any real receiver circuit exhibits some degree of nonlinearity. It is precisely this degree of nonlinearity we wish to measure by making IMD measurements. The most troublesome of the intermodulation products are the so-called third-order products. These are the $2F1-F2$ and $2F2-F1$ signals shown on the hypothetical spectrum analyzer of Fig 4. If you were using a receiver with similar IMD performance to copy a weak signal at or near one of these intermod frequencies, you would suffer some interference. The higher your receiver's IP_3 value (in dBm), the lower these third-order products—and the consequent interference—will be. Notice that the third-order IMD product ($2F1$-$F2$) in Fig 4 is shown as 80 dB below the two signals $F1$ and $F2$. If the power levels of $F1$ and $F2$ were to decrease by 10 dB, the power levels of the third-order IMD products would decrease by 30 dB. Because the levels of the third-order IMD products are dependent on the input signal level as well as the nonlinearities of the system, the third-order intercept or, IP_3, is a more useful figure of merit for system performance; it is independent of the signal amplitude.

Fig 5 is a graphical representation of the IP_3 concept. If the intercept point is known, the level of the third-order intermodulation product can be determined from the graph. In the example in Fig 5, if two tones at −38 dBm are applied to this hypothetical receiver system with an IP_3 of +3 dBm, their fundamental signal amplitudes would measure +40 dB over S9 and the third-order intermodulation products would measure S1.

Making the Measurements

To make third-order IMD measurements, connect the

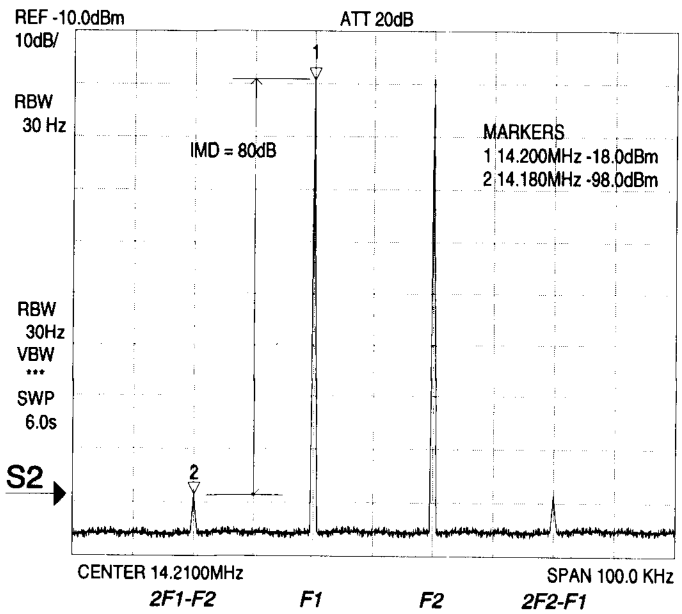


Fig 4—Typical third-order IMD products as seen on a spectrum analyzer.

Appendix A: Why is It Called Third Order?

The reason the particular type of distortion I have been referring to is called third order has to do with the derivation of the mathematical model for an imperfect amplifier. You will not need to be a math professor to follow this explanation, nor is it necessary that you be completely comfortable with a bunch of trigonometric relations. Just keep in mind that ω_1 and ω_2 are the two input signal frequencies in terms of radians per second and are equivalent to $2\pi f_1$ and $2\pi f_2$. Watch what happens to these terms as the power series is expanded for the cubic (third-order) term.

All amplifiers have some amount of distortion. In contrast to most general-purpose wide-band or video amplifiers, the outputs of RF or IF amplifiers are generally filtered. As shown in the amplifier configuration below, the filters effectively remove harmonically related signals caused by the nonlinear behavior of the amplifier. However, the third-order products are typically within the band-pass of these filters and therefore of particular concern.

If an amplifier had no distortion, its transfer function would be:

$$V_{out} = A_0 + A_1 V_{in}$$

where A_0 is just the dc offset and A_1 represents the coefficient of the desired linear gain. Because most real amplifiers do have some distortion, their transfer functions can better be represented by a power series polynomial:

$$V_{out} = A_0 + A_1 V_{in} + A_2 V_{in}^2 + A_3 V_{in}^3 + A_4 V_{in}^4 \cdots$$

For

$$V_{in} = V_1 \cos(\omega_1 t) + V_2 \cos(\omega_2 t)$$

the desired first order term, $A_0 + A_1 V_{in}$, gives the fundamental products

$$V_{out} = A_0 + A_1 V_1 \cos(\omega_1 t) + A_1 V_2 \cos(\omega_2 t)$$

The second order term, $A_2 V_{in}^2$, determines the second order products

$$A_2 V_{in}^2 = \frac{A_2 V_1^2}{2} + \frac{A_2 V_2^2}{2} +$$
$$\frac{A_2 V_1^2}{2}\cos(2\omega_1 t) + \frac{A_2 V_2^2}{2}\cos(2\omega_2 t) +$$
$$\frac{A_2 V_1 V_2}{2}[\cos(\omega_1 t + \omega_2 t) + \cos(\omega_1 t - \omega_2 t)]$$

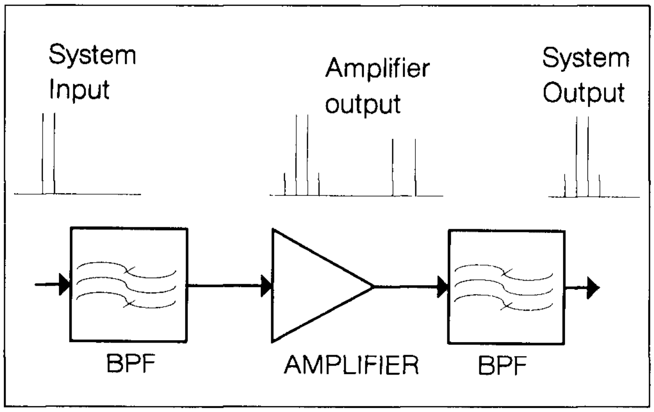
The first line shows the dc terms, the second line shows the second-harmonic terms, and the last line has the second-order IMD terms.

Here is where it gets interesting, the third-order term, $A_3 V_{in}^3$, gives us

$$A_3 V_{in}^3 = \frac{3A_3}{2}\left[V_1 V_2^2 + \frac{V_1^3}{2}\right]\cos(\omega_1 t) + \frac{3A_3}{2}\left[V_1^2 V_2 + \frac{V_2^3}{2}\right]\cos(\omega_2 t) +$$
$$\frac{A_3 V_1^3}{4}\cos(3\omega_1 t) + \frac{A_3 V_2^3}{4}\cos(3\omega_2 t) +$$
$$\frac{3A_3 V_1^2 V_2}{4}[\cos(2\omega_1 t + \omega_2 t) + \cos(2\omega_1 t - \omega_2 t)] +$$
$$\frac{3A_3 V_1 V_2^2}{4}[\cos(2\omega_2 t + \omega_1 t) + \cos(2\omega_2 t - \omega_1 t)]$$

The first line consists of terms at the fundamental (input signal) frequency. The second line again shows harmonic terms—third harmonics this time. The last two lines give the two third-order IMD products.

The difference terms in the third-order products are the troublemakers, and they usually turn up right next to a weak signal you are trying to copy. Notice that as the amplifier approaches the ideal, the coefficients (A_2, A_3,...A_n) would approach zero.



test receiver and attenuator set to the two-tone IMD source as shown in Fig 6. If you don't already have a set of step attenuators, you can build your own from designs in *The ARRL Handbook*. To get a feel for what to expect, begin by setting the attenuators for a combined value of 20 dB. Now tune the test receiver from 14.175 to 14.245 MHz; most receivers will have a noticeable third-order intermod at 14.180 and 14.240 MHz, plus the two very strong fundamental signals at 14.200 and 14.220 MHz. Your receiver should be set to either USB or LSB mode with the AGC on, RF attenuator to 0 dB and any preamplifier off. You may argue that the AGC should be off when making the measurements so that the RF and IF amplifiers will be at maximum gain. In principle I would agree, but the problem is that the S-meter will probably not be working with the AGC off, and it is required in order to make the necessary measurements.

Now that you have found the intermod signals, begin the measurement process by calibrating the receiver's S-meter. It is only necessary to calibrate the S-meter at one low-level value, say S1 or S2. This is done by setting the attenuators to a combined value of approximately 100-dB (–110 dBm) and tuning the receiver to either 14.200 or 14.220 MHz. Adjust the 1-dB step attenuator until the S-meter reads exactly S1 or S2. The choice of which to use depends on how responsive the meter is to a 1-dB attenuation change; some receiver's meters will not respond well at S1, so try S2 instead. Try to avoid using anything higher than S2 because the AGC will affect the linearity of the receiver's input circuits. The idea is that you should be able to resolve a

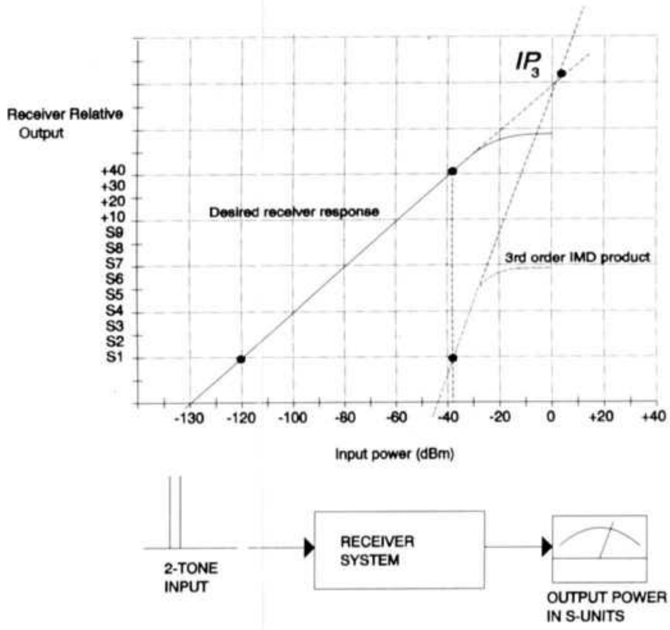


Fig 5—Third order intercept.

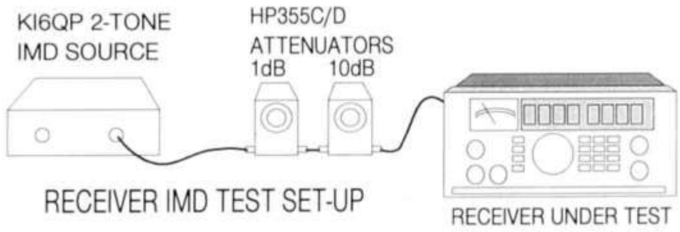


Fig 6—IMD measurement setup.

Table 1—Receivers Tested.

Manufacturer	*Model*	*P_A (dBm)*	*P_{IM} (dBm)*	*IP_3 (dBm)*
Drake	R-4C	−29	−92 at S3	+2.5
ICOM	IC-R70	−30	−96 at S1	+3.0
ICOM	IC-765	−29	−93 at S2	+3.0
ICOM	IC-781	−19	−95 at S1	+19.0
Kenwood	R599D	−50	−107 at S1	−21.5
Kenwood	TS-830S	−42	−107 at S1	−9.5

reference level within ±1dB of variation in input signal strength at some very low level near the receiver's noise floor. Now you should have an absolute signal level corresponding to a particular S-meter value. Record the signal level (P_{IM}) as −10 dBm minus the combined attenuator values.

Now, tune the receiver to either 14.180 or 14.240 MHz and increase the input signal level by decreasing the value of attenuation until the IMD signal is equal to the previously established S-meter calibration point. Record this signal (P_A) as −10 dBm minus the combined attenuator values. The IMD at this particular signal level is the difference in attenuator values $P_A - P_{IM}$. The IP_3 is equal to P_A plus half of the IMD value. Sounds confusing, so let's do an example:

With the receiver tuned to 14.200 MHz, let's say an S2 reading requires an attenuator setting of 88 dB. Therefore, P_{IM} = −10 dBm −88 dB = −98 dBm. And with the receiver tuned to IMD frequency of 14.180 MHz, the attenuator setting is found to be 18 dB. Therefore P_A = −10 dBm −18 dB = −28 dBm. So, IP_3 =[−28 dBm − (−98 dBm)] / 2 + (−28 dBm) = +7 dBm. This is derived from the more generalized form of intercept point:

$$IP_n = \frac{nP_A - P_{IM_n}}{n-1}$$

which in this particular example would look like:

$$IP_3 = \frac{3(-10\,\mathrm{dBm} - 18\,\mathrm{dB}) - (-10\,\mathrm{dBm} - 88\,\mathrm{dB})}{3-1} = 7\,\mathrm{dBm}$$

You can use either method you prefer. I like the first example because I can do it in my head. As another example, consider the hypothetical spectrum analyzer shown in Fig 4 and see if you can determine its IP_3 in your head: −18 dBm +80 dB/2 = +22 dBm.

Table 1 shows how a few familiar receivers compare in terms of IP_3 performance. Use this as a benchmark when testing your receivers.

For more insight on IMD characteristics, switch the test receiver's preamplifier on, if it has one, and run the test again. You should notice IP_3 decrease by a value approximately equal to the gain the preamp added. This is a good reason for leaving the preamplifier off! Try it again, but this time switch in the receiver's attenuator. The IP_3 should increase—at a detriment to the receiver's sensitivity. You may notice that either of the two IMD products ($2F1-F2$ or $2F2-F1$) is significantly lower than the other. This happens in most receivers as a result of the way the IMD products add in cascaded front-end stages. You should assume, however, that a receiver system can be no better than worst case (ie, don't average the two IP_3 results).

Conclusion

I hope you have found this article interesting and useful.

Sources

International Crystal Manufacturing Co, Inc (ICM), PO Box 26330, 10N Lee, Oklahoma City, OK 73126-0330. Tel: 800-725-1426. Specify a holder capacitance of 23 pF and order all four crystals with a matched series resistance of 15 Ω. Four crystals at the time of this writing were $40, including shipping.

Digi-Key Corporation, 701 Brooks Ave South, PO Box 677, Thief River Falls, MN 56701-0677. Tel: 800-344-4539. All parts except the crystal should be available from Digi-Key; no minimum purchase required; service charge on orders under $25.

The techniques described here should give you further insight into receiver design details as well as helping you select a commercial unit. Bare PC boards as well as assembled and tested units will be available soon. For more information contact the author.

Notes

[1]Rohde, Ulrich L., DJ2LR, "High-Dynamic Range Active Double-Balanced Mixer," *Ham Radio*, November 1977.

[2]Rohde, Dr. Ulrich L., KA2WEU, "Testing and Calculation Intermodulation Distortion in Receivers," *QEX*, July 1994.

References

Gruber, Mike, WA1SVF, "*QST* Product Reviews: A Look Behind the Scenes," *QST*, October 1994.

Reisert, Joe, W1JR, "High Dynamic Range Receivers," *Ham Radio*, November 1984.

Rohde, Dr. Ulrich L., DJ2LR, "Recent Advances in Short Wave Receiver Design," *QST*, November 1992.

Makhinson, Jacob, N6NWP, "A High-Dynamic-Range MF/HF Receiver Front End," *QST*, February 1993.

Rohde, Dr. Ulrich L., KA2WEU, "Key Components of Modern Receiver Design—*Part 1*," *QST*, May 1994.

Rohde, Dr. Ulrich L., KA2WEU, "Key Components of Modern Receiver Design—*Part 2*," *QST*, June 1994.

Rohde, Dr. Ulrich L., KA2WEU, "Key Components of Modern Receiver Design—*Part 3*," *QST*, July 1994.

Rohde, Dr. Ulrich L., KA2WEU, "Key Components of Modern Receiver Design: A Second Look," *QST*, December 1994.

Hawker, Pat, G3VA, "G3SBI's H-Mode Receiver Design," *Communications Quarterly*, Fall 1994.

Upcoming Technical Conferences

40th Annual West Coast VHF/UHF Conference

May 5-7, 1995, Sheraton Cerritos Hotel, Town Center, 12725 Center Court Drive, Cerritos, California.

For more information please call 714-990-9203 or fax 714-990-1340.

The Central States VHF Society Conference

July 27-30, 1995, Colorado Springs, Colorado.

Call for papers: Papers for inclusion in the conference proceedings for for presentation at the conference are due in early May 1995.

For more information contact Hal Bergeson, WØMXY, Program Chairman, 809 East Vermijo Avenue, Colorado Springs, CO 80903, tel: 719-471-0238.

1995 ARRL Digital Communications Conference

September 8-10, 1995, La Quinta Conference Center, Arlington, Texas—just minutes from Dallas/Fort Worth Airport. Co-hosted by Tucson Amateur Packet Radio (TAPR) and the Texas Packet Radio Society.

More information will be released soon, or contact the TAPR office at 8987-309 E. Tanque Verde Road #337, Tucson, AZ 85749-9399, tel: 817-383-0000; fax: 817 566-2544; Internet: tapr@tapr.org

Call for papers: Deadline for receipt of camera-ready papers is **July 21, 1995**. Contact Maty Weinberg at ARRL HQ (tel: 203-666-1541; fax: 203-665-7531; Internet: lweinberg@arrl.org) for infomation on submitting papers.

Microwave Update 95

October 26-28, La Quinta Inn, Arlington, Texas.

For more information contact: Al Ward, WB5LUA, 2306 Forest Grove Estates Road, Allen, TX 75002 or Kent Britain, WA5VJB, 1626 Vineyard, Grand Prairie, TX 75052-1405.

Up to Four Receivers on a Single Antenna

Connecting multiple receivers to one antenna won't work—unless you do it right.

By Michael Lass, DJ3VY

This article describes a system that can be used to connect up to four receivers to a single antenna. This can be useful for such applications as listening to different frequencies simultaneously, monitoring each sideband of a DSB signal independently or making comparisons between receivers.

At first glance, it may seem that connecting multiple receivers to a single antenna is a trivial matter. Just lash up a bunch of tee connectors to parallel the receivers across the antenna feed line. But this approach has potential problems. For one thing, you are almost guaranteed to have a poor impedance match to the feed line. And receivers typically have input impedances that are only nominally 50 Ω—in reality, they may be substantially different from 50 Ω—so connecting the receiver inputs in parallel will result in some receivers getting more of the input signal than others. And, of course, no receiver will get as much signal in this scheme as if it alone was connected to the antenna feed line.

Some of these problems could be addressed by using resistive splitters to connect the receivers, but mismatches will still occur, and resistive splitters are lossy. A better approach would be the use of hybrid couplers made from small transformers. These can split a 50-Ω source into several 50-Ω outputs while providing 20 to 50 dB of isolation between the receivers. But at best, connecting four receivers via hybrid couplers introduces a loss of over 6 dB to each receiver—and in practice it will be 0.5 to 3 dB more.

The solution, then, is to add an amplifier before the hybrid coupler. But such an amplifier has to be very good; it should not degrade the strong-signal or noise performance of the receivers. Fortunately, this amplifier doesn't need a lot of gain, as its job is just to overcome the losses of the hybrid coupler. A gain of about 8 dB should do it. We do need to concentrate on linearity in our choice of amplifier design, though. The measure of the amplifier's linearity is its third-order intercept point (IP_3). An oft-quoted value for a "good" IP_3 is +30 dBm, but I don't think that's quite good enough in environments where very strong signals are likely, as here in Europe. Substantially higher IP_3 values are needed than present amateur receivers exhibit—or promise.

The circuit presented here has been used with professional-grade receivers (Rohde and Schwarz EK085 and

43 Schlachte
28195 Bremen
Germany

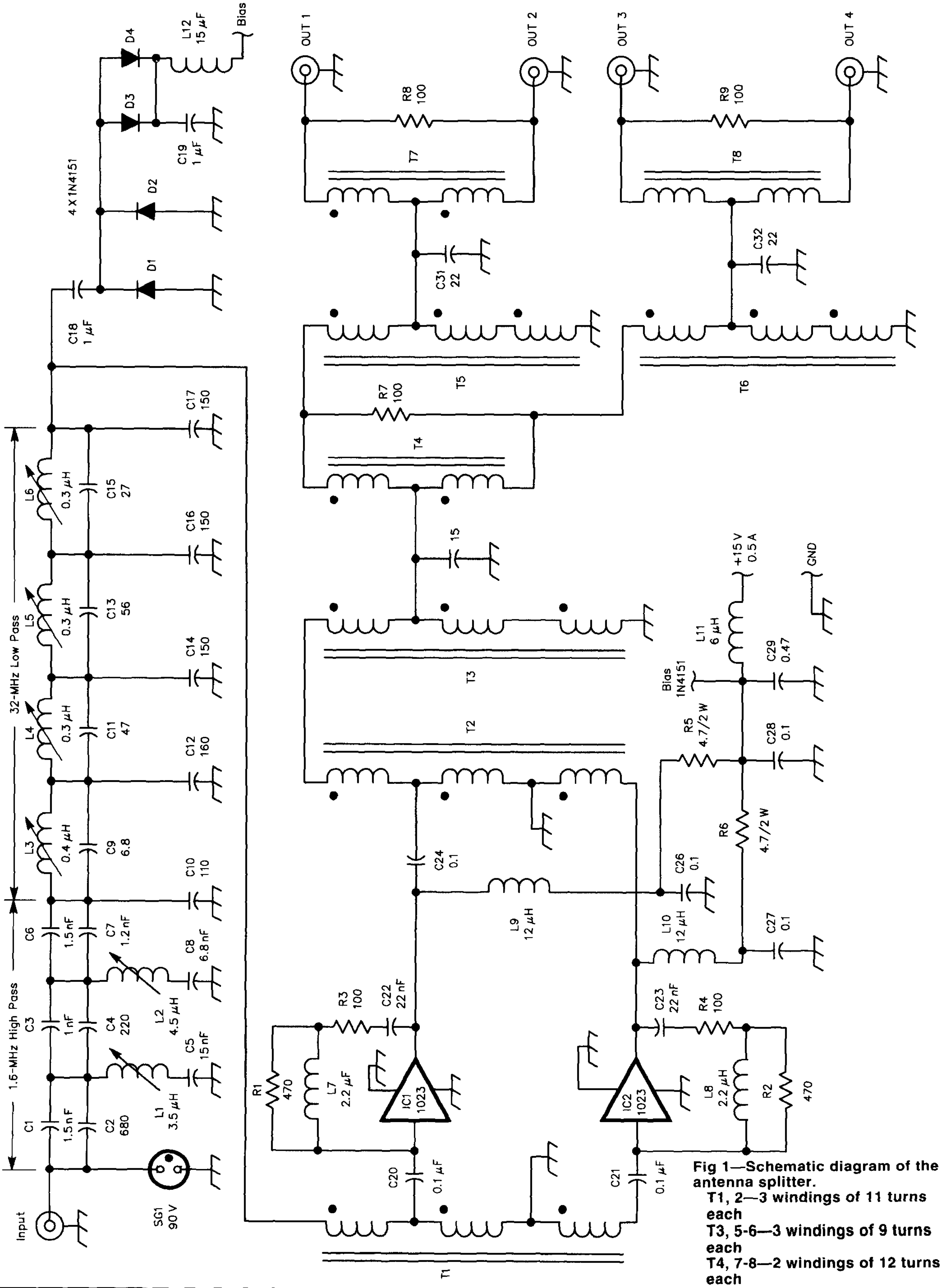


Fig 1—Schematic diagram of the antenna splitter.
T1, 2—3 windings of 11 turns each
T3, 5-6—3 windings of 9 turns each
T4, 7-8—2 windings of 12 turns each

EK890 models and a Telefunken E1800A) that have measured IP_3 values of between +35 and +42 dBm, and no degradation of performance is produced by this circuit. You can expect to spend about $200 to buy the components for this circuit.

Circuit Description

Fig 1 shows the circuit diagram, with the construction details shown in Fig 2. The central part of the antenna splitter consists of two push-pull driven Hewlett Packard MSA 1023 medium-power amplifiers, available from HP distributors. These amplifiers are designed for operation at up to several gigahertz and have a low noise figure, the right gain, a very large IMD-free dynamic range and suitable input and output impedances. Push-pull operation improves their performance, although it requires use of an extra transformer. But don't worry, the transformers are easy to wind.

To attenuate powerful MW broadcast signals and signals above 30 MHz, high- and low-pass filters are inserted between the antenna and the amplifier. Four voltage-limiting diodes are connected at the output of the filter to protect the amplifier from any transmitter signals accidentally applied to the input of the circuit. I haven't tested to see whether this will sufficiently protect the devices—for obvious reasons! If a highly selective antenna such as a beam is used, the input filters may not be necessary.

The two MSA 1023 devices are used as suggested in HP application notes. The bias for each amplifier is applied through a 4.7-Ω resistor (R5 and R6). Using separate resistors is good for thermal stability, which is needed since the total current consumption of the two devices is about 500 mA. The MSA 1023 inputs and outputs are connected to small-core transformers that ensure the correct phase and impedance relationships. In the original design, I discovered occasional instability at a frequency of 5 to 6 GHz using a spectrum analyzer. Since the gain of the amplifier was about 1.5 dB too high anyway, I cured this problem by adding a feedback network, which both eliminates the instability and helps to linearize the frequency response.

Three power splitters, built using toroidal cores, are used to split the amplifier output into four equal outputs. In theory, the loss through the splitters to each output should be 6 dB, a loss that is compensated for by the amplifier. If the system is built as shown, the frequency response should be quite linear. The cores of these transformers must not become saturated or nonlinearities will result, as has been noted in amateur literature previously. No such degradation of linearity was found in this circuit during testing with power levels of up to +20 dBm.

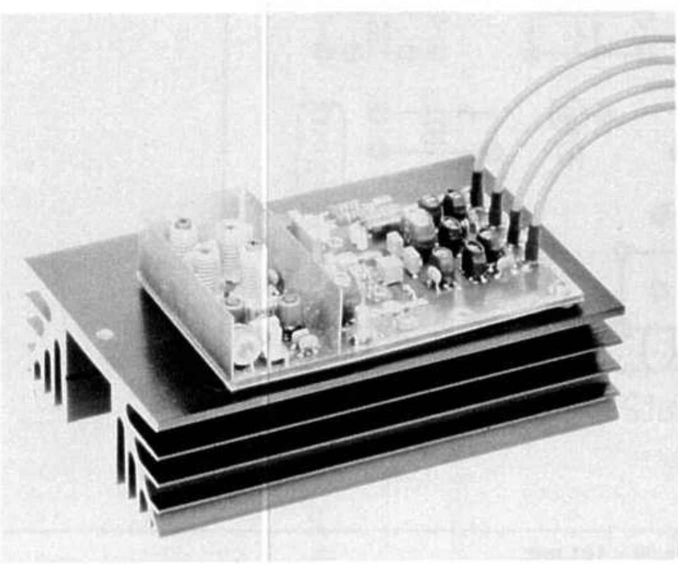


Fig 2—A rather large heat sink is used to mount the PC board.

Construction

Layout of the antenna splitter circuit is not critical as long as proper high-frequency techniques are observed. Use of a double-sided, plated-through PC board establishes the needed low-impedance ground reference. In my layout, I used as many plate-throughs as possible and used the ground leads for shielding. Details are in Fig 3.

The high- and low-pass filters are located very close to the amplifier input. Improved stop-band attenuation could probably be obtained by the use of additional shielding between the filter inductors, but the low interstage impedances in the filter make this unnecessary in practice. The high-pass part of the filter uses ferrite-core inductors, while the low-pass section uses adjustable coils from TDK. While the filter components aren't critical, the capacitors used should be either ceramic NP0 or silver-mica types to ensure long-term stability.

Mounting the two MSA 1023 devices is a bit tricky because they are inserted from the solder side of the board in order to achieve a good thermal connection to the heat sink. The specific type of heat sink used isn't critical as long as it provides a thermal resistance of 1 to 1.5°C/W.

All of the transformers are wound on Siemens N30 material toroid cores. Other types may be used if they have A_L values between 1000 and 2000. The winding technique is not critical; it doesn't require twisted wire.

My circuit-board layout supports use of SMB connectors or Tektronix probe-tip adapters (part numbers 131-27660-01 and 136-0352-02), but that's a matter of personal taste. I find these connectors ideal for use with RG-174 coax cable.

Testing and Alignment

If constructed without assembly errors, the splitter circuit should work immediately. Check to see that about 500 mA is being drawn from the 15-V supply and that about 13.6 V appears on the output pin of each MSA 1023. If that's the case, the amplifiers are probably working properly. The frequency response of the input filter is shown in Fig 4. You're welcome to replace the input filter with one having a different response if your requirements are different. Even without tune-up, chances are the filter will exhibit nearly this response. When the input filter isn't used, a frequency response of up to about 200 MHz (3-dB down) is available (Fig 5). If you want to use this circuit at VHF, you should use a low-pass filter to eliminate unwanted lower-frequency signals.

Performance

I tested the performance of this circuit using a spectrum analyzer and two high-quality signal generators. A hybrid combiner with a large core was used to combine the signal generator outputs. Don't try this using a resistive combiner, as the signal generators may cross modulate one another. My test setup exhibits an IP_3 of above +55 dBm, well above that needed to test this circuit.

At an amplification of 0 dB ±0.5 dB and at various frequencies in the 1.6 to 30-MHz range, the IP_3 of the an-

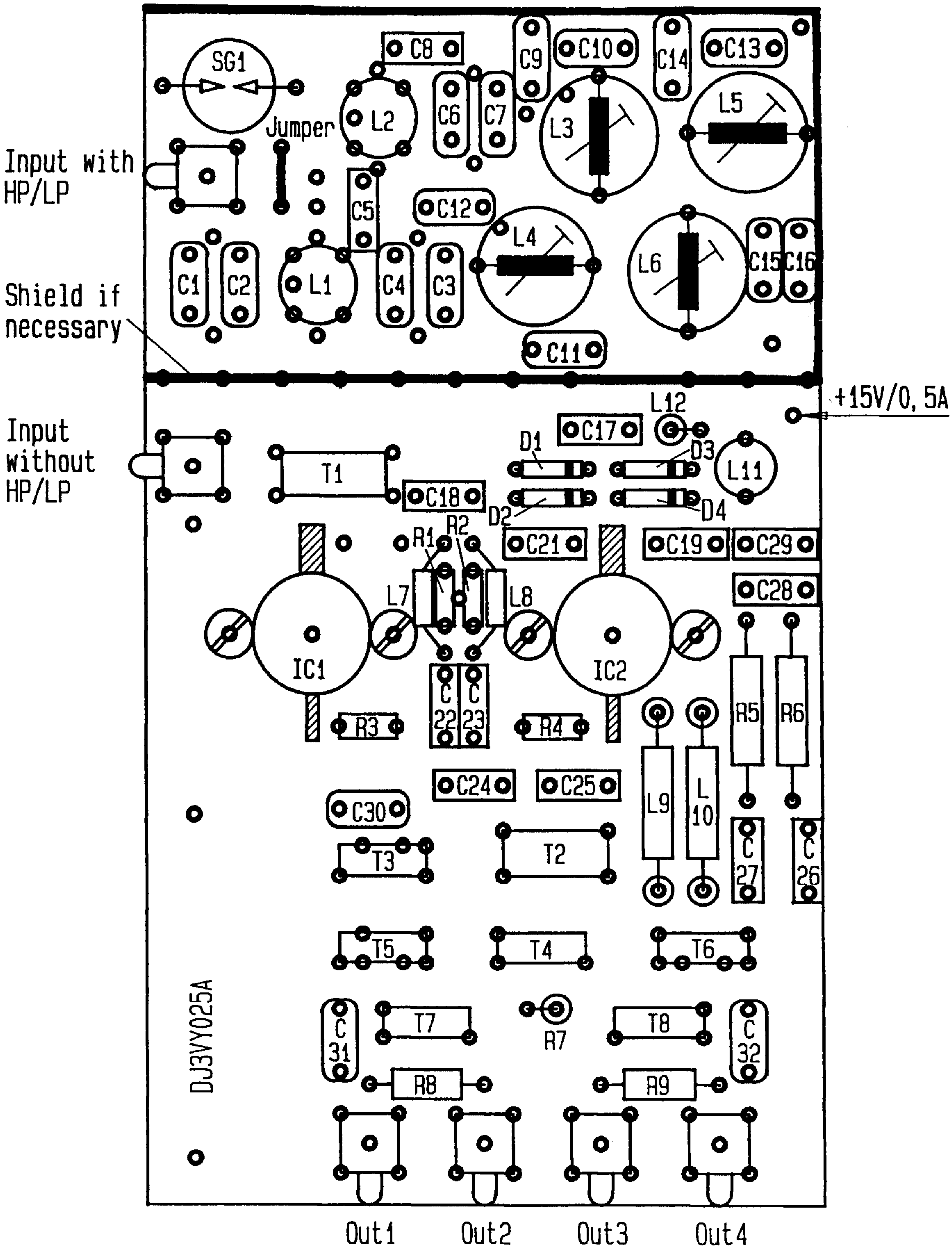


Fig 3—All components are placed on a PC board which measures 60 × 101 mm.

tenna splitter circuit measured as +42 dBm. (Two −10-dBm input signals were used to make the measurement.) The test setup is shown in Fig 6. Fig 7 shows the isolation between outputs of the splitter from 1 to 100 MHz. While the isolation might be improved, it is adequate in practice. The 1-dB compression point of the system was found to be about +25 dBm.

I wouldn't have minded if the IP_3 were a bit higher, as it is rather near the IP_3 of the best of my receivers. In practice, though, I've never seen any IMD-product signals, even on those frequencies known to be critical. If fact, the opposite is true: with this antenna splitter and good receivers, the bands are amazingly quiet.

Acknowledgments

My thanks to Manuela Dopieralla for translating this article from the original German and to Autje Wolpers for producing the original drawings.

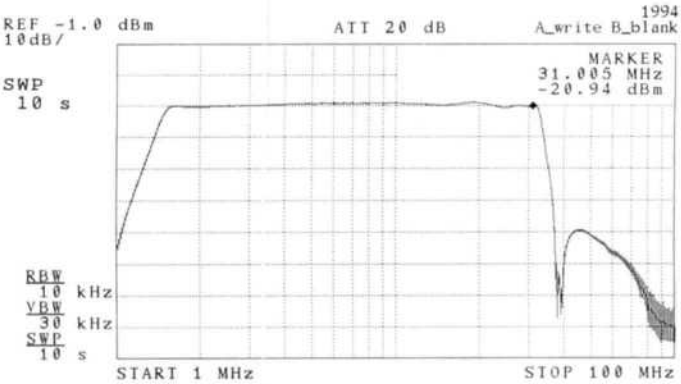


Fig 4—The frequency response is completely flat between 1.6 and 31 MHz.

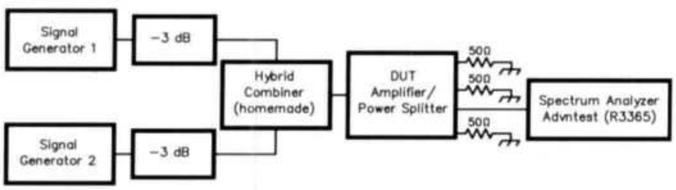


Fig 6—Two rather old SMLR signal generators from Rohde & Schwarz do an excellent job of IP_3 testing. Use an equivalent high-quality generator to do such tests.

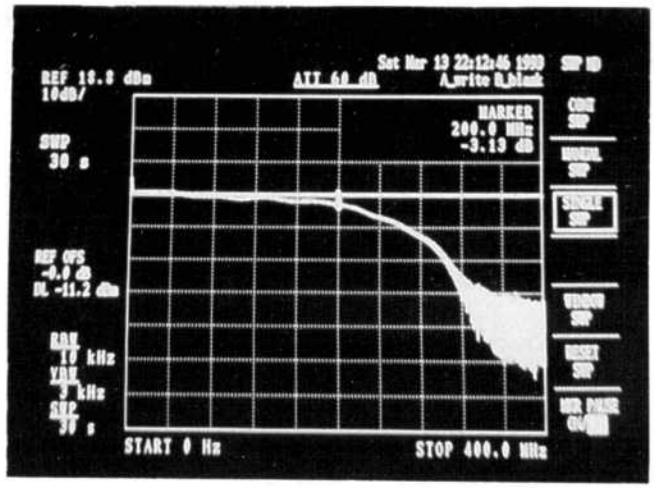


Fig 5—The antenna splitter frequency response without the low- and high-pass filters.

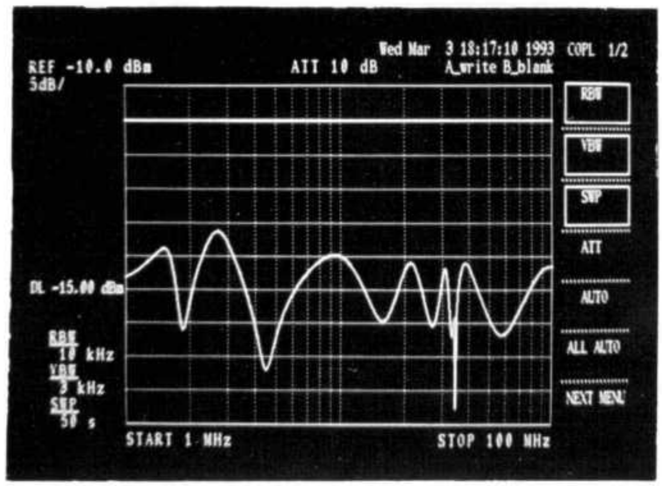


Fig 7—Measured isolation between the outputs.

Digital Communications

By Harold E. Price, NK6K

The Book of *Linux*—Chapter 1

Over the last few columns, we've developed a thread that suggests that we should not confuse the home/small office computing appliance (the device on which you keep your checkbook, do your taxes, your kids do their homework and you play *Doom*) with the packet-radio network device. Windows/DOS seems to be well established as the common platform for the home computing device, though IBM's *OS/2 Warp* and Microsoft's *Windows 95* will confuse this.

Windows/DOS may not be the best platform on which to host a packet-radio network device, however, and your computing appliance may not be the best hardware platform. If you want to use a second system and aren't limited to Windows/DOS, is there a better choice? Perhaps. A freeware UNIX derivative called *Linux* is making inroads into the amateur packet scene. UNIX is a true multitasking system, making it somewhat easier to develop network code and multiuser file and mail servers. UNIX also offers a true user login/shell facility. I invited *Linux* proponents to provide some details for us, and Bruce Perens, AB6YM, volunteered.

5949 Pudding Stone Lane
Bethel Park, PA 15102
email: nk6k@amsat.org (Internet)

Bruce has specialized in operating systems programming for media and communications since 1981, when he joined the pioneering NYIT Computer Graphics Laboratory. Today, Bruce works at Pixar, another graphics pioneer, where he has recently written software used to digitally restore vintage films such as *Snow White* and *My Fair Lady* and to perform effects on many current films.

Bruce's most active amateur mode is TCP/IP packet radio. He is on the board of directors of the Northern California Packet Association, for which he is an active participant in technical and FCC regulatory matters. I'm giving the rest of the column over to Bruce, to tell us about *Linux*.

Bruce on Linux

By now, most people have heard of *Linux*, the free-software clone of the UNIX operating system. *Linux* is really just the name of the "kernel," the software that provides services to programs and drives hardware devices. However, when people put that kernel together with the applications and utilities that make up a typical system, they call the result a "*Linux* system." I've been using a *Linux* system to operate my station for about a year now.

One of the major ways in which I use *Linux* is to communicate via packet radio. *Linux* provides raw AX.25 communication as well as TCP/IP over AX.25. Unlike most systems that support AX.25, *Linux* has AX.25 networking integrated into the kernel. The most important benefit of this is that many programs can use the networking features simultaneously. Programs that were written to use TCP/IP over the Internet automatically have the capability to use TCP/IP over packet radio added to them, even if the program's author has never heard of packet radio. For this reason, the interface I use to communicate via the Internet is the same interface I use to communicate on packet radio via TCP/IP. The only difference is that I'll start a program with an AMPR.ORG (amateur TCP/IP network) address rather than an address on the Internet.

A benefit of the multitasking and shared networking capabilities of *Linux* is that my system can run many different networking tasks simultaneously. For instance, I can download a file from one station while I'm connected to another station. If I use TCP/IP, file and mail transfers are automatic and can be scheduled to happen without my attention. Because the system is multitasking, I can also run my word processor and compile a new program at the same time I'm using packet radio.

Software

While *Linux* provides most of the same capabilities as *NOS*, KA9Q's DOS TCP/IP package, it does them in a different manner. *NOS* bundles all of the server and client programs for a diverse collection of services into one big program. *Linux* puts all of these facilities into separate small programs, all of which can be run simultaneously. This makes it much easier to change and add to the system. For example, if I want to modify the file transfer server, I can stop that server, edit the program, recompile, and restart using the new program, all without having to interrupt any of the other network services.

The most important difference between *Linux* and the programs that come with it and other systems such as Microsoft *Windows* is that the *Linux* system was built by thousands of people loosely collaborating via the Internet and working, for the most part, in their spare time. The second most important difference is that *Linux* comes with its source code. Why do you need source code? Many hams would refuse to buy a radio if they weren't able to get the schematic or service manual for that radio. They don't, however, apply that same standard to software, because so many manufacturers keep their source code under wraps. Thus, while they might be able to repair their radios, they are helpless to do anything but plead for attention from the manufacturer when software breaks. This is different for *Linux*: you can fix a program by yourself, though you generally don't have to. When I find minor bugs in my *Linux* system and post a message about it to an Internet *Linux* discussion group, someone who understands the program generally pops up and repairs the bug, often within a day. No commercial software that I've used has been able to meet this customer service record. The volunteers who publish the *Debian Linux Distribution* use a formal bug-tracking system like those of commercial companies and have logged and repaired 500 software bugs in the prerelease testing they've performed in the last few months. Their latest release is the result of a year-long collaboration by about 100 people.

Besides the benefits of having the source and being able to extend and repair the system, a major advantage that *Linux* derives from its UNIX heritage is that it's easy to program. This is important even for nonprogrammers: we amateurs depend on software written by volunteers, and many of those volunteers would rather program UNIX than another operating system.

Linux *can run many different networking tasks simultaneously.*

One more important benefit of *Linux*: when you copy the DOS you bought for your first computer over to your second computer, you're a software pirate. Not so for *Linux* users—the operating system and applications are free software, and you're allowed to copy them as much as you like.

When using TCP/IP on the air, my *Linux* system provides sophisticated services such as a World Wide Web server and client, a gateway between packet radio and the Internet and a packet BBS. For designing new hardware, there is a printed-circuit board program, several digital and analog circuit simulators and a symbolic mathematics program that can be used for antenna design and other electronic modeling applications. When I'm writing software, my *Linux* system provides compilers for C, C++, FORTRAN, and an assembler. There are interpreters for languages such as *Perl*, *Awk* and *Tcl / Tk*—even *BASIC*. I use software-maintenance tools such as RCS and Make.

Other programs I use are a satellite tracker, a CW practice program and a call-sign database. If I'm feeling artistic, there is 3-D rendering software and other image-manipulation programs, or I can make music with the sound board and MIDI software. And of course, there are the 1000 or so programs that one expects on any UNIX workstation. I generally interact with my *Linux* system though a text screen similar to the one that DOS uses. Other users prefer a windowing interface to text screens, and they use "The X Window System." I do switch to the window system when I run graphics programs such as the printed circuit board designer.

What did I pay for all of this software? Nothing. It was all available for free on the Internet. If I'd purchased it on a CD, it might have cost $30, still a tremendous bargain.

Are You Ready?

By now, you might have guessed that *Linux* may be too sophisticated for some amateurs. I'll offer some hints to help you decide if it's time for you to jump on the *Linux* bandwagon.

You might *not* like *Linux* right now if:

- You don't like to read the manuals of the software and hardware you buy.
- You've never built any of your own equipment, modified a radio, or plugged a board into your PC.
- You have only one computer and you want to run *Quicken* or other commercial applications on that computer.

In a few years, these reasons won't be valid any longer, as the pioneers will have done their job and *Linux* will be more palatable for the average ham.

You probably *should* use *Linux* if:

- You consider yourself an experimenter, or would like to be a pioneer.
- You run a full-service packet BBS.
- You write software.
- You build equipment.
- You're frustrated when you encounter a software bug, because if you just had the source code to the program you could fix it on your own.
- You read *QEX*.

Hardware

Linux also requires a reasonably powerful computer system. Don't bother asking computer store sales people about *Linux* hardware compatibility. Their job is to sell Windows systems to nontechnical people, not to understand *Linux*. Here are the minimum hardware requirements that I suggest:

- 386, 486 or Pentium processor.
- ISA-bus motherboard. VLB or PCI local bus is optional. PCI is swiftly be-

coming the standard local-bus interface.

- 4 MBytes of RAM. Get 8 MBytes if you want to run The X Window System.
- System cabinet with 150-W or higher power supply.
- 80-MByte disk and interface card. IDE, SCSI or the old MFM and RLL drives are fine. Don't bother with the disk interface cards that contain on-board cache memory—*Linux* provides a better cache. As I'm writing this, a 1 gigabyte (1000 megabyte) IDE disk can be had for $385.
- CD-ROM drive. Double-speed drives are $130 at this writing, quad-speed is down to $199.
- Floppy disk drive: 3.5-inch, 1.44MB or 5.25-inch, 1.2MB.

VGA or Super VGA display card. There have been some problems with supporting Diamond cards because they weren't releasing the details of their hardware—this isn't a *Linux* problem so much as it's a problem for The X Window System. Fortunately, Diamond is yielding to consumer pressure, and this issue may have been solved by the time this is published.

- Monochrome or color VGA or SVGA video monitor.
- Serial interface card. For best results, use one with 16550A-compatible UARTs.

To the above, you can add:

- Backup tape drive. Someday you'll wish you'd had one, and it'll be too late.
- Sound card.

For more detailed information on hardware compatibility, read the World Wide Web page: http://www.rahul.net/perens/Linux/Hardware.html or send an e-mail request to: Perens@Rahul.net for a copy of the hardware compatibility file.

And now, how to interface to the radios: for packet, any serial-interface TNC will work. The best serial TNCs have the fastest host bit rate, today this is 38,400 bit/s. You'll be operating the TNC in KISS mode, and the BBS software will run in your computer, so both the amount of memory in the TNC and its BBS capabilities—if any—are irrelevant.

The very best and fastest packet interface supported under *Linux* today is the Ontario *PI II* card. There is a driver for the Baycom *SCC* card, but this driver doesn't work on the *DRSI* cards at this writing.

How to Get It

To save other amateurs the effort of collecting all of these programs into one system, I am distributing a version of *Linux* with amateur-radio applications and many other programs called *Linux for Hams*. The distribution is available for free if you retrieve it via FTP, and for about $30 on CD-ROM. For information about how to get *Linux for Hams*, read the World Wide Web page: http://www.rahul.net/perens/LinuxForHams or send an e-mail request to: Perens@Rahul.net for the *Linux for Hams* introduction.

Following is a copy of the latest *HAM-HOWTO* for *Linux*, a document that describes software for amateur applications. ☐

Linux HAM-HOWTO, Amateur Radio Software List

Terry Dawson, VK2KTJ, terryd@extro.ucc.su.oz.au v1.5 14 Jan 1995

It is hoped that this list will assist Amateur Radio operators in finding and trying the various amateur radio software that has been written for, or ported to Linux. It is also hoped that as a consequence of this information being available that more amateur radio operators will choose Linux as the platform of choice for their experimentation, and that software developers will choose Linux as the platform for their software development, further expanding the role of operating systems like Linux in the Amateur Radio field.

Contents

1 Introduction.

This list was prompted by comments that had been expressed in the various Linux newsgroups about the number of amateur radio operators that were involved with Linux. It seemed to me that I was catching snippets of information here and there relating to development efforts taking place, but I never really knew where to locate either the person responsible for a particular piece of software, or the software itself. The list is growing as new packages are found.

This list is not limited to non-commercial software in any way. Taking a serious look at most amateur radio fields shows that most good developments are those that are designed by individuals and that commercial entities have taken up and disseminated to the mass market. I'd like to see the same happen for Linux support for Amateur Radio software too.

This list was originally called the RADIOLINUX list, but Matt Welsh suggested that there was no reason why it shouldn't be distributed with the Linux Documentation Project documents, so it has been renamed the

HAM-HOWTO and will be made available with the rest of the Linux Documentation. I make no apologies for the name.

1.1 Changes from the previous version

```
Additions:
        Ping/Pong convers server.

Corrections:
        Updated locations of some pieces of software.
        Updated version of SatTrack to 3.0
```

2 Where to obtain new versions of this list.

This list will be periodically posted to the **comp.os.linux.announce** newsgroup, and to the **HAMS** list on **niksula.hut.fi**.

It is also available from the following World Wide Web sites:

The Linux Documentation Project runs a Web Server and this list appears there as The HAM-HOWTO (http://sunsite.unc.edu/mdw/HOWTO/HAM-HOWTO.html).

Dennis Boylan N4ZMZ <dennis@nanovx.atl.ga.us> makes it available at the following three locations: *www.com* (http://www.com/linux/radio/index.html), *www.hboc.com* (http://www.hboc.com/linux/index.html) and *www.lan.com* (http://www.lan.com/linux/index.html)

John Gotts N8QDW <jgotts@engin.umich.edu> makes it available at: *www.engin.umich.edu* (http://www.engin.umich.edu/ jgotts/linuxhamsoft.html).

Alan Hargreaves VK2KVF <alan@dap.CSIRO.AU> makes it available in Australia at: *www.dap.csiro.au* (http://www.dap.csiro.au/RadioLinux).

Please let me know if you'd like to make it available somewhere too. I'd like to see it on some Web Servers that are accessible from radio.

3 Satellite Software.

The following software is for use in experimentation with Satellite communication.

3.1 MicroSat Ground Station Software

Author

John Melton, G0ORX/N6LYT, g0orx@amsat.org and Jonathan Naylor G4KLX, g4klx@amsat.org

Description

Microsat Ground Station software

Status

BETA. Version 0.9-Xaw released

System requirements

Alan Cox's kernel based AX.25 support ver 1.1.12 or better. X-Windows. The programs make use of the Athena Widgets and look much better with the 3D libraries.

Detail

This software allows you to use of a KISS tnc to directly communicate with the Microsat series of satellites. It provides an Athena Widgets based X-Windows interface, and allows you a comprehensive range of means of interacting with the satellite. The software should work with any window manager.

The software provides the following programs:

xpb:
broadcast monitor

xpg:
ftl0 file upload program, message upload program

xtlm:
telemetry display program

downloaded:
downloaded file list viewer

directory:
directory list viewer

message:
message preparation application

viewtext:
uncompressed ASCII text file viewer

viewlog:
display the contents of some log files

xweber:
special program for downloading webersat images

phs:
general purpose PACSAT header stripper

Where and How to obtain it.

John's software is available from:

ftp.ucsd.edu (ftp://ftp.ucsd.edu/hamradio/packet/tcpip/incoming/microsat-0.9-Xaw.tar.gz) or *ftp.funet.fi* (ftp://ftp.funet.fi/pub/ham/satellite/microsat/microsat-0.9-Xaw.tar.gz).

Please check for new versions.

Licensing/Copyright

GNU Public License. Freely redistributable. No warranty.

Contributed by:

John Melton, G0ORX/N6LYT, Alan Cox, GW4PTS, Jonathon Naylor, G4KLX

3.2 SatTrack - Satellite tracking program

Name

SatTrack

Author

Manfred Bester, DL5KR, manfred@ssl.berkeley.edu

Description

A VT100 and X11 based satellite tracking program.

Status

Version 2.0 is a release version.

System requirements

A vt100 terminal and or X11 server. A Maths Coprocessor for good performance, though it seems to work just fine on my 486sx25.

Detail

SatTrack provides a real-time or predictive display of Satellite orbit data. The current version uses a VT100 display to provide a text based interface to the data and an X11 based display to provide a graphical view of the orbit data.

Compiling the software under Linux is quite straightforward. Manfred has designed the Makefile to compile the software directly under your home directory, this is easy to change.

The steps I took were:

```
# cd /usr/src
# export HOME=/usr/src
# gzip -dc sattrack.V3.0.tar.gz | tar xvf -
# cd SatTrack/src
# vi Makefile
  {Comment SUN4 compile options}
  {Uncomment the linux options}
  {Select the options you want}
# make
```

Where and How to obtain it.

SatTrack can be found at: *ftp.jvnc.net* (ftp://ftp.jvnc.net/priv/kupiec/sattrack/sattrack.V3.0.tar.Z) or *ftp.funet.fi* (ftp://ftp.funet.fi/pub/ham/satellite/tracking/sattrack.V3.0.tar.Z).

Licensing/Copyright

Copyright (c) Manfred Bester. Permission is granted for educational, research and non-profit purposes. Prospective commercial users should seek permission from the Author. Read doc/COPYRIGHT for the actual copyright details.

Contributed by:

Manfred Bester, DL5KR

4 Shack Automation Software.

Software for simplifying tasks in the shack. Examples might include software for controlling the newer breed of radios, logging programs, QSL database, or antenna rotation.

5 Packet Radio

Software for use in conjunction with, or for facilitating packet radio.

5.1 Kernel Based AX.25 networking.

Author

Alan Cox, GW4PTS, iialan@iifeak.swan.ac.uk

Description

Software that allows the Linux Kernel to perform AX.25 networking.

Status

ALPHA. Quite stable though.

System requirements

Linux kernel 1.0 or later

Detail

Alan's software provides the programmer with a berkeley socket based interface to the AX.25 protocol. AX.25 sockets can opened for either connected, or connectionless modes of operation. Support to allow tcp/ip over AX.25 is provided. The user applications has been Jonathon G4KLX. The software comes in two parts, a kernel patch, and the user programs. The user programs included are:

axadd
to manipulate the AX.25 ARP table

axattach
to convert a serial device into a KISS device.

axl

an AX.25 listener designed to start a PMS when it receives an incoming connection. The PMS is still very new.

axsetcall

to change the callsign of a port.

beacon

generated beacon messages every 30 minutes.

call

A linemode AX.25 connection program. Call allows you to make connections to other AX.25 nodes. It provides file transmit and receive capabilities, and newer versions allow YAPP binary file transfers.

listen

a demonstration of how to use intercept AX.25 frames at the raw packet level. Useful as a building *block for packet tracing for example.*

Where and How to obtain it.

There are a number of different versions of the software. You must choose the one that suits your version of Linux kernel. The software is available from: *sunacm.swan.ac.uk* (ftp://sunacm.swan.ac.uk/pub/misc/Linux/Radio/)

More detail on where and how to obtain the software is provided in the *plain text version of the NET-2-HOWTO* (ftp://sunsite.unc.edu/pub/Linux/docs/howto/NET-2-HOWTO) or if you have WWW access at: *NET-2-HOWTO via WWW* (http://sunsite.unc.edu/mdw/HOWTO/NET-2-HOWTO.html).

Licensing/Copyright

Most of the software is covered by the GNU Public License, some of the software is Copyright by the Regents of University California Berkeley, and small portions of the user programs are Copyright Phil Karn KA9Q, whose copyright allows unrestricted use by Amateur Radio, Educational Institutions and Commercial KA9Q OEM license holders.

Contributed by:

Terry Dawson, VK2KTJ.

5.2 JNOS

Author

Brandon Allbery, KF8NH, bsa@kf8nh.wariat.org

Description

Brandon ported JNOS to Linux. His port is currently of the 1.09 (aka 1.08df) release of JNOS. Future versions will probably be based on Doug Crompton's evolution of 1.08df. Brandon suggests that JNOS for Linux is primarily of interest to people with existing DOS-based NOS configurations (especially server/switch configurations) who wish to switch to Linux or to escape the 640K barrier.

Status

The current version is ALPHA.4. Brandon is still tracking down bugs, as well as evolving new features.

System Requirements.

Any version of Linux along with *ncurses* 1.8.1 or 1.8.5 (the latter is preferred). Optionally you will require Linux networking (at least loopback) and *slattach* (kernels *pre*-1.1.13 or post-1.1.20 *required* for this).

Detail

If you include the Linux networking code in addition to JNOS, you can link the two by a slip link running over a pty, so that Linux can provide servers to your radio users. In this way you can easily add servers without having to build them into JNOS itself. Brandon has supplied the following list of known bugs:

- some servers seem to be causing unexplained exits.
- PPP is reported *not to work in ALPHA.4, although it worked in ALPHA.3*
- BBS forwarding when convers is compiled in but not configured causes core dumps.
- *the finger server* is getting bad filenames.
- the bbs W command is case-munging filenames.

Where and How to obtain it.

You can obtain JNOS for Linux ALPHA.4 at: *ftp.ucsd.edu* (ftp://ftp.ucsd.edu/hamradio/packet/tcpip/linux/j1091xA4.tgz) *ncurses* is available on most Linux ftp sites.

Licensing/Copyright

Brandon's modifications to JNOS are public domain. Most of *the* pre-existing NOS *code is copyrighted* and restricted to non-commercial use by the various contributors from Phil Karn on.

Contributed by:

Brandon Allbery, KF8NH

5.3 N0ARY Packet BBS for UN*X

Author

Bob Arasmith, N0ARY, *ported to Linux (and others) by Bob Proulx, KF0LW*, rwp@fc.hp.com

Description

A packet bbs implemented under UN*X.

Status

ALPHA. Bob is running version ARY-0.9 on both Linux and HP-UX. The next version from N0ARY which is called 4.0 is in alpha test/development release. Bob has not been able to bring this online yet and is still running 0.9+kf0uw mods. Bob has about half it working with Alan Cox's AX.25 kernel which would be the desirable combination.

System Requirements.

Linux installation, C Compiler, plus HAM radio TNC hardware.

Detail

This bbs has an excellent packet user interface. It has a compatible set of *commands with the* RLI bbs so users will be familiar with it immediately. It then extends the command set to be a very nice natural language style interface (e.g. "list at allus about KPC-3"). Many csh style bang commands are supported. Also included is a mail interface to provide a packet to internet gateway.

Currently you need to customize the source code for your installation so you need some C programming proficiency.

Where and How to obtain it.

For a Linux version or for Linux information send mail to rwp@fc.hp.com. For a SunOS version contact bob@arasmith.com. This code is not packaged *for distribution* yet *since it* is not past the alpha stage of development.

Licensing/Copyright

Copyright by Bob Arasmith, N0ARY, but freely redistributable.

Contributed by:

Bob Proulx, kf0uw, rwp@fc.hp.com

5.4 MBL/RLI message to NNTP and *email* converter.

Author

SM0OHI, pine@it.kth.se

Description

Software that will convert incoming MBL/RLI messages into either NNTP or RFC-822 formatted mail messages.

Status

Development, not yet released.

System requirements

Unknown

Detail

This software would be ideally suited to those who want to establish a mail and news gateway between conventional tcp/ip networks and the amateur radio mail network.

Where and How to obtain it.

Not yet availale.

Licensing/Copyright

Unknown

Contributed by:

SM0OHI

5.5 Single floppy disk AX.25 router.

Author

Alan Cox, GW4PTS, iialan@iifeak.swan.ac.uk

Description

A single floppy disk version of linux with enough software to allow a PC to act as an AX.25/IP router.

Status

Development, not yet released.

System requirements.

As for any Linux system a 386SX class PC or better.

Detail

No detail yet.

Where and How to obtain it.

Not yet available.

Licensing/Copyright

Unknown/Undecided.

Contributed by:

Terry Dawson VK2KTJ

5.6 TNT.

Author

Mark Wahl, DL4YBG, DL4YBG @ DB0BLO.#BLN.DEU.EU, wahlm@zelator.de

Description

A Hostmode terminal program for TNC's that support the WA8DED hostmode protocol.

Status

Version 0.8 is *reportedly* stable.

System requirements

TNC supporting WA8DED hostmode protocol, serial line.

Detail

TNT is a full featured hostmode terminal program. Compiling it on Linux is as difficult as untarring the source and typing 'make'. It provides both a 'dumb' terminal and X11 version. It comes with some comprehensive documentation which describes its features in some detail. Its main features are:

Multiple sessions:

TNT supports multiple packet radio connections on virtual screens that you can switch between. Each session window provides split screen (seperate transmit and receive text) operation with a status line. Commands can be entered in either a command session, or in any terminal session using a command key. Each of the virtual screens can be larger than the physical screen and can be scrolled around.

Data capture, file transfer and logging:

A number of options are available for logging received text to files. You can log transmit, receive or both to a log file. You can overwrite or append received data to existing files. You can use the 'autobin' protocol to transmit or receive binary files.

Remote Shell operation:

You can provide a shell to remote users so that they can access other programs on your computer. You can also run a program and direct its input/output to a channel so that users can use it.

Redirection of devices to a channel:

TNT allows you to redirect input/output from a channel to a device, a modem for example.

Umlaut conversion:

TNT provides umlaut conversion if necessary.

Remote mode:

Remote users can issue command themselves if allowed to do so.

Socket mode:

You can configure tnt so that it opens a socket for users to telnet to that allows them operation just as you have from the console. This is useful where your linux machine is on an ethernet and you want other terminal on the net to have use of your radio.

Where and How to obtain it.

The software is available by anonymous ftp from: *ftp.funet.fi* (`ftp://ftp.funet.fi/pub/ham/unix/packet/tntsrc08.tgz`). This site is mirrored at a number of places so you will probably find it at other places as well.

Licensing/Copyright

GNU Public License. Freely redistributable, no warranty.

Contributed by:

Steffen Weinreich DL5ZBG

5.7 IPIP encapsulation daemon.

Name

Mike Westerhof's IPIP encapsulation daemon.

Author

Mike Westerhof KA9WSB (original code), then Bdale Garbee N3EUA (port to BSD), then Ron Atkinson N8FOW and John Paul Morrison (port to Linux).

Description

A daemon that will allow you to use your linux machine as an IPIP encapsulating gateway.

Status

Ron is running this code himself, and the other version have been run quite successfully.

System requirements

KISS TNC, any version of Linux supporting RAW sockets.

Detail

If you've ever used an internet gateway from your amateur packet radio you have probably been connected by an encapsulation gateway of some descpription. Most run KA9Q's NOS and DOS, but others run Unix and this daemon. It allows you to encapsulate IP within IP, so that you can 'tunnel' IP connections over the Internet. The software allows you to connect a KISS TNC to your linux machine and to have all datagrams received on it carried across the Internet to a similar gateway. Ron has supplied a `Makefile` that will allow you to easily compile to software. It compiled without errors on my system running a recent version kernel. Ron has also written some `README` files which give you the detail necessary to compile the system and the original docum entation describes how to configure it.

Where and How to obtain it.

This software is available in the *sunsite.unc.edu ham apps directory* (`ftp://sunsite.unc.edu/pub/Linux/apps/ham/ipip.tar.gz`).

Licensing/Copyright etc.

Freely distributable, though Bdale asks that if you use the code and like it you might send him a QSL card or a postcard.

Contributed by:

Ron Atkinson, N8FOW

5.8 AXIP encapsulation daemon.

Name

Mike Westerhof's AXIP encapsulation daemon.

Author

Mike Westerhof KA9WSB, ported to Linux by Ron Atkinson N8FOW.

Description

A daemon that will allow you to use your linux machine as an AXIP encapsul ating gateway.

Status

Not properly tested, but assumed to work ok.

System requirements

KISS TNC, any version of Linux supporting RAW sockets.

Detail

This daemon is the partner to the IPIP encapsulation daemon. It allows you to encapsulate AX.25 frames in IP to carry them across the Internet. This is useful for linking AX.25 networks in remote areas. Ron has supplied a `Makefile` for linux, and with it the software compiled without errors on my system running a recent version kernel. Ron has also written some `README` files which give you the detail necessary to compile the system and the original documentation describes how to configure it.

Where and How to obtain it.

This software is available in the *sunsite.unc.edu ham apps directory* (`ftp://sunsite.unc.edu/pub/Linux/apps/ham/ax25ip.tar.gz`).

Licensing/Copyright etc.

Freely distributable so long as the original copyright notice is not removed.

Contributed by:

Ron Atkinson, N8FOW

5.9 Ping-Pong Convers Server

Name

Fred Baumgartens Convers Server for Linux.

Author

Fred Baumgarten, DC6IQ, <dc6iq@insu1.etec.uni-karlsruhe.de¿

Description

This is a version of the convers server that allows multiple users to simultaneously chat with each other in a round-table style conversation. It is compatible with the NOS convers servers, but provides extra facilities such as saving/restoring of Personal Text and Chanel Topics.

Status

In use of a number of popular convers servers, appears very stable.

System requirements.

Linux, GNU make, C compiler.

Detail.

Complete installation instructions are included in the `INSTALL` file that is included in the distribution.

Where and How to obtain it.

A distribution of this software is available in the *sunsite.unc.edu ham apps directory* (`ftp://sunsite.unc.edu/pub/Linux/apps/ham/convers-941122.tar.gz`). The home of the software is at *insu1.etec.uni-karlsruhe.de ftp server* (`ftp://insu1.etec.uni-karlsruhe.de/pub/hamradio/convers/convers`)

Licensing/Copright etc.

Presumably copyright Fred Baumgarten (not specifically stated), but some portions appear to be free for non-commercial use and copying provided the copyright notices stay intact.

Contributed by:

Terry Dawson, VK2KTJ

5.10 Wampes

A port of Wampes to Linux. Could someone send me details of the latest wampes release please ?

6 Morse Code

Software for use in conjunction with, or for facilitating Morse communication.

6.1 Morse trainer.

Author

Alan Cox, GW4PTS, iialan@iifeak.swan.ac.uk

Description

A Morse Code trainer that uses the PC internal speaker.

Status

stable, works quite well, unfinished.

System requirements

Linux, any version

Detail

Alan wrote this small program in only an hour. It is quite neat, and allows you to play morse at a range of speeds and frequencies through the PC internal speaker. You can specify the text to be played either from the command line, from a file, or the program is capable of generating random character groups. As it stands you must invoke the program from a Linux Virtual Console, as it relies on certain kernel calls to produce the sound, and these don't work as easily from an XTerm. As it sounds each character it lists the character in verbal form (Di, Dit, Dah etc.) to the screen. Alan is hoping that someone will take the code and enhance it with the features he has listed in the comments at the head of the source file.

Where and How to obtain it.

I've had a large number of responses from people seeking this code, so I've obtained Alan's permission to make it available. You can obtain it from *sunsite.unc.edu* (`ftp://sunsite.unc.edu/pub/Linux/apps/ham/GW4PTS.morse.tar.gz`).

Copyright/Licensing

GNU Public License 2, freely redistributable, no warranty.

Contributed by:

Terry Dawson, VK2KTJ, terryd@extro.ucc.su.oz.au

7 AMTOR Software.

Software for use in conjunction with, or for facilitating AMTOR.

8 PACTOR Software.

Software for use in conjunction with, or for facilitating PACTOR.

9 Slow Scan Television Software.

Software for use in conjunction with, or for facilitating Slow Scan Television.

10 Facsimile Software.

Software for use in conjunction with, or for facilitating Facsimile.

11 Design and Construction Software.

Software to assist in the design and construction of amateur radio related things. Antenna, Circuit Board, Filter, and QSL card design packages are all good candidates for this section.

11.1 Software Oscilloscope

Author

Jeff Tranter, Jeff_Tranter@Mitel.COM

Description

Scope is a simple software emulation of an oscilloscope. It graphically displays voltage as a function of time.

Status

ALPHA. First release.

System requirements

Sound card with input capability supported by the kernel sound driver. SVGALIB is used to do the display work.

Detail

Scope uses the `/dev/dsp` device to take audio in from the soundcard and displays it on the screen in a manner similar to an oscilloscope. Jeff claims Scope was written more for amusement value than for any serious purpose.

Where and How to obtain it.

You can obtain source, makefile and man page for Scope from: *sunsite.unc.edu* (`ftp://sunsite.unc.edu/pub/Linux/apps/circuits/scope-0.1.tar.gz`)

Licensing/Copyright

GNU Public License, Freely redistributable. No warranty

Contributed by:

Terry Dawson, VK2KTJ

11.2 irsim

Author

Dmitry Teytelman, dim@leland.stanford.edu

Description

An event-driven logic-level simulator for MOS circuits

Status

Version 8.6, production.

System Requirements

X-Windows.

Detail

irsim is an X11 based simulator for MOS circuits. It has two simulation modes, either *switch* where each transistor is modelled as a voltage controlled switch, or *linear* where each transistor is modelled as a resistor in series with a voltage controlled switch, and each node has a capacitance.

Where and How to obtain it.

You can obtain irsim from *sunsite.unc.edu* (`ftp://sunsite.unc.edu/pub/Linux/apps/circuits/irsim.tar.z`).

Licensing/Copyright

Freely Redistributable

Contributed by:

Terry Dawson, VK2KTJ

11.3 Spice

Author

University of California, Berkeley, ported by eoahmad@ntuix.ntu.ac.sg

Description

Spice is an analog circuit emulator.

Status

Stable release. Last fortran version produced.

System requirements

Unknown.

Detail

Spice allows you to design and test circuits in a computer modelled environment to see how they will behave without having to touch a soldering iron, or solder.

Where and How to obtain it.

You can obtain version 2g6 of Spice from: *sunsite.unc.edu* (`ftp://sunsite.unc.edu/pub/Linux/apps/circuits/spice2g6.tar.z`)

Licensing/Copyright

Copyright held by University California, Berkeley. Freely redistributable.

Contributed by:

Terry Dawson, VK2KTJ

12 Training/Educational Software.

Software to assist in education or training for amateur radio. Morse Code tutorials, technical examination database, Computer Based Training software, and the like are listed here.

13 Miscellaneous Software.

Software that I couldn't put anywhere else.

13.1 SunClock

Author

John Mackin, *john@cs.su.oz.AU*

Description

A clock that will show you instantly what parts of the globe are exposed by sunlight and what parts aren't.

Status

Released.

System requirements.

X-Windows.

Detail

sunclock is another of those desktop gadgets that most people think look nice but really don't have a lot of use for. I use sunclock to obtain an at-a-glance indication of the time anywhere in the world. In its iconic form it sits in a small Mercator projection. When maximised it produces the same image but obviously larger with slightly more detail. It also displays the date, local time and UTC. sunclock actually calculates mathematically what parts of globe are sunlit and which aren't. it seems quite accurate, so long as you assume the earth has no atmosphere.

Where and How to obtain it.

I haven't seen any precompiled sunclock binaries for Linux about, so check your nearest *archie* server. sunclock compiled straight out of the box for me.

Licensing/Copyright

Public Domain and may be freely copied as long as the notices at the top of sunclock.c remain intact

Contributed by:

Terry Dawson. VK2KTJ

13.2 Xearth

Author

Kirk Lauritz Johnson, tuna@cag.lcs.mit.edu, modified by Dimitris Evmorfopoulos, devmorfo@cs.ntua.edu

Description

A rotating earth for X-windows root window. *It has real life shading, and options for geostatic view,* and non geostatic view. A prettier and more modern version of *sunclock* but requires a bit more processing power.

Status

released, stable.

System requirements.

X-Windows.

Detail

xearth is much like *sunclock* except that it draws a view of the earth onto your root map in blue and green, as the earth would be viewed from space. You have a number of options in determining the behaviour of the view. You can either have it so that the same part of the earth is displayed, and the sun rotates, so you'll have varying light and shadow on the display, or you can have it shown as if you were travelling with the sun, so the whole of the globe is visible, and the earth rotates. You can specify the latitude/longtitude that will be the centre of the display. *xearth* is also capable of producing *gif* and *ppm* graphics output, so you can generate custom graphics of the globe. When run as your root map, you can adjust the interval of time between updates. On my 486sx25 you notice a small degradation in performance when it is recalculating, but it's not annoying.

Where and How to obtain it.

A precompiled *xearth* binary is available from: **sunsite.unc.edu**, or the source can be obtained from just about any X11/contrib directory. Try **export.lcs.mit.edu** if you can't find it elsewhere.

Licensing/Copyright

Copyright (C) 1989, 1990, 1993 by Kirk Lauritz Johnson. The copyright notice included states that *xearth* is freely redistributable so long as the copyright notice is left intact, and be included in documentation.

Contributed by:

Alan Cox, GW4PTS.

14 How to contribute or update an entry.

I'd like for this list to be as complete and up-to-date as possible. So I'm keen to hear about any developments or products that I don't already know about, or that the entry is obselete or outdated for.

What I'd like as a minimum set of requirement would be something like the following:

Name

The name of the software in question.

Author

Who wrote, or ported the software. An email address, or some other means of contacting them is also *essential.*

Description

A single line description of what the software does.

Status

An indication of the software's status. Is it still in testing? Is it a production release? Is it still in the design stage?

System requirements

What does the software require to run? Does *it require* X-Windows? Does it need a soundcard? Does it need a certain version of kernel? Does it need other software to support it?

Detail

I'm not keen on including a large amount of detail on each *piece of software as this would consume* a lot of time reading and trying to keep up to date. So instead what I'd like to list is what makes this software unique, anything special about it. Perhaps its most outstanding features, that sort of thing.

Where and How to obtain it.

If the software is freely distributable then ftp details would be great. If it is commercial software then the name of the company distributing the software, and an address or telephone number. If it is available only by some other means, say mail order, then details on where and how to obtain it.

Licensing/Copyright

Is the software Copyleft? Copyright? Shareware? Public Domain? Restricted in use in any way?

Don't worry if you don't know all of these details, just send me what you do know and I'll list what I can. I'd rather have an incomplete listing *than no listing* at all.

Please mail any contributions to:

terryd@extro.ucc.su.oz.au or **terry@orac.dn.itg.telecom.com.au**

I'd list a packet radio address too but I'm still not properly operational again yet after moving house.

15 Discussion relating to Amateur Radio and Linux.

There are various places that discussion relating to Amateur Radio and Linux take place. *They take place in* the **comp.os.linux.*** newsgroups, they also take place on the **HAMS** list on **niksula.hut.fi**. Other places where they are held include the **tcp-group** mailing list at **ucsd.edu** (the home of amateur radio tcp/ip discussions), and I believe there is an 'IRC' channel that sometimes is used to discuss them as well.

To join the Linux **HAMS** channel on the mail list server, send mail to.


```
linux-activists@niksula.hut.fi
```

with the line:

```
X-Mn-Admin: join HAMS
```

at the top of the message body (not the subject line).

To join the **tcp-group** send mail to:

```
listserver@ucsd.edu
```

with the line:

```
subscribe tcp-group
```

in the body of the text.

Note: Please remember that the **tcp-group** is primarily for discussion of the use of advanced protocols, of which tcp/ip is one, in Amateur Radio. *Linux specific questions should not ordinarily go there.*