Lead Feature

PSK31: A New Radio-Teletype Mode

*Part One, by Peter Martinez, G3PLX**

I HAVE BEEN active on RTTY since the 1960s, and was instrumental in introducing AmTOR to amateur radio at the end of the '70s. This improved the reliability of the HF radio link and paved the way to further developments which have taken this side of the hobby more into data transfer, message handling, and computer linking, but further away from the rest of amateur radio which is based on two-way contacts between operators.

There is now a gap opening up between the data transfer enthusiasts using the latest techniques and the two-way contact fans who are still using the traditional RTTY mode of the '60s, although of course using keyboard and screen rather than teleprinter. There is scope for applying the new techniques now available to bring RTTY into the 21st century.

This article discusses the specific needs of 'live QSO' operating, as opposed to just transferring chunks of error-free data, and describes the PSK31 mode which I have developed specifically for live contacts, which is now becoming popular using low-cost DSP kits, and which could become even cheaper as the art of using PC sound cards is developed by amateur radio enthusiasts.

WHAT IS NEEDED?

I BELIEVE THAT it is the error-correcting process used in modern data modes which make them unsuitable for live contacts. I have identified several factors; the first revolves around the fact that all error-correcting systems introduce a time-delay into the link. In the case of an ARQ link like AmTOR or PacTOR, there is a fixed transmission cycle of 450mS or 1.25sec or more, which will delay any keypress by as much as one cycle-period, and by more if there are errors. With forward-error-correction systems there is also an inevitable delay, because the information is spread out over a period of time. In a live two-way contact, the delay is doubled at the point where the transmission is handed over. I believe that these delays make such systems unpleasant to use in a two way conversation.

This is not so much a technical problem as a human one. Another factor in this category is concerned with the way that the quality of the information content varies as the quality of the radio link varies. In an analogue transmission system such as SSB or CW, there is a linear relationship between the two. The operators are aware of this all the time and take account of it subconsciously: they change the speed and tone of voice instinctively, and even choose the topic of conversation to suit the conditions. In a digital mode the relationship between the signal-to-noise ratio on the air and the error-rate on the screen is not so smooth. The modern error-correcting digital modes are particularly bad at this, with copy being almost perfect while the SNR is above a certain level and stopping completely when the SNR drops below this level. The effect is of no consequence in an automatic mailbox forwarding link, but can badly inhibit the flow of a conversation.

A third factor is a social one; with error-correcting modes you only get good copy when you are linked to one other station. The copy is decidedly worse when not linked, such as when calling CQ or listening to others. This makes it difficult 'getting to know' other people on the air, and there is a tendency to limit contacts to a few close friends or just mailboxes.

These factors lead me to suggest that there is a case for a transmission system that is *not* based on the use of error-correcting codes, when the specific application is that of live contacts. The continued popularity of traditional RTTY, using the start-stop system, is proof of this hypothesis: there is minimal delay (150mS), the flow of conversation is continuous, the error-rate is tolerable, and it is easy to listen-in and join-in.

IMPROVING ON RTTY

HOW, THEN, DO WE go about using modern techniques that were not available in the '60s, to improve on traditional RTTY? First of all, since we are talking about live contacts, there is no need to discuss any system that transmits text any faster than can be typed by hand. Secondly, modern transceivers are far more stable in frequency than they were in the '60s, so we should be able to use much narrower bandwidths than in those days. Thirdly digital processors are much more powerful than the rotating cams and levers of the mechanical teleprinter, so we could use better coding. The drift-tolerant technique of frequency-shift keying, and the fixed-length five-unit start-stop code still used today for RTTY are a legacy of the limitations of technology 30 years ago. We can do better now.

PSK31 ALPHABET

THE METHOD I have devised for using modern digital processing to improve on the start-stop code, without introducing extra delays

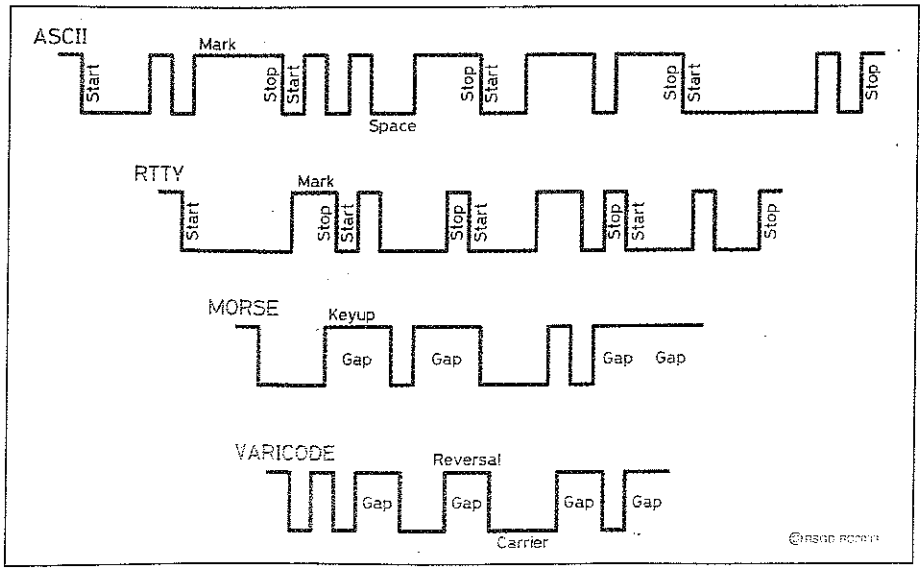

Fig 1: Showing the word 'ten' keyed in ASCII, RTTY, Morse, and Varicode.

*High Blakebank Farm, Underbarrow, Kendal, Cumbria LA8 8HP.

due to the error-correcting or synchronisation processes, is based firmly on another tradition, namely that of Morse code. Because Morse uses short codes for the more common letters, it is actually very efficient in terms of the average duration of a character. In addition, if we think of it in terms that we normally use for digital modes, Morse code is self-synchronising: we don't need to use a separate process to tell us where one character ends and the next begins. This means that Morse code doesn't suffer from the 'error-cascade' problem that results in the start-stop method getting badly out of step if a start or stop-bit is corrupted. This is because the pattern used to code a gap between two characters *never* occurs inside a character.

The code I have devised is therefore a logical extension of Morse code, using not just one-bit or three-bit code-elements (dots and dashes), but any length. The letter-gap can also be shortened to two bits. If we represent key-up by 0 and key-down by 1, then the shortest code is a single one by itself. The next is 11, then 101 and 111, then 1011, 1101, 1111, but not 1001 since we must not have two or more consecutive zeros inside a code. A few minutes with pencil and paper will generate more. We can do the 128-character ASCII set with 10 bits.

I analysed lots of English language text to find out how common were each of the ASCII characters, then allocated the shorter codes to the more common characters. The result is shown in **Table 1** on page 16, and I call it the 'Varicode' alphabet. With English text, Varicode has an average code-length, including the '00' letter-gap, of 6.5 bits per character. By simulating random bit errors and counting the number of corrupted characters, I find that Varicode is 50% better than start-stop code, thus verifying that its self-synchronising properties are working well.

The shortest code in Morse is the commonest letter 'e', but in Varicode the shortest code is allocated to the wordspace. When idle, the transmitter sends a continuous string of zeros. **Fig 1** compares the coding of the same word in ASCII, RTTY, Morse, and Varicode.

PSK31 MODULATION AND DEMODULATION

TO TRANSMIT VARICODE at a reasonable typing speed of about 50 words per minute needs a bit-rate of about 32 per sec. I have chosen 31.25, because it can be easily derived from the 8kHz sample-rate used in many DSP systems. In theory we only need a bandwidth of 31.25Hz to send this as binary data, and the frequency stability that this implies can be achieved with modern radio equipment on HF.

The method chosen was first used on the amateur bands, to my knowledge, by SP9VRC. Instead of frequency-shifting the carrier, which is wasteful of spectrum, or turning the carrier on and off, which is wasteful of transmitter power capability, the 'dots' of the code are signalled by reversing the polarity of the carrier. You can think of this as equivalent to transposing the wires to your antenna feeder. This uses the transmitted signal more efficiently since we are comparing a positive signal before the reversal to a negative signal after it, rather than comparing the signal present in the dot to no-signal in the gap. But if we keyed the transmitter in this way at 31.25 baud, it would generate terrible key clicks, so we need to filter it.

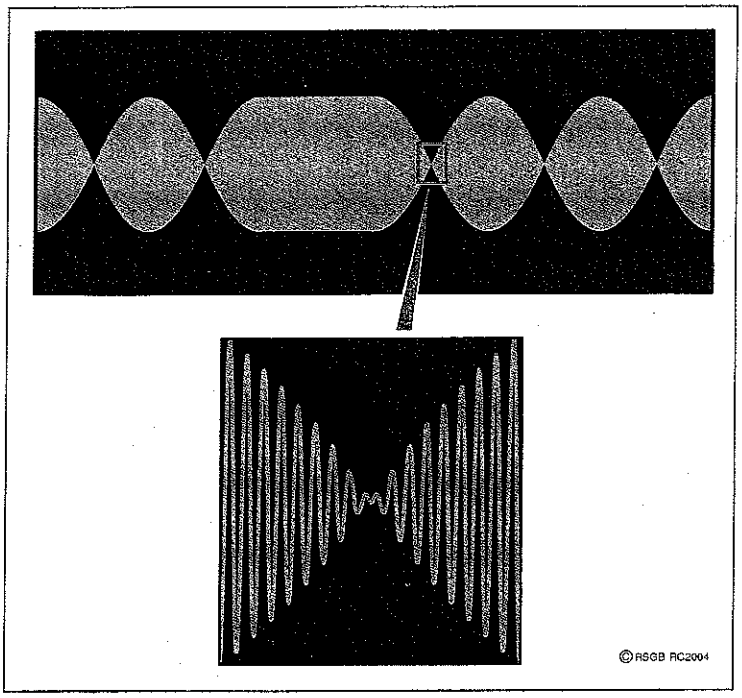

Fig 2: Showing the waveform of BPSK sending the Varicode 'space' symbol.

If we take a string of dots in Morse code, and low-pass filter it to the theoretical minimum bandwidth, it will look the same as a carrier that is 100% amplitude-modulated by a sinewave at the dot-rate. The spectrum is a central carrier and two sidebands at 6dB down on either side. A signal that is sending continuous reversals, filtered to the minimum bandwidth, is equivalent to a double sideband suppressed carrier emission, that is, to two tones either side of a suppressed carrier. The improvement in the performance of this polarity-reversal keying over on-off keying is thus equivalent to the textbook improvement in changing from amplitude modulation telephony with full carrier to double-sideband with suppressed carrier. I have called this technique 'polarity-reversal keying' so far, but everybody else calls it 'binary phase-shift keying', or BPSK. **Fig 2** shows the envelope of BPSK modulation and the detail of the polarity reversal.

To generate BPSK in its simplest form we could convert our datastream to levels of ±1V for example, take it through a low-pass filter, and feed it into a balanced modulator into which we also feed the desired carrier frequency. When sending continuous reversals, this looks like a 1V peak-to-peak sinewave going into a DSB modulator, so the output is a pure two-tone. In practice, we use a standard SSB transceiver and perform the modulation at audio frequency, or carry out the equivalent process in a DSP chip. We could signal a logic zero by continuous carrier and signal a logic one by a reversal, but I do it the other way round for reasons which will become clear shortly.

There are a variety of ways to demodulate BPSK, but they all start with a bandpass filter. For the speed chosen for PSK31, this filter can be as narrow as 31.25Hz in theory, but a 'brick-wall' filter of precisely this width would be costly, not only in monetary terms but in the delay time through the filter, and we are trying to avoid delays. A practical filter might be twice the baud-rate (62.5Hz) wide at the 50dB-down point, and have a delay-time of two bits (64mS).

For the demodulation itself, since BPSK is equivalent to double sideband, the textbook method for demodulating DSB can be used, but another way is to delay the signal by one bit-period and compare it with the direct signal in a phase comparator. The output is negative when the signal reverses polarity,

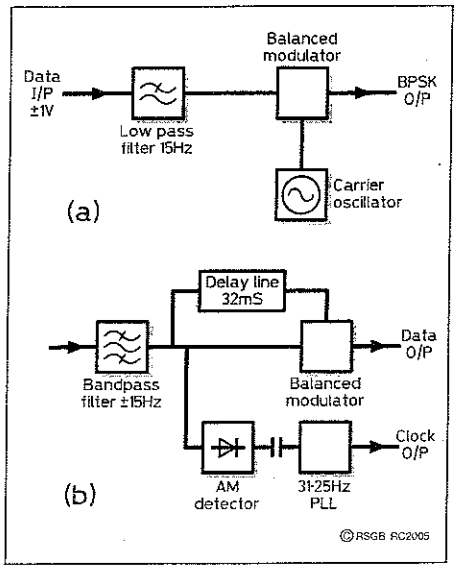

Fig 3: Block diagram of analogue BPSK modulator (a) and demodulator (b).

ASCII	Varicode	ASCII	Varicode	ASCII	Varicode
0	1010101011	+	111011111	V	110110101
1	1011011011	,	1110101	W	101011101
2	1011101101	-	110101	X	101110101
3	1101110111	.	1010111	Y	101111011
4	1011101011	/	110101111	Z	1010101101
5	1101011111	0	10110111	[	111110111
6	1011101111	1	10111101	\	111101111
7	1011111101	2	11101101	]	111111011
8	1011111111	3	11111111	^	1010111111
9	11101111	4	101110111	_	101101101
l/feed	11101	5	101011011	`	1011011111
11	1101101111	6	101101011	a	1011
12	1011011101	7	110101101	b	1011111
c/ret	11111	8	110101011	c	101111
14	1101110101	9	110110111	d	101101
15	1110101011	:	11110101	e	11
16	1011110111	;	110111101	f	111101
17	1011110101	<	111101101	g	1011011
18	1110101101	=	1010101	h	101011
19	1110101111	>	111010111	i	1101
20	1101011011	?	1010101111	j	111101011
21	1101101011	@	1010111101	k	10111111
22	1101101101	A	1111101	l	11011
23	1101010111	B	11101011	m	111011
24	1101111011	C	10101101	n	1111
25	1101111101	D	10110101	o	111
26	1110110111	E	1110111	p	111111
27	1101010101	F	11011011	q	110111111
28	1101011101	G	11111101	r	10101
29	1110111011	H	101010101	s	10111
30	1011111011	I	1111111	t	101
31	1101111111	J	111111101	u	110111
space	1	K	101111101	v	1111011
!	111111111	L	11010111	w	1101011
"	101011111	M	10111011	x	11011111
#	111110101	N	11011101	y	1011101
$	111011011	O	10101011	z	111010101
%	1011010101	P	11010101	{	1010110111
&	1010111011	Q	111011101	|	110111011
'	101111111	R	10101111	}	1010110101
(	11111011	S	1101111	~	1011010111
)	11110111	T	1101101	127	1110110101
*	101101111	U	101010111		

Table 1: The Varicode alphabet. The codes are transmitted left bit first, with '0' representing a phase reversal on BPSK and '1' representing a steady carrier. A minimum of two zeros is inserted between characters. Some implementations may not handle all the codes below 32.

and positive when it doesn't.

Although we could extract the information from the demodulated signal by measuring the lengths of the 'dots' and 'dashes', like we do by ear with Morse code, it will help to pick the data out of the noise if we know when to expect them. We can easily transmit the data at an accurately timed rate, so it should be possible to predict when to sample the demodulator output. This process is known as synchronous reception, although the term 'coherent' is sometimes wrongly used. To synchronise the receiver to the transmitter, we can use the fact that a BPSK signal has an amplitude-modulation component. Although the modulation varies with the data pattern, there is always a pure tone component in it at the baud-rate, and this can be extracted using a narrow filter or a phase-lock loop, or the DSP equivalent, and fed to the decoder to sample the demodulated data. **Fig 3** shows block diagrams of a typical BPSK modulator and demodulator.

For the synchronisation to work we need to make sure there are no long gaps in the pattern of reversals. A completely steady carrier has no modulation, so we could never predict when the next reversal was due. Fortunately, Varicode is just what we need, provided we choose the logic levels so that zero corresponds to a reversal and one to a steady carrier. The idle signal of continuous zeros thus generates continuous reversals, giving us a strong 31.25Hz modulation. Even with continuous keying there will always be two reversals in the gaps between characters. The average number of reversals will therefore be more than two in every 6.5 bits, and there will never be more than 10 bits with no reversal at all. If we make sure that the transmission always starts with an idle period, then the timing will pull into sync pretty quickly. By making the transmitter end a transmission with a 'tail' of unmodulated carrier, it is then possible to use the presence or absence of reversals to 'squelch' the decoder so that the screen doesn't fill with noise when there is no signal.

GETTING GOING

SO MUCH FOR the philosophy and the theory, but how do you get on the air with this mode? At the moment, the route to getting on PSK31 is to obtain one of several DSP starter kits. These are printed circuit cards, usually with a serial interface to a PC, marketed by DSP processor manufacturers at low cost to help engineers and students become familiar with DSP programming. A number of radio amateurs have started to write software for these, not just for RTTY but also for SSTV, packet, satellite, and digital voice experiments. They have audio input and output and some general purpose digital input/output. The construction work needed is limited to wiring up cables, building a power supply, and putting the card into a screened box. The DSP software is freely available, as is the software that runs in the PC to interface to the keyboard and screen, and can be obtained most easily via the Internet. It would certainly be possible to construct a PSK31 modem in hardware, although I know of no-one having done this yet, but probably the most promising hardware platform for the future will be the PC sound card.

PSK31 OPERATING

SINCE PSK31 performance is the same when calling, listening, or in contact, it's easy to progress from listening to others, to calling CQ, to two-way contacts and multi-way nets, but the narrow bandwidth and good weak-signal performance do mean learning a few new tricks. It's usual to leave the radio dial on one spot and fine-tune the audio frequency, listening through the narrow audio filter rather than the loudspeaker, and using an on-screen phase-shift display to centre the incoming signal within a few Hz. On transmit, since the envelope of the PSK31 signal is not constant (as is the case for FSK), it is important to keep the transmitter linear throughout. However, since the PSK31 idle is identical to a standard two-tone test signal, it is easy to set up. The worst distortion products will be at ±45Hz at a typical level of 36dB below PEP.

In part two of this article I will describe how I was persuaded to go back and take a second look at error-correction and describe how PSK31 has been extended to improve its handling of languages other than English. ♦

... to be continued

BY IAN WHITE, G3SEK*

SOFT STARTING

WHAT IS A 'soft start' circuit, and why should my power supply need one?

A SOFT START circuit is a way of turning on the mains supply gradually, to avoid stress on components due to a sudden voltage or current surge. In a typical transformer rectifier capacitor power supply (**Fig 1a**) at the moment of switch-on, the primary of the mains transformer looks like a very low resistance because the core is not magnetised. The larger the transformer, the lower the primary resistance, of course, and therefore the larger the switch-on surge. Mechanical movement inside the transformer is what's responsible for the loud 'thump' when you switch on, and obviously such stresses aren't doing the transformer any good - and likewise the mains switch. Beyond the transformer, the uncharged reservoir capacitor also looks like a short-circuit, and the initial current surge stresses all the components upstream of it. In a power amplifier using valves, there are also the heaters or filaments to be considered. These have a much lower resistance when cold than when up to temperature, so once again an uncontrolled switch-on causes a severe thermal and mechanical shock which will definitely lead to reduced life expectancy. The other problem is that a mains fuse or circuit breaker selected to give fast, sensitive protection in normal operation is unlikely to allow the equipment ever to start!

Because most components are rated to survive moderate switch-on surges, small power supplies can get away with a simple mains switch, and possibly a slow-blow fuse. The surge is bigger and more serious in larger power supplies. Although the peak current through any individual component is always limited by the resistance and inductance of all the other components upstream, starting with the mains wiring itself, it's better to avoid these stresses by deliberately limiting the current surge for the first few moments. That's what a soft-start circuit does.

A typical soft-start circuit places a resistor in series with the mains supply for a brief period during switch-on, a few seconds at most. Some kind of timing device then operates a relay which short-circuits the resistor until the equipment is switched off. Strictly speaking this is a 'step-start' circuit, but 'soft-start' is the more common name. **Fig 1b** shows an outline circuit. Values for the limiting resistor R1 range from about 500Ω down to about 20Ω, so you may need to experiment to see what suits your equipment. The resistance should not be so low that it provides very little surge-limiting effect at switch-on, but it definitely shouldn't be so high that the main current surge comes *after* the resistor is shorted. The power rating usually needs to be only about 50W, because the resistor is only dissipating power very briefly.

*52 Abingdon Road, Drayton, Abingdon, Oxon OX14 4HP
g3sek@ifwtech.demon.co.uk

If you're adapting an American design, you needn't follow their common practice of using a series resistor on each side to the transformer primary.

There are many different kinds of timing circuit, but they fall into two main groups. One measures a time period and then closes the shorting relay. The other kind senses the build-up of voltage at the 'downstream' side of the components that are being protected. **Fig 2a** shows the commonest and simplest of the timer-type circuits. D1-R2-C1 form a simple rectified mains supply for the 24V DC relay RL1, but the time constant of R2 and C1 delays the build-up of voltage across the coil of RL1. When RL1 finally closes, RL1a shorts the series resistor R1. The values given are adequate for a wide range of transmitter power supplies (regardless of output voltage). A disadvantage of this circuit is the continuous power dissipation in R2, typically 25W. **Fig 2b** shows a more sophisticated approach. This is representative of a large group of soft-start circuits that use one of the classic transistor/IC delay timers with a small independent power supply that is energised at switch-on. In this case the 555 timer starts when power is applied, and the output (pin 3) goes high after about 1 second, energising RL1 and short-circuiting R1 until all power is switched off.

Fig 2c shows a simple soft-start circuit that indirectly senses the build-up of output voltage in a transformer-capacitor-rectifier supply [1]. RL1 is a 24V DC relay that is supplied from two primary taps on the mains transformer that are nominally 20-25V apart. Bridge rectifier BR1 and series resistor R2 feed RL1 from the available AC voltage, which will build up slowly as the transformer is magnetised and the main reservoir capacitor is charged. In effect, the whole circuit acts as its own timing element. R2 is adjusted so that RL1 will pull in after a suitable time. Some experimentation will be needed, because the correct value for R2 will depend on almost all the other components acting together; about 200Ω (2W) would be a good first try.

Although some of these soft-start circuits appear delightfully simple, you do have to remember that if anything goes wrong, RL1 will never pull in, and then R1 may become very hot indeed. One way to deal with this is to choose a fairly short time delay, and protect R1 with its own slow-blow fuse FS2 that never normally has time to operate. Fig 2c shows the location for FS2, which could (and probably should) be used in all of the other circuits too. An alternative to a wire fuse would be a so-called 'resettable fuse', which is actually a Positive Temperature Coefficient (PTC) resistor. These components have a large, positive resistance change if they reach a certain temperature, and will then limit the current permanently to a safe value; they are re-set by removing the supply voltage and allowing them to cool down, and thus can be cycled indefinitely. PTC resettable fuses are in the Fuses section of the major component catalogues.

That leads us to another type of soft-start circuit, based on a Negative Temperature Coefficient (NTC) resistor all on its own (**Fig 2d**). This type of component has a high resistance when cold, falling to a low resistance when it has warmed up, and that's all you need to ensure a gradual soft start. Disadvantages are that component selection is needed because performance is load-dependant; the component won't work unless it is deliberately allowed to get hot in normal operation and is protected from draughts; and the residual resistance will affect the output voltage stability. The RS catalogue logically calls these components 'Inrush Current Suppressors' and you'll find them in the *Suppressors* section, but in the Farnell catalogue you need to look under *Resistors* for 'Thermistors, NTC'.

Last of all, let me repeat the advice to include a suitably rated mains filter at the input to all power supplies. A VDR should be connected between line and neutral to suppress mains spikes, but to minimise the risk of long-term deterioration it is best to connect this component downstream of the filter and mains switch.

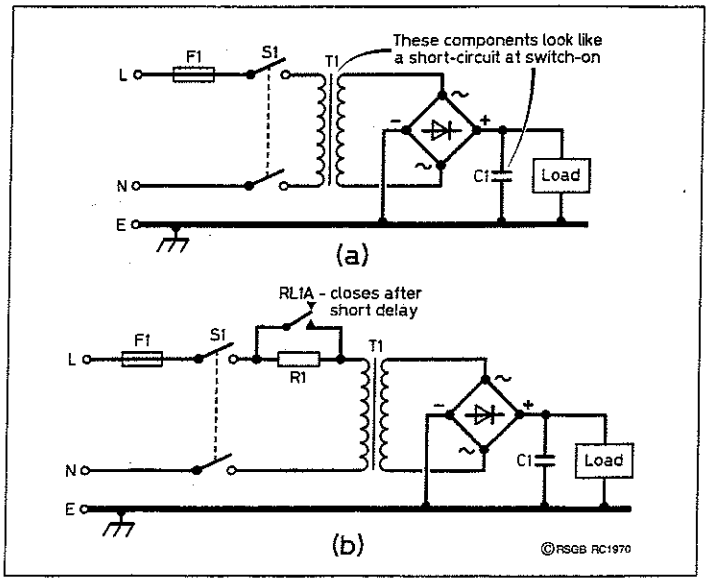

Fig 1: (a) At the moment of switch-on, transformer T1 and reservoir capacitor C1 appear as short-circuits, stressing all the components in the power supply. (b) Soft-start resistor R1 limits current surge and is then short-circuited.

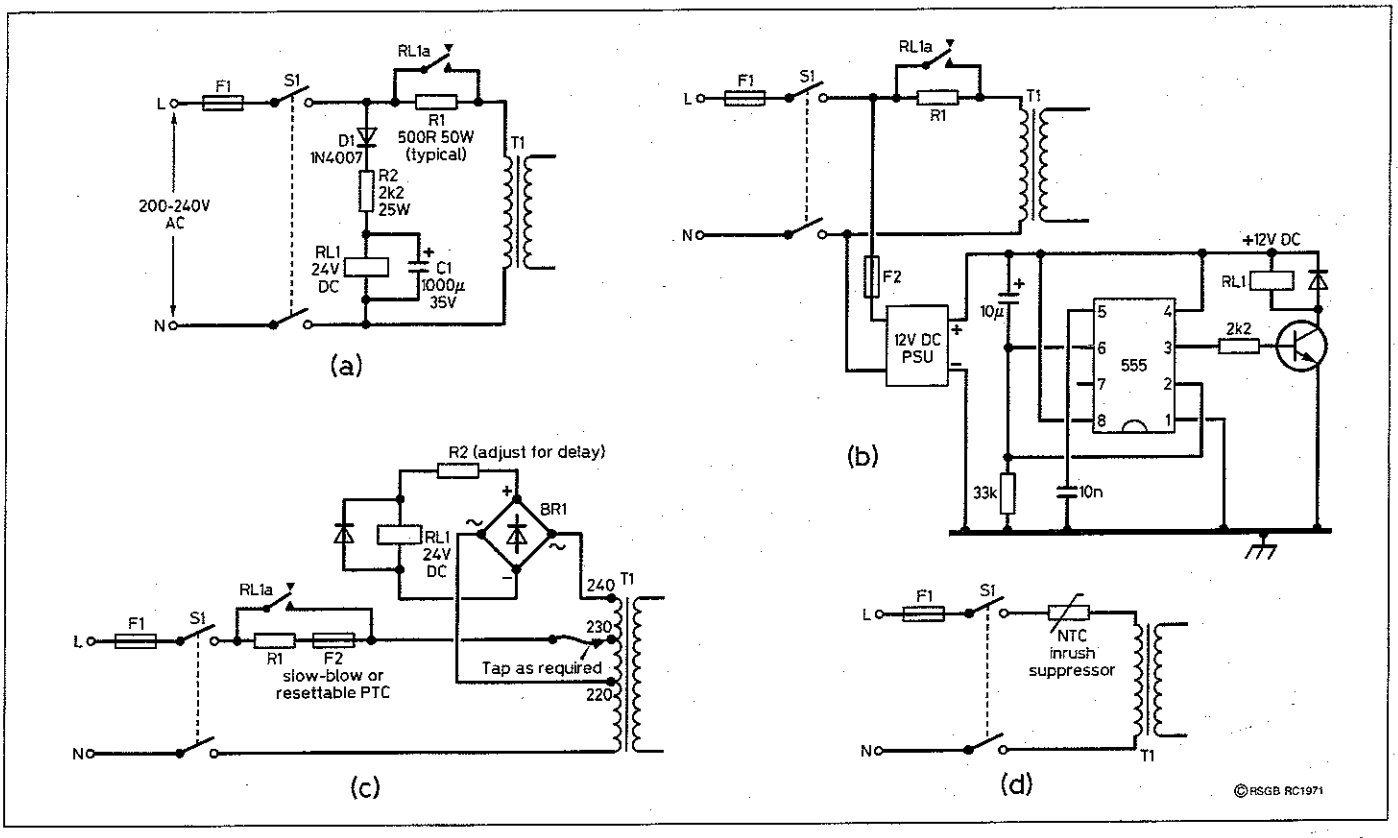

Fig 2: (a) A very common primary-side timer circuit. (b) One of many possible timer circuits using an auxiliary low-voltage supply. (c) Simple voltage-sensing circuit, but note the PTC protection for R1 which could also be used in the other circuits. (d) NTC resistor provides a very simple soft start, but with some disadvantages.

MODIFYING PC BOARDS

HOW DO I drill a PC board that already has components on, without breaking the drill bit?

IN ONE WORD, carefully. There's often a need for a few extra holes in an experimental PC board, or one that you're modifying, but the existing components on the board make the job much more difficult. For a start, hold the board firmly - not in a vice, but flat on the bench using a sheet of expanded polystyrene underneath. Press the underside of the board down into the sheet until it sits firm and level. To prevent the drill from skidding - and inevitably breaking - mark where the hole is to be with a sharp scriber. Dig a deep enough dimple to locate the point of the drill bit.

You need a drill that you can control very finely and does not put sideways strain on the bit - because it will break. A small high-speed electric drill in a bench stand is ideal, but even here you have to locate the board very accurately so that the bit goes straight into your starter mark - or else it will break. If you have to drill freehand, without the bench stand, the same electric drill is probably still the best option, but has to be used very carefully. Brace both wrists against the bench and move only your hands. Don't let your body weight come to bear on the bit, or else. . . well, you know! Also, avoid using any kind of hand drill with a side crank handle; that's a sure way to put sideways forces on the bit. When I can't be bothered to get out the drill and stand, I've had some success with a miniature Archimedean screw drill, where all the forces are along the line of the bit. . . but I've also broken lots of bits.

WIND LOADING

HOW DO I calculate the wind loading on my antenna and mast?

IN JANUARY 1995 this column showed an easy way to calculate wind loading, which simplified the calculation by treating all the antennas, rotator and mast as flat slabs. That approach maximises your estimates of the wind forces because it takes no account of streamlining effects, but it also tends to ensure that the whole installation is over-engineered. In a small installation, over-engineering doesn't cost much; but with larger antenna systems the costs of heavier-duty towers, rotators and masts escalate rapidly, and you literally cannot afford to over-engineer. Instead, you have to get the calculations right.

Computers to the rescue! A program called *YagiStress* by NI6W allows yagis and support booms to be optimised for mechanical strength and minimum weight. Tell *YagiStress* what kind of yagi you propose to build, including all the lengths and cross-sections of materials involved, and the program makes accurate estimates of the total weight and the static balance about the mast mounting point. Then it analyses the wind forces and aerodynamic balance of your proposed structure (using methods that have been checked against even more accurate computer codes), and calculates the loading on each component over a range of wind speeds. It even asks, "Would you like ice with that?" One of the main virtues of *YagiStress* is the ease with which you can change your ideas and re-analyse the effects, and there is also an interface with K6STI's antenna performance analysis programs. There is more information at http://freeyellow.com/members3/yagistress where you can also e-mail for a free demo version.

REFERENCE

[1] Variants of this circuit seem to have been re-invented several times. It has been featured in *Technical Topics* and also in *QST*.

THANKS AGAIN

THANKS AGAIN to all the readers and correspondents who make this column so interesting to write. Much of *In Practice* is based on other people's knowledge and experience, so thank you all for allowing me to pass it on. Happy holidays and best wishes for 1999. ♦

If you have new questions, or any comments to add to this month's column, I'd be very pleased to hear from you by mail or E-mail. But please remember that I can only answer questions through this column, so they need to be on topics of general interest.

Technical Feature

PSK31: A New Radio-Teletype Mode

Concluding part, by Peter Martinez, G3PLX*

PART ONE OF of this article discussed the requirements for a live-contact keyboard/screen communication system, and proposed the narrow-band PSK31 mode as a candidate for a modern equivalent to traditional RTTY. This mode has now been in use on the HF bands by a small but growing band of enthusiasts for about two years.

In this part I will describe two recent additions to PSK31.

A SECOND LOOK AT ERROR CORRECTION

AFTER GETTING PSK31 going with BPSK modulation and the Varicode alphabet, several people urged me to add error correction to it in the belief that it would improve it still further. I resisted for the reasons that I gave in part 1, namely that the delays in transmission, the discontinuous traffic flow, and the inability to listen-in, all make error correction unattractive for live contacts. There is another reason. All error correcting systems works by adding redundant data bits. Suppose I devise an error correcting system that doubles the number of transmitted bits. If I wanted to keep the traffic throughput the same, I would need to double the bit rate. But with BPSK that means doubling the bandwidth, so I lose 3dB of signal-to-noise ratio and get more errors. The error correction system will have to work twice as hard just to break even! It is no longer obvious that error correction wins. It is interesting to note that with FSK, where the bandwidth is already much wider than the information content, you *can* double the bit-rate without doubling the bandwidth, and error correction *does* work. Computer simulation with BPSK in white noise shows that when the SNR is good, the error correction system does win, reducing the low error rate to very low levels, but at the SNR levels that are acceptable in live amateur contacts, it's better to transmit the raw data slowly in the narrowest bandwidth. It also takes up less band space of course!

However, there was the suggestion that error correction could give useful results for bursts of noise which cannot be simulated on the bench, so I decided to try it and do some comparison tests. The automatic repeat (ARQ) method of correcting errors was ruled out, but forward error correction (FEC) which was added to modes such as AmTOR and PacTOR almost as an afterthought seemed to deserve a second look, provided the transmission delay was not too long.

I realised that comparing two systems with different bandwidths and speeds on the air would be difficult: adjacent channel interference would be different, as would the effects of multipath. There is, however, another way to double the information capacity of a BPSK channel without doubling it's bandwidth and speed. By adding a second BPSK carrier at 90° at the transmitter and a second demodulator in the receiver, we can do the same trick that is used to transmit two colour-difference signals in PAL and NTSC television. I call this quadrature polarity reversal keying, but everybody else calls it quaternary phase shift keying or QPSK.

There is a 3dB signal-to-noise penalty with QPSK, because we have to split the transmitter power equally between the two channels. This is the same penalty as doubling the bandwidth, so we are no worse off. QPSK is therefore ideal for my planned comparison experiment: the adjacent channel interference, the SNR, and the multi-path performance would be exactly the same for both.

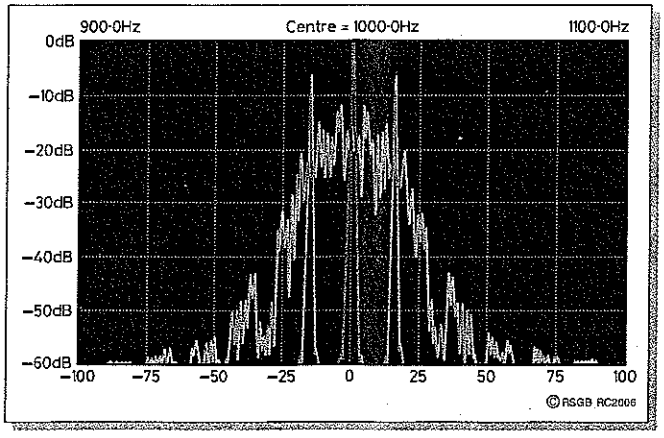

Fig 4: Showing the spectrum of the BPSK signal, idling and sending data, compared with an unmodulated carrier at the same signal level.

In the next section I will think of QPSK not as two channels of binary data, but as a single-channel which can be switched to any of four 90° phase-shift values. By the way, the clock recovery idea used for BPSK works just as well on QPSK, because the envelope still has a modulation component at the bit-rate.

QPSK AND THE CONVOLUTIONAL CODE

THERE IS A vast amount of available knowledge about correcting errors in data which are organised in blocks of constant length such as ASCII codes, by transmitting longer blocks, but I know of nothing that covers error correction of variable length blocks like Varicode. However, there are ways of reducing errors in continuous streams of data which have no block structure, and this seems a natural choice for a radio link, since the errors don't have any block structure either. These are called convolutional codes, and one of the simplest forms does actually double the number of data bits and is therefore a natural choice for a QPSK channel carrying a variable length code.

The convolutional encoder generates one of the four phase shifts, not from each data bit to be sent, but from a sequence of them. This means that each bit is effectively spread out in time, inter-twined with earlier and later bits in a precise way. The more we spread it out, the better will be the ability of the code to correct bursts of noise, but we must not go too far or we will introduce too much transmission delay. I chose a time spread of 5 bits. The table that determines the phase shift for each pattern of 5 successive bits is given in the appendix. The logic behind this table will not be covered here.

In the receiver, a device called a Viterbi decoder is used. This is not so much a decoder as a whole family of encoders playing a guessing game. Each one makes a different 'guess' at what the last 5 transmitted data bits might have been. There are 32 different patterns of 5 bits and thus 32 encoders. At each step the phase shift value predicted by the bit-pattern-guess from each encoder is compared with the actual received phase shift value, and the 32 encoders are given 'marks out of ten' for accuracy. Just like in a knockout competition, the worst 16 are eliminated and the best 16 go on to the next round, taking their previous scores with them. Each surviving encoder then gives birth to two children, one guessing that the next transmitted bit will be a zero and the other guessing that

*High Blakebank Farm, Underbarrow, Kendal, Cumbria LA8 8HP.