ZL1BPU LF Exciter

User Manual

M. Greenman CSc, ZL1BPU
July 2002

Contents

Notices

Source code for this project (the assembly code, include files etc, everything to reassemble the firmware) is thoroughly commented, and available from the author for $30. The source code is written in assembly language for the ATMEL AVRASM Windows compiler.

While the information in this manual is believed to be correct, and the author's prototype Exciter works to his entire satisfaction, the author accepts no liability for errors or failure of any constructor's copy of the design to perform as expected. Support is limited to the design as described, and to firmware purchasers as outlined above.

This Manual was written in Microsoft Word and ported to PDF format for distribution.

[1] zl1bpu@nzart.org.nz

1. Introduction

The ***ZL1BPU LF Exciter*** is an advanced unit, out-performing many conventional LF Exciters of the crystal controlled, VFO or PLL synthesis varieties. Despite the performance advantages, and its versatility, the unit is inexpensive, simple to build, has no adjustments, and is (mostly) easy to use.

The frequency reference for the unit is derived from a single stable high frequency crystal or TCXO (temperature controlled crystal oscillator), and therefore the transmissions are extremely stable. Because the unit uses a direct digital synthesis technique, the LF signals are also very clean. It is especially important in equipment for narrow band transmissions that associated carrier phase noise is very low.

The unit is self contained, 12V DC operated, and generates about 1W sine wave output from 10kHz to at least 250 kHz, and is useful from 7 kHz to 400 kHz, in steps of 0.085 Hz. Power level is DC adjustable over at least 60dB range. This feature, plus the built-in sweep generator, makes the unit a highly useful signal generator. Without the power amplifier, the output is 50mV RMS into 50 Ohm from zero to 400 kHz.

The ***ZL1BPU LF Exciter*** can be externally set-up and controlled by a computer or terminal, and has a built-in beacon system. The unit will transmit any (or all) of the following modes, using the built-in beacon, with no PC or other equipment required:

- Morse and other ASK modes, including QRS and QRSS
- FSK Morse
- DFSK Morse (also known as DFCW)
- MFSK single tone image modes, such as Sequential MT-Hell
- ASK image modes, such as Feld-Hell
- MFSK or IFK data modes, such as JASON

In fact, multi-mode and multi-frequency messages can be used, and the user can change the message any time, even remotely, using a PC or dumb terminal.

Using special PC software, (from the author or write your own), the unit will also transmit many modes from the PC keyboard in “real-time”. It is also possible to define your own modes, using a combination of frequency, offset, on/off keying and keying speed. The Exciter will generate:

- All the same modes as the beacon
- FSK and MFSK data, including RTTY
- ASK data
- IFK data, including JASON, using I2PHD/IK2CZL JASON V0.94 software
- Sweep Generation

To use the Exciter for PSK modes will require an external modulator, which could be conveniently interposed between the sine wave generator and the power amplifier. Commands sent to the Exciter during PSK transmission will cause unwanted changes in transmitted phase.

The Exciter "KISS" protocol is "packet ready", so the Exciter can be operated remotely via a packet radio or other serial data link.

Beacon Operation

To operate the ***ZL1BPU LF Exciter*** as a beacon you will need:

- An LF antenna and tuner – well of course!
- A 12V DC power source, capable of 200mA.
- For recording the beacon message and setting up, a dumb terminal or PC terminal capable of 9600 bps N-8-1.

Programming

To program the micro controller in the unit, you need the following (only once, during the construction and commissioning process):

- An AVR programmer. A simple parallel programmer[2] is recommended. Use the ATMEL ISP programming software.
- The EXCITER firmware, available from the author.
- A PC running Windows 3.1, 95 or 98 to run the programming software.

Real-Time Operation

In order to use the ***ZL1BPU LF Exciter*** for real-time QSOs, you will need a PC or other computer with relevant software. You could write your own, or use one of the DOS or Windows applications provided by the author. These are freely available from the website.[3] The author's software provides Morse, QRSS, DFSK and MT-Hell keyboard operation. You will need to use other software for reception. The author's software is intended for use in conjunction with ARGO[4] by

[2] See www.qsl.net/zl1bpu/micro or http://sharon.esrac.ele.tue.nl/mirrors/zl1bpu/micro for a suitable programmer. Alternatively, have a friend program the device for you.

[3] See "Where to get the Software", at the bottom of the "General Information" page.

[4] See www.weaksignals.com .

I2PHD and IK2CZL. In order to write your own software, you will need to understand the KISS commands, which are listed under “Serial Commands” in the Appendix.

Building the Exciter

The ***ZL1BPU LF Exciter*** is simple to build. There are four ICs and a handful of other components. The parts are not difficult to obtain. Construction is achieved on a simple project board about 100 x 150mm in area, with room to spare. No surface mount components are used. There are no adjustments to make.

There is only one front panel control, and four connectors – RF OUT, SYNC (or TX ON signal), DC power and RS232 communications.

- Level of expertise for construction – MODERATE

NOTE: ***No printed circuit board is available, and none is necessary.***

Fig. 1.1 Front view of the completed Exciter

2. Applications

There are four main uses for the ***ZL1BPU LF Exciter***:

- Stand-alone beacon keyer, modulator, VFO/exciter and transmitter, all in one.
- Real-time QSO transmitter or VFO/exciter
- Signal generator
- Sweep generator

Beacon Transmitter

Since the unit has over 100 bytes of user recordable message memory, just about any message you want, in any mode you want, can be recorded in. Indeed, since the beacon is "smart", you can send several different modes, by including mode changes, and even frequency changes in your beacon message![5]

Recording messages for standard LF modes, such as Morse, QRSS, and DFCW is straightforward using the "MAKEBCN" software provided by the author. This compiles and downloads the message, and allows you to set the keying parameters (speed, shift and operating frequency), even in the message if you wish. Recording messages that include the image modes is rather more complex, and needs to be done by hand input of the data bits. The author can code a mixed-mode message with callsign in MT-Hell or Feld-Hell in under 30 minutes, so it's not too difficult. JASON is the most complex, and really requires some method of capturing the output of the I2PHD/IK2CZL JASON program.

While the transmitter power level is not programmable, it can be set to any desired level via a front panel control (the only front panel control!). There are three programmable digital outputs, so it would not be difficult to use these to control power level, switch antennas, and so on. Since the message is stored in memory, just apply power, connect and tune the antenna, and the beacon will run with no further intervention. If you want to operate the beacon for a fixed period each day, simply operate it from an AC supply fed from a time clock! The message always starts at the beginning when the micro controller is reset.

Real-Time Transmitter

One watt is not much power for LF operation, and most users will prefer to drive a high power transmitter. The advantages of using this device as the exciter for a high power transmitter are the versatility of operating modes, extreme stability, precise frequency setting, and the very clean carrier generated.

[5] See the list of Beacon commands in the Appendix.

The ***ZL1BPU LF Exciter*** RF output is a clean sine wave with very low harmonic content. To operate a linear amplifier, simply connect the output to the transmitter input, set the output tap for 50 Ohm operation, and adjust the front panel output level control for sufficient drive. It is a good idea to include a high Q matching circuit between the Exciter and the transmitter, to further attenuate spurious products, such as the second and third harmonic, and the sampling clock frequency. However, given the operating bandwidth of most transmitting antennas, this will usually be quite unnecessary.

To operate a transmitter that requires square wave or twice-frequency drive, add the necessary squarer and diver circuitry to the output of the sine wave generator, and omit the 1W power amplifier. CMOS level 5V square wave drive is also available from the micro controller at PB7 (pin 19) or from programming header J1. You will ***definitely*** require a low pass filter on the transmitter output if you use square wave drive!

Real-time operation requires keyboard control. The Exciter is connected via a serial control link to a PC or other computer which runs a keyboard control program such as the author's EXC.EXE or Con ZL2AFP's simple Windows program. These programs send the necessary "KISS" commands to set the transmitted frequency, turn the transmitter on and off, and generate the on-air signals. They also allow you to change mode, and send from the keyboard or from pre-programmed messages. Such programs are designed to operate in conjunction with ARGO, which is used for receiving. In fact, you can even operate full duplex on just one computer, if the radio equipment will permit.

Signal Generator

The RF amplifier in the ***ZL1BPU LF Exciter*** has an extraordinary control range. By simply adjusting the drive level it is possible to reduce the output to fractional microvolts. There is little leakage from the Exciter, so it is possible to reduce the output way below the receiver noise threshold. A fixed 10dB or 20dB attenuator on the transmitter output is a good idea, in order to protect the receiver from damage.

Using any of the suggested PC software, you can tune the Exciter to any frequency you desire, within the range to 400 kHz. The output is substantially flat from 10 kHz upwards. If audio frequency capability is of interest, add a small line transformer across the output transformer L1. The 1W linear amplifier is actually an audio power amplifier.

It is not especially difficult to write your own program to control the transmitter or generate special complex transmission modes. This can be done on the oldest PCs using nothing more than GWBASIC, which came with early versions of DOS.

The LF Exciter is also fully compatible with the Signal Generator firmware,[6] which can be used where sine, square ramp or triangle waveforms are necessary. The Signal Generator also includes the sweep generator mode. Note that different control software is used since the serial commands are different.

Sweep Generator

The ***ZL1BPU LF Exciter*** uses a direct digital synthesis technique, realised in software, and so the frequency generated can be changed virtually instantly. The sweep step duration, frequency step and number of steps are all programmable. In the sweep generator mode, an internal timer changes the frequency (for example) every 5ms, with (typically) 20 steps per sweep, and generates precise frequency steps appropriately timed for use with an oscilloscope.

The sweep generator includes a SYNC output that is high only during the first frequency step. This is used as a synchronizing signal for the oscilloscope. With the time base of the oscilloscope set to (for example) 10ms/div, the display conveniently shows two frequency steps per division.

The sweep generator can sweep a range greater than 100 kHz, and operates over the full range of the generator. The minimum step size is 85 mHz, and the maximum about 5.5 kHz. These features make the unit very handy for checking out antennas, tuners, and even sweep IF filters (you can sweep in precise steps, and just about reach 455 kHz). Because the sweep can go from zero, it is also useful as an audio sweep generator, so you can sweep audio amplifiers, filters and SSB transmitters.

The sweep step resolution is the same as the frequency setting resolution. There can be from one (sweep off) to 255 steps, and the sweep dwell time (time per step) can be from 1ms to about 200ms. The versatility of the sweep generator is such that you can even generate test signals for RTTY and other FSK modes.

[6] See www.qsl.net/zl1bpu/micro/SIGGEN/Siggen.htm or http://sharon.esrac.ele.tue.nl/mirrors/zl1bpu/micro/SIGGEN/Siggen.htm

3. Theory of Operation

The Exciter uses software simulation of the Direct Digital Synthesis (DDS) technique to generate sine wave radio signals with very high precision.[7] The DDS system (whether in hardware or software) uses several building blocks – a ***high-speed adder***, a ***look-up table***, an ***analog to digital converter***, and a ***low pass filter***. There are three main functional sections to the Exciter unit – the ***micro controller***, a ***symbol clock***, and the ***power amplifier***.

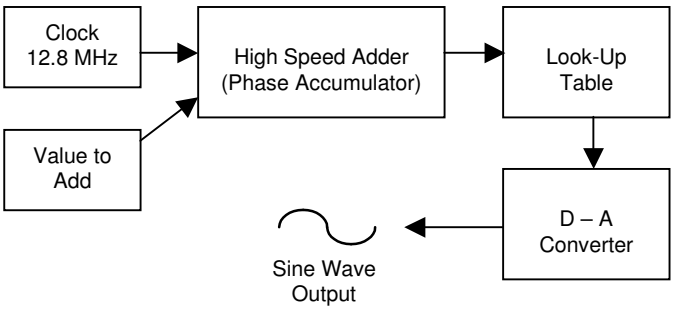

Fig. 3.1 DDS Block Diagram

The High Speed Adder

Early digital synthesis using digital techniques was achieved using resettable counters, the same technique still used today for phase locked loop synthesis. The counter can be made to give an output every "n" input clock events, where "n" must be between 1 and the maximum possible count of the counter. The trouble with this technique is that resolution is very limited, in fact limited to exact fractions of the clock, so that a very high frequency clock is necessary for even modest resolution.

[7] There is an excellent application note introducing Direct Digital Synthesis and function generation at ftp.thinksrs.com/PDFs/ApplicationNotes/dds.pdf

The high-speed adder technique gets around this limitation, and relies on the Nyquist Criterion for proper operation. Nyquist proposed that any periodic waveform could be reconstructed from at least two samples of the waveform per cycle (of the highest frequency component). In practice, more samples are required, as it is difficult to conceive sufficiently good interpolation filters to reconstruct the waveform accurately. Three samples seem to be sufficient.

So how does this affect the high-speed adder? Well, if we arrange to add a fixed number into a register at every (high frequency) clock event, and let the register "overflow" when it is full, the value in the register will appear to ramp up at a rate that depends not only on the clock frequency and size of register, but on the value added each time. (You can imagine the conventional resettable counter is simply an adder which only adds "1"). Such a register is called a *phase accumulator*, and the value added each time, the incremental change in phase, represents the requested operating frequency (see Fig. 3.1).

This technique allows the ramp represented by the phase accumulator count to progress (in effect change phase) for every clock event. If we add a positive number, the phase advances (the register ramps up); if we add a negative number, the phase retards and the register counts down. If we add zero, the phase stays constant.

The Look-up Table

The number in the phase accumulator register represents the current phase of the output signal. By using this value to look up a number in a table, it is possible to generate any periodic waveform, with a frequency that corresponds to the rate at which the counter overflows and repeats the same phase. In the ***ZL1BPU LF Excite***r, one table is used, containing a sine wave. Eight bit values are used. The phase accumulator is 24 bit, and after each phase step, the phase value (using the most significant 8 bits) is looked up in the sine table and the corresponding value placed on the outputs. (In the Signal Generator, the same technique is used, but any one of *four* different tables can be selected for different waveforms). Limiting the output and the phase lookup value to 8 bits has no appreciable effect on the output.

The D-A Converter

The micro controller output is a digital number representing an instantaneous sine value, which still must be converted into an analog value. This is the job of the Digital to Analog Converter. In the Exciter a simple resistive network does this job, and generates a 256 point waveform with very good linearity. The signal is

5V p-p and is completely flat in frequency response from DC to over 400 kHz. The quality of resistors used has a direct effect on the purity of the signal.

Frequency Limitations

The way the adder operates, low frequencies are generated by adding small values to the phase accumulator. The smallest increment is of course one. In the ***ZL1BPU LF Exciter*** design, a large phase accumulator (24 bit) is used, and so a step of one works out to 0.084 Hz. We call this the *DDS Resolution* (the mathematical derivation is covered in the next section). A step of zero of course results in a zero phase change (zero frequency or DC) output, a step of one gives a frequency of 0.084 Hz, and so on. Higher frequencies involve first fewer samples per step on the sine wave, and higher still, fewer and fewer samples per sine wave. The maximum frequency is limited by the Nyquist Criterion and practical low pass filters to about one third of the adder clock frequency, or about 400 kHz in this design. All frequencies from zero to 400 kHz can be generated with the same 0.084 Hz resolution.

As we approach 400 kHz, the output contains fewer and fewer samples per sine wave. If we continue on up in frequency (you can try this as there is actually no limit to the numbers you can give the Exciter), an image frequency is generated in addition to the desired output. This frequency ($f_{clock} - f_{desired}$), gets stronger and stronger, and closer and closer to the desired output, until at exactly one half the clock frequency, the image and desired output coincide.

It is actually possible to intentionally select the image frequency as an output, but very good band pass filters are necessary to exclude other frequencies, and this technique does not suit broadband operation for which this Exciter was designed.

Another interesting thing to try is to give the Exciter *negative* frequency values! The frequency value ("Value to Add" in Fig. 3.1) is a 24 bit number (0 to 16777215), and all the numbers above 8388607 represent negative frequencies. For example, a value of 16777215 will generate the same frequency as a value of 1, but the look-up table will be stepped backwards. This has no effect with a sine wave, which is of course symmetrical, but has a dramatic effect on ramp waveforms. Another odd thing about negative frequencies is that FSK shifts are reversed, and in sweep generator mode it is possible to sweep backwards or even through zero!

Frequency Resolution Mathematics

The DDS technique used in the ***ZL1BPU LF Exciter*** is borrowed from a signal generator by Jesper Hansen[8]. This highly efficient but tiny piece of software (only six instructions) performs the high-speed adder and look-up table functions in only nine micro controller clock cycles.

In the Exciter the micro controller clock frequency is very high – typically 12.8 MHz, and so the DDS executes one step every nine clock cycles, or at a rate of 12.8 MHz / 9 = 1.42 MHz.

The frequency resolution is also related to the size of the phase accumulator register. The Exciter uses a 24 bit register, and so the frequency resolution is given by the relationship:

$$\textbf{Resolution} = F_{crystal} / (9 \times 2^{24}) \text{ Hz} \quad \ldots[1]$$

With a 12.8 MHz crystal frequency, this works out at 0.0847710503472 Hz.

To calculate the value to send to the Exciter for any given frequency, you need to divide the frequency you need by this value, with all the many decimal places. Why so many decimal places? Well to generate accurate frequencies with all the precision necessary, we *need* the decimal places. Here's an example…

> We want to generate 136.000 kHz. Let's just divide by 0.084 and see what happens. The result is a command value of 1619048 (to the nearest integer). The frequency generated (converting back again) will be 137.248 kHz – well over one kHz in error.
>
> Using the correct division value, we end up at 135.999976 kHz, only 0.024 Hz in error. This small error is caused by the need to use integer numbers for the phase addition (remember the 0.084 Hz resolution).

We only need to think about these awkward numbers when entering frequencies manually. The PC software does all the hard work for you, with full resolution.

[8] See http://www.myplace.nu/avr/

Hexadecimal Calculations

The Exciter KISS commands use hexadecimal numbers.[9] These are easiest for micro controllers to manage, but mere humans will require some assistance!

> Take the previous example – transmitting a carrier on 136.000 kHz. From the above formula [1] we know the command value should be 1604321_{10}. Now that's $187AE1_{HEX}$. How do we know? The simplest way is to use a Windows calculator, or other calculator with a hexadecimal mode. Perform the calculation in decimal, and simply press the HEX button to convert the result, so that's the value to send the Exciter using the KISS "F" command - "F187AE1" (F commands always have six following characters).

We only need worry about hexadecimal numbers when entering frequencies manually. The PC software does all the hard work for you, calculating the numbers and converting to hexadecimal.

The Micro Controller

This is the heart of the Exciter. As well as providing the DDS functions described above, the micro controller has software to record messages, interpret commands, and control the DDS output. However, it cannot do all these things at once, and specifically, it cannot communicate with the user (receive commands, send data) while generating LF signals. So, you will notice that while you talk to it, the carrier generation stops. This is most obvious when controlling the unit manually, as the commands sent from a PC program are very brief, and those generated by the internal beacon and sweep generator are even briefer.

In effect there are four main sections of program, all apparently working independently, which make up the Exciter controller:

- The DDS synthesizer (generates the signal)
- The serial port command interpreter (operates the KISS protocol)
- The symbol rate timer (sends the next beacon data bit and operates the script interpreter)
- The sweep step timer (steps the sweep frequency, normally turned off)

[9] Hexadecimal (HEX) counts 0 – 15 instead of 0 – 9 for each "digit". The numbers are represented by 0 – 9 and A – F, and the hexadecimal "digit" is called a "nibble". A "byte" represents two nibbles, and can express numbers from $0 – 255_{10}$ ($0 – FF_{HEX}$).

This apparent program independence allows the operation to be very flexible (for example you can change the frequency, shift, mode, keying speed etc. even while transmitting), and also makes design of the micro controller firmware much more straightforward.

The Symbol Clock

When the beacon is operating, a slow timer (at data symbol rate) is required to trigger the next pass through the beacon control section of the firmware. (Each unique transmitted data event is called a “symbol”). Between these events the micro controller is busy generating RF. The timer could be generated inside the micro controller, but unfortunately, because of the very high crystal frequency, it is not possible to directly generate long enough periods for the slower LF modes (maximum would be about five seconds). It is possible to extend the time by simply adding several of these periods together, but it leaves unfortunate carrier interruptions whenever one of these events occurs. As a result, the signal has unwanted keying pulses superimposed, not what is wanted for narrow LF transmissions.

The solution is to use a slow external clock source (refer to Fig. 4.4). This could be crystal controlled (for example using a 32.768 kHz watch crystal), but it has been found that a simple stable RC oscillator is sufficient. In this design the oscillator runs at about 2 kHz and is divided down to provide a stable 64 Hz clock to the micro controller. The internal timer further divides this down, so the internal symbol clock can be from 64 Hz down to one clock every 1024 seconds. The range can be increased and decreased by selecting different divider outputs, but the range provided should cover all practical requirements.

The rate at which symbols occur is called the *baud rate*. To define the required baud rate, you have to specify the division ratio for the micro controller timer. Obviously, with a 64 Hz reference, one dot or data element per second will require dividing by 64 in the timer. Here’s the general formula:

Timer Value = 64 / baud rate …[2]

The number the timer needs is actually 65536 minus this number (it is a 16 bit timer), but the micro controller works this out.

For modes where you need to express the baud rate in “so many second dots” (dot period) as for QRSS, the formula becomes:

Timer Value = 64 x dot period …[3]

As an example, to generate 3 second dots, use a timer value of 3 x 64 = 192 ($00C0_{HEX}$). The values for the **K** command, which sets the keying rate, are always expressed as four characters. So, we send the command "**K00C0**".

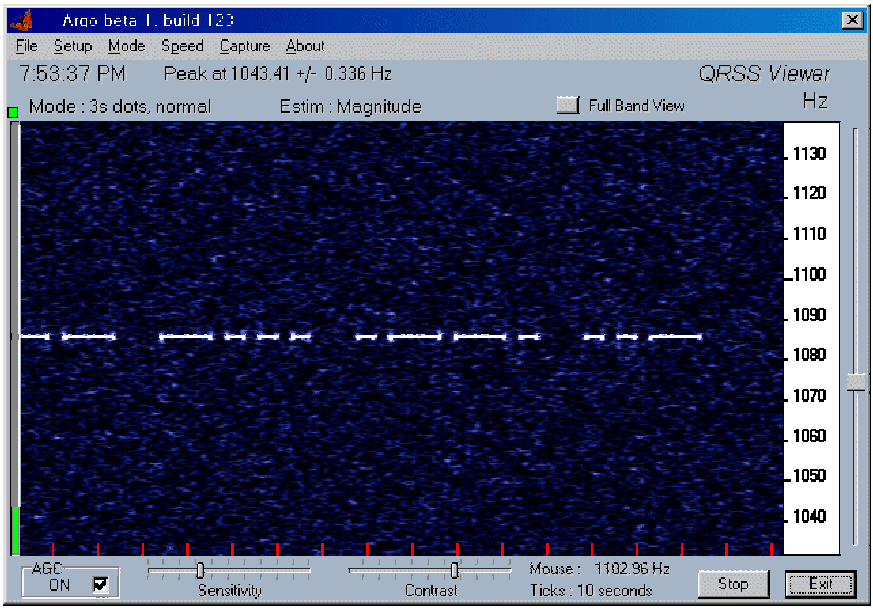

Fig. 3.1. QRSS with 3 second dots

Another example – to generate accurate JASON symbols (11.8 sec per element), use a timer value of 11.8 x 64 = 755 (**K02F3**). If your symbol clock is slightly off 64 Hz, you can easily tweak the value, since the timer has plenty of resolution.

For Morse code at normal speeds, we first need to work out the dot speed. The usual value for Morse dot speed is 1.2 x the speed in WPM. For example, for 12 WPM that's 12 x 1.2 = 14.4 Hz. There are two clocks per dot element, one for the dot, and one for the space, so we need to double the normal value used for Morse speed, so for 12 WPM we need a clock rate of 14.4 x 2 = 28.8 Hz. The nearest we can get to 28.8 Hz is 32 Hz (divide by 2, or a setting of "**K0001**", since the timer counts from zero), and the resulting speed will be 32 / 2.4 = 13.3 WPM.

The Power Amplifier

Good news – no mathematics here! The power amplifier uses a Philips TV audio amplifier, which has two special features – full power output to at least 250kHz, and a DC volume control with a very wide range. This design uses both of these features. See Fig. 4.4.

Input to the power amplifier (essentially a linear amplifier) is AC coupled from the digital to analog converter (D-A) output sine wave, and attenuated to the correct input level, about 1V p-p. The transmitter is keyed on and off, not by keying the amplifier, but by stopping the DDS generator, so the D-A output simply stops changing phase while the transmitter is off. This prevents "thump" which is caused by DC shifts. Keying transients are generated when using on-off keying of course, but have not been a problem. There are no transients when using FSK keying, as the Exciter generates phase continuous signals on frequency change.

The amplifier chip is designed to drive a balanced loudspeaker load, so for RF use, a balanced broadband transformer is used. Since the transformer DC resistance is very low, a blocking capacitor prevents DC imbalance in the push-pull output stage. The output has taps to allow it to drive loads from 50 Ohm down to fractions of an Ohm. All you need do is select the tap that gives the best output power. The chip is reasonably indestructible, and will shut down if overloaded. Experience has shown however that it does not like having high power from another transmitter pumped into its output!

The DC volume control input connects to the front panel power level potentiometer, so you can adjust the power smoothly from full output down to virtually nothing.

The power amplifier behaviour includes an effective low pass filter, so the power output and gain drop off quite quickly from 250 kHz upward. To use the Exciter at the upper end of its range you will need to use the output direct from the D-A converter, which is flat from DC to 500 kHz and beyond. There is sufficient output into a 50 Ohm load (50 mV RMS) to drive an NE612 type mixer directly.

4. Construction

Not many parts are required, and they should all be easily obtainable. Most components are non-critical, so any available brand and many different styles of device would be quite suitable. The parts list below specifies the parts used by the author, but in reality just about anything will do. The exceptions are:

- The micro controller U2 must be the AT90S2313-10PC (through-hole) or AT90S2313-10SC (surface mount).
- The amplifier U3 (TDA7052A) must be the "A" part (the TDA7052 has insufficient frequency response).
- The R-2R resistors R1-9 and R10-16 must be 1%.
- The frequency reference X1 (covered in Chapter 5).

	LF Exciter Parts List			
Cct Ref	**Description**	**Supplier**	**Part No**	**Comment**
Mechanical				
-	Circuit Board	Jaycar	HP9552	93 x 140mm spot board
-	Case	DSE	H2520	180 x 210 x 55mm, two identical parts, end panels supplied.
	Knob			To suit pot R40
	Panel Overlays, front and back			Laser printed and laminated
Sheet 1				
C1, C5	100nF mono ceramic 50V	DSE	R2001	
C2, C3	27 or 33pF NPO ceramic 50V	Jaycar	RC5317	C2, C3 can be adjusted for exact frequency trim
C4	100pF polystyrene 250V	DSE	R2819	
C6, C8	10uF 25V Al electro radial	DSE	R4315	
C9	10uF 16V Al electro	Jaycar	RE6066	
D1, D2	Diode, Si signal 1N4148	Jaycar	ZR1100	
J1	Header, 2.54mm pitch 2 x 5 pins (snapoff)	DSE	P2728	J-Tag programming connector
J2	RS232 connector, DB9F panel mount	Jaycar	PS0804	
R1 - R9	Resistor, metal film 0.5W 1% 2k00	Jaycar	RR0579	9 reqd
R10 - R16	Resistor, metal film 0.5W 1% 1k00	Jaycar	RR0572	7 reqd
R20, R24	Resistor carbon film 0.25W 5% 1k0	DSE	R1074	
R21, R23	Resistor carbon film 0.25W 5% 4k7	DSE	R1090	
R22	Resistor carbon film 0.25W 5% 100R	DSE	R1050	
R25, R26	Resistor carbon film 0.25W 5% 10k	DSE	R1098	
R29, R31	Resistor carbon film 0.25W 5% 10k	DSE	R1098	
U1	5V regulator, low power 78L05	DSE	Z6108	
U2	Microcontroller, AT90S2313-10PC	Atmel		Try Apex Electronics, Memec, or www.polykom.com.au
X1	Crystal, 10 to 16 MHz, 12 MHz preferred			Good quality high stability crystal, or use TCXO
Sheet 2				
C12, C14	0.47uF 50V mylar or polyester	DSE	R2130	
C13	220uF 25V Al electro radial	DSE	R4390	
C15	1nF 50V mylar or polyester	DSE	R2010	
D3	Diode, power 1N4002, 1N4004	Jaycar	ZR1004	
D4	LED 5mm red	DSE	Z4085	
D5	LED 5mm green	DSE	Z4087	
J3	Coaxial DC power connector, 2.1mm panel	Farnell	299-972	
J4, J5	BNC connector female panel screw thread	Jarcar	PS0658	
L1	Broadband transformer core	Jarcar	LO1234	Core TN25-15 or similar
L1	Broadband transformer windings			Pri 20T 0.7mm Sec 22T tapped from halfway every 2T
R27, R38	Resistor carbon film 0.25W 5% 1k0	DSE	R1074	
R35	Resistor carbon film 0.25W 5% 100k	DSE	R1124	
R36	Resistor carbon film 0.25W 5% 220k	DSE	R1132	
R37	Resistor carbon film 0.25W 5% 3k3	DSE	R1086	
R40	Pot, panel mount carbon film M0 linear	Jaycar	RP8524	
U3	Amplifier, BTL Audio 1W TDA7052A	Philips		Note - must be the "A" part
U4	Oscillator divider CMOS, HEF4060	Philips		

Table 4.1 The Parts List

The circuit board is a perforated fibre-board or glass-filled project board with copper spots, variously known as “spot board”, “matrix board”, or “vector board”. The glass-filled type with plated through holes is best. It is equally possible to construct the unit on stripe board (e.g. “Veroboard”), but the layout is unlikely to be as compact. It is best to choose a board and case that suit each other.

Layout is not critical, and observing the author’s layout in Fig. 4.1 and Fig. 4.2 will show that to a large extent the layout is suggested by the pin-outs of the devices.

Use IC sockets for all ICs except the regulator U1, as this allows for easy repair following accidents, and also reduces the risk of static damage during construction. The micro fits in a standard 0.3 in pitch 20 pin IC socket.

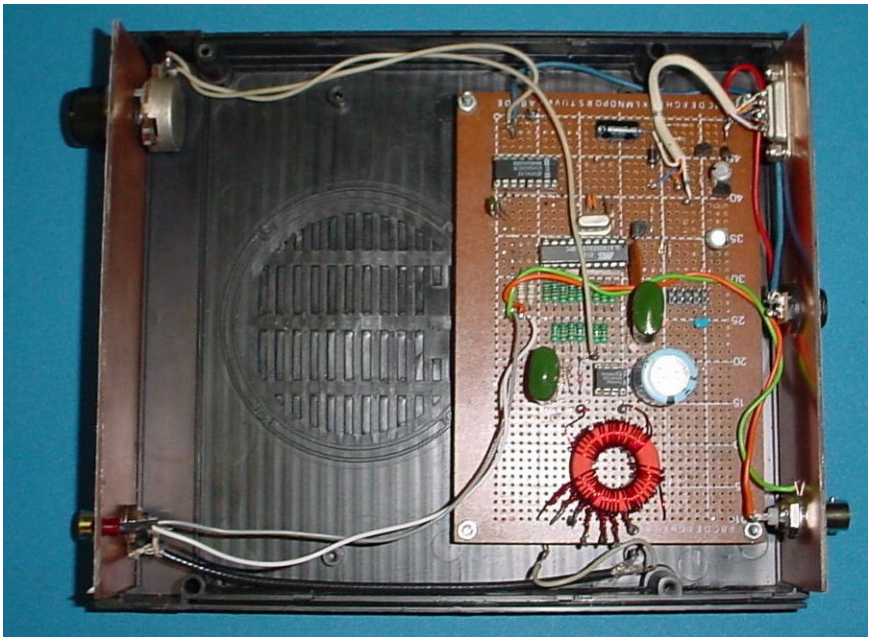

Fig. 4.1 Inside view of the LF Exciter

In this view the front panel is to the left, and supports the POWER control (top) and POWER LED, the RFOUT connector (bottom) and TX LED. The back panel to the right supports the RS232 communications connector (top), the power connector (centre) and the SYNC / TX output (bottom). There is plenty of room

remaining in this 210 x 180 x 55 mm case – perhaps for an LCD display controller board or a 10W power amplifier!

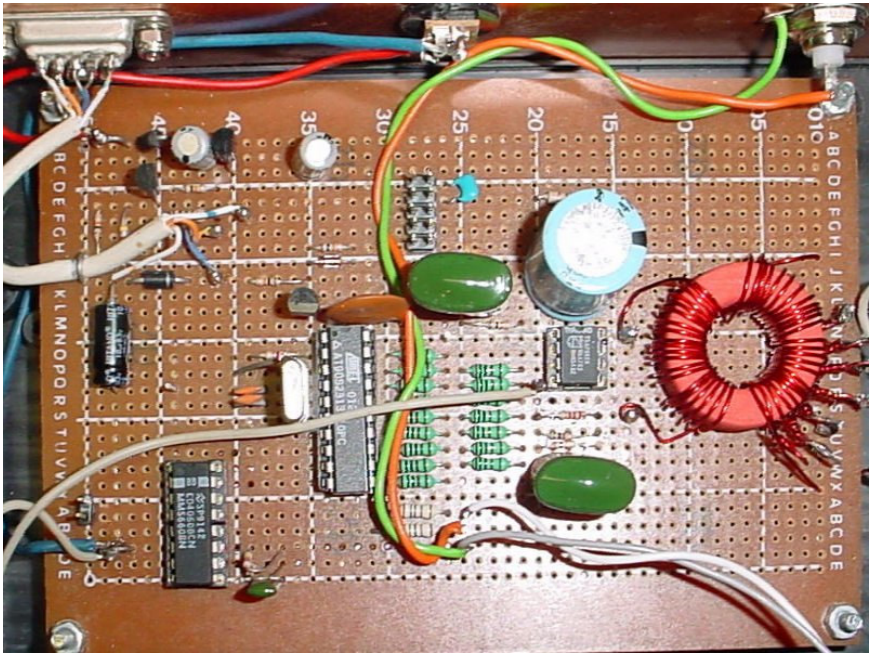

Fig. 4.2 Inside view of the LF Exciter

This view is a detail of the circuit board rotated, with power supply top left, RS232 circuit below it, and the symbol clock IC bottom left. You can clearly see the R-2R resistors laid out to the right of the micro controller, the crystal on the left of the micro, and the amplifier circuit to the far right. The amplifier output transformer with its taps is very obvious on the right. The transformer need not be this large – it is this size simply for convenience.

The connector above the micro is the programming header. It is point-to-point wired to the micro under the board, and is only used when you wish to change the firmware (NOT for recording beacon messages, which are loaded via the RS232 port).

Assembly

The order in which the board is assembled isn't especially critical, but do start with the IC sockets, and plan out where the R-2R resistors will fit (R1-9 and R10-16). The micro has a natural "IN" side and an "OUT" side in this design, which makes layout easy. In Fig. 4.3, pin 1 of U2 is bottom left.

In Fig. 4.3 the symbol clock generator U4 is at the bottom right, with its RC timing components C15, R35, R36 above. U4 pin 1 is bottom left on the chip. To the left at the bottom is the simple discrete transistor RS232 interface, wired to the connector on the back panel. Just above it is the small 5V regulator U1. No heat sink is required for the regulator.

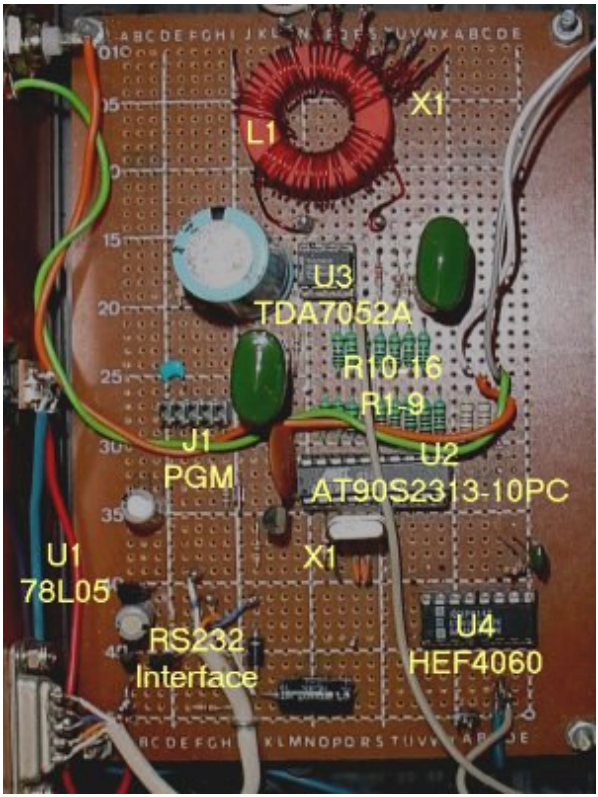

Fig. 4.3 The circuit board layout

Power amplifier U3 requires no heatsink. U3 is flanked by the capacitors C13 (on the left) and C14 (on the right). A short length of RG174 coax to the output socket is used to select the correct tap on output transformer L1 - simply soldered as needed.

Finally, the power busses are run under the board using tinned copper wire. The +12V DC input is at the bottom left in Fig. 4.3. (under the RS232 connector, and runs down from the rear panel connector halfway up on the left), and the 12V and 5V busses run up the left of the board. The 0V connection is visible at the bottom right of the board, where there' s a handy link for use as a test equipment ground, and the ground buss runs up the right of the board, with branches out to each device. The secondary of the output transformer is not grounded except at the front panel. I used single sided PC board for the front and back panels.

Checkout

Before applying power, check the board for shorts between the power rails, using an Ohm-meter. Apply power before fitting the socketed devices, and check for +5V, +12V and ground at the correct pins. Next fit the 4060 device (U4) and check that it oscillates (about 256 Hz on the test point on pin 7).

Now fit the micro controller U2 and check that it does not get warm. It won' t run, of course, until you have programmed it. Then program the device (see below), and using an oscilloscope or spectrum analyser, if you are lucky enough to have one of these, marvel at the signal at the point marked "DDS OUT". This is the synthesized LF signal, a 256 discrete point sine wave, sampled at 1.3 MHz. (When you are marvelling at the waveform, remember that all the 256 points do not appear in every sine wave cycle).

If you have used a 12.800 MHz crystal, the unit should run correctly as soon as it' s programmed. The output should be around 180 kHz, but the exact frequency won' t matter, as calibration takes place in the PC software. Other crystal frequencies will necessitate one small change to the micro controller stored values. This is covered below under programming. To check out the RS232 port, connect it up to a PC communications port set to 9600 bps (the Exciter' s default value). Run some terminal software such as Windows Terminal. When you apply power to the Exciter, the micro will send a little message saying:

<EXCITER D5>

or something similar. If the message is garbled, check that the computer data rate is correct, and if it is, go back and check the programming value for the data rate

in the micro controller (see the section on programming). To check that it is listening as well as talking, type **H** on the keyboard, and the Exciter should respond with its list of commands (the "Help" information).

Now you are in business! Fool around with the **F** command a little and try out different frequencies. At this point you might like to study the Serial Command List in the Appendix in order to understand how the unit is controlled.

If the output from the point labelled "DDS OUT" is not a reasonable sine wave, investigate the connections to the R-2R network resistors and check that the values are correct. Just one resistor wired incorrectly will destroy the nice sine wave. Having good resistors here makes all the difference to the performance.

Finally, when you are happy that all is well, and you have nice clean sine waves coming out, turn off the power, plug in the power amplifier U3, add a load to the output and with power applied and the carrier on (use the T command), adjust the taps for maximum power with the power level control at full output (maximum resistance). The power output should exceed 0.5 Watts from about 10 kHz to well over 250 kHz. If necessary, adjust the value of R37 so the amplifier just clips at full output. No damage will result if the load is not connected.

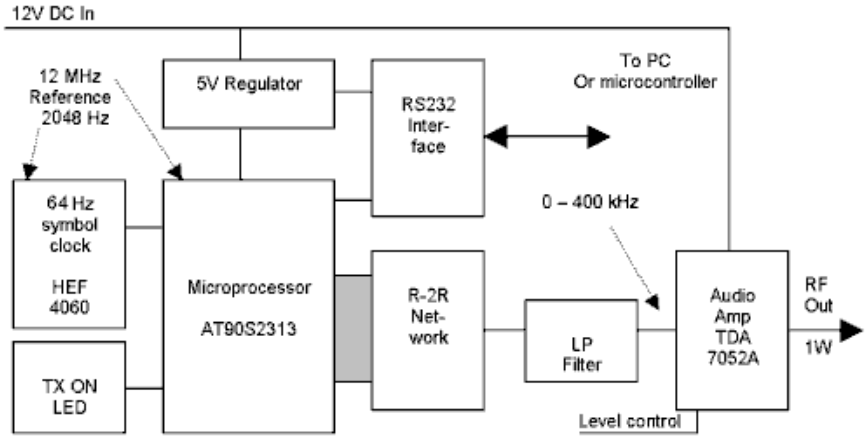

Fig. 4.4 The Exciter Block Diagram

5. Options

If you choose to use the Exciter for real-time QSOs, or as a signal generator, it will not really matter what crystal frequency you use, nor what is stored in the micro message memory used by the beacon generator. All you need to do is ensure that the communications rate is set for the crystal you use. You should use a good quality crystal, not a cheap micro crystal. If you do not wish to patch the micro controller setup, use the default frequency, 12.800 MHz.

If however, you wish to use the built-in beacon, you will have to decide what the main purpose of the unit is before you choose the crystal frequency, and whether you need a simple crystal, or a TCXO.

- For precise 0.1 Hz steps, use 15.0994944 Mhz
- For hexadecimal precise steps (1/16 Hz), use 9.437184 MHz
- For really accurate JASON transmissions, use 12.700 MHz
- For most other purposes, use 12.800 Mhz.

TCXO references are easily obtained at 12.800 MHz, and would be the most appropriate reference to use for a beacon operated at a remote site, or where very high stability is necessary, such as for QRSS120. A good quality crystal and 30 minutes warm-up will suffice for JASON.

Crystal Frequency

The crystal X1 can be any frequency from 10 MHz to 16 MHz, provided (a) the micro controller will operate that fast (all the ones tried so far have operated at 16 MHz), and (b) the frequency allows an accurate 9600 bps communications rate to be generated.

When choosing a crystal, remember that the maximum frequency that can be generated cleanly is about 1/30th of the crystal frequency – for example about 430 kHz using a 12.8 MHz crystal. Since the power amplifier response drops off above 250 kHz, this is only an issue if you plan to use the unit without the amplifier or with a lower frequency crystal.

Once the crystal is chosen, the communications rate to the PC, which depends on this crystal, has to be set. The best rate to use is 19200 bps, as this provides high performance with minimal difficulty with PC speed, lead lengths and so on. For some crystal frequencies lower speeds will be necessary, as the speed error increases with speed. For example, with a 10.0 MHz crystal, 9600 baud is the highest practical speed. The default value is 9600 bps with a 12.8 MHz crystal.

In order to operate the unit with I2PHD and IK2CZL's JASON, you must use 9600 bps, so if you contemplate using JASON at any stage, stick with 9600 bps. The calculation for the micro controller UART divider goes like this:

UART Divider = ($f_{crystal}$ / UART Speed x 16) -1 …[4]

For example, the value for 9600 bps using a 12.8 MHz crystal is (12800000 / 9600 x 16) –1 = 82.333_{10}. Choose the nearest integer value (82_{10} or 52_{HEX}). To check the speed error, recalculate back the other way, using the following formula:

UART Speed = $f_{crystal}$ / (16 x (UART Divider +1)) …[5]

Using the value 82, this gives a speed of 9638.5 bps, about +0.4% in error. Any error less than 0.5% is acceptable. If the error is higher, use a lower communications speed, or change the crystal.

Beacon Message

The beacon messages are stored in compiled form, in other words, in a form easily understood by the micro controller. There are several different formats used:

- Morse in "Murphy" format (for ASK, QRSS, FSK and DFSK Morse)
- Bit maps (eight bits high) for Feld-Hell and MT-Hell modes
- JASON format (frequency offset) for JASON IFK
- A special embedded command format

The format for Morse is the widely used "Murphy" format, where dots are represented by "0", dashes by "1" and a "1" follows the last element. Inter-element and inter-character spaces are generated automatically. A complete list of Morse characters in this format is given in the Appendix. One character is sent per byte, and the format applies to all Morse-related modes; ASK, FSK and DFSK.

There are two graphics modes, Feld-Hell (amplitude vs. time) and MT-Hell (Frequency vs. time). The same Hell graphics font can be used for both modes, but they way the characters are transmitted differs. In Feld-Hell, just one frequency is used, and the dots are either transmitted, or not transmitted, with very precise timing. The receiving software has to scan vertically and horizontally, displaying signal strength as brightness. Only slow Feld-Hell is possible with the Exciter, and special receiving software will be required. In the example in Fig. 5.1, the text is transmitted at 32 dots per second (**K0001**), and the receiving software scans at four columns per second. The signal was very weak, hence the noise on

the signal. Feld-Hell is not an especially sensitive mode, but is quite tolerant of man-made noise.

Fig. 5.1. Weak Feld-Hell transmitted by the Exciter

In MT-Hell, each dot in the column is transmitted on a different frequency, and the signal is received with a spectrogram program such as ARGO (see Fig. 5.2). Because there is no receiver column scanning, timing is very relaxed. MT-Hell is not as sensitive as many of the modes received with ARGO, but is a lot more fun!

Hell format consists of eight dots per byte, which creates one column. The LSB bit of the byte represents the lowest dot in the column. It is normal to use five or seven active dots per column – for lower case use the lowest two dots for "descenders", as on "y" and "q". There will almost always be two unused dots at the top of the column (these can be used for graphics). Each character will take several bytes to express – from three for an "I" or "T" to five for a "W" and more for graphics. If each column is repeated several times the text slows, but readability improves. In Fig. 5.1, four identical bytes were transmitted per column, and the font used five active dots per column.

A simple font (upper case only), suitable for Feld-Hell and MT-Hell, is included in the Appendix. MT-Hell can also be transmitted live from the EXC.EXE program, which uses advanced dot timing for higher speed and seven dots per column for better character shape (see Fig. 5.2).

There is no "compiler" for JASON, and there is not even a code table, since the data transmitted is incremental, and depends on what was transmitted before! The best way to work out what needs to be loaded into the beacon for JASON mode is to use the I2PHD / IK2CZL JASON V0.94 software, set it in "ZL1BPU format", and have it transmit to another PC running a terminal program. The software will transmit "**T**", followed by a series of "**Axx**" commands. Type in the

message you need, set the software in transmit mode, and write down the “xx” values that appear on the other PC. These are the values to program into the beacon (just leave off the “A”).

Control functions can be added to the beacon message. For example, you can indicate the end of the message, a change of keying mode, and changes of carrier frequency, FSK shift, and baud rate (keying speed). You can even set outputs for controlling external equipment. These commands always start with a hexadecimal “F”. The beacon commands are listed in the Appendix. Note that they are NOT the same as the KISS protocol commands used to control the unit via the serial port, although similar in effect.

The compiler provided by the author (MAKEBCN.EXE) can compile only Morse messages and embedded commands. MAKEBCN is very handy, because as well as generating the beacon message, it downloads it to the Exciter, making the process really easy. If you wish to add other modes to the message, you will need to work out the complete message on paper first, and laboriously enter it with the **B** command.

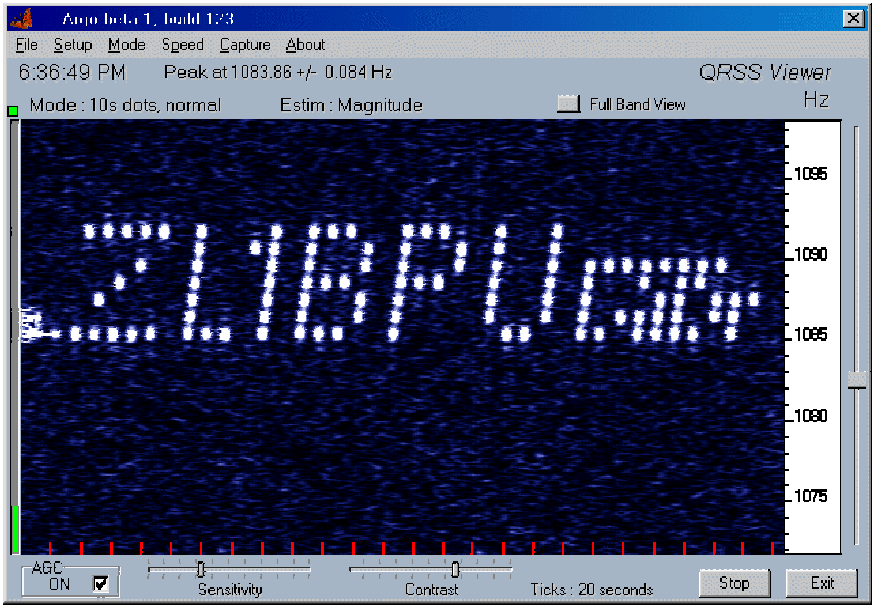

Fig. 5.2 An ARGO picture of the LF Exciter sending MT-Hell

Controlling the Outside World

There are three controllable outputs, PD2, PD3, and PD4. These can be controlled by direct serial command (the **P** or PORT command), or from the beacon script. This latter feature is really valuable, because it allows you to operate at different power levels, on different antennas, or even on different frequencies (with switched antenna tuner settings).

The outputs are CMOS level, and can drive LEDs directly. (The outputs can sink or source 20mA - see the circuitry for the TX LED in the schematic). In order to operate relays, a driver is required. The circuit below is recommended.

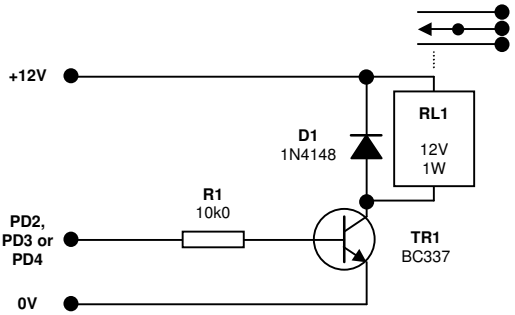

Fig. 5.3 Suitable relay driver

Controlling the outputs is very simple, but be aware that the PORT command and the beacon script command are binary, and control all the outputs at once, so you have to check all the bits you need to decide what number to send. The simplest application (controlling one relay connected to PD2) requires only sending two commands: **P0** (or FB00) for off and **P1** (or FB01) for on.

6. Programming and Setup

This procedure assumes you use the same programming tool as the author[10]. Since this is free, it's a reasonable assumption! If you use other tools, you will obviously need to be familiar with them.

You will need:

- The ATMEL ISP AVR programmer, ISP.EXE V2.65 or similar.
- A programming cable.[11]
- The EXCITER firmware, EXC_Dnn.HEX and EXC_Dnn.EEP, available from the author ("nn" is the firmware version, currently 5A).
- A PC running Windows 3.1, 95 or 98 to run the programming software.

The .EEP file is not strictly necessary, although it includes default values for 12.8 MHz crystal operation, and a demonstration beacon message (listed in the Appendix).

Connect the DB25 end of the cable to the PC printer (parallel) port, and the 10-way header to the Exciter programming header. Make sure the cable is the right way round on the header – no damage will be done if it is wrong, but nothing will work.

If you have two computers, or can run two applications at once, connect up the Exciter serial port to a PC serial port. You will need a "modem" serial cable (pin-to-pin with no crossover), NOT a "null modem" cable. Run a Terminal Emulation program (Windows 3.1 Terminal is ideal), setting it to the correct port, and the speed you intend to use. This will allow you to monitor progress during programming.

Run the ISP software ISP.EXE. There is also a DOS version; the most recent version supports 32 bit operating systems. The author recommends V2.65, which will work on a 486 PC running Windows 3.1, as well as later computers.[12] Make a project (from the menu, **Project/New Project**), selecting AT90S2313 as the device.

---

[10] ISP.EXE V2.65, available from:
user.cs.tu-berlin.de/~sirhenri/sides/up_avr/avrisp.zip.
The latest version is available from www.atmel.com/atmel/products/prod203.htm

[11] See www.qsl.net/zl1bpu/micro/ for a suitable cable.

[12] There have been problems with port timeout errors with later versions.

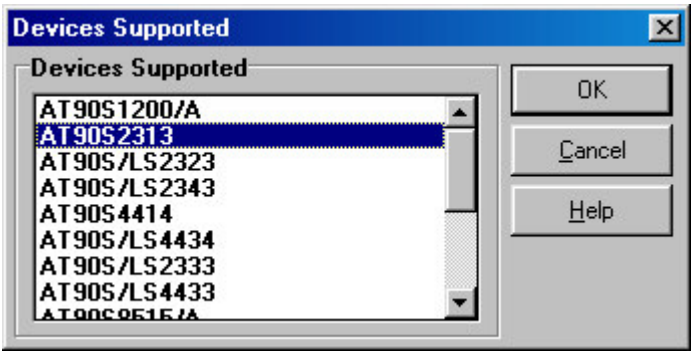

Fig. 6.1 Select the device

Press OK and then on the “Manager” tab of the next screen, give the project a title, and then from the main program menu select **Project/Save Project** to save the project file.

The next step is to check for communications with the micro. Select **Options/Change Printer Port…** and you should see a "Port available" message. If all is well, select **CANCEL**. If not, try changing the port in case it isn' t LPT1.

Next, select **Program/Read EEPROM**, which should be something safe to do (it won’t matter what you do if the micro is already blank). If all is well, the data from the micro will fill the “Data EEPROM Memory” window:

Fig. 6.2 The “Data EEPROM Memory” window

You can see this window by clicking on it, or from the menu **Window/EEPROM Data Memory**.

If the cable is faulty or not connected, you are likely to see an error message like:

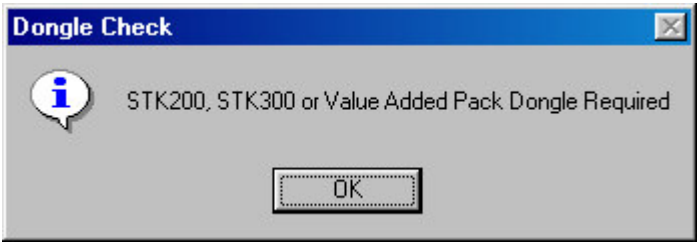

Fig. 6.3 Oh dear, no / faulty programming cable!

If the cable is connected, but there is a fault at the Exciter (perhaps the programming header is wired incorrectly, is plugged in backwards, or there is no power applied to the Exciter), you will see a different message:

Fig. 6.4 No communications with device to be programmed

This message also appears if you have selected the wrong device in the Project Manager.

If everything worked fine, you are in a position to try programming. The software implicitly loads files into the current window, so in order to load the executable firmware, select the "Program Memory" window, by clicking on it, or from the menu **Window/Program Memory**.

Next select **File/Load…** and locate and select the executable file (EXC_Dnn.HEX). The file is in "Intel HEX" format. You are now ready to program the device.

From the menu, select **Program/Erase** to erase the device, then **Program/Program Device** to program it. It should take 10 – 20 seconds, depending on the speed of your computer. If the device is faulty or not blank, it may take longer and will report errors.

At this point the micro starts talking to the serial port, and you should see a line of garble on the Terminal screen. If you wish to try the demonstration message, select the ”EEPROM Data Memory” window (menu **Window/EEPROM Data Memory**). Select **File/Load…** and locate and select the EEPROM file supplied (EXC_Dnn.EEP).

From the menu, select **Program/Program EEPROM** to load the message. It should take just a couple of seconds. If you are using a 12.7 or 12.8 Mhz crystal, you will immediately see the Exciter start-up message appear on the Terminal screen a couple of times.

The final step applies if you are using a different crystal frequency, or are using 12.7 / 12.8 MHz, but don’t want to load the demonstration message. The ”EEPROM Data Memory” window will probably show all values as “FF”, so leave them as they are (it won’t matter what they contain). If the EEPROM in the device is blank, reading it will restore all values in the window to ‘FF”. Next, count along to the 8th byte in the first row, and change it to your required data rate (see Crystal frequency” in Chapter 5). The example below shows setting to 9600 bps with a 12.8 MHz crystal:

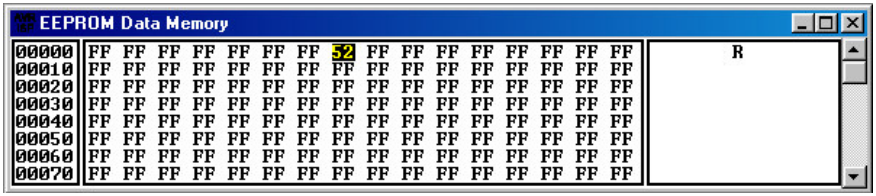

Fig. 6.5 Patching the Serial Comms data rate

Then program the EEPROM as described above. If the message on the Terminal is still garbled, check your maths again, and check the Terminal data rate. Table 6.1 gives the value to load for a range of common crystals and communication speeds of 9600 and 19200 bps. Where no value is given for 19200 bps, the error would be too great for reliable communication (> 0.5%).

Crystal MHz	9600 bps	19200 bps
9.437184	0x3D	-
10.000	0x40	-
12.000	0x4D	0x26
12.700	0x52	-
12.800	0x52	-
15.0994944	0x62	0x30

Table 6.1 Typical data rate values

7. Operation

So now you should have a working Exciter! Check that the serial communications work both ways, by typing "**H**" on the Terminal keyboard: you should see a list of commands. Type "**R**" and the unit will respond with the current settings. (The **P**, **T**, and **X** settings are not reported).

Kiss Commands

While you have the Terminal program operating with the Exciter, you should explore KISS mode. Sure, you don't need to operate the Exciter in this way (unless you actually enjoy hexadecimal numbers!), any more than you need to use KISS mode with a packet radio TNC, but it is useful to have a play so that you understand how the ***ZL1BPU LF Exciter*** communicates. All the computer programs that control the Exciter use these commands.

There are twelve commands, which are summarized in the Appendix. Let's look at them in detail. To start with, almost all of them provide a response – the only exceptions are the "**T**" and "**X**" commands. Response is omitted here to allow for clean sending at the highest speeds, for example ASK Morse at 30 WPM. The computer programs can (and do) generally ignore the responses, although they can be useful for checking that the command has been understood. Some programs simply display the response without error checking.

The commands are all a single letter (upper or lower case), followed by zero to six other letters or numbers. These are always hexadecimal, i.e. 0 - 9 or A – F (or a – f). No other characters are accepted or needed, and no "Enter" is needed, although "Enter" (CR) and "Linefeed" (LF) are tolerated. This is important, because it allows the Exciter to operate over a packet-radio link.

Unexpected or misunderstood information results in the Exciter sending a "?" response. This occurs with "Enter" as well, but has no effect on operation, except that a partially entered command will be abandoned. "Enter" is therefore useful for terminating commands entered in error.

Before the commands are studied in detail, it is important to understand the concept of ***resolution steps***. As discussed earlier, the basic resolution of the ***ZL1BPU LF Exciter*** is dependent on the crystal used. It's typically about 1/12 Hz, and when any of the following commands that relate to frequency are used, it is steps of this size, the ***resolution steps***, that are implied. Every value entered represents a number of resolution steps, so the frequency or frequency increment that results will be the number entered multiplied by the resolution.

A ADD

Syntax: **Axx**

ADD an offset of xx resolution steps, where xx is "00" to "FF" in two hexadecimal characters. The offset is not cumulative, so adding another offset adds to the base frequency, not the current offset frequency.

This command can be used in conjunction with **T** and **X** to generate ASK, FSK and MFSK transmissions in a wide variety of modes. The command has no useful effect on ASK modes such as MODE 1 (ASK Morse) and MODE 5 (ASK HELL), or on MODE 6 (MFSK/IFK mode) where offsets are set directly from data.

In Sweep Generator mode, ADD sets the sweep dwell time (time per step). The resolution is about 1/12 milliseconds and range is under 1ms (1ms = **A**0C) to over 20ms (20ms = **A**F8). See the comments about Sweep Generator mode under the **K** command.

B BEACON

Syntax: **B xx xx xx xx … FF** ~

Enter a new BEACON message. The command is used to enter beacon data, representing text and graphics. This becomes the beacon “script”, and the content of the data depends on the mode to be transmitted. Up to 120 bytes of data and commands can be entered.

The data consists of HEX-ASCII character pairs. <CR> and <SPACE> are permitted but not stored, and the last character pair must be "FF". Data entry mode is terminated by the tilde character " ~ ". The message is permanently stored and the unit is reset on exit.

Four types of information are used to “create” a message. These are (a) Morse text, coded in “Murphy” format; (b) bit map information for graphics modes; (c) differential frequency offset data for JASON mode, or offset data for MFSK and FSK modes; and (d) embedded script commands, which allow the beacon to change mode and various other parameters during the message. These four types of information are covered later in the chapter.

F FREQUENCY

Syntax: **Fhhmmll**

Set the Exciter transmit FREQUENCY to this value times the resolution. The value "hhmmll" is a 24 bit binary number expressed as six hexadecimal characters, "000000" to "FFFFFF". The useful range is 000001 (about 1/12 Hz) to 480000 (about 400 kHz). Values above this will cause the signal to have bad aliases (spurious signals), or have a "negative" frequency.

In sweep mode, FREQUENCY sets the frequency of the first step, the one that corresponds to the SYNC signal. Following steps will normally be higher in frequency. If the FREQUENCY value used is negative (1000000_{HEX} – required value), the sweep steps will be negative, i.e. down in frequency rather than up.

H HELP

Syntax: **H**

The Exciter sends a simple HELP message, listing all the available KISS mode commands. Once the message has been sent, the Exciter resets, so the start-up message also appears.

The syntax for each command is given in the list.

K KEY

Syntax: **Knnnn**

The KEY command sets the beacon mode keying rate (baud rate). The value "nnnn" is hexadecimal, "0000" to "FFFF". The beacon mode uses its own baud rate generator, the “symbol clock”. This clock is generated externally to the micro, and is normally 64 Hz (15.625 ms).

The value given for the **K** command is the number of 15.625 ms periods required per symbol, so the resolution is also 15.625ms. The formula for calculating the required value is:

KEY = (64 / baud rate) - 1 …[6]

Hence **K**0001, the highest speed possible, gives a baud rate of 32 baud, and can be used for 26 WPM Morse. 3 sec dot QRSS requires **K**00C0 (10 bits/minute)

and JASON (11.8 sec symbols) **K**02F0. By the way, to calculate the baud rate for Morse, use this formula:

BAUD = 1.2 x WPM …[7]

This formula is *not the same* as the normal formula used for Morse speed (2.4 x dot speed) because there are *two symbol events per dot element* (the dot and following space).

So, for the Exciter, sending Morse, you can use the following formula:

KEY = 53.3 / WPM …[8]

For QRSS use this formula:

KEY = 53.3 x QRSS (secs) …[9]

Of course, you then have to convert the value to hexadecimal to send it to the Exciter.

Here are some typical settings:

Morse, 12 WPM	K0004
Morse, 5 WPM	K000B
QRSS3	K00C0
QRSS10	K0215
QRSS120	K1900

Table 7.1. Speed settings

In Sweep Generator mode, **K** sets the frequency increment per step. The units are resolution steps (as for the **F** command), but range is limited to from **K**0000 to **K**FFFF (over 5 kHz/step).

It may not seem logical to use this command for the frequency step size, but the most obvious command to use (A) has only 8-bit resolution, offering only 0 – 255 resolution steps (21 Hz). This would have seriously limited the usefulness of the sweeper. In effect, the functions of these two commands have been swapped in Sweep Generator mode in order to improve the step resolution. Sweep dwell time does not need high resolution.

M MODE

Syntax: **Mn**

This command tells the Exciter to change beacon MODE. If the mode is zero, the beacon is turned off, otherwise the beacon is turned on in one of the six operating modes. The modes are:

M0 **BEACON OFF** In this mode the Exciter is a CW generator that can be controlled in frequency by the **A** and **F** commands, on and off by the **T** and **X** commands, and into Sweep Generator mode with the **W** command.

M1 **ASK BEACON** The Exciter transmits ASK (on-off) keying, using Morse. The message sent is contained in the beacon script, which can also modify the behaviour and even the mode of the beacon. In ASK mode the keying speed is controlled by the **K** command. This mode uses the "Murphy" interpreter to convert script data to Morse.

M2 **FSK BEACON** The Exciter transmits FSK (frequency shift) keying, using Morse. The message sent is contained in the beacon script, which can also modify the behaviour and even the mode of the beacon. In FSK mode the keying speed is controlled by the **K** command. As with other FSK modes, the **F** command sets the idle frequency, while the **A** command sets the "key down" FSK shift. The key down shift is up in frequency unless the carrier frequency is negative. This mode uses the "Murphy" interpreter to convert script data to Morse.

M3 **DFSK BEACON** The Exciter transmits DFSK (often called dual frequency CW) keying, using Morse. The message sent is contained in the beacon script, and this mode uses the "Murphy" interpreter to convert script data to Morse. The keying speed is controlled by the **K** command, while the **A** command sets the FSK offset. Dots are transmitted on the nominal frequency (set by the **F** command), while dashes are shifted up in frequency (or down if the carrier frequency is negative). In this mode intra-character gaps are omitted if the following element is different, and all elements are of dot length.

M4 **MFSK BEACON** The Exciter transmits MFSK (multi-frequency) keying, using graphics. This is in effect a Sequential Multi-tone Hellschreiber mode. The message sent is contained in the beacon script, and must be coded using a graphics image, or text in graphics format. A suitable font is included in the Appendix. Each script byte represents one column of

graphical data, and the data is sent LSB first, representing the lowest frequency dot (set by the **F** command). If any bit is "1", it is transmitted with full dot duration (set by the **K** command), while if the bit is "0", silence is transmitted for the duration of one half dot. Bits following the LSB are transmitted at increasing frequency (step size set by the **A** command), or decreasing frequency if the carrier frequency is negative.

M5 **HELL BEACON** The Exciter transmits ASK keying, using graphics. The message sent is contained in the beacon script, and must be coded using a graphics image, or text in graphics format. A suitable font is included in the Appendix. Each script byte represents one column of graphical data, and the data is sent LSB first. All dots are sent at the same frequency (set by the **F** command). If any bit is "1", it is transmitted with full dot duration (set by the **K** command), while if the bit is "0" silence is transmitted for the duration of one full dot. This mode is completely compatible with slow Hellschreiber. "DX Mode" can be achieved by storing each column of data in the script two or more times.

M6 **MFSK/IFK BEACON** The Exciter transmits MFSK (multi-frequency) or IFK (incremental FSK) keying, using data. The message sent is contained in the beacon script, and must be coded using the required data format. For example, IFK encoded using the JASON technique. Each script byte represents one transmitted symbol, and the data is sent as an offset frequency from the nominal carrier frequency, (set by the **F** command). Appropriate values are from 0x00 to 0xEF, and operate in resolution steps.[13] Symbols are transmitted at increasing frequency (similar to setting an offset with the **A** command), or decreasing frequency if the carrier frequency is negative.

Each of the modes interpret the script data in a different way, so make sure you select the correct mode for the beacon data, or rubbish may result. The best way to achieve this is to include the appropriate mode command in the message.

[13] Therefore for JASON, there should be *three* resolution steps per JASON IFK step. If JASON is operated correctly in serial port mode with the "ZL1BPU protocol" enabled, the IFK commands will be multiplied by three in the PC software, and the Exciter will send perfect JASON. If these commands are captured, they can be loaded into the beacon script.

P PORT

Syntax: **Pp**

The command turns on the output ports PD2, PD3 and PD4 according to the content of **p**. Only the lowest three bits of **p** are significant, so although values higher than 7 will work, it is best to stay within the range 0 – 7.

Outputs are standard CMOS logic outputs and can be used to control many things – for example transmitter power level, antenna switching or front panel LEDs.

PD2 corresponds to bit 0, PD3 to bit 1, and PD4 to bit 2. Use the following table to set the outputs:

P	binary	PD4	PD3	PD2
0	000	OFF	OFF	OFF
1	001	OFF	OFF	ON
2	010	OFF	ON	OFF
3	011	OFF	ON	ON
4	100	ON	OFF	OFF
5	101	ON	OFF	ON
6	110	ON	ON	OFF
7	111	ON	ON	ON

Table 7.2 Output port control

R REPORT

Syntax: **R**

A simple one-line REPORT of the status of the Exciter. It lists the current values for the **A**, **K**, **M**, **W**, and **F** commands.

S STORE

Syntax: **S**

This command is used to STORE operating parameters in EEPROM so they are retained over power-down. When power is restored to the Exciter, these default values are loaded before operation begins. The values stored are FREQUENCY, OFFSET, MODE and KEY.

If the MODE stored is **M0**, the Exciter will start as a conventional CW exciter, with the key down. If any other MODE is stored, the Exciter will enter a beacon mode and start executing the beacon script when power is applied.

The Sweep Generator **W** setting is not stored, so the unit cannot default to Sweep Generator mode. **W** defaults to zero at power-up. The other parameters used by sweep mode are stored.

T TX

Syntax: **T**

The TX or TRANSMIT command turns the transmitter on if it is off. It has no effect on any other parameter. The Exciter does not echo this command.

TX is used in conjunction with **X** (RX) to control the Exciter externally. For example, to send ASK Morse, a series of “T X T X T X T X “ commands (correctly timed, of course) can be used to send perfect Morse up to at least 35 WPM.

W WIDTH

Syntax: **Wmm**

Sets the WIDTH of the Sweep Generator mode by setting the number of sweep steps. The value "mm**"** is two hexadecimal characters representing a value "00" to "FF", (0 to 255), although widths of greater than 30 or so steps aren’t generally useful for conventional sweep generation.

The start frequency for the sweep is set by the **F** command, sweep dwell time by the **A** command, and the sweep step size by the **K** command.

Typical values are **W0A** for 10 steps and **W14** for 20 steps, giving one or two steps per scale division on the oscilloscope screen. Small values of **W** can be used to simulate FSK and MFSK modes. For example, audio frequency 50 baud 170 shift RTTY with 2125 Hz MARK can be simulated with **W**02 **A**F8 **K**07E5 **F**0062B1.

W01 has no effect. **W00** turns sweep mode off.

X RX

Syntax: **X**

The RX or RECEIVE command turns the transmitter off if it is on. It has no effect on any other parameter. The Exciter does not echo this command.

RX is used in conjunction with **T** (TX) to control the Exciter externally.

Beacon Mode

In beacon mode, the data and commands are received internally, from the *beacon script*, which is stored in the EEPROM memory. Operation is exactly the same as sending external commands, except that the data is processed internally. The beacon in the ***ZL1BPU LF Exciter*** is very powerful. The *command interpreter* that receives the data from the stored message, and interprets it, can handle coded Morse, graphics and data, in addition to a range of commands – in fact just about anything but text. The message is in a *compiled* form, designed for easy interpretation by the micro, and how the data is interpreted depends on the current mode. The PC program MAKEBCN is capable of compiling and recording beacon messages in the required format, at least for Morse modes. For more information on the beacon modes, see Chapter 5, "Beacon Message".

There are six beacon mode commands that can be used in the message script. In general, these work in the same way as their KISS command counterparts. The differences are that they are executed automatically, and the syntax is different.

The commands can be in any order, and can be distributed throughout the message. No transmission time is lost when commands are processed, but of course they do use up message space. Here is an example that sets the equivalent to KISS commands **A08 K00C0 M2**:

FD 08 FE 00 C0 F2

The commands consist of a *command byte*, which will contain the command, followed by zero to three *data bytes*. The commands always start with "F", and so values of message data starting with "F" are not permitted where they might be interpreted as a command (this has not proved to be a disadvantage). Commands that are misunderstood are ignored, but be aware that following data belonging to a misunderstood command may well be processed as other commands or message data. There is great potential for getting strange results, but a quick check of the data from the point where the message goes crazy will usually pin-point the problem.

There is a list of sample script data in the Appendix. The script commands are described below in hexadecimal format, where two hex characters represent each byte.

MODE

Syntax: **Fn**

This is a family of six commands, **F1** to **F6**. These commands set the beacon mode to **n**, where **n** has the same meaning as the KISS MODE command. **F0** is not used, as this would turn off the beacon and defeat further script processing.

Using these commands, you can switch modes in the middle of a beacon message. It will generally be necessary to follow mode commands with changes to SHIFT and SPEED.

PORT

Syntax: **FB pp**

Sets the three output ports, PD2, PD3 and PD4, according to the content of **pp**. The format is the same as the KISS PORT command, i.e. only the three least significant bits are useful, and they control the same outputs.

FREQ

Syntax: **FC hh mm ll**

Sets the frequency to $hhmml l_{HEX}$ resolution steps above zero, in other words the format, range and resolution are the same as the FREQUENCY command. This command allows you to QSY in the middle of a message, for example to create a dual-frequency beacon, or to reverse the direction of keying shift by switching to negative frequency.

SHIFT

Syntax: **FD nn**

Sets the carrier frequency offset to nn resolution steps. This command works in the same way as the KISS ADD command, i.e. the offsets are based on the nominal carrier frequency, and do not accumulate as additional shifts are processed.

This command has no effect on modes 1, 5 and 6. A shift of 1 Hz is achieved with **FD 0C**.

SPEED

Syntax: **FE nn nn**

Sets the beacon keying speed, i.e. the speed at which symbols are transmitted. The value of "nnnn" can be from 0000 to FFFF. The baud rate is 64/nnnn, and the definition and use is exactly the same as the KISS KEY command.

END

Syntax: **FF**

Marks the end of the beacon script. The next data processed will be at the start of the script.

Message Recording

Recording a beacon script (message) is simplicity itself. However, as you will have discovered from the command list and the fact that the message is pre-coded, generating the message isn't quite so straightforward!

To generate messages manually (which is the only way at present for graphics messages), it is best to work out and write down the message first. Simply write the bytes as pairs of characters, separated by spaces. Don't put too many on each line (say eight pairs), as it is easy to lose your place when typing them in.

Messages that contain only commands and Morse can be made and downloaded directly by MAKEBCN, but other modes require more ingenuity. You can store the message in a file - then download the file you create using a Terminal program, but there must be some way to send the message slowly (such as a "pacing" option), as the micro is not capable of receiving and storing messages any faster than about 100 bytes/second (the EEPROM write process is slow).

Enter the commands as described in the section above, and the Morse data from the table in the Appendix. Graphics for MT-Hell (Mode 4) and Feld-Hell (Mode 5) come from the font table in the Appendix. Each character consists of several bytes, and note that some characters require more bytes than others, because the font is proportional (different widths for different characters).

The best way to work out a JASON message (Mode 6) is to use the JASON software by I2PHD and IK2CZL to create it for you. The process was discussed under "Beacon Message" in Chapter 5. You can also do the same for Hell modes, using EXC.EXE.

You can make a multi-mode message by simply putting sections of message together one after another, along with the necessary mode change commands. With the correct commands, any mode can follow any other mode with no interaction.

One way or another, if your message contains more than just Morse and commands, you will have to convert it to HEX and write it down. Then, with the message written out in HEX in front of you, and the Exciter connected to a Terminal program and operating, press "**B**" and start entering your message. Once the message has been successfully entered, enter "FF" as the last byte, and send "~" (tilde) to tell the recording process to quit.

You will very likely make a mess of it, perhaps missing out one character, or entering one twice, so be patient and try again. To quit the message, enter "~" and try again. When you abort a partially written message, the micro will reset, so you need to press "**B**" again to start over. You can type spaces or "ENTER" in the message, and they will be ignored. There is no way to see what you have recorded, without sending the message via the beacon, or using the ATMEL programming tools to peep inside the micro.

Sweep Generator Mode

The Sweep Generator started life as a simple fixed rate 20 step sweeper (as in the Signal Generator), but it became apparent that at the cost of some minor complexity, considerable versatility could be attained.

The Sweep Generator has four programmable properties – the start frequency, the number of sweep steps (it has discrete steps, not a linear sweep), the frequency step size, and the dwell time, the time spent on each step. The first step is always the lowest frequency (unless you set the frequency to be negative). Three of the Sweep generator commands are normal mode commands recycled, while the third is unique to Sweep mode. The Sweep Generator commands are described briefly below, and in more detail earlier in the Chapter under KISS Mode.

A ADD

Sets the dwell time (time per step).

F FREQUENCY

Sets the sweep start frequency, exactly as in normal mode.

K KEY

Sets the frequency step size, in resolution steps.

W WIDTH

Sets the number of discrete steps in the sweep, or turns off the Sweep mode.

Perhaps one of the most useful features of this Exciter, Sweep Generator mode allows you to sweep antennas, filters and antenna tuners for correct adjustment. When used with a Return Loss Bridge (like an SWR meter with no meter), or an Impedance Bridge, you can plot the performance of your antenna and feed system very simply.

Unlike a conventional sweep generator, this unit generates discrete frequency steps, and they are also very precise in frequency.

There can be from 2 to 255 sweep steps, although Sweep mode is best used with 20 discrete frequency steps (**W14**).

The sweeping occurs continuously, with each frequency generated for the set dwell time, typically 5ms (**A3E**). The TX LED and SYNC output are turned on for the duration of the first step. Using an oscilloscope, trigger the sweep off the SYNC output, and with the sweep set to 10ms/div, the 20 steps will nicely line up across the screen. **A1F** works well with a 5ms/div time base.

You can also make the sweep generator sweep backwards, by setting a negative start frequency, although why one would need to do this isn' t obvious! It will also sweep through zero, but once again, that' s not much use. For audio sweeping, try setting the start frequency to zero (**F000000**), and use a narrow sweep increment (e.g. **K0498** for 100 Hz/step). So, here's an example – sweep from zero to 2000 Hz in 20 steps, with 5ms on each step:

A3E F0000 K0498 W14

You can also write a PC program to control the Sweep Generator however you wish. All the sweep generator parameters can be adjusted while the sweep generator is running. There might be the odd hesitation, if for example you set it

to 10 steps while it is already on step 19! Remember also that the generator will always stop while you actually enter the commands.

Another hint – there is a short transient every time the frequency is stepped, so the fewer steps and the slower the step rate, the cleaner the results will be.

For high resolution sweeping of very narrow filters (such as crystal filters) slow sweep rates are necessary. This in turn may make use of a storage oscilloscope or other means of recording the signal obigatory. The resolution and extreme stability of the ***ZL1BPU LF Exciter*** Sweep Generator are perfect for sweeping such difficult applications, something most conventional (even expensive) sweep generators struggle to achieve.

8. Software

Micro controller Firmware

There is only one executable file required, **EXC_Dnn.hex**, where "nn" is the version number. As explained earlier, the only hardware dependent parameter is the serial communications data rate (affected by the crystal frequency), which is programmed separately during setup. Hence there is only one "build" of the firmware. While an EEPROM image **EXC_Dnn.eep** is also supplied, this is not necessary, and only sets the default crystal frequency parameters and loads a demonstration beacon message.

PC Control

Several control programs are available. For KISS operation, any dumb terminal at 9600-N-8-1 will suffice. For portable and field use, the author uses a small DOS palm-top computer with built-in terminal emulation (HP 200LX) with great success, and there are many DOS terminal programs that work well on DOS computers and laptops (remember Procomm?)[14]. For Windows computers of all types, **Win 3.1 Terminal** is recommended (it works well on Win 3.1, 95, 98 and 2000), and is much easier to use than the one normally supplied with the later versions of Windows. By the way, most packet radio programs will also work well.

For general beacon use (messages that are Morse based), use **MAKEBCN.EXE**, which offers real-time control of frequency, can turn the transmitter on and off, but most importantly, can accept a text message file, compile it into commands and Morse in Murphy format, and download it to the Exciter.

For real-time QSOs using a PC to control the Exciter, there are two choices; the DOS program **EXC.EXE** by the author, and the Windows program **WinEXC.EXE** by Con ZL2AFP (not documented here). They are generally similar. The DOS program will operate under Win 3.1, 95 and 98 in a DOS box. Furthermore, the program can be run concurrently with ARGO or another Windows spectrogram program for reception.

[14] There is a wide variety of such programs available for free download at simtel.man.szczecin.pl/pub/msdos/commprog/

For real-time JASON operation, you can use JASON V0.94 by Alberto I2PHD and Vittorio IK2CZL. This software supports the ***ZL1BPU LF Exciter*** command protocol for transmitting, so with a serial cable connected to the Exciter, you can transmit from the Exciter and receive (using the sound card) directly from the LF receiver.

Calibration

The actual frequency of the Exciter reference crystal does not usually matter, as the software used to control the Exciter can calibrate out or compensate for any error. Only with KISS-mode manual control will you need to know the actual frequency accurately, that is unless you simply adjust the frequency (with the **F** command) until the frequency counter or receiver indicates that the transmission is where you need it!

Start by editing the file EXC.SET, used with all the PC control programs, placing the nominal crystal frequency (in Hz) in line 3 of the file. Using one of the PC control programs (see descriptions below), operate the Exciter, and cause it to transmit on a convenient frequency by giving the PC program the commands for that exact frequency. For example, if you use a nominal 12.8 MHz crystal, send the command to operate the Exciter at 128000 Hz (1 / 100^{th}) or 160000 Hz ($1/80^{th}$). Read the transmitter output frequency with a good frequency counter, and note the reading. Multiply this by the ratio between the nominal crystal frequency and the actual operating frequency (100 or 80 in the examples). Then edit the EXC.SET file, and replace the value on the third line with this new value.

Now, when you run the PC control program again, it will read the EXC.SET file again, and exactly compensate for the error in the crystal. The better the actual transmitted frequency can be read, the better the program's control accuracy.

If you have chosen a crystal to give exact 1/10 or 1/16 Hz steps, you will need to trim the crystal. Set the nominal crystal frequency (the value that gives the correct steps) as precisely as possible in the EXC.SET file, set the transmitter to operate on a convenient frequency, such as 200000 Hz, and adjust the trimmer for exactly that reading. Don't tweak the value in the file again.

Note – in all cases, measure the Exciter output, not the crystal frequency. Measuring the crystal frequency will cause the oscillator to shift, and maybe even stop oscillating.

Remote Control

None of the software described below suits remote operation of the ***ZL1BPU LF Exciter***. However, you *can* operate the Exciter remote using KISS commands – either via a modem, or via a Packet Radio TNC in connected or unconnected mode. Connected would be best, as that spares the Exciter from receiving unnecessary and confusing data. Turn off all TNC reporting and unconnected messages.

With the exception of message programming (the **B** command), all commands will operate with a 9600 bps link to the TNC. In order to use remote reprogramming, you should use a 300 bps link to the TNC, and preferably keep the message size down to one packet's worth.

The Exciter can be controlled remotely quite satisfactorily with a conventional packet radio program – and it will also send back the normal KISS mode responses. You may need to be patient if the channel is busy – probably best to use a little used channel. Real-time keying operation above QRSS60 would not be practical, since timing accuracy is lost over a packet radio link.

MAKEBCN

This DOS program has three main functions:

1. Real time control of the transmitter, turning it on, off, and sending a basic Morse ID message. This allows you to tune and test the Exciter.

2. To ' compile' a text file message into a beacon message, which it then "downloads" to the Exciter message memory. This message can include text as well as frequency, mode, speed and FSK shift commands.

3. It allows the current parameters (frequency, mode, speed and shift) to be changed, and when required, to be permanently stored for instant start-up when power is applied, even without the computer connected.

Requirements

As well as your ***ZL1BPU LF Exciter***, you will need a computer capable of running a DOS program, with one serial port capable of operating at 19200 bps. Most fast 486 or better computers would be appropriate (but bear in mind it may also require to operate a sophisticated receive program at the same time). You need an RS232 modem cable to connect the PC serial port to the Exciter.

Any EGA or VGA monitor will do, and memory and disc requirements are minimal. The program will operate from a floppy disc just fine, if the EXC.SET setup file is also on the disc

Using the Program - Setup file

The setup file EXC.SET is shared with the more elaborate Exciter control program EXC.EXE, which is capable of real-time programmed multi-mode operation. This file sets, among other things, the Exciter clock frequency, and the PC serial port and baud rate. These things need to be set before you run the program. Just edit the file with a text editor, making sure that the exact same number of lines are left intact, and that the values match those of your PC and Exciter. The closer you can set the Exciter clock frequency (typically 12800000 Hz), the more accurately the programs will control the frequency settings of the Exciter synthesizer. See the section on calibration earlier in this chapter.

EXC.SET needs to reside in the same directory as the MAKEBCN.EXE program, and can be shared by both programs.

Using the Program - The Immediate Commands

When the program is running, look closely at the menu commands on the right side of the screen. The first set is operated by keys on the main keyboard, and are immediate or program commands.

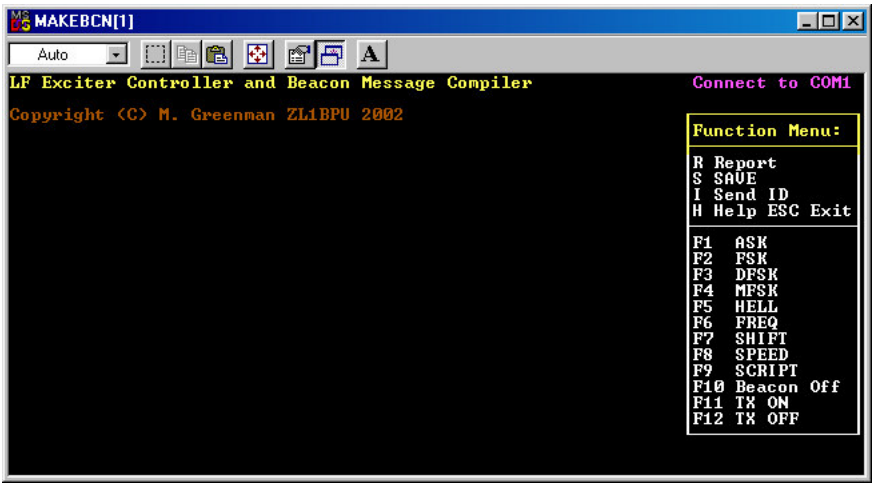

Fig. 8.1 The MAKEBCN screen

R Report

Sends a request to the Exciter for the current settings. These are then displayed on the bottom line of the screen, in typically cryptic Exciter hexadecimal! This is a useful command to check that the Exciter is responding correctly to the computer program. This is exactly the same as the Exciter '**R**' command.

S SAVE

Instructs the Exciter to store the current parameters, including the frequency, mode, speed and shift, to be used when power is next applied to the Exciter. If the mode set is M0, the Exciter will simply operate as a carrier generator, responding to manually sent commands. If the mode is any other, it will operate as a self-contained beacon, even when power is removed and reapplied.

I Send ID

This is a program command. The program asks you for a short line of text (such as ' de ZL7ABC'), which is then sent as ASK Morse at about 8-10 WPM. If the Exciter is in the middle of a beacon message, this command will temporarily take over the Exciter, and will leave the TX off at the end of the message, so the following beacon text element may be missing. Any short line of text can in fact be sent. The keying speed will depend on the speed of your PC.

H Help

Sends a short help message to the PC screen.

ESC Exit

Quits the program and closes the files and communications port.

Using the Program – The Exciter Mode Commands

Now look at the lower group of commands on the MAKEBCN.EXE screen (Fig. 8.1). These directly control the LF Exciter, and are accessed via the ' Function Keys", usually in a row at the top of the PC keyboard. The first five are mode commands, and will instantly switch the Exciter into a beacon keying mode. The three modes ASK, FSK and DFSK are all Morse modes; the other two are ' bit mapped' graphics modes, used to send simple images or various MFSK or Hellschreiber modes. All modes except HELL can be received on a Spectrogram display. JASON mode is not included since it is not supported by MAKEBCN.[15]

[15] You need to use the JASON software by I2PHD and IK2CZL

The F10 key returns the Exciter to mode M0, allowing only local or manual control with the beacon switched off.

F1 ASK

Sends on-off keyed Morse from the in-built beacon message. The message is stored inside the Exciter memory as Morse characters.

F2 FSK

Sends frequency-shift keyed Morse. The "key-up" periods are sent at the set frequency (set by the FREQ command), and the dots and dashes are sent at a higher frequency, raised by the amount set by the SHIFT command.

F3 DFSK

Sends dual frequency keyed Morse. Dots are sent at the set frequency, while dashes are sent at the offset frequency defined by the SHIFT command. Importantly, the dashes are the same length as dots, and spaces between elements are omitted if the elements are different making them unnecessary. This mode is easy to read visually, and is much faster than ASK or FSK Morse.

F4 MFSK

A bit-mapped mode. Sends each bit of each data byte in the message at a different frequency. If the bit is one, the carrier is on, if zero, it is off. The SHIFT command sets the difference between the frequencies. The lowest significant bit is on the nominal set frequency, and is sent first. Subsequent bits are raised further in frequency by the SHIFT, even if not transmitted. Zero (not transmitted) dots have shorter duration than one (transmitted) dots.

You could capture the output of this mode to create an MT-Hell beacon message.

F5 HELL

A bit mapped mode. Can share the same font information as MFSK, but can benefit from specially designed characters. The speed setting is critical to the appearance of the text. The dots are sent in the same order as MFSK, but all on the same set frequency. Cannot be received on a spectrogram, needing a HELL-style amplitude-sensitive receiver.

You could capture the output of this mode to create a Feld-Hell beacon message.

F10 Beacon Off

Disables the built-in beacon keyer. Allows the Exciter to be directly controlled with the F6, F7, F11 and F12 commands, by the ' I' ID command, or by the EXC.EXE program.

Using the Program – The Exciter Settings

These commands allow direct control and beacon mode settings to be changed. These settings will be lost on power-down, unless the SAVE command is sent. These same commands (and the Mode commands, of course) can be embedded in the beacon message, so if the Exciter does not seem to obey any of the commands you set, check that it is not being countermanded by the beacon message!

F6 FREQ

Allows the Exciter synthesizer carrier frequency to be set with great precision. The resolution is less than 0.1Hz. The value is entered in Hz. This value is then divided by the synthesizer resolution (depends on the clock frequency), converted to hexadecimal and sent to the Exciter. Fortunately the user does not need to worry about that! Negative frequencies can be set, and result in the correct carrier frequency, but the SHIFT works in the opposite direction in FSK, DFSK and MFSK.

F7 SHIFT

Sets the offset used for FSK and DFSK modes, and the incremental offset per bit in MFSK mode. The range is zero to about 25Hz. Values of about 30 divided by the dot period are recommended.

F8 SPEED

Sets the dot period in seconds. The program assumes the dot clock is 64Hz. The range is from about 0.05 to 1000 seconds. For "normal" speed Morse, use the following table:

Period	Command	Speed
0.03 sec	K0002	35 WPM
0.05 sec	K0003	25 WPM
0.07 sec	K0004	15 WPM
0.10 sec	K0006	12 WPM
0.15 sec	K000A	9 WPM
0.20 sec	K000D	6 WPM

Table 8.1. Keying Speed

F11 TX

Turns the transmitter on. If used with the beacon running, will only remain active until the next beacon ' off' element.

F12 RX

Turns the transmitter off. If used with the beacon running, will only remain active until the next beacon ' on' element.

Using the Program – Making and Sending Scripts

The F9 Function Key provides a means of making and downloading beacon messages. These are otherwise fairly laborious to make, involving looking up each Morse character in a table, and entering the values manually with the Exciter "B" command, which is both boring and prone to error. The SCRIPT command works in two phases.

First, the requested file is opened, read and converted into Morse (and commands if included, see later). The size of the message is checked to see if it will fit into the Exciter memory, and will be rejected if too large. Next, the message will be slowly sent to the Exciter, and when it has been stored, the command will be terminated and the Exciter will revert to what it was doing before.

The Exciter stores Morse very efficiently, one byte per character, so quite large messages can be sent. In addition, all the other direct commands can be embedded in the script message, and will be converted into beacon message commands. You can therefore change mode, change shift (which is necessary when you change FSK speed or use MFSK, for example), change keying speed

and even frequency, so you can for example set up a dual frequency beacon. Text to be converted to Morse is simply typed into the file - use spaces for pauses. Both upper and lower case are recognised, all punctuation and some symbols and prosigns are included. The commands are included in the script using the exact same syntax as the Exciter commands, except they are prefaced with a ' $" character:

SHIFT $Ann

0 - 255 resolution steps (about 0 - 25 Hz). Value in HEX 00 - FF.

FREQUENCY $Fhhmmll

0 - 2^{24} resolution steps. value in HEX 000000 - 400000. (Use the F6 key to find out what the current settings are.)

KEYING SPEED $Knnnn

0 - 65536 dot clocks per dot period. See the F8 key above for more information. The value is in HEX 0000 - FFFF.

MODE $Mn

Sets beacon keying mode, M1 to M5. M0 is not a valid script option. M6 is not supported.

A tilde character ' ~' is used by the program to identify the end of the message. This allows you to add other text and comments later on in the file that are not converted into the message. Here is an example of a simple message text file for use with MAKEBCN. Note that the end-of-line (“Enter”) characters are converted to spaces:

```
$M1$F22CE6E$K0006 QRP BEACON DE ZL1BPU ZL1BPU RF72KU AR
$M3$A7E$K00C0 ZL1BPU ~
```

In this example, once the script is downloaded and the beacon started, it first sets the mode to ASK ($M1), the frequency ($F22CE6E), the keying speed to about 12 WPM ($K0006), sends an ID message, then changes to DFSK mode ($M3), 10 Hz shift($A7E), three second dots ($K00C0), and sends my callsign. Finally the command is terminated (~), and will repeat from the start. The program automatically stores a final terminator in the Exciter memory which forces the beacon to start again at the beginning of the message.

Unless the message text file is in the same directory as the MAKEBCN.EXE executable, when the F9 SCRIPT command asks for a file name, you will need to specify the exact and complete path to the text file, in addition to its name.

Other Features

The program reports all commands at the bottom of the screen. It also continually monitors responses from the Exciter, and will display them in a long line at the bottom of the screen. The responses are not error checked, or interpreted in any way by the program. If communications are lost, the program reports an error at the bottom of the screen. All these messages are periodically erased (every 10 seconds) to keep the screen tidy.

The ' Connect to Com...' message at the top right corner of the screen reminds you which port the software is expecting to see the Exciter on. This and the serial communications data rate are read from the setup file EXC.SET.

This program does not at present allow bit-map files (for MFSK, MT-Hell and Feld-Hell, modes M4 and M5) to be ' compiled' and downloaded. You will have to make them manually and laboriously, or use EXC.EXE to create them in a manner similar to that described for JASON.

Source code for the MAKEBCN program is available from the author.[16] The price includes email support.

[16] Email zl1bpu@nzart.org.nz .

EXC

This DOS program has many similarities to MAKEBCN, but is intended for real-time control, in other words, as well as the program controlling the transmitter, when you type into the program it sends what you type. EXC provides:

1. Real time control of the transmitter, turning it on, off, and sending text from the keyboard in any pre-defined mode.

2. Sends a choice of pre-defined messages from file.

3. Sets keying parameters (speed, shift, modulation technique) according to the selected mode. These modes are also pre-defined in a file.

WinEXC by Con ZL2AFP is essentially a Windows implementation with the same functionality, so the following comments also apply to it.

Requirements

The requirements are the same as for MAKEBCN. The same setup file, EXC.SET is used. (WinEXC also uses the EXC.SET file). Modes are defined in the file, along with the pre-defined messages.

Using the Program - Setup file

Most of the setup file (the first five lines) is exactly the same as for MAKEBCN. What's different is the pre-definition section that follows. There are 10 pre-defined modes, which you can alter if you wish. There must be always 10 mode lines. Each line contains the name of the mode, the modulator to use, the frequency offset to use, and the keying speed, as follows:

"MODE NAME", MODULATOR, OFFSET, SPEED

The mode name must be in quotes, and be 8 characters or less. The modulators are referred to by a number, "MOD" in the table below:

MOD	KEYING	CODE	COMMENT
0	ASK	MORSE	Conventional on/off keyed Morse
1	FSK	MORSE	FSK Morse, keydown is shifted up
2	DFSK	MORSE	Dual shift Morse, dash is shifted up
3	MT-HELL	GRAPHIC	Multi-tone Hellschreiber
4	MFSK	ITA2	MURRAY, visually decoded MFSK

Table 8.2. EXC's Modulators

OFFSET is the number of DDS resolution steps FSK shift or per MFSK step. SPEED is the number of seconds per symbol (per data bit or dot), although MFSK modes use much smaller dots for the same FFT receiver speed (say one third to one quarter) and are therefore less sensitive.

The default modes and their names are arranged so the transmission looks correct at the corresponding ARGO speed, for example QRSS3 and DFSK3 are readable on ARGO 3 SEC mode.

Examples:		
	MORSE, 10 WPM	"MORSE",0,0,0.1
	QRSS, 3 SEC DOTS	"QRSS3",0,0,3
	QRSS, 10 SEC DOTS	"QRSS10",0,0,10
	DFSK, 3 SEC DOTS	"DFSK3",2,32,3
	DFCW, 10 SEC DOTS	"DFSK10",2,10,10
	MT/HELL, 3 SEC	"MTHL3",4,32,0.6
	MURRAY, 3 SEC	"MURR3",4,32,1

(MURRAY is a fun mode using 5-bit ITA-2 code transmitted for visual reception, like reading paper tape).

The final four lines in the setup file contain the four pre-programmed messages. There is one message per line, and they must be surrounded in quotes. The messages can be 255 characters long (but remember they may take ages to send!), but only the first 20 or so are shown on the screen. When these messages are used, they are not copied into the transmit buffer, but send the selected message over and over again until you quit the program or change from message to keyboard.

Using the Program – The Screen

There are three areas to the screen (see Fig. 8.2). The top and centre show the current operating mode and message, the current frequency setting, and the transmit status (TX/RX). You can see the lists of modes and messages, from which you select in turn with a function key. The frequency is displayed to 1 mHz resolution, and is the actual carrier frequency, achieved by converting your requested frequency to hexadecimal, sending it to the Exciter, and converting this value back to decimal. In this way the frequency accurately reflects the digitization limitations of the Exciter.

Below, in the row starting with the Copyright message, is the range statement. This gives the operating frequency range of the software and Exciter – the upper limit depends on the crystal frequency set in EXC.SET. Below this line is the current data sent to the Exciter. To the right side of this line is a small buffer which shows the last 22 or so characters sent to the Exciter.

At the bottom, in the box, is the keyboard buffer. This is where you type text to be sent. The program features “KOX” (Keyboard Operated Xmit), so the transmitter goes on and starts sending as soon as you start typing. The transmitter goes off (back to “receive”) when the keyboard buffer becomes empty. Below that again is a list of the function keys.

The program contains no graphics, so will operate in a Windows DOS box. This is handy as it allows you to run ARGO at the same time, to monitor your transmissions, or to receive during a QSO. Just remember to return “focus” to the EXC program, by clicking on its window with the mouse, in order to start typing (and transmitting) again.

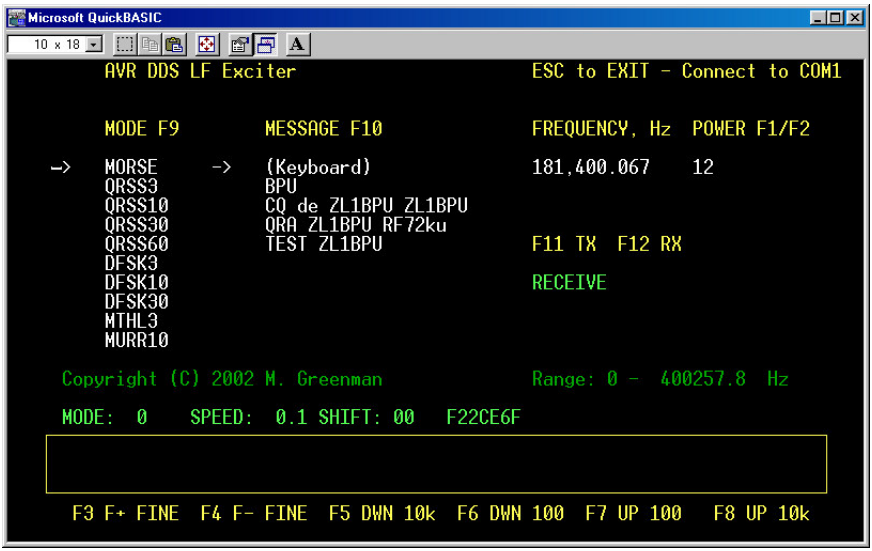

Fig. 8.2. The EXC screen

Using the Program – Commands
The program uses function keys for commands, leaving most of the keyboard free for real-time typing. The commands are:

F1	Increase power
F2	Decrease power
F3	Increase frequency (fine, mHz resolution)
F4	Decrease frequency (fine, mHz resolution)
F5	Decrease frequency 10 kHz
F6	Decrease frequency 100 Hz
F7	Increase frequency 100 Hz
F8	Increase frequency 10 kHz
F9	Step to next mode (default is first in list)
F10	Step to next message source (default is keyboard)
F11	TX on
F12	TX off
ESC	Closes the program

(Note – F1 and F2 only work if the output ports PD2, PD3 and PD4 are wired to an R-2R network to control the amplifier gain. See the PORT command under KISS commands in Chapter 7. This program is not intended to provide real-time control of the output ports.)

If you change mode in the middle of a message (or with text in the keyboard buffer), nothing nasty will happen, but there may be partial characters transmitted in the new or old mode. Similarly, if you change messages slowly, there may be partial messages sent (the first character of each message). The messages always start at the beginning. In this sense the keyboard buffer is no different to the fixed messages. It is a good idea to include a space in the fixed messages to force a silent period at the message start.

WinEXC

This program, created by (and available from) Con Wassilief ZL2AFP[17] seeks to replicate the operation of EXC, but in a true Windows environment. As you can see in Fig. 8.3, it offers just about the same functionality, with drop-down lists and mouse clicks instead of function key operation.

WinEXC uses the same EXC.SET file as MAKEBCN and EXC, and it must reside in the same directory or folder as the executable program. The program is very small, is much nicer looking than EXC.EXE, and it makes no nasty changes to the Windows Registry!

[17] Contact Con at zl2afp@internet.co.nz

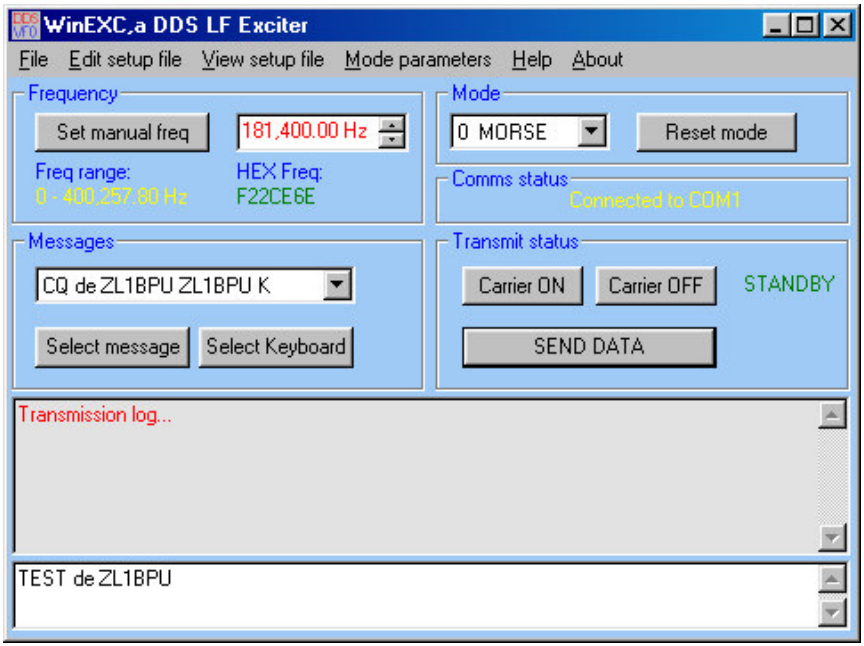

Fig. 8.3. The WinEXC screen

9. Appendices

Serial Commands (KISS commands)

Axx ADD
Add offset of xx resolution steps, where xx is "00" to "FF. In Sweep Generator mode, sets the sweep dwell time (time per step). Resolution is about 1/12 milliseconds and range is under 1ms (1ms = A0C) to over 20ms (20ms = AF8).

B BEACON
Command to enter beacon text, where up to 120 bytes of data and commands can be entered. The data consists of HEX-ASCII character pairs. <CR> and <SPACE> are permitted, and the last character must be "FF". Data entry mode is terminated by the tilde character " ~ ". The message is permanently stored and the unit is reset on exit.

Fhhmmll FREQUENCY
Set frequency to this value times the resolution. The value hhmmll is a 24 bit binary number expressed as six hexadecimal characters, "00" to "FF".

H HELP
Simple help message listing these commands (also resets microcontroller).

Knnnn KEY
The beacon mode keying baud rate, where nnnn is 0x0000 to 0xFFFF. Resolution is 31.25ms, but is dependent on the symbol clock generator frequency (typically 32 Hz). Hence K0001 gives a baud rate of 32 baud - 3 sec dot QRSS requires K00C0 and JASON (11.8 sec symbols) K02F0. In Sweep Generator mode, sets the frequency increment in resolution steps (as for the F command), but range is limited to K0000 to KFFFF (over 5 kHz/step).

Mn MODE
Sets the beacon mode, where n is a number 0 to 6.

0	Beacon Off	Continuous carrier controlled by A, F, T, W and X commands
1	ASK	On-Off single frequency keying, beacon message in Morse
2	FSK	Continuous carrier FSK keying, beacon message in Morse
3	DFSK	Dual frequency on/off keying, beacon message in Morse
4	MFSK	Eight frequency bit-mapped scanned MFSK mode (e.g. MT-Hell)
5	HELL	On-Off single frequency bit-mapped keying (e.g. Feld-Hell)
6	MFSK/IFK	MFSK data frequency shift in resolution units (e.g. JASON)

Pp PORT
Sets Port D outputs PD2, PD3, PD4 according to value least significant bits of value "p" as follows:

0	PD2=0 PD3=0 PD4=0	4	PD2=0 PD3=0 PD4=1
1	PD2=1 PD3=0 PD4=0	5	PD2=1 PD3=0 PD4=1
2	PD2=0 PD3=1 PD4=0	6	PD2=0 PD3=1 PD4=1
3	PD2=1 PD3=1 PD4=0	7	PD2=1 PD3=1 PD4=1

"p" values above 7 work, but upper bits are ignored.

R REPORT
Requests a message giving the current settings. Reports A K M W and F.

S STORE
Store current settings. Saves FREQUENCY, OFFSET, MODE and KEY. Sweep mode settings are not saved.

T TX
T turns the transmitter on.

Wmm WIDTH
Sets number of sweep steps, where mm is "00" to "FF" in two hexadecimal characters. Typical value is W14 for 20 steps. 50 baud 170 shift RTTY with 2125 Hz MARK can be simulated with W02 AF8 K07E5 F0062B1. The start frequency for the sweep is set by the F command, sweep dwell time by the A command, and the sweep step size by the K command. W00 turns off the sweep.

X RX
X turns the transmitter off.

Beacon (script) Commands

F1	Mode 1 ASK Morse
F2	Mode 2 FSK Morse
F3	Mode 3 DFSK Morse
F4	Mode 4 MFSK Graphics (Sequential MT-Hell)
F5	Mode 5 ASK Graphics (Feld-Hell)
F6	Mode 6 MFSK/IFK Data (Jason)
FB pp	Set port pins (see the **P** command)
FC hhmmll	Set frequency (see the **F** command)
FD nn	Set FSK shift or MFSK increment (see the **A** command)
FE nn nn	Set keying speed (baud rate - see the **K** command)
FF	End of script

The FB, FC, FD and FE commands are functional replicas of the corresponding KISS commands.

Script Examples

If you load all this data in using the **B** command, the beacon will send everything, one mode after another! The complete message will not of course all fit in the 112 byte memory, so just use the bits you want to try out, and place “FF” at the end.

`FC 20 A6 E9`	Set operating frequency (181.4 kHz, using a 12.8 MHz crystal)
`F1 FE 00 04`	10 wpm ASK mode Morse
`01 09 02 01`	DE
`01 13 12 3E 11 16 0C 01`	ZL1BPU

F1 FE 00 C0	3 sec dot ASK mode QRSS
01 11 16 0C	BPU
F2 FD 40	3 sec dot FSK Morse
01 11 16 0C	BPU
F3 FD 40	DFSK mode
01 13 12 3E 11 16 0C	ZL1BPU
F6 FE 02 F0	JASON mode, 11.8 baud
00 1B 1E 1B 24 1B 2A 18 1E 12 1B 15 18 12 24 0C 0F 03 0C 00 0C 03 18 00 03 00 0C 06 0F 03 18 06 1F 0C 15 0C 18 12 24 21	ZL1BPU BCN [RF72KU]
FC 20 A6 00	Change frequency slightly
F4 FE 00 10 FD 18	MFSK image mode, MT-Hell 3 sec
01 41 45 49 51 41 00	Z
7F 01 01	L
20 7F 00	1
7F 49 49 36 00	B
7F 48 48 30 00	P
7E 01 01 7E 00	U
F5 FE 00 03	Feld-Hell mode, 1 sec columns
01 41 45 49 51 41 00	Z
7F 01 01	L
20 7F 00	1
7F 49 49 36 00	B
7F 48 48 30 00	P
7E 01 01 7E 00	U
FF FF	End of message

Morse Coding

Using “Murphy” coding, each dot is represented by “0” and each dash by “1” packed right-to-left in a single byte. A “1” is placed in the next free bit to the left to signify end of character, and remaining bits to the left remain “0”. As each element of the character is transmitted, the data byte is shifted right and the carry transmitted (and followed by inter-element spaces) until the remaining data is 0x01 (i.e. “00000001”).

Coding is arranged so that automatic inter-character spaces are correct at three dot-lengths, and since “space” is coded as no data elements, but followed by an inter-character space, the inter-word space is six dot-times.

Some characters are interpreted as inter-character spaces (0x01), while others are ignored completely (0x00). The table below is represented in hexadecimal. Lower case characters should be coded as upper case.

SP	01	0	3F	A	06	Q	1B
!	00	1	3E	B	11	R	0A
“	52	2	3C	C	15	S	08
#	00	3	38	D	09	T	03
$	C8	4	30	E	02	U	0C
%	01	5	20	F	14	V	18
&	01	6	21	G	08	W	0E
‘	5E	7	23	H	10	X	19
(	2D	8	27	I	04	Y	1D
)	6D	9	2F	J	1E	Z	13
*	01	:	47	K	0D	[	00
+ AR	2A	;	35	L	12	\ SK	68
,	73	<	00	M	07	]	00
-	61	= BT	31	N	05	^	00
.	6A	>	00	O	0F	_	6C
/	29	?	4C	P	16	`	00

Table 9.1 The Morse Coding Table

Hell Font

The suggested font is a 5x7 low resolution Hell font. Characters are listed in ASCII order, starting at ASCII 32 (space). Five bytes are shown for each character, represented in HEX. Trailing spaces can be ignored, making the font proportional. Always include one byte of 0x00 (zero) between characters to act as a character space. For double-width characters, repeat each byte - for Feld-Hell, four or more repeats. The example in Fig. 5.1 has four identical bytes per column.

Note - this is NOT the same font as that transmitted by EXC.EXE!

SP	00,00,00,00,00	0	0E,13,15,19,0E
!	00,1D,00,00,00	1	00,10,1F,00,00
“	00,18,00,18,00	2	03,15,15,15,09
#	0A,1F,0A,1F,0A	3	15,15,15,0A,00
$	08,15,1F,15,02	4	1E, 02,07,02,00
%	19,1A,04,0B,13	5	1C,15,15,15,02
&	0A,15,0D,02,05	6	0E,15,15,15,02
‘	00,10,18,00,00	7	10,10,17,18,00
(	00,0E,11,00,00	8	0A,15,15,15,0A
)	00,11,0E,00,00	9	08,15,15,15,0E
*	15,0E,1F,0E,15	:	00,05,05,00,00
+	04,04,1F,04,04	;	00,0A,0B,00,00
,	00,01,06,00,00	<	04,0A,11,00,00
-	04,04,04,04,04	=	00,0A,0A,0A,00
.	00,03,03,00,00	>	00,11,0A,04,00
/	01,02,04,08,10	?	08,10,15,14,08
@	0E,11,15,15,08	P	1F,14,14,14,08
A.	0F,14,14,14,0F	Q	1F,11,15,13,0F
B	1F,15,15,15,0A	R	1F,14,16,15,09
C	0E,11,11,11,00	S	08,15,15,15,02
D	1F,11,11,11,0E	T	10,10,1F,10,10
E	1F,15,15,15,11	U	1F,01,01,01,1F
F	1F,14,14,14,10	V	18,06,01,06,18
G	0E,11,11,13,0A	W	1E,01,06,01,1E
H	1F,04,04,04,1F	X	11,0A,04,0A,11
I	00,11,1F,11,00	Y	10,08,07,08,10
J	02,11,1F,10,00	Z	11,13,15,19,11
K	1F,04,0A,11,00	[	1F,11,11,00,00
L	1F,01,01,01,00	\	10,08,04,02,01
M	1F,08,04,08,1F	]	11,11,1F,00,00
N	1F,08,04,02,1F	^	00,08,10,08,00
O	1F,11,11,11,1F	_	01,01,01,01,01

Table 9.2. The Hell Font table

EEPROM Memory Locations

Most of the EEPROM memory is dedicated to the beacon message. Using the table below it would be possible to program beacon messages via the programming software, which might be easier if the message was complex. There are also a few interesting locations before the message. Addresses and contents are given in HEX.

Address	Default	Content
0000	00	-
0001	00	-
0002	20	High byte of default operating frequency
0003	E8	Mid byte of default operating frequency
0004	33	Low byte of default operating frequency
0005	00	Default mode
0006	18	Default offset (FSK offset, MFSK step size)
0007	52	UART data rate (9600 bps)
……		
0010	FF	Start of beacon message
……		
007F	FF	Highest possible location of beacon message

Table 9.3 EEPROM Locations

Brief Exciter Specifications

Spectral Purity

Second harmonic -42dBc, third harmonic -50dBc (no low pass filter).
All harmonics at least -50dBc (with low pass filter).
Below 20 kHz all harmonics -60dBc or better.
Random spurii better than -60dBc, close-in noise below -60dBc within 1 Hz. Sampling rate 1.11 MHz (Nyquist frequency 550 kHz).
Sampling clock and alias image at 1.4 MHz -42 and -54dBc respectively.

Operating Range

0.08 Hz to 400 kHz, in any step size, with a resolution of 0.08 Hz.
Power amplifier response drops off below 7 kHz and above 250 kHz.
500mW power bandwidth 10 kHz to 200 kHz, and depends mostly on the output transformer and coupling network.

Power Output

Amplifier: Up to 1W into 8 - 50 Ohm load. Power level adjustable over more than 60dB range using a DC control pot. Direct: Output without power amplifier is 5V p-p into 1M Ohm, or 1V RMS into 1k.

Digital Outputs
8 bit sine data to D-A converter. By changing the sine table, it is possible to generate push-pull non-overlap square wave drive at 20mA with any one of four user selected duty cycles. The eight square wave CMOS 5V level outputs have sub-microsecond rise-times. It is also possible to use two 4-bit sine/cosine tables to generate two signals with constant and accurate 90° phase relationship. This is useful with quadrature modulators or demodulators, e.g. using the phasing technique.

Output Ports
Three digital outputs can be controlled for external use - controlled via script or serial command.

Power Supply
+12V DC to +15V DC at about 300mA. Supply need not be regulated. Key up 10mA, key down up to 150mA. Will operate at reduced output down to 7V.
No standby power required for memory retention.

Commands
Twelve immediate mode user commands. Six script commands. Scripts are downloadable and used for beacon messages.

Serial Interface
RS232, TXD and RXD only, at up to 19200 bps (user selected), no parity, eight bit data, one stop bit. The synthesizer stops while processing commands.
Data rate is stored in EEPROM - there is only one version of the firmware.

Beacon Messages
112 bytes of user EEPROM message memory, containing coded data and commands. Memory is sufficient for about 15 words in Morse, 8 in JASON, or three words in S/MT-Hell or Feld-Hell. 16 bytes of EEPROM are used to store the current user settings for power-up beacon or normal mode restoration.

Physical
Can be built on a prototype board about 100 x 75mm and will fit in a TNC sized box. The Exciter will operate from 0 to 70°C and will tolerate thermal and mismatch overload. The micro controller can be programmed and reprogrammed in circuit.

Modes
ASK, FSK, DFSK, MFSK, IFK, using Morse, image or data. Six beacon modes.

10. Glossary

ASCII American Standard Code for Information Interchange. The standard method for coding an alphabet of letters, numbers and symbols in computers. There are 128 characters in the alphabet.

ASK Amplitude Shift Keying. On-Off keying (as in Morse) is the most common form of ASK.

Baud The measure of speed of transmission of data events. The baud rate is the speed at which the smallest entity of a transmission system (a *symbol*) is transmitted.

Beacon An automated transmission operated as a broadcast for test, identification, telemetry transmission and experimental purposes.

DDS Direct Digital Synthesizer. A frequency synthesis technique where the RF signal is directly generated by numerical methods.

DFSK Differential Frequency Shift Keying (sometimes mis-named DFCW). A method of sending Morse where dots and dashes are the same duration but on different frequencies.

EEPROM Electrically Eraseable Programmable Read-Only Memory. A type of memory which retains stored values when power is removed.

Feld-Hell An image mode where text is sent as dots, like a dot matrix printer does. Each column is scanned vertically, bottom to top, left to right. Visible dots are sent key-down, non-visible spaces as key-up. The mode is extremely immune to noise, since the data is interpreted by eye. The name derives from the inventor (Rudolf Hell), and the first major use of this exact mode, for military “field” transmissions, in the 1930s. See *MT-Hell*.

Firmware Program which operates in a fixed-program device such as a micro controller. See *software*.

FFT Fast Fourier Transform. A mathematical technique used to convert signals from the time domain to the frequency domain, in the same manner as a spectrum analyser.

FSK Frequency Shift Keying. Morse sent this way has key-up on one frequency, and key-down on another.

Hexadecimal Numbers expressed to base 16, which are more readily understood by a computer than decimal (base 10).

HEX-ASCII A method of representing *hexadecimal* values by a number 0-9 or letter A-F. Two such characters are used to express each byte, “00” to “FF”.

IFK Incremental Frequency-shift Keying. A type of MFSK where the information sent is not in the actual frequency transmitted, but the incremental frequency difference between the event and the previous event. This technique is resistant to frequency drift.

KISS	"Keep it Simple, Stupid!" A term used to describe simple control protocols for microprocessors and micro controllers which are human readable as well as machine readable.
LF	Low frequency. The region between 30 kHz and 300 kHz.
LSB	Least Significant Bit. In a byte or word of data, the bit with the lowest weighting, the right-most bit.
MFSK	Multiple Frequency Shift Keying. A method of coding data as different (usually sequentially transmitted) frequencies, to represent more than one bit of data per event. FSK is simply two-frequency MFSK with one bit per event.
MT-Hell	An image mode where text is sent as dots. Each column is scanned bottom to top, left to right. Each row of dots is sent on a different (slightly higher) frequency, and non-visible dots are not transmitted. The mode is extremely immune to noise and ionospheric effects, since the data is received using an *FFT* technique (such as ARGO), and interpreted by eye. There are two types, concurrent, and sequential. This Exciter sends the sequential (one dot at a time) version, which does not require a linear transmitter. See *Feld-Hell*.
PLL	Phase-locked loop. A type of frequency synthesizer.
PSK	Phase Shift Keying. Data is encoded as shifts of carrier phase.
QRS	Slow ASK Morse. (QRS means "please send slower").
QRSS	Very slow ASK Morse.
Software	Program which (in this context) operates in a general-purpose computer, where the program can be loaded or changed at will.
Source Code	Human-readable program instructions which are also readable by a program (compiler or interpreter) which converts the program into native micro controller or computer instructions.
Symbol	The smallest data entity of a digital transmission. A symbol may contain one or more (or even fractional parts of) bits of digital information. See *baud*.
TCXO	Temperature Compensated Crystal Oscillator. An inexpensive but high performance type of crystal frequency reference. Temperature variations in the crystal are compensated electrically, rather than eliminated by close temperature control.
UART	Universal Asynchronous Receiver Transmitter. A device in a computer designed to transmit and receive asynchronous (stop-start) serial data.