Appendix A: Building an inverter

A.1. What is an inverter?

Electrical power is usually transmitted and used in the form of alternating current. However, some kinds of electrical generation and storage devices produce direct current, examples being PV modules and batteries. An inverter is a power electronic apparatus which converts DC to AC, allowing the DC power from these generators to be used with ordinary AC appliances, and/or mixed with the existing electrical grid.

A.2. Why build an inverter?

During the course of this project, the need was identified for a novel type of inverter, which could be dispatched as part of the REDMan system. In practice this meant that it should have a computer interface that could accept commands to set the output power. No inverter of this kind was available on the market, and it was not even remotely economical to have one custom-made by an outside contractor. So, there seemed to be two possible courses of action: buying a commercial inverter that lacked the required facilities and modifying it to suit, or building an inverter from scratch. Modifying a commercial unit seemed attractive at first, but after a few inquiries [1] it became apparent that manufacturers kept the details of their inverters confidential. They were not prepared to release any information on the circuitry of their inverters or the computer firmware that controlled them, even for academic purposes. This is obviously wise practice in a commercial scenario, but it would mean that modifying their product would be a matter of reverse-engineering it; in other words, poring over the circuit boards with a magnifying glass, and trying to reconstruct the firmware from the raw machine code extracted from ROM chips. This would not only be a very difficult job, but probably an offence under intellectual property law.

On the other hand, while building an inverter from scratch might seem more difficult, the whole reverse engineering issue would be avoided, and more importantly, the plans could be placed in the public domain where they might be of use to other researchers.

A.3. Design objectives

The most important objective, and the whole purpose of the exercise, was that the inverter must be controllable in real-time by a computer. It must accept commands telling it how much power to transfer from DC to AC at a given instant.

It would also have to be reliable. It is all too easy to create a unit that performs OK on the test bench, but fails in the field. The unit would have to operate successfully for the duration of the experiment – at least one month.

Efficiency was also a concern. If the experiment was to give realistic results, the efficiency would have to be representative of the efficiencies of contemporary commercial inverters. Since these can exceed 90%, this could be challenging.

Safety and power quality were also crucial. Any risk to personnel caused by malfunctioning of the inverter would be completely unacceptable, as would disruption to other electrical equipment caused by interference from the inverter. UK electricity companies drafted the G77 standard, defining the required safety features, and maximum level of distortion, permissible for equipment connected to their grid, and ideally the design would meet these. However, this is a fairly strict specification, and there would always be the danger that meeting it would not be economical in terms of time and money. N.B: at the time of writing, the UK standard was harmonised with the US standard, IEEE P929 [2].

The final consideration was the power rating of the apparatus. It would be wise to make it at least the same, if not more than, the expected peak power from the RE sources in the experimental system. The PV arrays totalled 260 W and there were also two 100 W wind turbines.

A.4. Specification

Parameter	**Min**	**Typ**	**Max**	**Unit**
Power quality:	**(as per G77)**			
Current THD			5	%
P.F.	0.95 lead		1.00	
DC Injection			5	mA
Protection:	**(as per G77)**			
Voltage range	216	230	253	V
Frequency	47	50	50.5	
Disconnect time			5	S
Reconnect time	3			Minutes
Performance:				
Efficiency		85	90	%
Power control tolerance			1	%
Output power	300			W

Reconnect time is to be reckoned as time after supply is restored within limits. Disconnection to be by mechanical contacts to IEC 255, not electronic means.

A.5. Basic design choices

Now that the specification is known, the task is to design the inverter circuit that will meet it. There are a number of different possible circuits, but fundamentally, all inverters work by using switches to periodically reverse a direct voltage. So, the two main design choices to be made are: what sort of switches to use, and what control algorithm to use for switching them on and off.

A.5.1. Switching technologies

Historically, inverters have been made with every kind of switching apparatus, such as rotating or vibrating mechanical contacts, gas-filled electronic valves, and thyristors (SCRs). However, in contemporary use, the field is led by two special kinds of transistor.

The first kind is the Metal-On-Semiconductor Field-Effect Transistor (MOSFET). This device has a very rapid switching action, and can be designed with a low resistance so that it will pass high currents efficiently, provided that the voltage it has to stand in the ‘OFF’-state is low. MOSFETs designed to withstand high voltages have a much higher ‘ON’-state resistance, making them less efficient. Whatever the voltage rating, MOSFETs are electrically robust, and difficult to destroy by excessive voltage or current.

Complementing the MOSFET is the Insulated Gate Bipolar Transistor (IGBT). When designed for high ‘OFF’-state voltages, this outperforms MOSFETs, although the MOSFET is still best at lower voltages. IGBTs switch rather slower than MOSFETs and are not quite as resistant to damage by overloads [3].

Given these advantages and disadvantages, the actual device chosen will depend on what sort of inverter circuit is chosen (this determines the voltages and currents imposed on the devices) and on what control algorithm is chosen (this determines the speed at which switching must be performed)

A.5.2. Circuit topologies

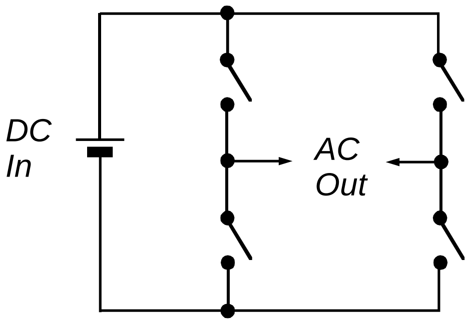
The most simple and well-known kind of inverter is as shown in Fig A.1. It consists of four switches which connect the DC supply (symbolised by a battery) across the output terminals, first in one sense, then in the other. In this way, the voltage is periodically reversed.

Voltage conversion is often required where an inverter is used. This is the case in the present application, where 24V DC must be converted to 230V AC. The circuit of Fig A.1 has a fixed output that is determined by the voltage of the DC source. There are two common ways of circumventing this, the simplest being to apply the AC output of the inverter to a transformer, and so step it up or down to the desired voltage. This circuit is shown in Fig. A.2. A more complex method is to change the voltage of the

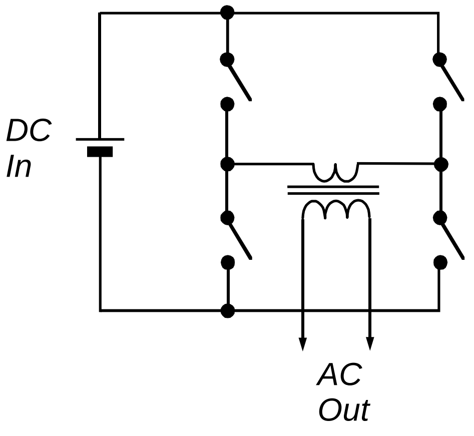
DC source instead, by means of a DC-DC converter. This is made of an inverter (sometimes called a *chopper* in this application) followed by a transformer, followed in turn by a rectifier, as shown in Fig.A.3. Although this is more complicated, it has certain advantages. When the inverter output is fed through a transformer, the transformer must be designed to operate at the inverter's output frequency. In the case where the inverter operates at 50 Hz, the transformer can be rather bulky and costly. When a DC-DC converter is used, its operating frequency can be made different to the eventual output frequency. By using a very high frequency, such as 50 or 100 kHz, a much smaller transformer is needed to handle the same amount of power, making the finished apparatus lighter, more compact, and cheaper. But, since the inverter stage connects directly to the line in this circuit, DC injection could be a problem.

When an inverter circuit is used to drive a transformer, there are extra possibilities in terms of topology. The most common circuit uses a centre-tapped primary winding and cuts the number of switches required from four to two. ("Inverter 1" in Fig. A.3 is of this kind.) While this saves money on switches, it makes the transformer less efficient, because each half of the winding is passing current only 50% of the time. Hence for a given mean current (which determines the power output) the RMS current (which determines the losses) will be higher, and so for a given design efficiency, the transformer must be bigger and more expensive. This must of course be weighed against the fact that in the full-bridge circuit, the current must pass through two switches in series. In practice, the centre-tap circuit is very commonly used where the DC source voltage is low and the operating frequency is high.

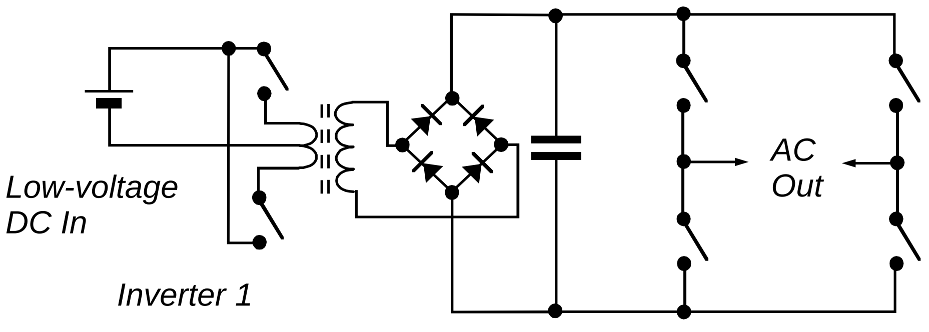
NB: A point should be made here on terminology. The functions of voltage changing and DC-to-AC conversion are normally combined in the same apparatus. Even though it is made of an inverter and a transformer, or even two inverters, a transformer and a rectifier if a DC-DC converter is used, it is customary to refer to the whole apparatus as an 'inverter'. To muddle matters even further, the digital logic gate performing the 'NOT' function is also known as an inverter, even though it has nothing to do with converting DC to AC.



(Fig A.1: Basic inverter circuit)



(Fig. A.2: Inverter with transformer)



(Fig. A.3: Inverter with DC-DC converter)

A.5.3. Control algorithms

From the point of view of control, these different inverter circuits are more or less interchangeable. The precise details of exactly which switches must be operated vary from circuit to circuit, but the scheme that controls *when* they are to be operated tends to be the same.

The most basic is the algorithm described earlier. To generate one half-cycle of the output, the inverter is switched to one polarity; for the next half-cycle it is switched to the opposite polarity. This generates a square output waveform whose peak amplitude (NB: peak and RMS are same for square wave) is equal to the DC source voltage. This is very easy to implement, but a square wave will not satisfy the requirement for low distortion. (The total harmonic distortion of a square wave is around 55%)

The next step in complexity is to arrange for a period in the cycle when the output voltage is zero. In the full-bridge inverter circuit, for example, this is achieved by turning S1, S3 on, hence shorting the AC terminals together. The result is a square wave with pieces missing, which can be arranged to have a peak-to-RMS ratio the same as a sine wave. This is very useful when the inverter is used to power a collection of normally mains-driven apparatus, which includes some appliances functioning according to the RMS voltage, and others requiring the peak voltage to be correct. With an ordinary square wave, the peak:RMS ratio is always 1, so both conditions cannot be satisfied at once. The harmonic distortion of this ‘modified sine’ wave is also less than a square wave (at around 25%) but this is still too high to meet the specification.

Therefore, it will be necessary to turn to more advanced methods. The most popular and most efficient way of creating a genuine sine-wave output is by pulse-width modulation (PWM). This starts with a sinusoidal modulating wave at the desired output frequency, and a triangular carrier wave at the desired switching frequency. These two waveforms are fed to a comparator: an electronic comparison circuit whose output is ‘HIGH’ if the instantaneous value of the modulating wave is greater than that of the carrier wave, and ‘LOW’ otherwise. The result is a train of pulses repeating at the carrier frequency, with the width of each pulse proportional to the value of the modulating wave. A spectrum analysis of this waveform would show that it contained a component at the modulating frequency, a component at the carrier frequency, and the harmonics of the carrier frequency. This pulse train is used to operate the inverter’s switches, so that a high-power replica of it emerges from the inverter’s AC output terminals. A low-pass filter is then used on this to remove the carrier frequency and its harmonics, while letting through what turns out to be a very good reconstruction of the modulating wave. See [4] for more information.

The task of the low-pass filter is eased by making the difference between carrier and modulating frequencies very large. By using MOSFET switches, which perform very well at high frequencies, it is easily possible to have a carrier frequency of, say, 50 kHz, and so the carrier can be greatly attenuated, while the desired 50 Hz component is unaffected, using only a simple second-order filter. In this way, it is theoretically easy to meet the distortion spec.

The PWM generator can be simplified even more by using hysteresis (aka bang-bang) control, which is a technique borrowed from commercial inverters that drive induction motors. In bang-bang control, the carrier wave is dispensed with, and the modulating waveform is compared directly with the AC output. The result of the comparison is used to control the power switching stage: If the output is too great, the power switch is turned OFF so that it begins to decrease, and if it is too small, the switch is turned back ON. This can be thought of as forcing the inverter to generate its own carrier by self-oscillating, and for it to work efficiently, there are two necessary conditions. Firstly, the comparator must have hysteresis (a dead band where no action is taken) and secondly there must be a low-pass filter included between the inverter switches and the output which is being controlled. If there were no filter, the output would change immediately the power switch changed state. Between them, the hysteresis band size and the filter cutoff frequency determine the effective carrier frequency.

The bang-bang approach is also well-suited to grid-intertied operation. The conventional form of PWM generation is not very suitable, because the modulating input controls the output voltage. So, the output of a classical PWM inverter appears as a voltage source. Now, the grid is also a voltage source, and so there are two voltage sources connected together by the very small reactance of the inverter’s output filter. Thus, tiny changes in the magnitude or phase of either voltage would cause large and dangerous current surges.

However, by sensing the inverter’s instantaneous output current, and using bang-bang control acting on this, then the inverter appears as a current source instead, and the problem is avoided. The necessary low-pass filter takes the form of an inductor in series with the inverter’s output terminals.

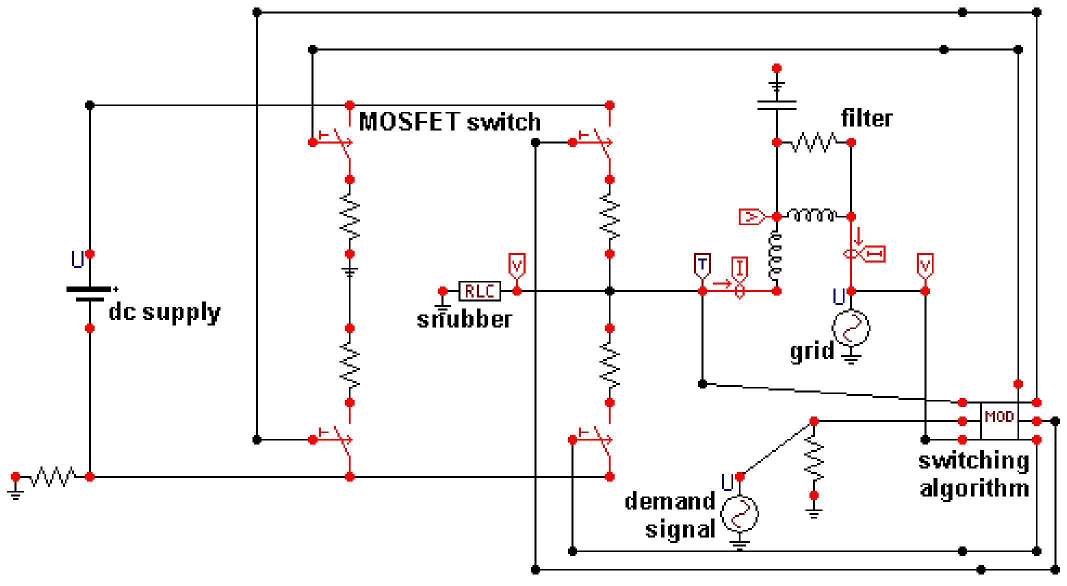
A.6. Simulation

An inverter of the design described above was simulated using ATP, the Alternative Transients Program [5]. At this early stage, the precise design details were not yet known. For this reason, and also to save computing resources, a simple representative circuit was simulated. The model used was a combination of two parts: a schematic of the switching circuit created with the ATPDraw graphical front end, and a description of a bang-bang controller algorithm/circuit made with ATP’s MODELS language. The ATP simulator solved the electrical circuit, and also executed the controller algorithm at every timestep of the solution. This model was mainly used as a design aid, to explore the effects of different kinds of output filter and different hysteresis bands. The results gave confidence that the proposed design could be made to work and give good performance. Figs. A.4-A.7 show the schematic diagram, the control algorithm description, and some sample waveforms. The circuit modelled here uses a second-order LC output filter.

A.7. Practical design issues

Once the basic topology and strategy of control had been developed, the next step was to design a practical electronic circuit around them.

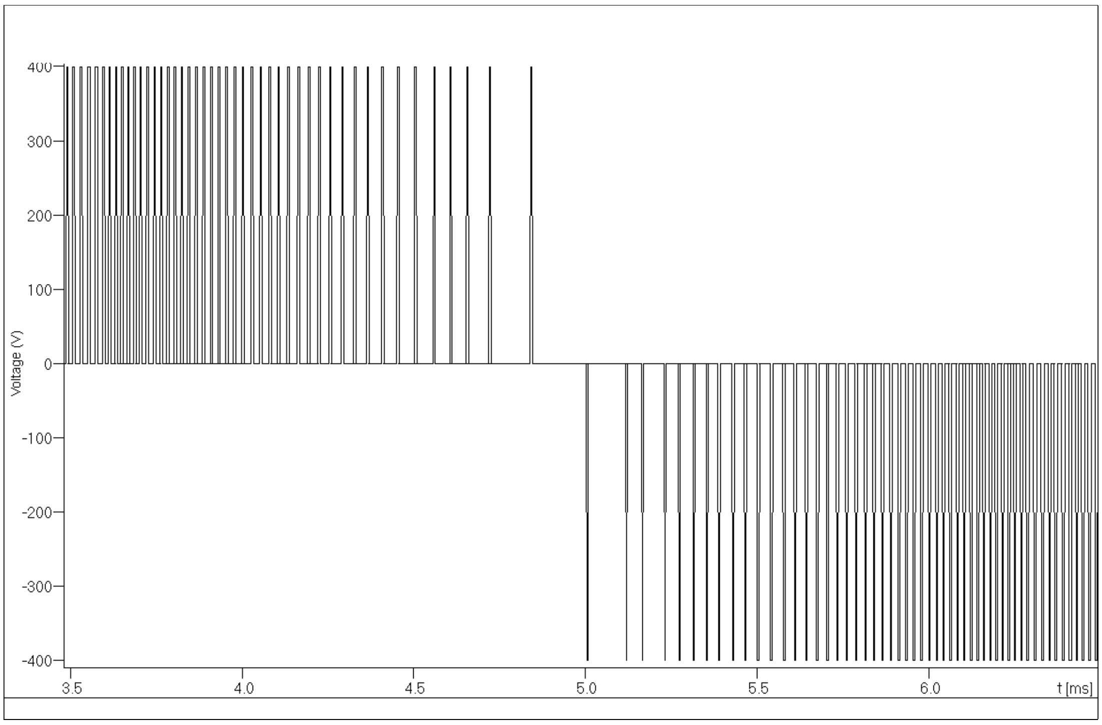
The main problem was to find a way of sensing inverter output current for the bang-bang control. The nature of the challenge was this: In accordance with feedback control theory, the error performance in a system of this kind is mainly limited by the sensor. The sensor would have to be accurate to within a few per cent to meet the specification, with a low DC offset being vital to prevent DC injection to the line,



(Fig. A.4: ATPDraw schematic of the inverter model)

```
MODEL bang3
DATA
  db                      -- dead band magnitude (amps)
INPUT
  IL,VDEM,VGRID                   -- inductor current, demand current
OUTPUT
  S1,S2,S3,S4             -- to TACS switches
VAR
  S1,S2,S3,S4
HISTORY
  S1 {dflt:0}
  S2 {dflt:1}
  S3 {dflt:1}
  S4 {dflt:0}
  IL {dflt:0}
EXEC
  if VGRID>0              -- first do left hand bridge (50 Hz)
  then
    S1:=0
    S2:=1
  else
    S1:=1
    S2:=0
  endif
  if IL>(VDEM+db)                 -- now do right-hand half (hysteresis)
  then
    S3:=0
    S4:=1
  endif
  if IL<=(VDEM-db)
  then
    S3:=1
    S4:=0
  endif
ENDEXEC
ENDMODEL
```

(Fig. A.5: ATP MODELS description of bang-bang controller)



(Fig. A.7: Simulated switching waveform: close-up shown, note time scale)

(Fig. A.8: Simulated line current waveform)

or saturation of the output transformer if one was used. It would also require high bandwidth so that the bang-bang control would work properly. Finally, it would need excellent immunity to interference from the inverter’s own HF output voltage.

A current transformer would seem attractive, because of its immunity to interference. Unfortunately, the sensor needs to be DC-accurate, and transformers do not respond to DC at all. This could probably be got round by ingenious circuitry, but even then the transformer would have to be physically large if it was to handle the 50 Hz component of the inverter output without saturating. Its leakage inductance might then cause problems with accurate measurement of the HF components.

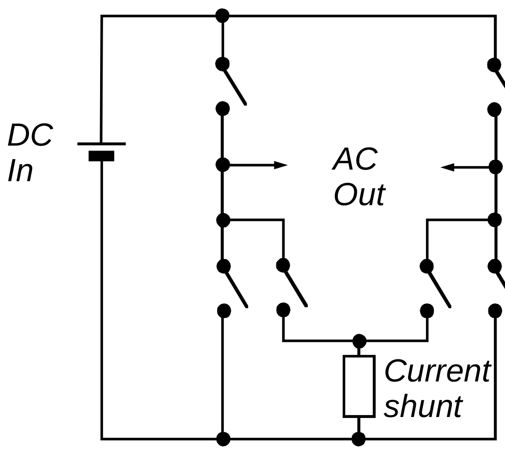
A natural contender, then, was the Hall-effect current transformer. It can respond to DC, and has a high bandwidth. Unfortunately, though, the devices investigated all had a very large DC offset, which ruled them out.

The remaining option was the humble sense resistor. This had been ruled out, because it would have to be placed in series with the inverter’s output. In this situation, the small voltage across the resistor would have to be separated from a large high-frequency common-mode signal.

This seemed to be a major problem. Solutions like opto-isolation were investigated and ruled out on grounds of complexity or inaccuracy. It was finally solved by a modification to the switching circuit (See Fig. A.8)

It works because the full-bridge (as it is usually set up) has only one side driven with HF at a time. The sides change roles with every half-cycle. The inactive side has the lower switch turned on continuously, in effect earthing the un-driven end of the load.

So, two extra transistors were added solely for this function. Instead of taking the load current directly to earth, they were arranged to divert it through a current-sensing resistor. Crucially, one end of this resistor is now earthed, solving the common-mode interference problem. A simple op-amp differential amplifier finishes off any remaining common-mode due to voltage drops in the ground paths.



(Fig. A.8: Modified full-bridge inverter for easier current sensing)

A.8. Control circuits

A.8.1. PWM generator

This is perhaps a misleading title, because the bang-bang method of control causes an oscillation involving the entire circuit. So, it is hard to say just which part is actually the PWM generator. For the purposes of this discussion, though, it will be defined as the part from which the PWM waveform first emerges, which means the comparator circuit which compares the modulating wave to the measured current. The comparator is U2A of Fig. A.9. It compares the DAC output (smoothed slightly by C2 and buffered by U1B) with the voltage across the current shunt, amplified ten times by differential amplifier U1A, and smoothed to eliminate high-frequency interference by R5, C1. R6, R7 and R8 set the amount of hysteresis, while R9 is a pull-up resistor required due to the design of the comparator chip (open-collector output) The PWM waveform appears at U2A output, and is fed to the power switching circuitry via a switching logic circuit, described next.

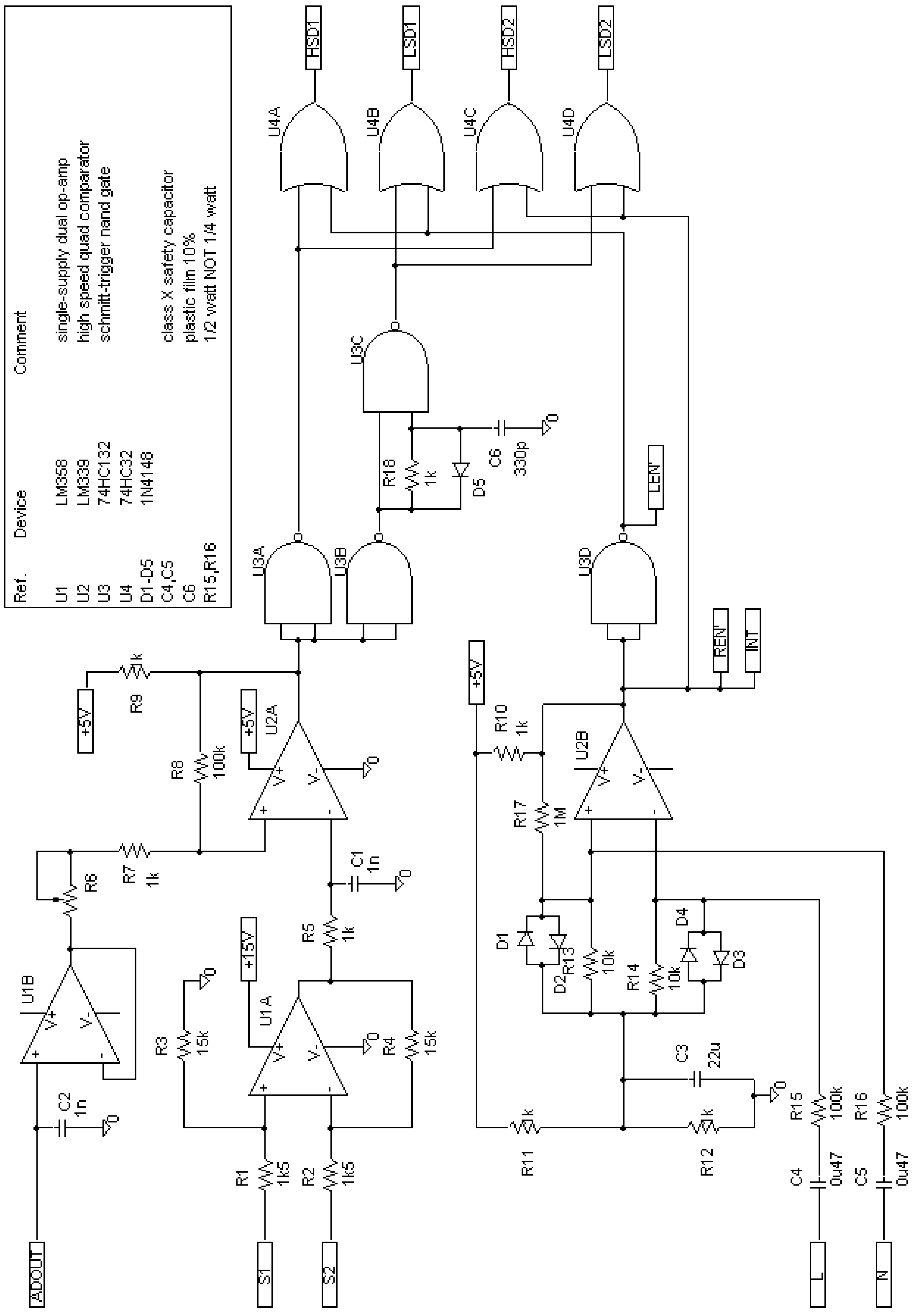
A.8.2. Switching logic

The main issue to be addressed is how to drive the switches in the proper sequence. There are six separate switching elements, and yet the PWM comparator only gives one output. Some sort of additional logic is required to sequence the operation of the switches.

With reference to Fig A.10, the basic strategy of operation is this. During one half-cycle of the grid voltage, the PWM waveform is applied to S1, S2. S1 receives the normal waveform, and S2 receives its inverse, so that S1 is ON when S2 is OFF and vice versa. The result is an amplified copy of the PWM waveform at the junction of S1 and S2.

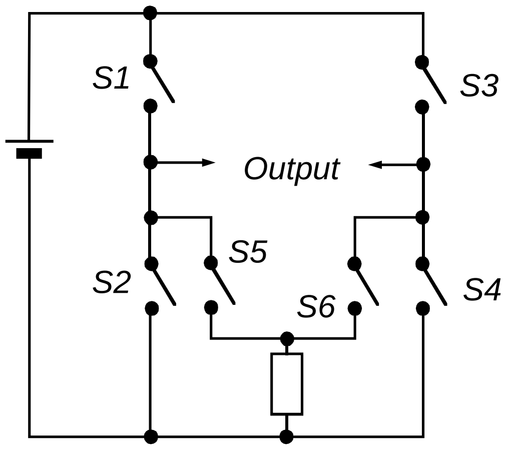
While this is happening, S3 and S4 are both OFF. S6 is permanently ON, connecting the other output terminal to ground.

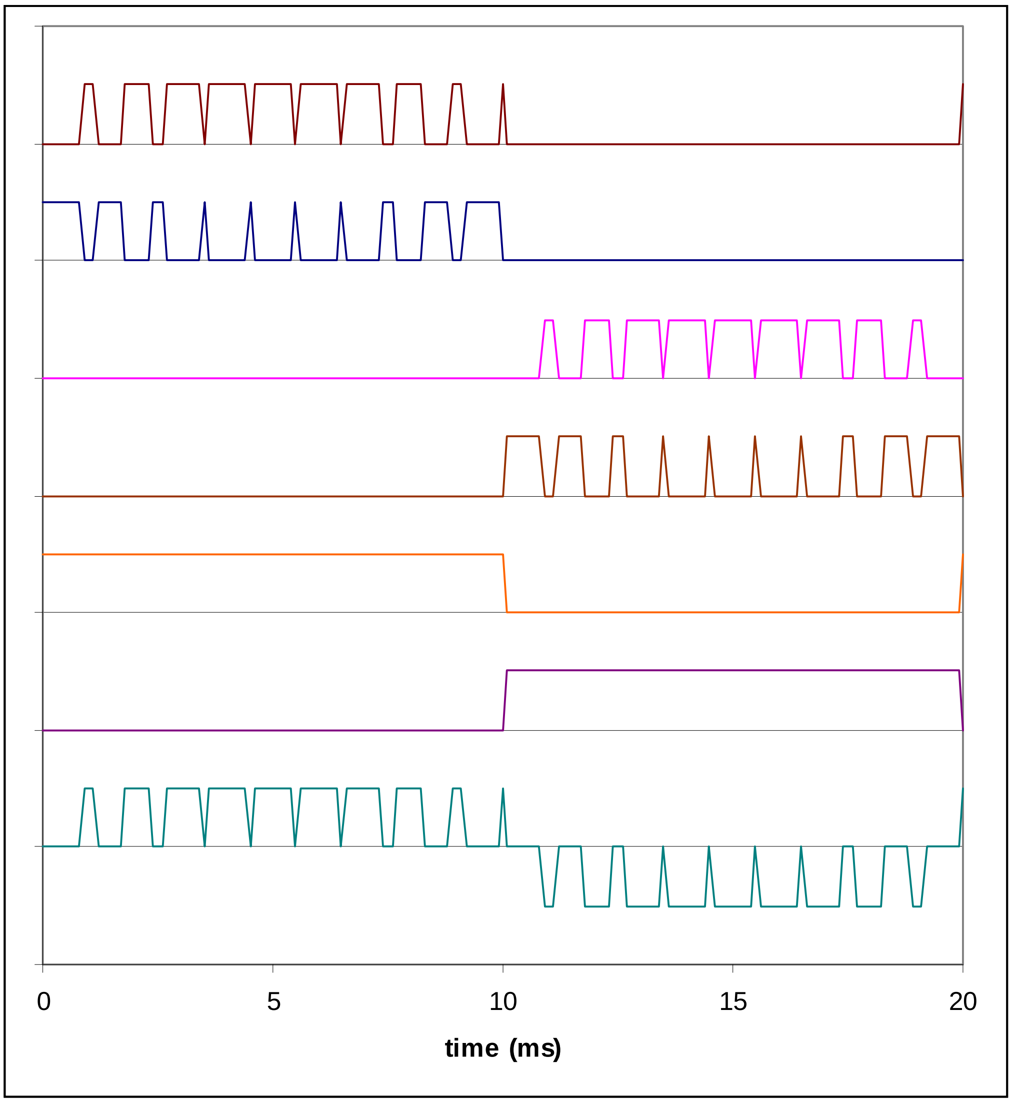
During the next half-cycle, the roles must be swapped over. S3, S4 receive the PWM signal, while S1, S2 are off, and S5 is ON.



Ref.	Device	Comment
U1	LM358	single-supply dual op-amp
U2	LM339	high speed quad comparator
U3	74HC132	schmitt-trigger nand gate
U4	74HC32	
D1-D5	1N4148	
C4,C5		class X safety capacitor
C6		plastic film 10%
R15,R16		1/2 watt NOT 1/4 watt

(Fig.A.9: PWM generator and switching logic circuits)





(Fig.A.10: Switching circuit and simulated waveforms. Carrier frequency reduced for clarity)

These rules can be turned into a simple collection of digital logic gates by using well-known techniques. Firstly, the inputs to the circuit must be defined: they are P, the PWM pulsetrain, and G, the polarity of the grid voltage. (G=0 when grid is negative and 1 when it is positive) Let the outputs be S1, S2, S3, S4, S5, S6, and let a '1' indicate that the corresponding switch is closed. A truth table can then be drawn. In the interests of clarity, this will not be done in the conventional manner; instead of the usual 0 or 1, the outputs can also be P or P'.

G	S1	S2	S3	S4	S5	S6
0	0	0	P	P'	0	1
1	P	P'	0	0	1	0

(Table A.1)

From this it is easy to write Boolean expressions for each output:

$$S1=PG \quad \textbf{(Eq. A.1)}$$

$$S2=P'G \quad \textbf{(Eq. A.2)}$$

$$S3=PG' \quad \textbf{(Eq. A.3)}$$

$$S4=P'G' \quad \textbf{(Eq. A.4)}$$

$$S5=G \quad \textbf{(Eq. A.5)}$$

$$S6=G' \quad \textbf{(Eq. A.6)}$$

Before proceeding from here to an actual logic circuit, there is one detail that must be taken care of. S1, S2 are in series across the DC bus, as are S3, S4. If both switches in either pair should turn on simultaneously, the result is a short across the DC bus. Even if this only happens for a matter of microseconds, a very high current can momentarily flow, which leads to inefficient operation and possible damage to the circuit. To avoid this destructive "shoot-through", it is normal practice to arrange a small delay in the turning-on of each switch, so that the previous one has ample time to turn off. The precise nature of the delay required depends on the switching speed of the power circuit, which will be discussed in more detail in a subsequent section. For now, suffice it to say that by the nature of their driver circuits, S1 and S3 will naturally turn on and off somewhat slower than S2, S4. Therefore it is sufficient to delay the turning-on of S2, S4. The logic expressions for these can be rewritten:

$$S2=Q'G \qquad \textbf{(Eq. A.7)}$$

$$S4=Q'G' \qquad \textbf{(Eq. A.8)}$$

Where Q’ is the same as P’ but with a delay introduced in each low-to-high transition. Components R18 and C6 introduce this delay in the circuit.

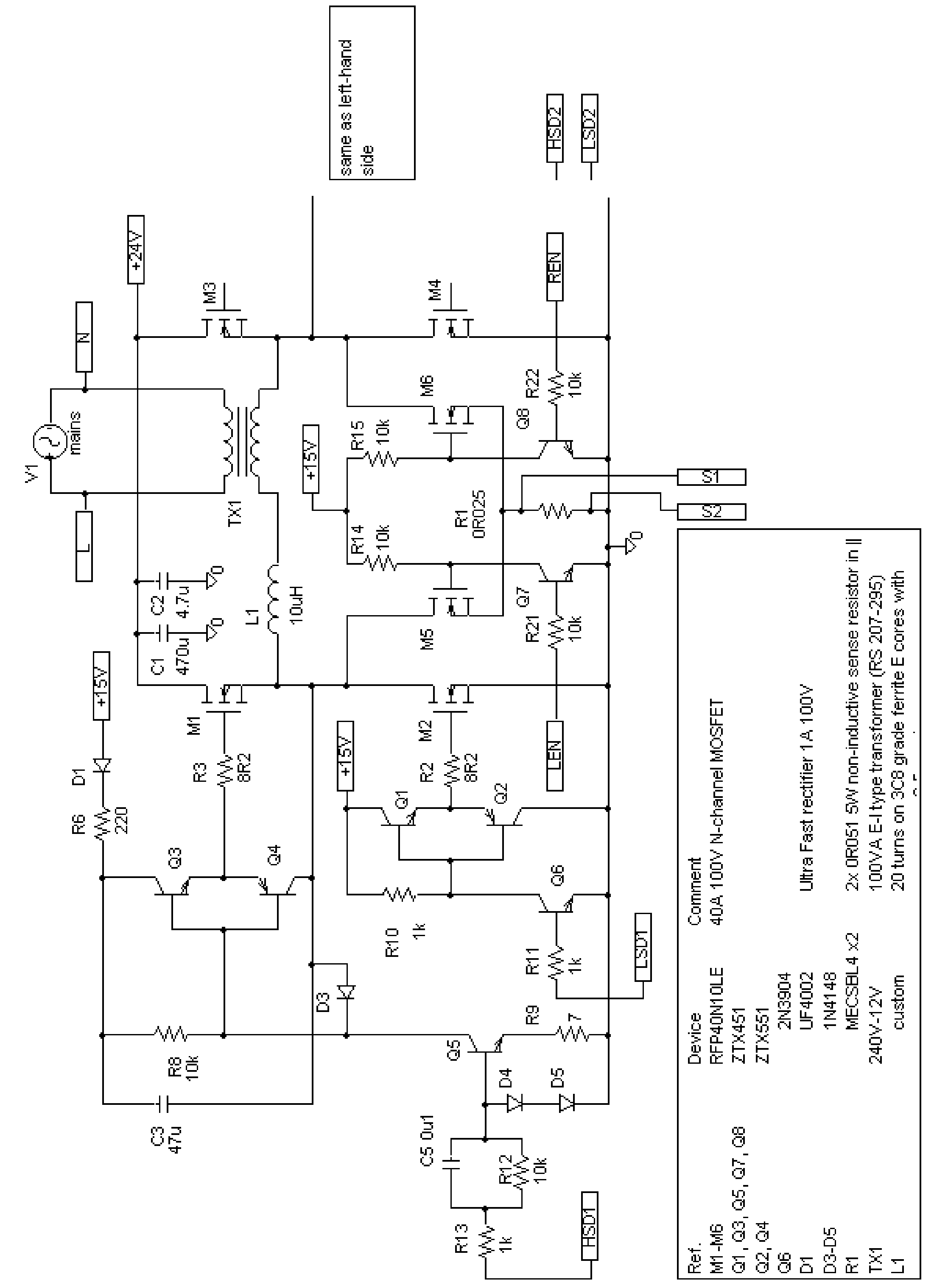
Both switching logic and PWM generator are shown in Fig. A.9. The grid voltage signal G is derived from U2B and associated components. These act as a differential comparator measuring the voltage between live and neutral of the mains input. It can be thought of as a differential amplifier with very high gain, so that the output is a square wave. C4, C5, R14, R16 attenuate the mains voltage to a safe level.

A.8.3. Drive circuits

There are a few peculiarities involved in actually applying these drive signals to MOSFETs, which will be explained here. A certain amount of specialised drive circuitry is needed, and the circuit used is shown in Fig. A.11, to which the following explanation refers.

The first difficulty is that the gate of a MOSFET has considerable capacitance. It is essential to turn the MOSFET on and off as quickly as possible, and so high peak currents are required to charge and discharge the gate capacitance. In order to provide the current, a complementary pair of transistors Q1 and Q2 are used as emitter follower buffers. These are special transistors designed for the application and can supply peaks of up to 2 A. MOSFETs M5 and M6 do not have this drive circuit because they only operate 50 times per second and so do not require high-speed switching.

The second difficulty is that the drive signal must be referred to the MOSFET source. In the case of M2, M4, M5, M6 this is not a problem since the source is grounded. But M1 and M3 have their sources connected to the output terminals. When they are turned on, the source voltage will rise to the DC bus voltage, and so to keep them switched on, the gate voltage must be kept higher than the DC bus. This is taken care of by a ‘bootstrapping’ circuit, which supplies the gate drive voltage from a



Ref.	Device	Comment
M1-M6	RFP40N10LE	40A 100V N-channel MOSFET
Q1, Q3, Q5, Q7, Q8	ZTX451	
Q2, Q4	ZTX551	
Q6	2N3904	
D1	UF4002	Ultra Fast rectifier 1 A 100V
D3-D5	1N4148	
R1	MECSBL4 x2	2x 0R051 5W non-inductive sense resistor in ll
TX1	240V-12V	100VA E-I type transformer (RS 207-295)
L1	custom	20 turns on 3C8 grade ferrite E cores with

(Fig. A.11: Circuit diagram of driver stage. One half only shown.)

capacitor, C3, connected to the source. This capacitor is charged from the auxiliary 15 V supply via R6, D1 during periods when M2 is on and the source is grounded. A side-effect of this circuit is that it slows down the switching action compared to the non-bootstrapped version. This is because the voltage at Q5 collector must swing by a larger amount, and hence the Miller effect in Q5 will be greater. The network C5, R12 injects extra base current to compensate for this as much as possible, but it is still somewhat slower.

It should be noted that these drive circuits all invert the signal: a HIGH input signal turns the associated MOSFET OFF. This is taken care of by modifications to the switching logic.

A.8.4. Reference generator

If this circuit is to produce a sinusoidal output, it requires a sinusoidal modulating wave as a reference. In order to connect to the grid, the reference must be phase-locked to the grid voltage. Also, the power output is controlled by varying the magnitude of this wave, so to ensure accurate results, the magnitude must be stable, and controllable in an accurate and repeatable manner.

Various ways of generating this were investigated, and there seemed to be two attractive methods. The first was to use traditional analogue techniques. The reference would be generated by a sinusoidal voltage-controlled oscillator (VCO) with stable output amplitude. A phase-locked loop (PLL) would be used to synchronise this with the line voltage. To vary the power level, an analogue multiplier would be used to multiply the reference waveform with a DC power command voltage. This would ultimately be derived from a digital-to-analogue converter (DAC) under control of a computer running dispatching software.

The main competitor was a microcontroller-based system, where all the processing is done by a computer program, and a DAC outputs the finished reference waveform. This system could be called direct digital synthesis (DDS) In this, the reference waveform is stored digitally as a look-up table (LUT) in memory. At regular intervals, successive cells are read from the LUT, and sent to the DAC. However, for this application, the basic DDS is elaborated somewhat. To implement power control, each value from the LUT is simply multiplied by a power command value, read in from the host computer via some kind of digital interface. Synchronisation is achieved by sensing the zero-crossings of the grid voltage, and having the program adjust its timing until its own zero-crossings coincide with them; this is just a digital version of the phase-locked loop used in electronics.

The choice between these two techniques is not difficult. Although the analogue system is conceptually easier to understand, there are serious challenges associated with making a VCO whose output amplitude remains constant to within 1%, and a multiplier which is similarly stable. The digital version, while being somewhat more troublesome to construct and de-bug, avoids these problems altogether, and so is naturally superior.

A.8.5. Choosing a processor

The first task was then to choose a suitable microprocessor, from the hundreds of types available. In order to do this, a quick estimate was made of the computing power required. Firstly, to meet the 5% distortion target, 8-bit precision would be ample. This then gives an idea of the number of points required per cycle; if the output can only take 256 (2^8) possible values, then there is no point in sending out more than 256 points per quadrant of the sine wave.

Secondly, for each point, two 8-bit numbers are to be multiplied together. Unless the processor has a hardware multiplier (and only more complicated and powerful processors do) this is quite an intensive task; the number of instructions required is at least the square of the bit depth of the smaller of the two numbers. Therefore, each point will need at the very least 64 instructions; say 100.

So, given that there are 50 cycles per second, and 4 quadrants in each cycle, the processor needs to execute approximately: 50*4*256*100=5,120,000 instructions per second.

Now, to estimate the amount of memory required: The LUT will contain around 256 values, each of which will consume one word of memory. Then, there are 100 instructions executed to produce one data point. (This is not strictly accurate; the program might consist of 10 instructions, looped 10 times. However, it is adequate for a rough guess.) There will also be code for the phase-locked loop, and for reading in power commands. This is assumed to be 100 instructions or less; otherwise, it would not have time to execute once per zero-crossing. Finally, there will be perhaps another 200 instructions for setup and error-detection. Each instruction will also use one word, so giving a total of 656 words.

Therefore, a suitable microcontroller would have around 700 words of non-volatile memory, and an execution speed of about 5 million instructions per second (MIPS).

Arizona Microchip's PIC16F84-10 chip looked quite attractive, due to its simple reduced-instruction set computer (RISC) architecture, low price, and ease of use; it can be programmed using an ordinary PC and the very simple NOPPP ('No-Parts PIC Programmer'). However, while it had an ample 1,024 words of memory, its maximum speed was only 2.5 MIPS. This would be fast enough, though, if the precision were reduced to 7-bit, and 64 points used per quadrant instead of the 128 implied by the precision. This would still provide a good enough waveform to meet the distortion spec.

A.8.6. Firmware

The firmware program is listed at the end of this Appendix, and an explanation of the code is also given.

A.8.7. Support circuitry

The PIC does not quite do everything by itself. It requires a few supporting components; the circuit is shown in Fig. A.12. The quartz crystal is a standard part: the frequency of 9.8304 MHz may seem odd, in fact it was a convenient multiple of 50 Hz.

(50 cycles per second

*2 half-cycles per cycle

*128 D/A conversions per half cycle

*192 instructions per conversion

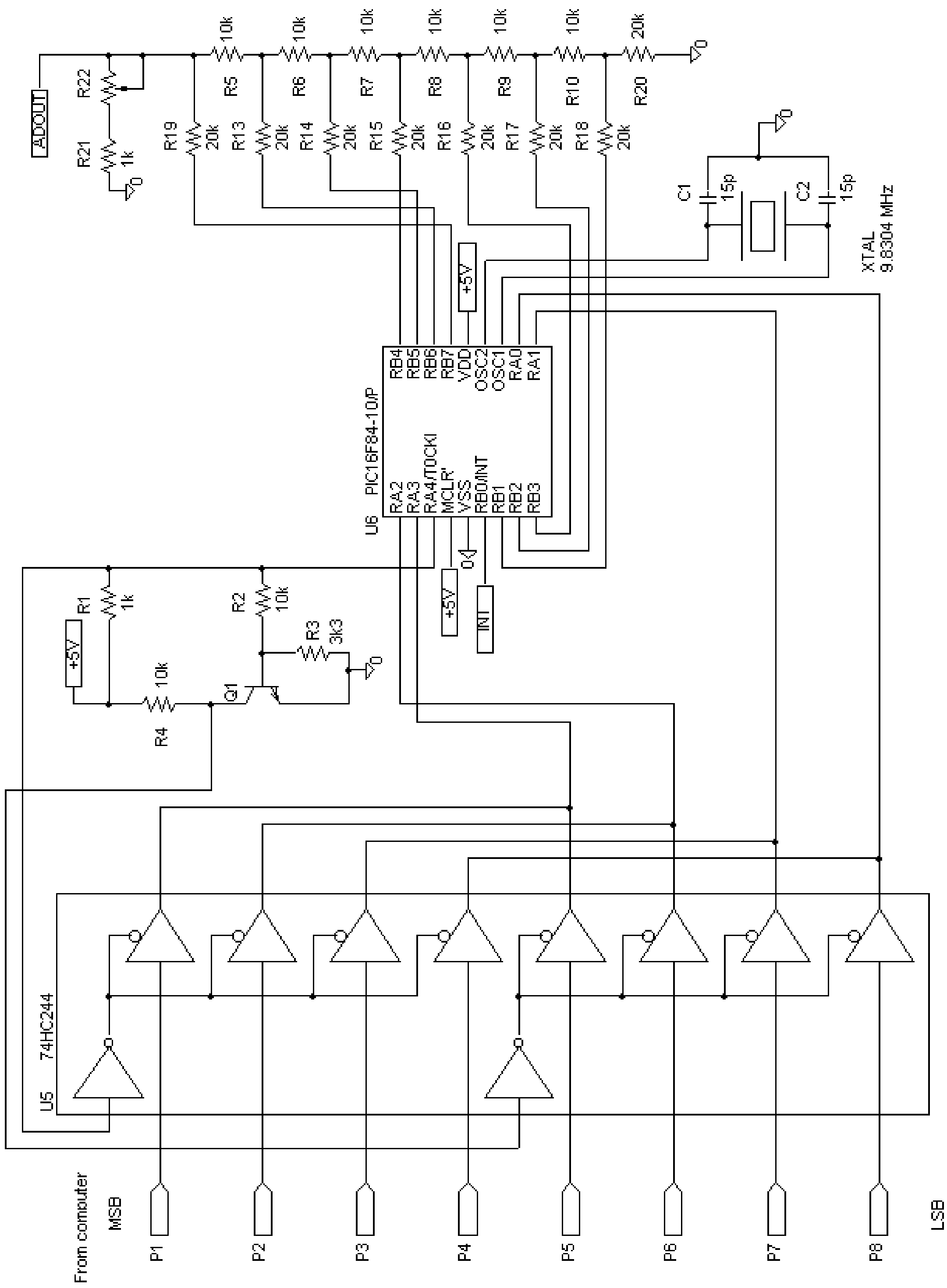
*4 clocks per instruction =9,830,400)

Also needed is a digital-to-analogue converter, which is formed by resistor network R5-R20, a classical R-2R ladder. R21, R22 allow the output voltage to be adjusted.

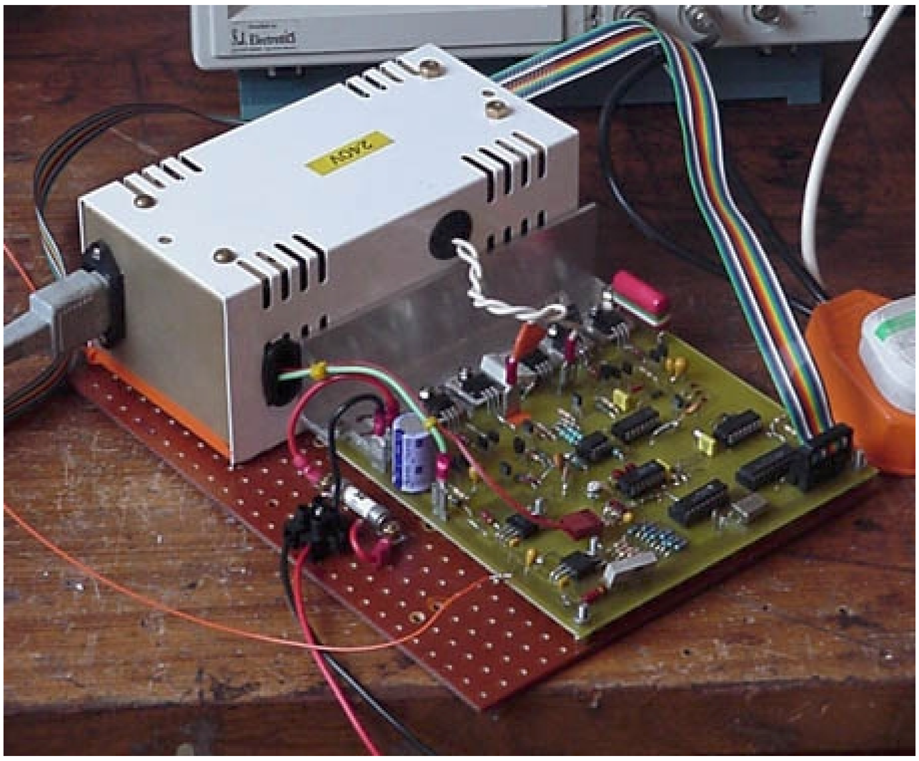
R1-R4, Q1, and U5 multiplex the 8-bit parallel input down to 4 bits in order to save I/O pins. U6 is the microcontroller itself.

A.9. Building and testing the Mark One

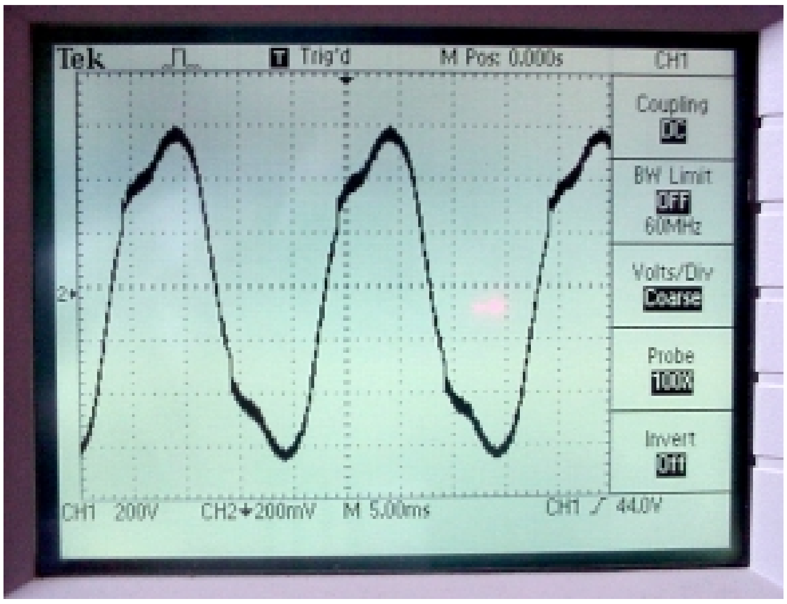
This design evolved over a period of a few months. The various subsystems were tried out on breadboards in the lab, and once they seemed to be functioning happily in isolation, it was time to build a prototype and try out the whole system. The emphasis at this stage was not on perfect performance or error-free operation, but simply to provide a proof of concept. A printed-circuit board (PCB) was made and stuffed with components, and the various stages of the circuit were tested in isolation, before connecting everything together. Inevitably, a number of design flaws were discovered, and changes had to be made to the circuit.



(Fig. A.12: Circuit of microcontroller and support components)



(Fig. A.13: Mark One inverter under test)



(Fig. A.14: Line current of Mark One operating at 100% output. Y Scale: 200mA/div)

A.10. Lessons learnt from Mark One

A.10.1. Odd spikes

The most puzzling anomaly in the Mark One's operation was a strange disturbance in the current waveform. When the switching devices change state, the rate of change of current is supposed to reverse. This it did, but accompanied by a very large transient (a 'spike') which disturbed the bang-bang control system quite severely. To allow proper operation, the spikes were reduced by low-pass filtering the current signal, but the true cause was discovered quite by accident.

If it is to work correctly, the inverter requires a filter inductance in series with the load. Quite a small inductor had been used, wit the intention of using the leakage inductance of the transformer to help with the filtering. Unfortunately, the transformer winding also has capacitance to the core, which is earthed. In use, one end of the winding was connected directly to the unfiltered HF output, and the very high rate of change of voltage (dV/dt) caused large transient currents to flow to earth via the winding-core capacitance. Swapping the leads around, so that the inductor was in series with the end of the winding having most capacitance to the core, reduced the problem considerably. A more permanent fix would be to use two inductors, one for each winding end.

A.10.2. Excessive losses

The Mark One also suffered from excessive losses. It was only 85% efficient at rated output. No single component was really responsible; the losses were equally spread amongst the transformer windings, transistors, and current sense resistors.

A.10.3. Too much distortion

Also, it did not meet the 5% distortion target. The source of the distortion turned out to be the transformer; a commercial unit designed with economy in mind. Discussions with a manufacturer of transformers revealed that it is common practice to design for a peak flux density of 1.5 Tesla, which is actually greater than the saturation point of the core material, 1.3 T. Therefore, the core saturates towards the end of every cycle, reducing the inductance dramatically and causing spikes of magnetising current. A more conservative transformer design would be the obvious solution.

A.10.4. Latch-up

The final insult was that it occasionally suffered latch-up. This was a frustrating condition where once in a while all six MOSFETs would turn on simultaneously as soon as power was applied to the circuit. The result was a complete short-circuit across the DC bus, normally followed by some kind of small explosion. The root cause of this was that the MOSFET driver circuits were inverting, i.e. a HIGH input to the drive transistor base turns the MOSFET off. If the power to the driver circuits were to come up before the power to the logic circuits, then the drivers would start operating while all their inputs were still low. The solution would be to sequence the power supply rails so that the driver circuits were powered up last of all.

A.11. Mark Two

The Mark One inverter had served its purpose as an experimental prototype. However, it was not powerful enough, and in any case had design flaws which would require serious revision. It seemed that the best course of action was to build another inverter.

With these shortcomings in mind, work began on the Mark Two. The most important goal for this was extra power; 600 watts instead of the Mark One’s 100. Achieving this extra power while meeting the 90% efficiency target required some careful planning. The first step was to draw up a loss budget.

A.11.1. Loss budget

Up to 10% of the power can be lost. Now, due to the design of the inverter’s output circuit, every component passes the same current. This is assumed to be 50 A (600W at 12V: the reason for using 12V will be discussed later) So, the maximum allowable circuit resistance is the value which will dissipate 10% of 600W, when passing 50A. From Ohm’s law this is 0.024Ω.

Now, it is assumed that half of this resistance is in the transformer primary and secondary lumped together. So, all the other parts must come in at under 0.012Ω.

A.11.2. Transistors

‘UltraFets’ made by Intersil were prime candidates. They are inexpensive devices with a very low on-state resistance; only 0.006Ω. Four were used in parallel in each switching position; a total of 24 devices. Since the current flows through two sets of switches in series, the total resistance will be 0.003Ω.

A.11.3. Current sense resistor

A commonly-available 30A 75mV meter shunt was chosen. This has a resistance of 0.0025Ω.

A.11.4. Wiring

The power connections were specified as copper sheet 2mm thick by 12mm wide. The total length is about 150mm. The resistance of this is 0.0001Ω: small enough to ignore altogether. (The inductance was not- but that is another story)

A.11.5. Transformer

Bearing in mind the requirement for a ‘clean’ magnetising current, a custom transformer had to be constructed. An off-the-shelf 625VA toroidal transformer, with 230V primary and two 40V secondaries, and 5% volt-drop at full load (regulation) was chosen as a base.

First of all, the magnetising current was measured. Although the mains voltage was near 250V on the day of the experiment, the current was very low; less than 30mA RMS. It would have been difficult to measure with more accuracy because the switch-on surge would have destroyed a sensitive meter. The transformer was also silent in operation with no buzzing. In any case, if the magnetising current had been higher, the transformer could have been modified by adding extra primary turns.

Next, 10 turns of wire were placed on it and the voltage on this winding measured; 4.99V. Thus, each turn gives 0.5V.

But, what voltage should the new winding be? The inverter will malfunction if the DC terminal voltage ever drops below the peak AC voltage. Now, the DC voltage is nominally 24. But, being supplied from a lead-acid battery, it could drop as low as 20.

So, the peak AC should be a little less than $\frac{20}{\sqrt{2}}$ =14V. Leaving a little more room for the inverter's internal 10% voltage drop (design efficiency is 90%) the result is around 12V, therefore 24 turns.

Next, calculate the thickness of secondary conductor required. Assume that the 5% volt drop is shared equally between the primary and secondary, so that the volt drop in the new secondary should be 2.5%. It is also known that at full power, the current in the secondary will be nearly 50 A. So, it is possible to calculate the resistance of the secondary that will drop 2.5% of 12V= 300mV when passing 50 A. It is 0.006Ω.

Next, knowing the core dimensions, the length of one turn can be calculated; 0.18m. Therefore, 23 turns will use 4.14m of wire. The resistivity of copper is 1.72 x 10^{-8} Ω–m; so if it is to have a resistance of 6 mΩ, a conductor of this length must have a cross-section of 1.19 x 10-6 m^2, or 12 mm^2. A single copper wire of this size would be very difficult to handle, so a number of smaller wires in parallel is preferable. 2mm diameter wire is easily available, and using four strands of this gives a cross-section of 12.6 mm^2. However, there appeared to be plenty of room on the core and so it should be possible to be conservative, and use five.

So, the existing secondaries were unwound, and replaced with five 24-turn windings of enamelled copper wire, 2mm diameter, connected in parallel.

A.11.6. Filter components

Say that the switching frequency is not to exceed 30 kHz. Now, it was observed from the ATP simulation, and experiment with the Mark One, that the highest switching frequency occurred when the instantaneous line voltage was about half its peak value: that is, 12V.

$$\frac{di}{dt} = \frac{V}{L} = \frac{12}{L}$$

(Eq. A.7)

Now say the system is running at a hysteresis of 5% of the peak current; 3.5A. One half switching cycle can then be calculated as the time taken for the current to change by 3.5A.

So,

$$\int_{T}^{T+\tau} \frac{di}{dt} dt = 3.5$$

(Eq. A.8)

Substituting 12/L for di/dt (Eq. A.7) and performing the integration:

$$\frac{12\tau}{L} = 3.5$$

(Eq. A.9)

thus t=0.29L, and f=1/2t=1.72/L

so to have 30kHz, L=1.72/30000=57 μH

So, the requirement is for two 30 μH inductors to handle 71A without saturating. Also, their combined DC resistance must not exceed 0.006 Ω. The options were somewhat limited; it was necessary to use off-the-shelf cores because having custom magnetic assemblies made would be too expensive and time-consuming. The largest ferrite available was Ferroxcube's ETD39, and for each inductor, two of these core assemblies were stacked to double the core area. These were wound with a 5-turn coil made of 5mm dia. copper pipe with 0.7mm wall thickness, and assembled with an 0.8mm (approx.) airgap between core halves.

With these coils, the switching was somewhat faster than ideal: 70-100kHz. This suggested that the inductance was too small. However, since the inverter did not seem to be suffering from excessive switching losses, and larger inductors would have been a major problem, requiring custom-made ferrite assemblies, it was decided to use them anyway.

A.12. Assembly and snagging

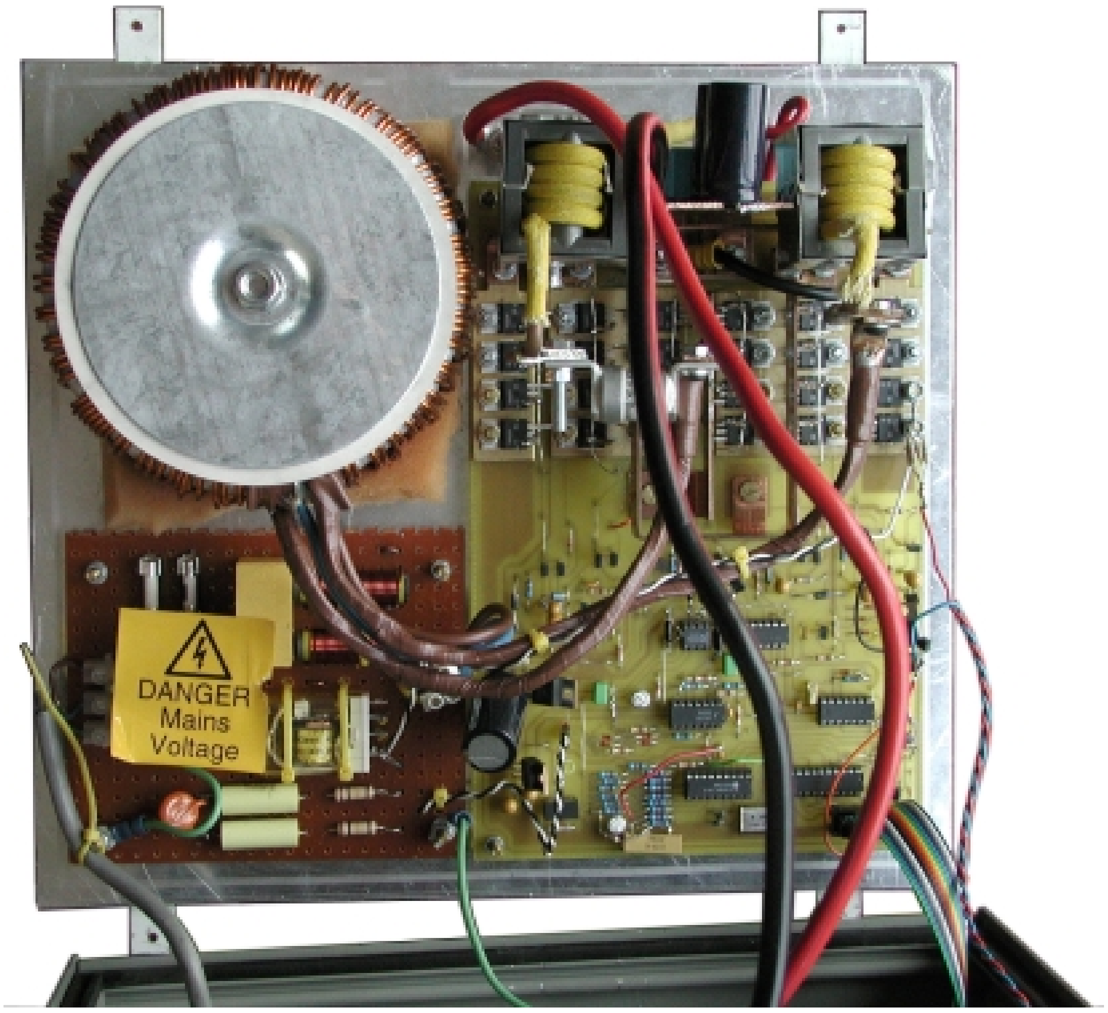
Most of the circuitry was put on a PCB, which was a modified version of the board used in the Mark One. There were three main revisions. First was an extra negative supply (generated by a small DC-DC converter) for the current sense amplifier. This removed the rail-to-rail requirement, so allowing a wider choice of op-amps. Second was a transistor switch allowing the microcontroller to disable all the MOSFET drivers at once, so curing the latch-up problem, and allowing the inverter to be totally shut down. Third was a great enlargement of the power circuit, with 24 MOSFETs instead of the original six. In fact, the power circuitry could not be put directly on the PCB, because the copper was not thick enough to carry the current. So, the board was used only for mechanical support, and the current was carried by brass and copper bus strips joined together by nuts and bolts. Considerable thought was put towards laying out the power circuitry, to give the shortest current paths and smallest current loops possible, and hence minimum resistance and inductance. The drain leads on the MOSFETs were not used, contact being made through the metal tab of the package instead. This gave a lower-impedance electrical contact, and also better thermal contact.

The latter was not of great significance, though, because the transistors were not expected to dissipate very much power in this high-efficiency design. In fact, there were no actual heatsinks as such; the brass connecting strips that carried the transistors were just made a little larger and thicker than electrically necessary, to help in carrying the heat away. (See Fig. A.17) This design decision proved to be reasonable, with transistor case temperatures not exceeding 60 ^{O}C, in a 20 ^{O}C ambient without forced air cooling. Unfortunately, the current shunt ran rather hotter than this, since it was wedged underneath the circuit board and overloaded beyond its design rating. This problem was mitigated by raising the circuit board up so that air from the cooling fan could circulate underneath it.

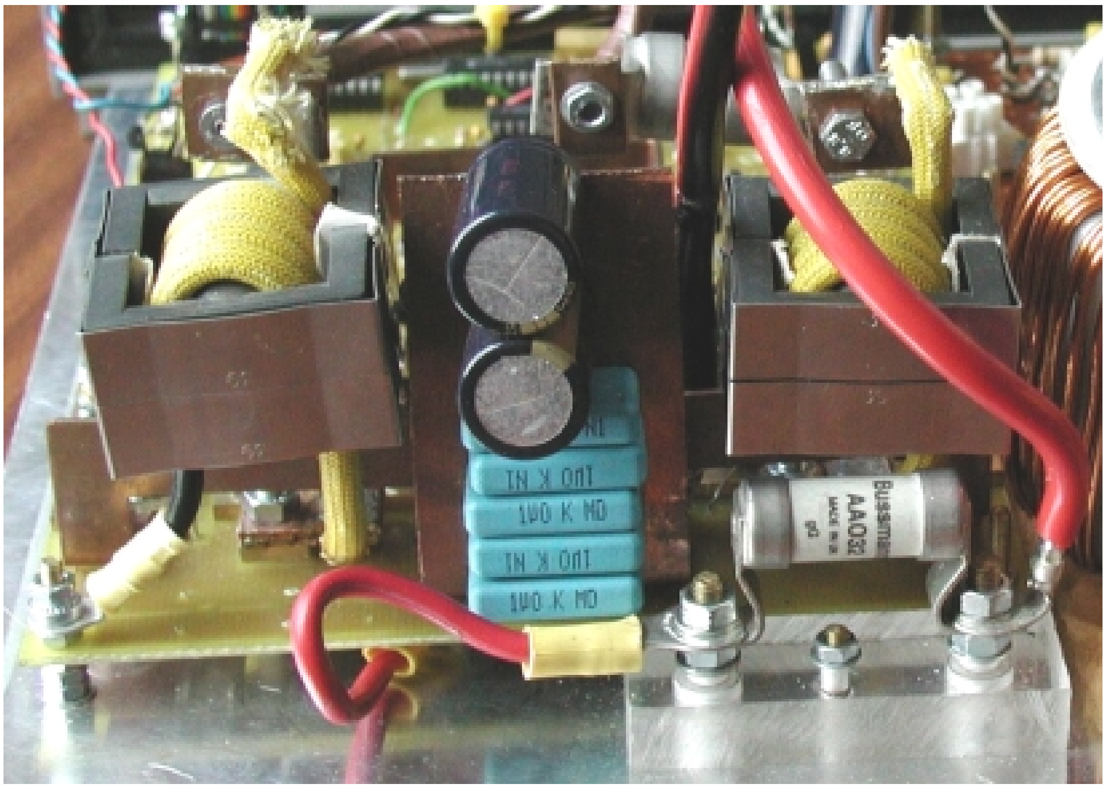
The main circuit board, transformer, and an extra board (containing mains relay, EMI filter, and parts of the zero-crossing detector) were mounted to an aluminium baseplate. This was installed in a casing that had once housed a variable-speed motor drive, as shown in Fig. A.17. The cooling fins, while they look the part, do not actually do anything; cooling is by forced air, sucked in through the fan, and exiting through a grill on the opposite side of the case (not visible in figure). Two high breaking capacity (HBC) fuses were also fitted, one 32 amp in the DC circuit, and one 63 amp in the low-voltage AC circuit.

The inverter was tested in this state. The original electrolytic capacitors used for decoupling were found to be inadequate straight away; they were overheating and there was excessive DC bus ripple, enough to cause the circuit to malfunction. They were augmented by six pulse-rated plastic film capacitors in parallel (each 1μF, 63V) and all the decoupling capacitors were relocated onto two copper strips right at the DC input terminals, instead of being on the PCB as before. This cured the ripple problem. The electrolytics still heated up, but not excessively. Fig A.16 shows the decoupling arrangement.

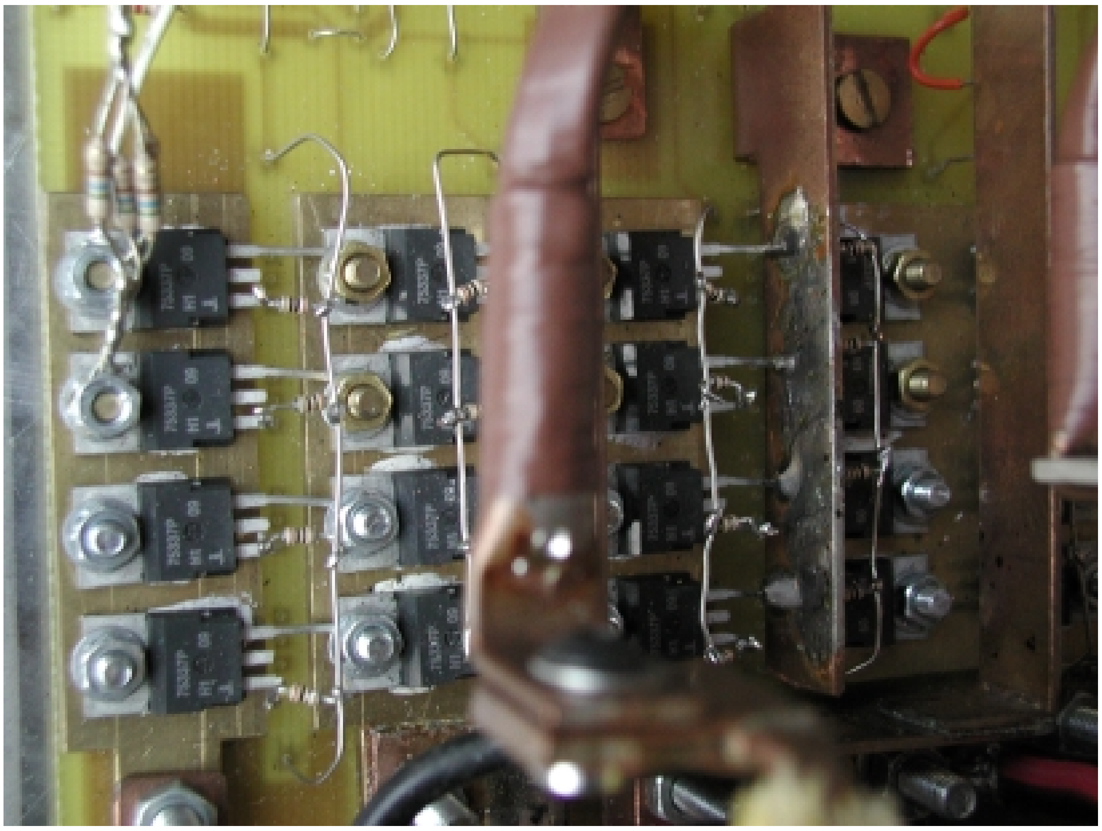
Later, additional circuitry was added; a controller for the cooling fan so that it only started when the inverter was running, and an undervoltage trip for the DC bus. The reason for using an undervoltage trip was that if the DC bus voltage fell too low, the inverter would act as a rectifier instead (due to body-drain diodes in the MOSFETs) and start to backfeed the DC side from the AC side. This might cause unexpected behaviour of the circuit and possible damage. The undervoltage trip made sure that the inverter would be completely shut down in a safe manner before the DC voltage could fall to a potentially dangerous level.



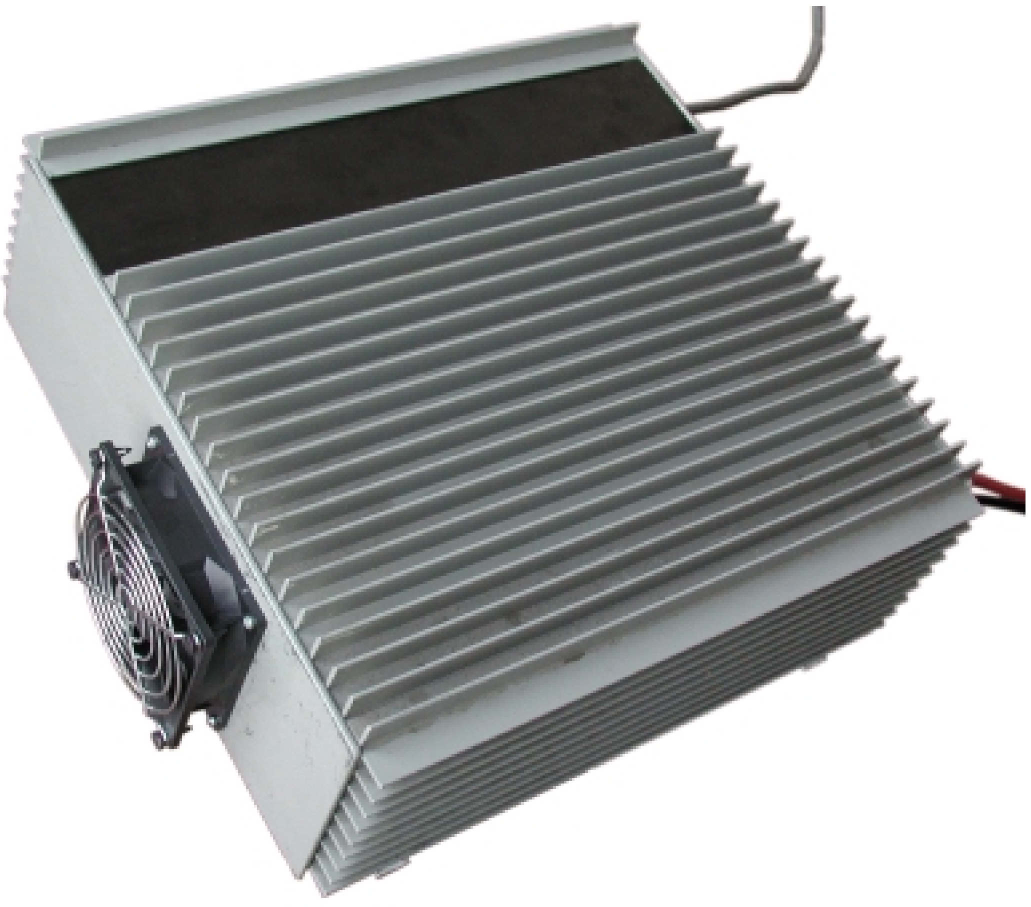
(Fig. A.15: Internal view of Mark Two inverter. Overall size approx. 250 x 300mm)



(Fig. A.16: Filter chokes and DC bus decoupling capacitors)



(Fig. A.17: Detail of power circuit showing transistors and busbars)



(Fig. A.18: Mark Two inverter assembled)

A.13. Testing the Mark 2

Once the unit seemed to be operating satisfactorily, it was time to conduct some tests. First and most important was the efficiency/power control test. For this, the inverter was powered from a series pair of 12V lead-acid batteries, with facilities for measuring DC voltage and current draw. The AC output was fed back to the local grid via a digital power meter (made by Elcontrol, Italy: type VIPD)

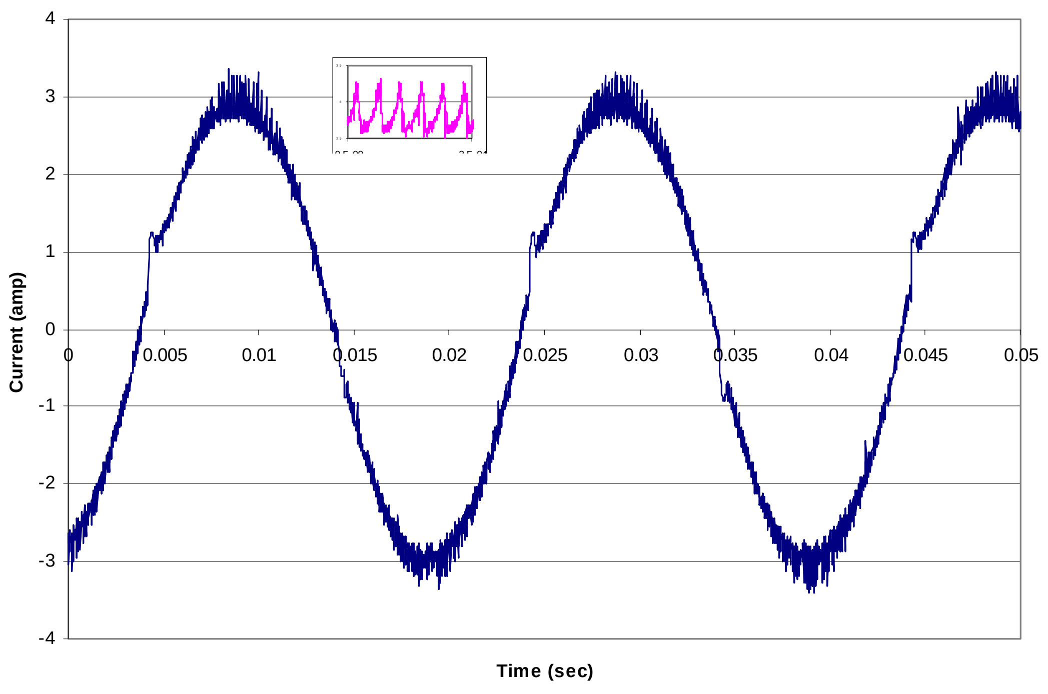
The results were not terribly encouraging. It was immediately obvious that the inverter could not meet its rated power spec: although it had been designed for 600W, it proved impossible to get any more than 570W out of it, and this accompanied by some very sinister crackling noises coming from the filter chokes. A 550W limit was set on subsequent tests, to reduce the possibility of any damage, until the problem could be found.

As tests continued it became obvious that it would not quite meet the efficiency spec either: the efficiency at 550W was only 86%. The peak efficiency was 90% at 260W output.

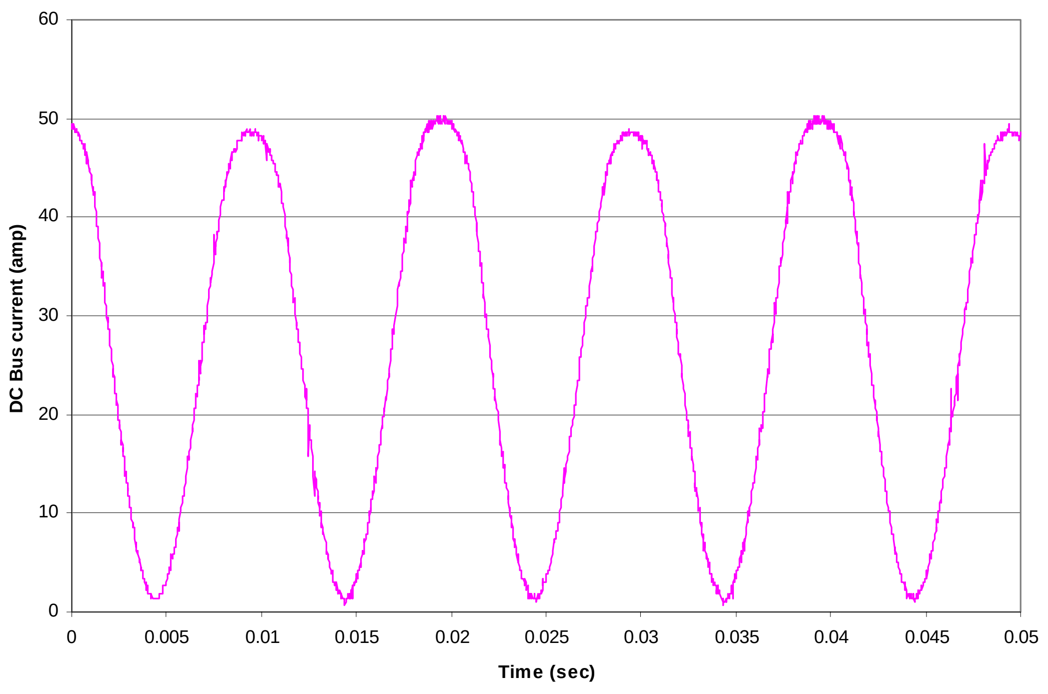
Harmonic distortion of the AC line current was somewhat higher than spec. too, with a measured 5.8% at 550W.

There was also a non-linearity of power with power control. The power began to compress at higher levels, and it seemed likely that this was related to the inability to meet rated power.

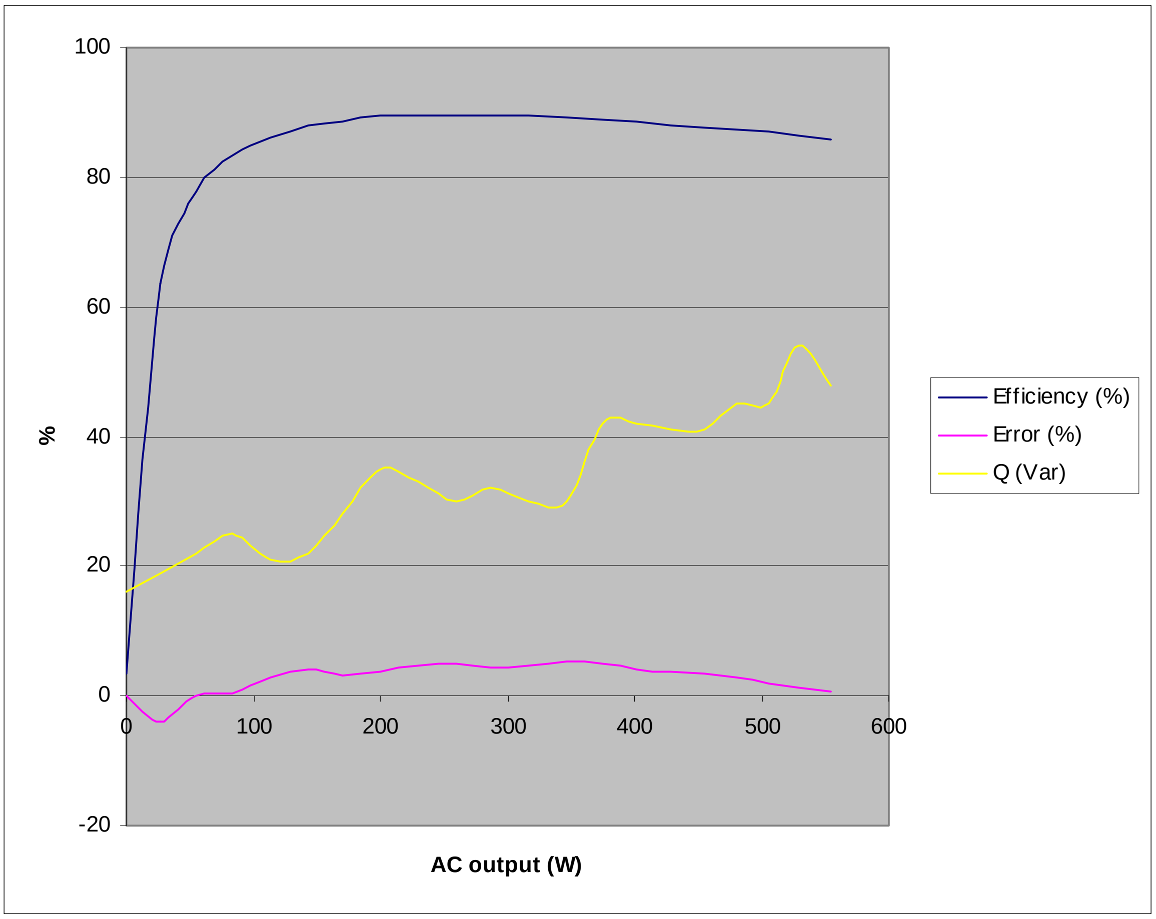
Finally, to add insult to injury, the power output was not stable with time, varying by around 15W (3% of the rated power) at full power. It seemed likely that this was a thermal effect, probably heating of the current shunt, which could be expected to heat up since it was operated at high current in a confined space, and being made of copper would have a considerable tempco. of resistance. This would explain the compression issue, too. Note: a retest of the inverter one year later (see Section 9.4) found that this power instability effect was no longer present. Therefore, there are grounds for suspecting that the true cause was not heating.



(Fig. A.19: Mark Two inverter line current at 550W output. Inset shows waveform at current peak, magnified to show 200 us)



(Fig. A.20: Mark Two inverter DC input current, also at 550W output)



(Fig. A.21: Mark Two efficiency/reactive power/control error)

A.14. Lessons learnt from the Mark 2

The lack of efficiency was the most intractable problem. Again, from an examination of the temperatures of the various components while running under load, it seemed that no one part was contributing excessively to the losses. There were probably two main causes: an underestimation of resistance across the whole circuit, by not taking into account the skin effect, and an ignorance of iron losses in the transformer, which would also be aggravated by the high-frequency currents and any DC component in the output.

The premature power limiting was also puzzling. Considerable time was spent in investigation of this, without any real success. In the end, the most likely explanation was transient dips in the power supply rail, caused by circuit inductances and the extremely high rate of rise of current (up to 700A/μs) in the rail during switching. Transients of 10-20 volts were observed, across only the inductance of a 80mm x 15mm x 2mm thick copper strip. A complete solution of this problem would have required scrapping the existing power stage, and redesigning it to a more compact layout, which was not feasible due to time pressures. Derating the unit to 550W proved to be an acceptable workaround.

The excessive THD was probably a result of too large a hysteresis band, coupled with inadequate low-pass filtering of the output.

So, in conclusion, the Mark Two inverter almost met the specification, but not quite. Considerable time and effort were devoted to improving the performance, but in the end, the above figures represented the best possible without a redesign. There was no time to embark on a Mark Three, and so the Mark Two unit was the only hope. It should be borne in mind that the original goals represented the standard obtained by the very best commercial inverters. Even though it did not meet the spec., the Mark Two still outperformed many units currently on the market. Also, the only shortfall which actually affected standards compliance was the distortion. This only required marginal improvement, which could easily be done by adding an off-the-shelf EMI filter, for instance. Therefore, it was still quite acceptable for experimental use.

A.15. Protection

Once the basic operation of the inverter had been proved, it was time to consider a protection system. The specification has already been quoted, and is quite clear on all points, except for “loss of mains protection”. This seems to be an additional category apart from voltage and frequency limits. Since the G77 spec was written, this has been defined more clearly in the American standard IEEE P929 [2]. It is now formally known as “anti-islanding protection”.

Islanding is the condition where a network containing embedded generation, and demand, comes disconnected from the rest of the grid. because of a blown fuse or tripped breaker, and continues to function on its own, with the embedded generation supplying the local demand. The implications of this have been discussed in previous chapters; it is assumed for the present discussion, as the electricity authorities do, that it is an undesirable condition and must be prevented. The first line of defence is to have over/undervoltage and frequency trips, which work because inverters (and ours will be no exception) look to the grid to determine the voltage and frequency. If the grid is cut off, the inverter will lose its timing, and the frequency will go wrong. And, if the load does not match the inverter output, the voltage will go wrong too. Either case will result in a trip. Unfortunately, there are conceivable situations where the load does indeed match the inverter output. (The REDMan system is one example) Worse, there are combinations of inductive and capacitive loads which will resonate at the line frequency. This is fairly common, in fact, because power factor correction works in exactly this way. If an inverter is islanded while supplying a load that matches its output, with a strong enough resonance at 50 Hz, the voltage and frequency will stay within limits, and it will keep on going.

It was to address this situation that dedicated anti-islanding protection was invented. One popular anti-islanding algorithm, SFS/SVS [6], is available in the public domain. It functions by deliberately introducing instability; if the voltage departs suddenly from its mean value, the inverter is caused to change its power output, in such a way as would amplify the disturbance. A similar algorithm is used for frequency; the inverter tries to change its own output frequency to amplify any deviations in grid frequency. The thinking behind this approach is that the inverter is basically made to behave like a hooligan and fight the grid instead of working with it. Because the grid is bigger, it always wins. But as soon as an island develops, the inverter wins; the voltage and frequency go wildly out, and the voltage/frequency trips shut the inverter down.

Of course, this approach assumes that inverters have an insignificant effect on the grid. Therefore it is doomed to become a victim of its own success. If enough inverters with SFS/SVS were installed in an area where the grid was weak, they would actually be able to out-fight it, with tragi-comic consequences. Aggressive anti-islanding protection like this can never be a part of any plan for significant penetration of embedded generation.

So what protection scheme was opted for? Considering the situation, as discussed here and in previous chapters, it seemed that the only pressing need was for a rudimentary loss-of-mains detection. The microcontroller chosen limited the options rather, because there were no I/O pins left, and no onboard analogue/digital converter, so fitting an AC over/undervoltage trip would have been a very fiddly business. Eventually, the method settled on was a sensitive over/underfrequency trip (this was a firmware job and needed no hardware mods) which resulted in instant loss-of-mains tripping in every scenario tested.

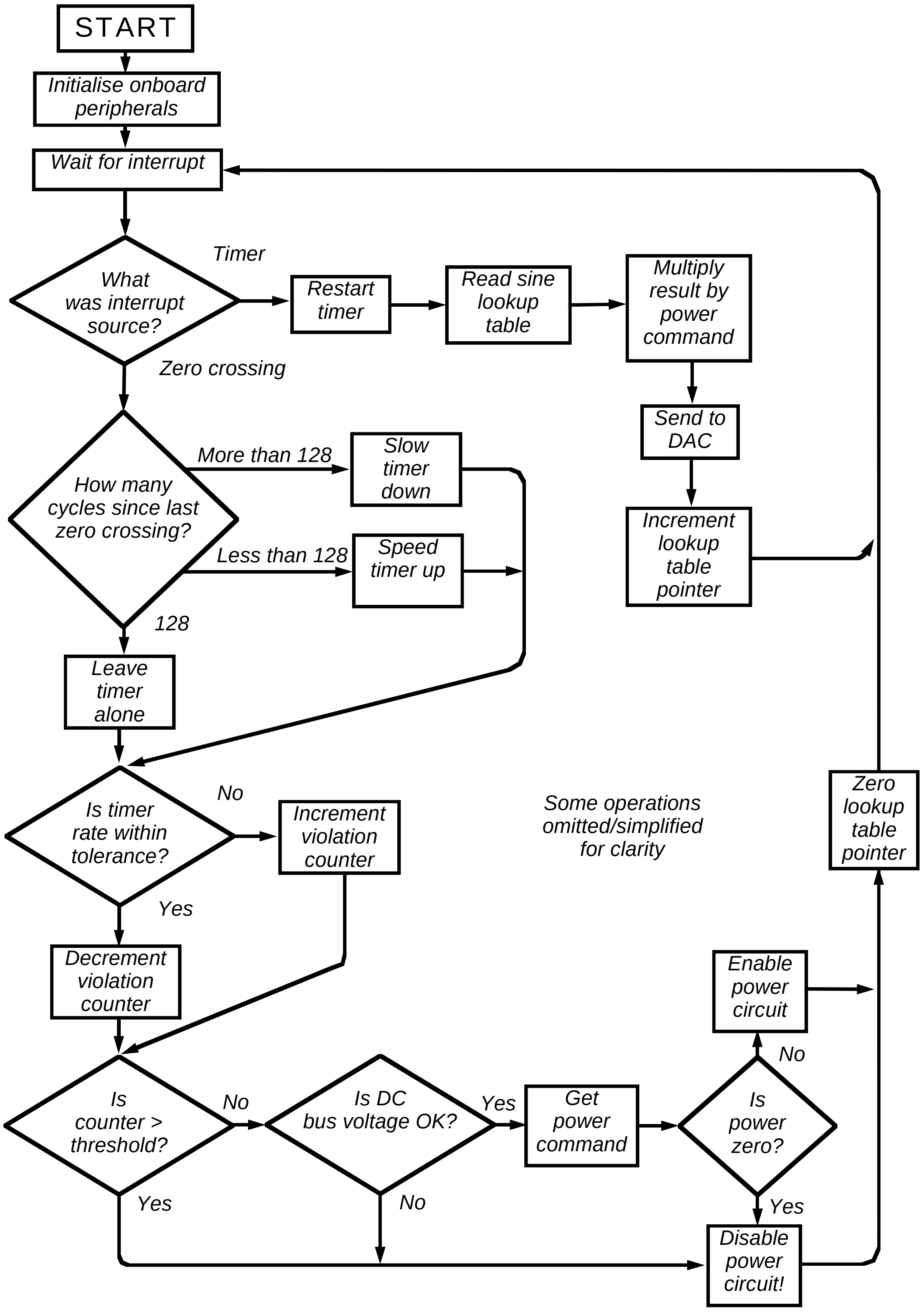
A.16. Notes on firmware

The firmware has three major jobs to do: locking to the mains frequency, scaling the output according to the power command, and measuring the frequency for protection purposes. The frequency measuring and locking is the most complicated part. It is based around an interrupt which is triggered at every zero-crossing of the mains voltage. (The interrupt pin is normally edge-sensitive, and a programming trick is used to make it sensitive to both positive and negative edges.) Another interrupt is triggered by the onboard timer, and causes a D/A conversion to happen. The frequency of this interrupt is nominally every 192 instruction cycles, but it can be changed by programming different values into the timer. This is what the frequency locking routine does; it tries to adjust the timer value so that 128 timer interrupts happen in the space of one zero-crossing interrupt. In order to get finer frequency resolution, the timer value is dithered by adding one to it on the first ‘n’ interrupts in every cycle of 128. (0<=n<128)

The frequency measuring works by keeping an eye on ‘n’ and the timer value, which (assuming the frequency locking is working) form a 13-bit number inversely proportional to the mains frequency. This is compared to high and low limits, and a violation counter is incremented every time the limits are broken. This counter is slowly decremented all the time, so that in normal use it will not reach the threshold. But if the rate of violations becomes excessive, the threshold will be broken. This causes the inverter to shut down immediately. Shortage of I/O pins mandated some dirty tricks here. The same pin that drives the input multiplexer is used for shutdown: the multiplexer only needs narrow pulses to operate it and the shutdown circuit is deliberately made too slow to respond to these. Undervoltage on the DC bus (which is potentially catastrophic) also causes a shutdown. The undervoltage detector circuit connects to a pin that is normally the LSB of the D/A converter. A series resistor is used so that in normal operation the PIC pin overdrives the detector output, and the D/A works properly. The pin is briefly reconfigured as an input at every zero crossing to sample the detector.

Scaling is done by a software multiplying routine, because the PIC used has no hardware multiplier. This uses well-known arithmetic techniques to multiply two unsigned 8-bit numbers together, taking about 77 instruction cycles to do so (which leaves about 115 instructions per conversion for everything else) The least significant 9 bits of the result are discarded in this application.

A.17. Flowchart


START
Initialise onboard peripherals
Wait for interrupt
What was interrupt source?
Timer
Restart timer
Read sine lookup table
Multiply result by power command
Send to DAC
Increment lookup table pointer
Zero crossing
How many cycles since last zero crossing?
More than 128
Slow timer down
Less than 128
Speed timer up
128
Leave timer alone
Is timer rate within tolerance?
No
Increment violation counter
Yes
Decrement violation counter
Some operations omitted/simplified for clarity
Zero lookup table pointer
Is counter > threshold?
No
Is DC bus voltage OK?
Yes
Get power command
Is power zero?
No
Enable power circuit
Yes
Disable power circuit!
Yes
No

A.18. Listing

This program may also be downloaded from the ESRU website at http://www.esru.strath.ac.uk/

```
;*******************************************************
; GRID INTERTIED INVERTER FIRMWARE
; VERSION 3.03
; WITH PROTECTION
;*******************************************************
; STRICTLY (C) 2000-1 STEPHEN J.CONNER ESQ BENG
; SO HANDS OFF
;*******************************************************
; HI-RES VERSION
; INTENDED FOR PIC16(C|F)84(A)-10
; WITH 9.8304 MHz CRYSTAL
;*******************************************************

;*********************
;  ASSEMBLER SETUP
;*********************

; What kind of processor are we using
       LIST    P=16F84

; include file gives names to special function registers
#include       "P16F84.INC"

; processor config flags -
; watchdog on, code protection off, xtal oscillator, etc
        __CONFIG _WDT_ON & _HS_OSC & _PWRTE_ON
        __IDLOCS H'1234'

; code origins
#define LUTBASE 0x300          ; lookup table base address
#define        RESVEC  0x00           ; power-on reset vector
#define        INTVEC  0x04           ; vector for one and only interrupt
#define        LUTPCH  (LUTBASE/D'256')      ; high bits to poke to PC

; Magic numbers
#define FCY     D'100'         ; mains frequency Hz
#define FTOL    D'3'           ; mains freq tolerance (in 128ths of a cycle)
#define FOSC    D'9830400'            ; crystal frequency Hz
#define SPC     D'128'         ; wave steps per cycle
#define CPS     ((FOSC/4)/(FCY*SPC))          ; clocks per wave step
#define LAT     D'18'          ; total timer interrupt latency
#define MAXLEN  D'104'         ; maximum number of instructions in a step
#define ISAT    (SPC+D'14')           ; index saturation value
#define        TMR     (D'256'-(CPS-LAT))     ; initial preload for timer allowing for
latencies
#define TMAX    (D'256'-(MAXLEN+LAT+D'10'))   ; the absolute max timer setting
#define TREF    (D'128'+(MAXLEN/2))    ; the timer reference point to aim for
#define THI     (TMR+FTOL)            ; upper and lower tolerance bands
#define TLO     (TMR-FTOL)
#define TARGET  (SPC-1)        ; the target step index value
#define CTR     D'255'         ; number of cycles to elapse before run
#define CTS     D'10'          ; number of frequency violation cycles before shutdown

; Protection states
#define SYNCING 1              ; in process of locking to grid
#define READY   2              ; locked and ready to start power stage
#define RUNNING 3              ; power stage started
#define TRIPPED 4              ; tripped out
#define ASLEEP  5              ; shut off due to being idle for a while

; Working registers
power   equ     0x0c           ; Throttle setting (output power control)
index   equ     0x0d           ; current index into lookup table
H_byte  equ     0x0e           ; multiplier working regs
L_byte  equ     0x0f           ; as above
mulplr  equ     0x10           ; working reg for multiplier
count   equ     0x11           ; multiplier loop count
```

```
timer   equ     0x12            ; what the timer value should be based on...
errc    equ     0x13            ; phase lock coarse error
errf    equ     0x14            ; phase lock fine error
tmrtmp  equ     0x15            ; timer saved value
wtemp   equ     0x16
stemp   equ     0x17
run     equ     0x18            ; ready to go when run=0
pstat   equ     0x19            ; protection status
fvi     equ     0x1A            ; frequency violation counter
cyc     equ     0x1B            ; cycle counter

;*********************
;   LOOKUP TABLE
;*********************
;
; contains a half cycle of rectified sine wave
; note: the dt statement generates a retlw instruction for each data item
; note: there is extra dummy data at the end
        org     LUTBASE
table   addwf   PCL,F           ; jump to the right place in the table
        dt      .6,.13,.19,.25,.31,.37,.44,.50,.56,.62,.68,.74,.80,.86,.92
        dt
        .98,.103,.109,.115,.120,.126,.131,.136,.142,.147,.152,.157,.162,.167,.171,.176
        dt
        .180,.185,.189,.193,.197,.201,.205,.208,.212,.215,.219,.222,.225,.228,.231,.233
        dt
        .236,.238,.240,.242,.244,.246,.247,.249,.250,.251,.252,.253,.254,.254,.255,.255
        dt
        .255,.255,.255,.254,.254,.253,.252,.251,.250,.249,.247,.246,.244,.242,.240,.238
        dt
        .236,.233,.231,.228,.225,.222,.219,.215,.212,.208,.205,.201,.197,.193,.189,.185
        dt
        .180,.176,.171,.167,.162,.157,.152,.147,.142,.136,.131,.126,.120,.115,.109,.103
        dt      .98,.92,.86,.80,.74,.68,.62,.56,.50,.44,.37,.31,.25,.19,.13,.6,.0
        dt
        .128,.128,.128,.128,.128,.128,.128,.128,.128,.128,.128,.128,.128,.128,.128,.128

;*********************
;  RESET VECTOR
;*********************
        org     RESVEC

; interrupts off until we are all set up
        bcf     INTCON,GIE
        goto    init

;*********************
;  DO-IT-ALL HANDLER
;*********************
        org     INTVEC

; interrupts are stopped automatically when entering handler
; find out where the interrupt came from
; zero crossings get priority
        btfsc   INTCON,INTF             ; if a zero crossing interrupt
        goto    zero
        btfsc   INTCON,T0IF             ; if a timer interrupt
        goto    wave

; if a cosmic ray hits the interrupt circuitry...
; we should cater for the possibility
        retfie

;*********************
; zero crossing interrupt service routine
; uses a simple digital PLL to keep locked into the 50Hz
; and reads the throttle setting
; protection also dealt with here
;*********************
;
; deal with the timer ASAP
zero    movf    TMR0,W          ; get the current timer
        movwf   tmrtmp
        movf    errc,W          ; and restart it
        movwf   TMR0

; coarse lock
; this adjusts timer speed until we get the correct number of steps per cycle
        movf   index,W
        bcf    STATUS,C
        sublw  TARGET         ; (target-index)
        btfsc  STATUS,Z              ; if the index is exactly target, Z=1
        goto   fine           ; do fine lock
        btfss  STATUS,C              ; if the index is greater than target, C=0
        goto   toohi          ; execute 'too high'

; too low (executes if none of above conditions true)
        incf   errc,F         ; increment error
        goto   fprot          ; and skip over fine lock

; too high
toohi  decf   errc,F         ; decrement error
        goto   fprot          ; and skip over fine lock


; once the coarse lock has worked -the fine lock is brought into play
; TREF is subtracted from the measured TMR0 value. If TREF > TMR0, i.e.
; TREF-TMR0 > 0, then the timer needs to be faster, so we increment the error
; register. Otherwise we decrement it
fine   movf   tmrtmp,W
        bcf    STATUS,C
        sublw  TREF
        btfss  STATUS,C              ; test TREF-TMR0
        goto   fneg

; if positive or zero
        incf   errf,F         ; increment it
        btfss  errf,7         ; if error is now greater than 7F
        goto   fprot          ; (skip rest of routine if it's not)
        clrf   errf           ; reset it to zero
        incf   errc,F         ; and increment coarse error
        goto   fprot          ; end of 'if zero or positive' code

; else if negative
fneg   movlw  1
        bcf    STATUS,C
        subwf  errf,F         ; take 1 away from errf
        btfsc  STATUS,C              ; has it rolled over from 00 to FF
        goto   fprot          ; if not skip next instruction
        decf   errc,F         ; decrease coarse error

; now timer calculations are finished- test if frequency in tolerance.
; This is done by checking the timer preload (errc) which of course depends
; on the frequency
; Now there are 128 steps in each cycle, 100 cycles per second, and the timer runs
; at 2,457,600 ticks per second. So, we'd expect there to be 192 ticks in each
; cycle. But, the timer interrupt has a latency of 14 ticks, etc. Fear not, the
; value which the timer should take is calculated in the defines- it's TREF

; This routine is vaguely based on Sandia Labs' SFS protection system
; increase frequency violation counter every time errc strays outside tolerance
fprot  movf   errc,W
        bcf    STATUS,C
        sublw  TLO
        btfsc  STATUS,C              ; is errc less than TLO
        goto   finc
        movf   errc,W
        bcf    STATUS,C
        sublw  THI            ; or is it more than THI
        btfss  STATUS,C              ; skip if it is not
        goto   finc
        goto   fok

finc   incfsz fvi,W          ; test if this will roll over
        incf   fvi,F          ; if it won't then increment
        goto   ftest          ; skip decrement

; else, if frequency was OK, decrement fvi. Unless it's zero!
fok    movf   fvi,F          ; move fvi to itself
        btfss  STATUS,Z              ; in order to test for zero
        decf   fvi,F

; if the counter hits 'CTS'- Bring the show down!
ftest  movf   fvi,W
```

```
        bcf     STATUS,C
        sublw   CTS
        btfsc   STATUS,C                ; skip if fvi greater than CTS
        goto    fnotrip        ; branch to 'no trip' if fvi within limit

        bsf     PORTA,4        ; otherwise immediately cut out power stage
        movf    pstat,W        ; test if status != tripped ie trip has just occurred...
        sublw   TRIPPED        ; because we only want to reset fvi once...
        btfsc   STATUS,Z                ; and not every time we do this test...
        goto    fnotrip        ; which would lock us up for good
        movlw   TRIPPED
        movwf   pstat          ; status = tripped
        movlw   CTR            ; reset fvi.
        movwf   fvi


; if the counter hits 0- Start up again!
fnotrip movf    fvi,W          ; next instruction skipped if fvi=0
        btfss   STATUS,Z
        goto    ttest
        movlw   RUNNING        ; set status to running
        movwf   pstat

; now get the new throttle setting for this cycle
; a 8 bit setting multiplexed in as two lots of 4 bits due to shortage of
; IO pins
ttest   movf    pstat,W        ; don't execute this if we are tripped
        sublw   TRIPPED
        btfsc   STATUS,Z
        goto    nothrot
throt   bsf     PORTA,4        ; set multiplexing bit
        nop                    ; wait
        nop
        movf    PORTA,W        ; read 4 most significant bits into W
        andlw   B'00001111'             ; mask
        movwf   power          ; put into power register
        swapf   power,F        ; swap nibbles
        bcf     PORTA,4        ; clear multiplexing output
        nop                    ; wait
        nop
        movf    PORTA,W        ; read 4 least significant bits into W
        andlw   B'00001111'             ; mask
        iorwf   power,F        ; OR with existing contents of power reg

; simple sleep mode
        btfsc   STATUS,Z                ; is throttle setting zero?
        bsf     PORTA,4        ; then cut out power stage

; finally saturate errc
; this is needed so the timer can't be set to pump out interrupts faster than the
; program can handle them
nothrot movf    errc,W
        bcf     STATUS,C
        sublw   TMAX
        btfsc   STATUS,C                ; is errc greater than TMAX
        goto    skpsat         ; if not skip the next bit
        movlw   TMAX
        movwf   errc           ; if so let errc=TMAX

; reset the step index
skpsat  clrf    index

; toggle the interrupt edge trigger bit (the next interrupt will come on the opposite
; edge)
        bsf     STATUS,RP0              ; our business is in bank one
        btfsc   OPTION_REG,INTEDG       ; if bit is set
        goto    clear          ; clear it
        bsf     OPTION_REG,INTEDG       ; otherwise set it (because it was clear)
        goto    tend           ; and skip the next line...
clear   bcf     OPTION_REG,INTEDG       ; which would just clear it again
tend    bcf     STATUS,RP0              ; back to bank zero

; wreck any interrupts that might have happened meantime
        bcf     INTCON,INTF             ; clear zero crossing interrupt
        bcf     INTCON,T0IF             ; clear timer interrupt

; send zero to PORTB (since this code runs instead of wave step handler)
```

```
        clrf    PORTB

; now portb=0 we can test that DC input is ok
; this is dodgy- we hooked a comparator to one of the port pins normally
; used as output (ie time multiplexing)
        bsf     STATUS,RP0
        bsf     TRISB,1                 ; briefly turn rb1 to an input
        bcf     STATUS,RP0
        nop                             ; wait for things to settle
        nop
        nop
        btfss   PORTB,1                 ; if the DC is out of spec this =1
        goto    dctstok                 ; so if not =1 skip to ok
        movlw   TRIPPED
        movwf   pstat           ; status = tripped
        movlw   CTR             ; reset fvi.
        movwf   fvi
dctstok bsf     STATUS,RP0
        bcf     TRISB,1
        bcf     STATUS,RP0

; return the hard way - resetting the program (and eventually overflowing the stack-
; but this is not a problem since the stack is a circular buffer)
        bsf     INTCON,GIE
        goto    main

; timer interrupt service routine
; living dangerously we ENABLE interrupts when executing this code
wave    bcf     INTCON,T0IF             ; clear the interrupt that got us here
        bsf     INTCON,GIE              ; interrupts on

; first calculate the timer value- this needs explained
; the timer value is made from the coarse error plus a simple dither arrangement
; the timer is bumped up by one if the current step index is less than the fine
; error
; if you think about it- this controls the period in increments of 1/128 of a step
; we compute errf-index. if this is +ve or 0 then the timer is incremented
; note the higher the error values- the FASTER the timer will go
        movf    errc,W          ; save errc into timer
        movwf   timer
        movf    errf,W
        bcf     STATUS,C
        subwf   index,W         ; errf - index
        btfsc   STATUS,C                ; if errf < index
        incf    timer,F         ; add one to timer value
        movf    timer,W
        movwf   TMR0            ; now load the timer

; load up some registers
        movlw   0x08            ; set loop count for multiplier
        movwf   count
        movf    power,W         ; transfer current power setting to
        movwf   mulplr          ; temp register (multiplier will mangle it)

; fetch the right lookup table entry
        movlw   LUTPCH          ; load PC high bits latch
        movwf   PCLATH          ; because the table is in a different page
        movlw   LUTBASE         ; put base address in W
        addwf   index,W         ; add current wave index to W
        call    table           ; here goes nothing

; we return from lookup table with wave step value in W
; multiply by throttle value
; using microchip example multiplier code (mul8x8)
; Multiplier is in mulplr, multiplicand is in W. Answer comes out
; in H_byte (the lower 8 bits are ignored)
; It takes 73 cycles
        clrf    H_byte          ; clear working regs from last time
        clrf    L_byte
                bcf     STATUS, C               ; Clear the carry bit in the status Reg.
mloop           rrf     mulplr, F               ;  Rotate  the  multiplier  right  (through
carry bit)
                btfsc   STATUS, C               ; Test the carry bit - if not zero
                addwf   H_byte, F               ; then add W to the high byte
                rrf     H_byte, F               ; Rotate high byte right (with carry from
addwf)
                rrf     L_byte, F               ; rotate low byte right (through carry)
```

```
               decfsz  count, F              ; decrement count and test - if not zero
               goto    mloop         ; then loop again

; Test to make sure we are in run mode
        movf    pstat,W
        sublw   RUNNING
        btfss   STATUS,Z
        goto    clr

; Send out to D/A (LSB will be ignored by PORTB) unless not in run mode
        movf    H_byte,W
        movwf   PORTB
        goto    noclr

; if not in run mode- send zero instead
clr     clrf    PORTB

; increment wave step index for next time, testing for overrun
noclr   movlw   ISAT           ; value to saturate index to
        subwf   index,W
        btfss   STATUS,Z              ; if zero flag not set...
        incf    index,F        ; increment index

; all done
        clrwdt                 ; keep watchdog timer happy
        retfie                 ; retfie also re-enables interrupts

;****************************
;  POWER-ON INITIALISATION
;****************************
; initialise working registers
init    bcf     STATUS,RP0            ; bank 0
        clrf    power
        clrf    index
        clrf    errf           ; set up initial values
        clrf    cyc
        movlw   TMR
        movwf   errc           ; for timer speed
        movlw   CTR
        movwf   run            ; countdown to start
        movwf   fvi
        movlw   TRIPPED
        movwf   pstat          ; protection algorithm status

; setup timer
        bsf     STATUS,RP0            ; bank 1
        bcf     OPTION_REG,T0CS       ; timer mode not counter
        bsf     OPTION_REG,PSA ; prescaler to watchdog timer
        bcf     OPTION_REG,PS2 ; set for 1:1 prescaling
        bcf     OPTION_REG,PS1 ; gives nominal 18ms WDT
        bcf     OPTION_REG,PS0
        clrwdt
        bcf     STATUS,RP0
        bsf     INTCON,T0IE           ; enable timer overflow interrupt
        clrf    TMR0           ; preload timer

; setup PORTA
        movlw   B'10000'              ; output latches to known state
        movwf   PORTA          ; (RA4=1 to hold power stage disabled)
        bsf     STATUS,RP0            ; bank 1
        movlw   B'01111'              ; all inputs (TRISA=1) except RA4
        movwf   TRISA

; setup PORTB
        bcf     STATUS,RP0            ; bank 0
        clrf    PORTB          ; clear output latches
        bsf     STATUS,RP0            ; bank 1
        movlw   B'00000001'    ; all outputs (TRISB=0) except RB0
        movwf   TRISB
        bsf     OPTION_REG,NOT_RBPU   ; turn off internal pullups
        bsf     OPTION_REG,INTEDG     ; interrupt on rising edge
        bsf     INTCON,INTE           ; enable RB0 as interrupt source

; start interrupts and we got ourselves a convoy
        bcf     STATUS,RP0            ; bank 0
        bsf     INTCON,GIE            ; global interrupt enable
        goto    main
```

```
;*********************
;     MAIN CODE
;*********************
; this doesn't need to do anything - all the work is done by interrupts.
; Having the processor in an endless one-instruction loop might increase interrupt
; latency (because goto takes 2 cycles) so I give it several hundred
; pointless instructions
main    nop                     ; no operation
        org     (LUTBASE-1)             ; then execute a stack of empty memory
        goto    main            ; loop forever

; phoo-ee
        END
```

A.19. References

1. Personal communications with Trace Engineering and Statpower, 2000
2. ‘P929 Recommended Practice for Utility Interface of Photovoltaic (PV) Systems’, IEEE, Sep 1998
3. ‘Power MOSFET Design’, Taylor, B.E., Wiley, 1993, various.
4. ‘Power Electronics’, Lander, C.W., McGraw-Hill, 1993, pp. 198-216
5. Alternative Transients Program website, Michigan Technical University, http://www.ee.mtu.edu/atp/, 2001.
6. ‘Development and Testing of an Approach to Anti-Islanding in Utility-Interconnected Photovoltaic Systems’, Stevens, J. *et al*, Sandia National Laboratories, 2000
7. ‘Switched Mode Power Supplies in Practice’, Kilgenstein, O., Wiley, 1993.
8. ‘Electromagnetism for Electronic Engineers’, Carter, R.G., Chapman & Hall, 1992
9. ‘The Art of Electronics’ 2nd ed., Horowitz, P., Hill, W., Cambridge University Press, 1989
10. ‘Utility-interactive photovoltaic power conditioning system with forward converter for domestic applications’, Matsui, K. et al., IEE Proc.- Electr. Power Appl, Vol. 147, No. 3, May 2000