PicATUne

- an intelligent antenna tuning unit

Part five, by Peter Rhodes, BSc, G3XJP

Issue 7.0 Nov 8

IN THIS LAST part, the architectural considerations behind PicATUne's design are covered, culminating in specific detail of the operator interface - and finally some ideas on installation.

ARCHITECTURAL DESIGN

THE SYSTEM, hardware and software thinking that went into PicATUne is briefly discussed here since if you understand some of the background thinking, the operator instructions which follow will make more sense in that context.

Also, it is a relatively simple task to design different ATU hardware to use with my software - or your own software with my hardware - or you could be inspired to design your own tuner from scratch for your specific operational needs.

Or, if you are thinking of buying a commercial offering, this may inspire some aspects to look out for.

SYSTEM CONSIDERATIONS

There are a number of gross options in conceiving how to automate an ATU.

QST published a basic design [8] which is representative of the KISS approach. It is billed as 'easy to build' - which certainly looks likely - and 'makes time-consuming knob twisting and roller cranking a page in your station's history book.' It uses low voltage capacitors and only has a coax output socket, which makes me somewhat suspicious. What it certainly does *not* do is remember the matching solutions. It has to re-discover them from scratch each time you change frequency. This seems a pity, because even us humans write down the settings of a manual ATU on a piece of paper. However that does remove a lot of systems complexity - such as the need to measure frequency - and it may represent a good compromise for some. But to pick another tune, "A KISS is just a KISS .....".

From the outset, I decided that committing solutions to memory for later use was money well spent - but I did expend a lot of effort trying to keep the memory size (and cost) down.

The issue is this. If you want to cover the *entire* HF spectrum, then a lot of memory is needed to retain the full detail of unique solutions at sufficiently small frequency intervals. So much so, that there are architectural implications. It may be easier to settle for storing a limited number of solutions - or partial solutions, but then refining them in real time - every time - during the first few seconds of operation on a given frequency. Since this latter has to be done quickly to be effective, you probably need phase detection to tell you instantly which way to go; and certainly full VSWR measurement since you cannot rely on a steady carrier to match on. Maybe the bigger memory works out cheaper?

If, however, you are designing exclusively for the amateur then the total HF allocation only adds up to some 5MHz (if you are generous) which immediately divides memory needs by six. But more significantly, it does allow retention of the full exact matching solution at frequent intervals - with no topping up required - at an acceptable price.

Having spent weeks trying various data compression techniques - and encoding 11 bits of capacitor settings into an 8-bit byte (by sacrificing resolution at higher capacitances), you can imagine my language when the latest catalogue from a well known UK distributor arrived showing the price of 32k EEPROM to be less than that for 16k the year before. That's progress! What I had actually saved was about 20% of my software from being devoted to the memory interface alone.

HARDWARE ASPECTS

The great debate was whether to put the intelligence (ie PIC) at the shack end - or out there in the remote ATU.

The former has the great virtue of making the operator interface easy to implement since switches and LEDs are readily to hand. The difficult aspect is the need for a data link - probably serial in practice - for the relay switching commands; and the need for some decoding at the far end. A two PIC solution was implemented (one at each end of the link) to handle the communications as well as the ATU application. It worked well until any serious level of RF was applied - and then it collapsed in a heap of tangled '0's and '1's on the link.

The philosophical issue was that if it were necessary to run even one extra wire up to the ATU (for power or data) then one might as well run many extra cores. But I badly wanted to get to a simple and convenient installation (ie only a coax link) and without all the inherent hazards of adjacent logic and RF signals.

So I built a version with no user interface at all (and no incremental features). It relied on detection of a carrier running for 5 seconds as the cue to go and find a match - which works fine unless you want to use a data mode - and you were never quite sure when it had finished the matching process.

Then serendipity struck. By chance I miscoded the Tx frequency measuring software which left it still running on receive - and noted a broad band spectrum of tones on my receiver as the counting gate opened and closed.

Twenty minutes later, the remote PIC was sending CW - simultaneously on all frequencies in the HF spectrum. Complete flexibility to send any messages back down the coax - and with licensed HF operators as the target audience, well, no problems there. And thanks to Keith, G3OHN for explaining the Fourier maths needed to get the required spectral distribution.

Multiplexing the DC power up the coax presented few problems either. The remaining issue was how to get operator commands up to the remote ATU.

I briefly toyed with the idea of asking you to send CW to it. I think I could have made it work, but whereas it is fair to expect anyone to copy slow machine-sent CW with a restricted set of possible short messages, expecting people to send it might well have limited the appeal. And without a lot of software (for which there is no space) a PIC is very unforgiving of scruffy CW. Especially when it also has to ignore real CW QSOs. Anyway, it was all very clumsy and unfriendly.

The idea of interrupting the power to the PIC was inspired immediately after one of the frequent power failures round here. This provides exactly one command - no more, no less - so it soon developed to the concept of using that one command to pick choices from a PIC-driven menu.

I was now only short of one signal back from the ATU to denote completion of the matching process - which can't be CW since the Tx/Rx is on transmit at the critical moment. I first implemented it by pulsing on and off a small mismatch - which could be observed on the shack SWR bridge, but eventually settled for continuously toggling all the relays and detecting the resultant regular current pulses in the shack.

At the end of the day, this flexible user interface allowed the project to proceed to fruition at all, since I was able to exploit it for software development. One of the issues with PIC development on the cheap is that unless the application has a means of communicating with you, it is difficult to make much progress. Trying to peer at some LEDs through a pair of binoculars at night - and climbing on the roof to upload the latest software from my laptop by day - is not an easy development environment!

SOFTWARE ASPECTS

It would be nice to think that the software development proceeded classically. That is, understand the problem, develop a solution, code and test. Nice theory!

In real life I started off developing some chunks of code to perform utilities I just knew I would need. The ability to measure frequency, VSWR, detect phase, to read and write from memory - and above all, a fast matching algorithm. Then I struggled to integrate these into a working system at the same time as coping with radical shifts in ideas on hardware architecture. The software, the hardware and I were going round in endless loops for weeks.

As with all software development, coding is a pretty mechanistic task. It is understanding exactly how you want the system to behave in *all* circumstances that is the real issue.

Ironically, it was my wife (who has never written a line of code in her life) who sorted it out. By the simple expedient of sitting me in front of a manual ATU and saying "... show me exactly how a human does it. Why do you turn that knob first? Why do you overshoot the minimum reflected power when using that knob?" And, tellingly, "... how did you manage without a phase detector thingy?" Ah well, a lesson learned! The moral is, you don' t have to understand ' software' . You don' t even much have to understand the solution. But you do have to understand the problem.

It also goes to illustrate that even if you have never written software for a living and you are not being paid by the hour, you can still get there if you are persistent and enjoy the challenge.

For your interest - and especially if you want to write your own - **Table 2** shows how much code it finally took to implement the functionality.

MATCHING ALGORITHMS

A few words about the process of finding a match are in order. Although once the installation is stable this may never be needed again, it is certainly critical in getting to that point.

Any matching algorithm is some trade-off of speed versus the certainty of finding the best solution. There are exactly 526,400 relay combinations to try. The software could get round these in a few milliseconds each were it not for the need to allow for a relay settling time of about 15mS. So the ' try everything' approach would, by simple multiplication, take about 3 hours to find the solution. But it would definitely find the best one!

It might be thought that a quick coarse pass through each of the four impedance options - looking for any sort of SWR dip - would then determine which was the best impedance, thus quartering the size of the problem. In practice this does not work because you can find some sort of SWR dip - especially for an already near 50Ω load - for all four impedance options. Some of these dips are due to spurious resonances and some are down to the fact that the solution is truly on the borderline. Some are wide shallow dips, some are narrow, deep and easy to miss. In fact they come in all shapes and sizes. But the only way to find which is the right impedance is to pursue its dip in detail to the best match.

My final algorithm of many does just that. It finds the best match for each impedance by using first coarse steps to find any dips. Picking the best one, it then alternately dithers the L and C values up and down to find the required direction to produce a better match. If the match starts to get worse, that direction is quickly abandoned. If there is no change (which is what happens most of the time) that direction is pursued - since you never know when a dip is just around the corner. But little settling time is allowed for the relays, relying on the hardware to smooth the resultant DC reflected power level. However as soon as any hint of improvement is noted, the algorithm slows down to allow full relay settling. And once the bottom of the dip is passed, its all over!

There are other subtleties but the result of all of this is the best solution for each of the four impedance options - and the best one of all is the winner. The time to do this varies with the load but is typically somewhere about 20 seconds.

This is a necessary improvement on 3 hours but must carry some inescapable risk. The risk occurs early on; that in going for the best dip after the coarse pass the wrong one is picked. This risk is minimised by keeping the coarse steps small - which, of course, in the limit takes you back up towards 3 hours again.

22	PIC general overheads
19	General timing
93	Memory & bus management
66	CW send routines
55	Frequency measurement
48	Frequency to memory mapping
20	Reflected power measurement
63	' Restore from Before'
260	' Match from Scratch'
69	Menu management
107	Mode U utilities (QED)
120	Status reporting
71	Optional Bits (A-E)
1013	**Total**

Table 2: Lines of code to implement PicATUne functionality. Divide by 10 to give percentage.

I have tested the algorithm against a wide range of reactive dummy loads - and real antennas - at different frequencies. The algorithm does not always produce the same result (there are often two or more genuinely good answers), but it has never got it truly wrong yet. If it should happen to you, my suggestion would be to let it try again, perhaps at a slightly different (and preferably higher) power level. By its nature I can' t anticipate the problem; if I could I would code it out.

But for sure, if your Tx contains any significant spurii (harmonic or otherwise) then although it will always indicate unity SWR into a dummy load, it will play havoc with any reflected power measurement into a real antenna.

Peter Hart reviewed the SGC SG-231 Smartuner [9] using a 100W light bulb as a dummy load. As he points out, this varies in impedance between 50Ω and 500Ω with increasing applied power. The SG-231 passed the test of matching it, but I suspect PicATUne would not - except by luck. My algorithm does not anticipate the antenna changing impedance with applied power - on the contrary, it assumes that the antenna impedance remains constant at any given frequency.

The same load was used to judge internal heat dissipation and hence efficiency. Without any implication that the SGC is less than efficient (I have never tested one) it is worth pointing out that any ATU will be efficient into this sort of load. As the graphs in Part 1 show, the real stress areas into real antennas do not lie around the 500Ω region for an L-match. Nor do they for any other configuration.

OPERATIONAL USE

PicATUne HAS ONE NORMAL default mode of operation (and 8 other modes used variously for training and maintenance). Each mode is named for the CW character sent when first switching to it - and that letter is also a prompt for its function.

NORMAL USE - MODE K

Entered at power on, this is the default mode for operational use. Whenever you hear a **K**, it means - unsurprisingly - over to you. All other modes revert to K on completion. Once you have a stable installation, you need never leave this mode.

In Mode K, PicATUne constantly monitors the frequency of your transmission - and repeatedly fetches the pre-stored solution from memory for that frequency - and applies it immediately on a break in transmission. A morse dot or a snatch of SSB speech is sufficient; merely pressing the PTT switch is often enough. This process is called ' Restore from Before' .

In practice, a ' break in transmission' occurs when the SSB waveform takes a dip at normal speech frequencies ie roughly every 10mS. An actual pause in speech is not needed. On CW, the gap between morse elements is used.

During each such brief burst, the software measures the frequency - twice normally, but four times if a change is detected - and only if it gets the same result each time, looks up the matching solution in memory for that frequency. If none is found, it searches both up and down from that centre frequency till it finds the nearest solution - or the band edge(s). This entire process is essentially instantaneous in human time frames - and of course, if you haven' t actually changed frequency there is no net effect.

The software contains a zero crossing detector which minimises relay switching under load. It waits for the brief break - and was implemented purely as a means of making the relays affordable. So, if you are using a constant carrier mode, eg a data mode or FM, you absolutely must send a quick burst after you change frequency to allow switching.

Following any change in band you will hear **R K**. Thereafter no further status is reported while you remain on that band - but a different solution will be applied as necessary should you change frequency within the same band.

Note that PicATUne makes no judgement about the quality of the solution. As a matter of design philosophy, you are in total control; and it is up to you to decide if the SWR is too high - and to train PicATUne to a better solution for that frequency (assuming one exists).

If - very unusually - there is no stored solution for any frequency in the band then PicATUne will send a constant and annoying **X K** sequence. This sequence ceases on pressing the Command Switch.

USER SELECTABLE MODES

THESE MODES ARE made available by pressing the Command Switch while in Mode K. PicATUne responds by sending the user selectable mode letters **M U S I C** - an easily remembered acronym. If none of these characters is selected, operation continues in Mode K.

To summarise what follows, the modes and their initials are:-

- **M**atch from Scratch
- **U**tilities
  - **Q**RS or **Q**RQ
  - **E**rase solutions
  - **D**ummy load
- **S**tatus
- **I**nhibit
- **C**onfigure

MODE M

' Match from Scratch' . This is the mode you use to command PicATUne to find and remember a new matching solution.

On entering Mode M, PicATUne sends **R** to acknowledge your command followed by **M** continuously repeated.

You then supply a steady carrier for about 20 seconds. The time varies depending on the nature of the load. As for power level, 10W is ideal. 5W is acceptable and 2W will often produce a result. Anything over 40W risks saturating the detector, especially on the LF bands - which will merely have the effect of prolonging the matching time. PicATUne is in ' quiet tune' during matching so for 10W in, you will be radiating 40mW. This is not much, but it is not nothing!

Whatever power level you use, *do not alter it during the matching process*. If you should get the power level badly wrong, immediately stop transmitting - which aborts the process - and start again. (You can do this deliberately while setting the power level in the first place.)

After detection of the carrier, PicATUne finds the best match and stores the resultant solution against the frequency in use. Throughout the matching process you will note that your SWR bridge reads close to unity - and the intensity of the Command Unit LED varies. When it starts to flash at about 1Hz, you know matching is complete and you can stop transmitting.

PicATUne then reverts to Mode K with the matching solution applied - and you will hear **R K**. If you have not held the key down long enough, you will hear **M** continuously repeated. You need to either re-apply carrier for long enough to find the solution - or you can press the Command Switch to abort and return to Mode K.

Up to 1,000 different solutions may be stored, sufficient for a potentially different solution every 5kHz throughout the 9 bands 160m through 10m - (slightly larger band allocations than the present USA allocations are assumed). If you use PicATUne outside these allocations, the results are not predictable.

MODE U

Utilities mode brings up **R U** followed by **QED**, the initials of the three utility modes.

Mode Q (QRS or QRQ) allows you to toggle CW speed between about 12wpm and about 20wpm. On first use, PicATUne is set to the slower speed. Even if your CW is truly appalling, this will soon seem very slow. When you select Q, the speed will immediately change and operation reverts to Mode K. Your speed selection is remembered even after powering off.

Mode E Erase memory contents mode. This erases all stored matching solutions. Use it if you change your antenna installation, thus invalidating the solutions.

Before erasing, you will be asked to confirm - **CFM** - by pressing the Command Switch. If you do, you will hear four **E**s, at about one second intervals, one for the successful erasure of each 8k memory block. There is no way back!

Mode D Dummy load mode is for your general convenience - and that of other operators. It switches in the dummy load with the L-match and antenna grounded to give vanishingly small radiation.

The character **D** is sent continuously while Mode D is activated - and is exited to Mode K by pressing the Command Switch.

MODE S

Status mode reports the current PicATUne settings. It uses binary values, with a dit representing a ' 0' and a dah representing a ' 1' . The most significant bit is sent first.

Besides obvious value in commissioning the L, C and Z information is useful in determining if PicATUne is matching in a ' risky' region. More about that later.

L	**6 bits**	which L1 turns in use
C	**11 bits**	which capacitors in use
Z	**HI LO ONLY L ONLY C**	
FREQ	**13 bits**	frequency ÷ 5
BITS		
BIT A	**ON** or **OFF**	
BIT B	**ON** or **OFF**	
BIT C	**ON** or **OFF**	
BIT D	**ON** or **OFF**	
BIT E	**ON** or **OFF**	

Fig 23 allows you to perform an approximate conversion of L1 turns to inductance.

The frequency bits have the normal binary weighting but need multiplying by five to give the answer in kHz. Some examples are:-

3.7MHz	00010	11100100
14.2MHz	01011	00011000
28.5MHz	10110	01000100

Note that this is not a frequency standard in any real sense. Its sole purpose is to let you check that PicATUne is getting the frequency about right.

Pressing the Command Switch at any time aborts Mode S and PicATUne reverts to Mode K with **AR K**.

MODE I

Inhibit mode. This mode inhibits all activity and explicitly prevents PicATUne from changing the matching solution.

In this mode the PIC chip itself goes to SLEEP and ceases all activity including its 4MHz clock, thus preventing the possibility of any internally generated noise finding its way into your receiver. Since all the serious PIC activity occurs while you are transmitting, this latter is not a realistic risk - but better safe than sorry.

This mode is particularly useful if other strong transmitters are in the immediate vicinity, since there can be enough pick-up on your antenna to cause PicATUne to react to a strong inbound received signal.

Pressing the Command Switch gets you back to Mode K.

MODE C

Configure mode allows you to specify the behaviour of the five optional output bits. Why you might want to is covered in a moment.

Of these, Bit A is a simple on/off toggle switch. Firstly, you will hear **BIT A** followed by **ON** or **OFF**. If you select it, the switch immediately toggles and PicATUne reverts to Mode K. The switch setting is remembered during power off.

If you make no selection against Bit A, PicATUne will then play out 4 bits (B-E) against each of 9 bands, in the sequence 160m-10m. A typical ' line' is:-

20m N Y N N

This example states that if you were to operate on 20m, Bit C would be set.

To alter a given setting, simply select it as it goes past and PicATUne will toggle it Y/N and then replay the entire line for confirmation before continuing. Thus to change all four settings (in any order) you would make four passes through the line.

There are a total of 36 settings (9x4) to provide maximum flexibility for your application. The net result of altering any of these 36 configuration settings is first applied after the next burst of transmission. They are all remembered during power off.

SWR PROTECTION

An inherent problem with any auto-ATU is that of presenting your Tx with SWR spikes when changing bands. For example, if you are operating on 80m and change to 40m, the 80m solution is likely to present a very high SWR to your Tx on 40m. This only lasts for a few milliseconds while the ATU measures that something radical has happened - and fetches and applies the 40m solution. But during that few milliseconds, your PA transistors can exhibit their fuse-like properties.

I am not aware of any commercial auto-ATU which does other than rely on SWR protection in your Tx to save the day. Ironically, if it works to shut down the PA quickly and hard, then there may not be enough RF energy reaching the ATU to allow measurement of the new frequency.

PicATUne has two defence mechanisms:-

1. If very high reflected power is measured in normal use, PicATUne will switch instantly to its dummy load. This in turn allows your Tx to develop full power - allowing PicATUne to measure the frequency, find the solution, apply it and then remove the dummy load. This all takes milliseconds and unless you have a hard fault on your antenna, you are unlikely to be aware of it.

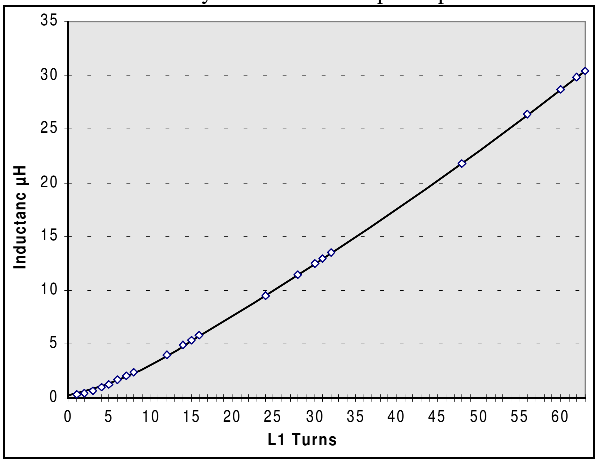

Fig 23: Measured inductance of my coil. Yours may vary from this, but the difference is unlikely to be significant in plotting your operating point on the graphs in Part 1.

2. If your PA is unprotected - or you don' t wish to rely on it every time you change bands, proceed as follows:-

- Change bands on your Tx/Rx but before transmitting, press the Command Switch to bring up the MUSIC menu.
- Transmit a morse dot or utter a brief word. You are on dummy load, so nobody will hear you. Immediately key-up.
- If you continue to hear the main menu, then you have not transmitted enough power even to be noticed. If you hear a succession of **T**s, then you have not applied enough power for long enough for accurate frequency measurement. Apply more, and the Ts will cease. (Or press the Command Switch to escape completely.)
- Normally however, you will hear **R K** and you will have changed bands without ever presenting your Tx with other than 50Ω - even for an instant. Nor will you have radiated. For both these reasons, this is the preferred way to change bands.

GENERAL INTERFACE ISSUES

Although PicATUne is simple to use, there are some issues to watch out for:-

- If you make a mistake in menu selection, the simple rule is - do nothing! PicATUne will find its own way to a stable situation - normally Mode K.
- If you want to hear CW messages from PicATUne, you must be listening! PicATUne cannot tell if your Tx/Rx is on receive - or on Tx but with no output. If you are operating SSB without VOX or CW without break-in, you will miss any message sent at key-up time if you do not lift the PTT line briefly.
- If you are using a narrow Rx filter, PicATUne' s CW note may fall outside your passband.
- An interesting phenomenon arises operationally when you are by chance on a frequency that is spot on the transition between two different solutions. You may find PicATUne flips back and forth between them, especially on SSB where your actual voice frequencies of the moment determine the transmitted frequency. This is not a problem and can be safely ignored. But if you are within earshot of PicATUne, the noise of relay switching can be a nuisance. You can always engage Mode I (Inhibit) to stop it happening. Equally this *may* be evidence that you have got too many closely spaced - and unnecessary matching solutions.

PIC YOUR ANTENNA

THE OPTIONAL BIT outputs provide a unique and powerful means of automatic remote antenna selection. They are designed to drive external relays - though a few extra ones could be judiciously accommodated within the PicATUne enclosure. It is up to you to use relays which can stand the strain in your application.

Bit A is switched directly from the Command Unit, providing remote *manual* antenna swapping. It is designed to drive a relay which diverts the coax input to the L-match instead to a further coax socket and thence to an already well matched alternate antenna. The L-match is thus bypassed, but will continue to be set. So when you switch back to the matched antenna, it will be instantly available.

If this ' alternate' antenna is in fact the only one you want to use on a given band, it would be better to configure one of the four frequency sensitive bits (Bits B-E). You would then *automatically* switch to this antenna whenever you used that band.

Here are a other few ideas for those frequency sensitive bits:-

- Nested mono-band Quads or Yagis. Feed them with one coax run up to PicATUne near the masthead and then configure the bits to route a separate short coax lead to each antenna - as a function of frequency.
- A mast with both beam and wire antennas. Much as above, you can specify which antenna is to be selected on which band.
- On any band (particularly the LF bands) if PicATUne has not enough matching range, you can arrange to selectively switch in some external reactance to improve the match.
- Many antennas benefit from different earthing arrangements or counterpoise lengths on each band. These too can be switched in automatically.

This feature is very versatile, but suffice it to say if you have no interest in it, all you have to do is ignore it.

BALANCED ANTENNA OPTION

IF YOU WANT to feed balanced loads - yet still retain an unbalanced capability - you need to insert a choke balun in the coax feed from the SWR head to the L-match input. This is made possible because the whole L-match is floating at RF - except for the braid of this coax.

I taped together four 3/8" ferrite rods, 145mm long to form a square(ish) cross section and then wound 23 turns of RG58 round this. This assembly fits in the space on the copper side of the RF deck sensor section - between the SWR head and the casing.

Another approach - but for balanced configurations only - is to fit an external 1:1 balun at the antenna and counterpoise terminals. This practice is employed by the commercial ATU manufacturers, but I confess doubts about its effectiveness. Such baluns do not work well in the presence of a reactive component.

FIRST USE

CONNECT UP any antenna of convenience to PicATUne and place it where you can see and hear it. Keep your power level to no more than about 10W and set the spark gap to a few thou until you gain confidence.

Run through all the menu options to gain familiarity. Specifically try out the erase mode (Mode E) since you may well not want to do so later once you have some real matching solutions stored.

Then use Mode M to find some matching solutions - preferably on different bands - and practice subsequent band changing.

TRAINING PicATUne

ONCE YOU ARE confident PicATUne is functional, mount it in its target location and connect up the antenna(s). Choose a pleasant day and fit PicATUne not weatherproofed to give you access to the spark gap. Remember to use Mode E first to get off to a clean start.

Starting on the lowest HF band of interest, check that you can obtain a reasonable match on all the HF bands. If you have any problems, pay particular attention to the quality of your counterpoise or ground plane or RF earth - depending on the type of antenna you are using.

Ultimately if your antenna is too short PicATUne will not be able to match it. A possible way round this was just discussed.

If your antenna system is near half wave resonant and end-fed - or full wave and centre fed - there may be portions of the band which will not produce a good match. See Part 1, Fig 1. Altering the antenna length slightly either way should fix it - and in general, longer is better.

For each band, start at the band bottom edge and use Mode M to ' Match from Scratch' . Then move up the band, checking the SWR. As soon as it starts to become unacceptable, use Mode M again to find a new solution. Repeat until you get to the top band edge. With some antennas, the same match will cover the whole band. With others, especially if the antenna is naturally near resonance, you will need many different - potentially radically different - solutions.

Then (using Mode S) plot your operating points onto the graphs provided in Part 1. Alternatively, the Mode S data may be entered directly into the QBASIC utility. This vital step ensures you are not operating in a danger zone - or at least that you understand and accept the limitations.

All the above caution is wise practice but in reality there are few issues unless you want to push the power to (but not beyond) the design limits.

As a final check, turn your power up to the normal operating maximum and check that there is no flashover on the spark gap at any frequency. You will be able to see the arc at night from a distance and hear it on any broadcast receiver. If there is, open up the gap as little as you need to, but to absolutely not more than 75 thou (2mm) - ie 1.5kV. If the spark gap is flashing over you will generate substantial TVI and BCI. So you must check that there is no evidence of this at full power - on all the bands.

Finally, complete the weatherproofing and enjoy!

ACKNOWLEDGEMENTS

THIS PROJECT was tested by Alan, G3TIE; Keith, G3OHN; Peter, G3XJS; and David, G4FQR. Without them and the power of instant information and software exchange over the Internet, it would never have seen fruition. My thanks to them for their invaluable help in verifying the drawings, making suggestions for this article and getting the software features tested and honed.

My thanks to Steve, G4ZBV who took all the photographs and to my wife, Fran, for her active contribution and endless patience.

REFERENCES

[8] ' An Automatic Antenna Tuner: the AT-11' , byDwaine L. Kincaid, WD8OYG. *QST*, Jan 1996.

[9] 'SGC SG-231 Smartuner' , reviewed by Peter Hart, G3SJX *RadCom*, Feb 2000.