```
MGspec.txt        created 12/13/83 sjg

Update log:  Random fixes - 17:30 13Dec83/roy
             Everything in the world 2/21/84 sjg
```

Minnie Subsystem Top-level Specification

Abstract

The Minnie subsystem is a very-large-scale integration implementation of a specialized sound synthesizer intended for use in the 3600 family product line. This memo provides software and hardware descriptions of that subsystem.

Preface

This memo has two sections:

Section 1, which describes the role the Minnie fulfills in the 3600 project, and

Section 2, which describes the hardware itself, along with software interface information.

TABLE OF CONTENTS

1.0 Minnie OVERVIEW

$2

The Minnie was conceived as an extremely low-cost sound synthesis IC intended for use in game cartridges for the 3600 home video game system, augmenting the sound generation capabilities of the TIA chip present on the 3600 board. As such, it faces severe constraints on package size, die size, external component count (preferably zero), and general producibility. The overriding concern has been to produce a synthesis subsystem whose total cost, installed on the cartridge, would be less than $1.00. Thus, an "add 'til full" design approach was used, to good effect. Secondary concerns were to produce the highest quality sounds possible within the first constraint, with particular attention being paid to:

(w)

1. Versatility - many sounds are available, with ROM-based waveforms ~~easily~~ redefined on a per-game basis
2. Musicality - musically useful features, such as fine frequency resolution, wide ranges of timbre variation, and careful choice of noise generation techniques
3. Listenability - a wide dynamic range generation technique and output section assure low distortion and high quality previously unattainable at this price point.

1.0.1 SOUND SYNTHESIS FEATURES:

1. The output audio will be a mix of three distinct voices.

2. The mixed output will have a sample rate of 28Khz (useable upper frequency around 10 to 12 Khz), and have a dynamic range of 10 bits.

3. Each voice will have adequate frequency resolution (effective 16 bits) for music all the way from 27Hz to the highest useable frequency (more resolution at higher frequencies).

4. There will be two waveforms stored in on chip ROM. Each waveform is selectable by any voice. Additionally, three fixed waveforms (triangle, square, and sawtooth) are available, giving a total of 7 waveforms useable by any of the three voices in any combination. 5?,

5. Each waveform consists of 64 eight bit samples.

6. Each voice will have independently controllable phase relative to other voices.

7. Each voice will have independently controllable amplitude (six bits of control, provided as three bits of exponent and three bits of mantissa).

8. Each voice will have independently controllable noise modulation, to be described below. Four bits of control allow sixteen levels of phase-modulation noise, ranging from nearly imperceptible dither to wide-spectrum filtered noise effects. This effect will vary widely depending upon which waveform and base frequency is selected. Dither

1.1 OVERVIEW OF THE DIGITAL OSCILLATOR

The approach taken to synthesis in the Minnie bears some discussion at this point, before delving into the hardware and control issues. This involves the idea of storing a sampled waveform and playing it out at some given frequency.

The Minnie generates each voice by the use of a digital oscillator driving a waveform table. Each of the 2 ROM-based waveforms is stored as 64 8-bit samples, representing one complete cycle of the waveform. The oscillator is created by adding a constant value, called the frequency constant, to a variable that we call the index at fixed intervals (28 KHz). By varying the value of the frequency constant, the period of time it takes for the index to "wrap around" (overflow) can be controlled. This effectively sets the period of repetition of the waveform, and thus its frequency.

In the Minnie, the frequency constant and the index are 16-bit values, and the clock frequency is 28 KHz. Thus, once every 37.5 uS the index is updated by adding the frequency constant to it, and then the 6 most significant bits are decoded to determine which of the 64 samples is to be output.

For the Minnie, if the frequency constant is set to 0001H, the resultant oscillator frequency will be 2.34 Hz. Increasing the value will increase the frequency, as governed by the formula

    28000Hz * (freq. const. / 65536) = output frequency

In the Minnie, the values for frequency constant and index are accessable to the programmer at all times. This allows some creative things to be done, such as setting relative phase between oscillators by adjusting the index values, frequency sweeps under program control, and so on. All of the control information for each of the three oscillators is accessable to the programmer via the 32 registers to be described below.

2.0 INTRODUCTION

The Minnie chip is intended to handle supplementary sound synthesis chores for games designed for the 3600 base unit/home computer family. It will reside on the plug-in software cartridges used with the base unit, and will communicate with the 3600's 6502 processor via a number of registers. Its output audio signal is mixed with that from the TIA chip resident within the base unit, allowing many more effects than are possible with the TIA alone. The output audio will be passed through a pin on the cart connector and all audio mixing will be handled on the 3600 board, with the normal RF modulation section sending the mixed audio to the user's TV audio section. This allows the extended sound features to be used in a transparent fashion, with no external connections required.

It is intended that the Minnie be controlled by data values written to a group of 32 registers mapped into XXXXXX in the 3600 processor's address space. These registers are divided up into four blocks of eight. The first three blocks are referred to as Voice Blocks, as each controls the characteristics of one of the three voices, and the final block is referred to as the Chip Control Block. These registers will be described in more detail in section 2.3.1 below.

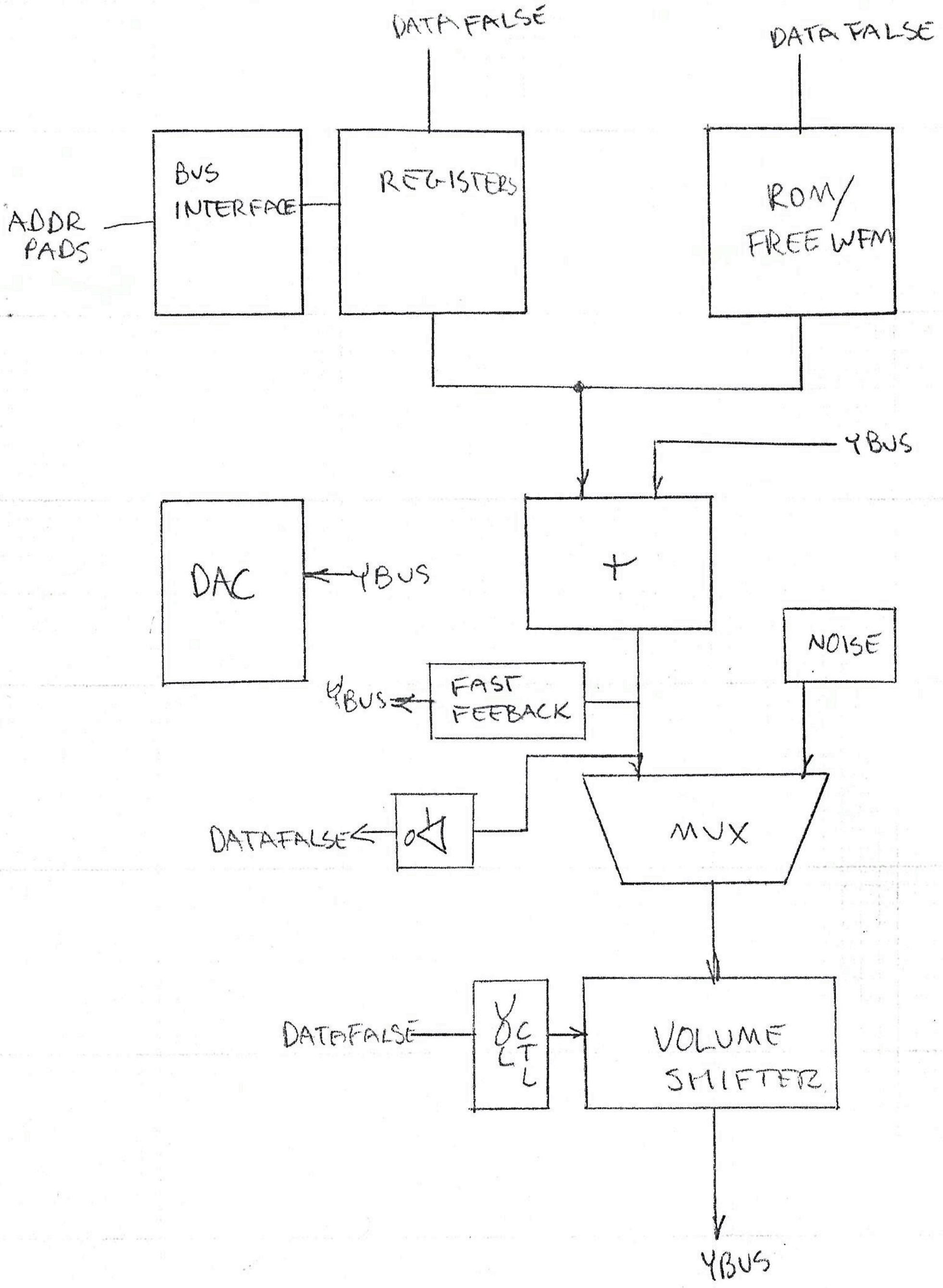
12 Modules?

The unit is made up of ~~five basic blocks~~. These are the general-purpose microcoded data path/ALU used to implement the digital oscillator/accumulator functions, the ROM waveform storage section, the analog output section, and the 3600 processor's bus interface control logic, and the RAM array containing the 32 data and control registers. These sections will be described in detail below.

The chip is tied together with three major data buses. These are the adders' Xbus and Ybus, and the DataFalse bus used to communicate with the external data pads.

2.0.1 Minnie BLOCK DIAGRAM

See fig. 2.0.1.


DATA FALSE
DATA FALSE
BUS INTERFACE
REGISTERS
ROM/ FREE WFM
ADDR PADS
YBUS
DAC
YBUS
+
NOISE
FAST FEEBACK
YBUS
DATAFALSE
MUX
DATAFALSE
Y C CTL
VOLUME SHIFTER
YBUS

[illegible]

2.1 DATA PATH/ALU INTRODUCTION

The Minnie's real power lies in its microcoded 10-bit data path. The data handled by this data path is generally in two's-complement form, and thus the path has been optimized for two's complement operations.

There is only one data path, which is then time-division multiplexed among all three voices by the microcode.

The actual data path consists of a 10-bit parallel adder and a sign-extension shifter used for scaling of values. Associated with the adder are two input buses, called the Xbus and the Ybus. It is the peculiar configuration of sources that can drive these two buses that provide the Minnie's power. Fig. 2.0.1 shows that the Xbus can be driven by data values from the register array or by values looked up from the waveform ROM. The Ybus can be driven by the output of the volume shifter, or via a fast feedback path from the adder's output. This feedback path can either provide a straight signal path, or a right-shifted value, allowing for fast shift-and-add multiplies.

Between the adder and the sign-extension shifter there exists a port to the chip's internal DataFalse bus, allowing adder results to be communicated to the register array, ROM, or external data bus pads.

Associated with the sign-extension shifter are an input source multiplexer and a polynomial counter, used to provide pseudorandom noise data. This section drives the internal Ybus with its results.

2.1.0.1 DATA PATH/ALU BLOCK DIAGRAM

See fig. 2.1.0.1.

?

2.1.1 Minnie MICROCODE ALGORITHM

```
**********************************FIX ME!!*************************************
```

17 states? clock rate?

The microcode takes the Minnie's innards through a fairly simple loop once every 37.5 uS, in order to calculate the 10-bit sound sample to output to the DAC for the upcoming interval. This involves doing three 16-bit adds (one per voice) to implement the basic oscillator functions, three 16-bit adds to implement noise modulation, three fetches from the internal ROM, three 8-bit shift-and-add multiplies along with the associated volume scaling of each sample, and two 10-bit adds of the scaled samples to create the final aggregate sample.

As the Minnie is operated directly from the 1.8 MHz clock, this allows for 64 internal microcode states for each complete sample calculation.

```
***********
so, let's add in a nifty section describing the code in detail.
***********
```

The XXXX states left over are used to acommodate accesses by the 6502. These make it possible for the processor to interrupt the flow of the sound generation calculations without causing an audible slip in the timing.

2.1.2 THE ADDER

The adder is a conventional 10-bit two's-complement design. The only variation from the norm for this section is the addition of logic allowing a carry to be detected from the 8th bit for multiple-byte adds. This carry is controlled by the internal microcode.

The adder takes its input from its X- and Ybuses, and its output can drive either the fast feedback/shift path, the internal DataFalse bus, or the sign-extension shifter's input multiplexer directly.

2.1.3 THE SIGN-EXTEND SHIFTER

The sign extend shifter is used to reduce the magnitude of values applied to its inputs. It can perform a right shift of 0 through 7 places under the control of a 3-bit control field.

Associated with the shifter are an 8-bit input multiplexer and a 15 stage poly counter noise source. The multiplexer allows input to the shifter to be taken from either the output of the ROM/free waveform logic, or the random number output from the poly counter under microcode control.

2.2 REGISTER ARRAY

The register array is used to contain the information necessary to control the behavior of the Minnie. There are 32 registers in a contiguous area based at XXXXXX in the 3600 processor's address space. These are divided into 4 blocks of 8 registers, with each block associated with some single function. The first three blocks are identical in function, and are dedicated to controlling the three voices produced by the chip. These are referred to as Voice Blocks. The final block, referred to as the Chip Control Block, is used to support test modes, reset functions, and the like, and is not intended for normal use.

The register layout is as follows:

Voice Block 1

0	FREQ1L
1	FREQ1H
2	VOL1
3	TIMBRE1
4	INDEX1L
5	INDEX1H
6	not used
7	T (reserved for internal use)

Voice Block 2

8	FREQ2L
9	FREQ2H
A	VOL2
B	TIMBRE2
C	INDEX2L
D	INDEX2H
E	not used
F	T (reserved for internal use)

Voice Block 3

10	FREQ3L
11	FREQ3H
12	VOL3
13	TIMBRE3
14	INDEX3L
15	INDEX3H
16	not used
17	T (reserved for internal use)

Chip Control Block

18	CREG
19	not used
1A	not used
1B	not used
1C	STROBE
1D	not used
1E	not used
1F	not used

The registers are not intended to be readable. There does exist a read function, but it is intended for test fixture use only, and attempts to use it in a real-world application are NOT RECOMMENDED.

2.2.1 VOICE BLOCK

The following section delves into the layout and function of each of the 3 Voice Blocks used to control the three voices produced by the Minnie.

2.2.1.1 FREQnL

This register is used to contain the low byte of the frequency constant for Voice n.

2.2.1.2 FREQnH

This register is used to contain the high byte of the frequency constant for Voice n.

2.2.1.3 VOLn

This register is used to hold the volume control data for voice n. The format for data written here is as follows:

MSB							LSB
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
ZER	VOL5	VOL4	VOL3	VOL2	VOL1	VOL0	N/U

The MSB must be written with a zero, or unexpected things will come out (go for it, hackers!). The LSB is not used.

2.2.1.3.1 VOICE VOLUME CONTROL

Volume control of each voice is accomplished by using all of the capabilities of the data path to scale the 8-bit sample fetched from ROM to the appropriate value. The Minnie uses a six-bit volume control field, contained in the Volume register in each Voice Block, and interprets this field as three bits of exponent and three bits of mantissa. Thus, given an 8-bit sample S, and a six-bit volume control field VOL5-VOL0, the sample scaling is calculated by the following operation:

(4V5 + 2V4 + 1V3)

Sscaled = S(1 + VOL2/2 + VOL1/4 + VOL0/8)(2 exp ~~(VOL5,VOL4,VOL3~~))

The low-order term is calculated by doing shift-and-add operations via the fast feedback path, and then the result is run through the sign-extend shifter for the exponent shift. In this way, the maximum resolution is preserved.

The result of this calculation is one of the three voice samples, and is added to the total (T) register in preparation for presentation to the DAC.

2.2.1.4 TIMBREn

This register is used to hold the waveform select and noise modulation control data for voice n. The format for data written here is as follows:

```
  MSB                                                                  LSB
 bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0

  NZE3    NZE2    NZE1    NZE0    N/U     WFM2    WFM1    WFM0
    \       |       |      /                \       |      /
  Noise modulation control bits             waveform select bits
```

2.2.1.4.1 NOISE MODULATION CONTROL

The noise modulation scheme used on the Minnie utilizes a phase modulation technique. In essence, this method works by adding a random number of a controllable magnitude to the index just before a sample is fetched. This will introduce a certain amount of phase jitter. Large amounts yield a signal that sounds like slightly pitched pink noise, while small amounts yield a just perceptible imperfection in frequency stability.

The four-bit noise control field in the Timbre register in each Voice Block is used to select 0 through 16 bits of noise to be added to the index.

2.2.1.4.2 WAVEFORM SELECT CONTROL

A given waveform is selected as follows:

WFM3-0	Waveform selected
0	ROM waveform 0
1	ROM Waveform 1
2	not used
3	not used
4	off (outputs a -1)
5	sawtooth
6	square
7	triangle

2.2.1.5 INDEXnL

This register is used to contain the low byte of the waveform index for Voice n.

2.2.1.6 INDEXnH

This register is used to contain the high byte of the waveform index for Voice n.

2.2.1.7 T REGISTER

The T register is used by the microcode as a place to store the intermediate results of the summation of the three voice samples into the final output sample. It is cleared automatically at the end of each sample cycle, so that writing to it will not really produce any useful results. All three

T registers point at the same hardware register, threby giving the programmer the choice of three places to write to with no useful results.

2.2.2 CHIP CONTROL BLOCK

These registers are used to control the chip during testing, and are not recommended for normal use. When written, they control the state of various test bits as shown below. When read, they relay the contents of the internal data bus to the processor bus.

2.2.2.1 CREG

Rethink

Test mode control register

MSB   LSB

bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0

nc 0 nc 0 RESET MC POLYRST Clear ~~TESTROM~~ FREERUN nc 0 nc 0

****
add in a spiffy section on what the test stuff does here.
****

2.2.2.2 STROBES

These bits, when written, cause something interesting to happen sometime soon. We're not sure just what, though.

```
 MSB                                                            LSB
bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0
```

2.3 ROM/FREE WAVEFORM LOGIC

Waveforms to be produced by the Minnie are handled in two ways. Up until this point, it has been assumed that all waveforms were stored in the on-chip ROM. For very complex waveforms this is a reasonable approach, but one that requires that substantial area on the chip be used merely for storage. It turns out that simple waveforms such as square, sawtooth and triangle waves can be mathematically derived on the fly in a very simple way, using the addressing circuitry already present in the ROM. Thus, it was possible to get the three simple waveforms essentially for free (in terms of chip area), in addition to the two ROM waveforms. Hence, the name "free" waveforms.

The "free" waveforms are generated by combinational logic tacked onto the side of the ROM, but appear to the user as if they were normal ROM-based data. No weirdies should be encountered in their use.

Waveform data is all encoded in two's-complement form, and it will be fairly easy to redefine the contents of the ROM if needed for a special effect or unique sounds for a new game.

A given sample of a waveform is selected in the following way. The Timbre register in each Voice Block contains a 3-bit waveform select field, which is decoded to pick out the target waveform. If it is one of the ROM-based waveforms (0, or 1), then the 6 MSB's of the high byte of the index for the current voice are applied to the 6 least significant address inputs to the ROM. The 3 waveform select bits are applied to the MSB address inputs to the ROM, addressing one byte. This byte is then processed by the volume contol algorithm described above in section 2.1.3.1, and added to the T register.

In the case of the free waveforms (5,6, and 7), the actual index is processed. To produce waveform 5, the sawtooth, the 8 MSB's of the index are passed through to the shifter as data, unchanged. Waveform 6, the square wave, passes only the MSB of the index. Waveform 7, the triangle, uses the 8 MSB's of the index. It uses the MSB to exclusive-OR the 7 next most significant bits, shifted up by one. A zero is shifted up into the LSB. The values thus obtained are then treated exactly as the ROM data above.

2.4 6502 BUS INTERFACE

The 6502 bus interface has the responsibilty for decoding the synchronous bus signals from the processor, and providing a data path for access to the various control and data registers on the chip via the internal DataFalse bus. It also provides logic that handles arbitration of processor accesses versus the synthesis operations on the Minnie's internal data path.

2.4.0.1 BUS INTERFACE BLOCK DIAGRAM

See fig. 2.4.0.1.

2.4.1 BUS INTERFACE LOGIC

Due to the synchronous nature of the 6502's bus, and the basically asynchronous (to the 6502) nature of the Minnie's data path, not all operations are performed immediately. Further, due to chip die size limitations, reads of the data contained in the registers on the chip are limited to testing purposes.

Writes to Minnie registers are performed immediately, suspending operation of the chip's internal data path, whereas reads require 2 cycles. Thus, in order to read a register, it is necessary to do a preliminary read of the location, followed by the real read of the location. The preliminary read shuts down the internal data path, and moves the data to the bus driver circuitry. However, any data that shows up during this preliminary read is erroneous and should be discarded (it consists of the last byte of data read, assuming that it hasn't decayed, for the curious). The next read receives the actual data.

Needless to say, reads of the registers on the chip are NOT RECOMMENDED during normal operation, and read-modify-writes are RIGHT OUT. This ability has been provided primarily for testability reasons, overhead has not been minimized, and the information above has been included for completeness only. The chip was not intended as a random number source, 17-byte scratchpad RAM, or 96-byte lookup table ROM. Sorry.

2.4.2 BUS ACCESS ARBITRATION LOGIC

Bus access arbitration is made fairly simple by virtue of the fact that the processor always wins. In the case of a write, the processor's write is recognized, the internal processing path is stopped, and the data form the processor's bus is immediately applied to the internal data bus and strobed into the internal register directly. In the case of a read, the internal data path is stopped, the data to be read is applied to the internal bus, and then latched at the external bus driver pads, ready to be read out on the next read.

Reads and writes are not destructive, insofar as the sound generation process is concerned. There are only two ways that processor accesses can audibly disrupt the sound generation process. One is the obvious problem of changing waveforms, phase, or such while a sound is in progress. This will result in an audible change in the output sound, although it will not be at all objectionable.

The other is to make too many accesses in one 37.5 uS (28 KHz) cycle. The internal microcode has enough spare time to allow up to 16 single-cycle accesses in one 28 KHz tick. Exceeding this number will cause the Minnie to repeat a sample on all its voices, essentially slipping one sample in time. This is also not likely to be objectionable, as it will be hard to even hear a single-sample slip. Also, it will be hard to get the 6502 to write to the innie that many times in that period.

2.5.0 ANALOG OUTPUT SECTION

The analog output section is responsible for converting the 10-bit output sample into an analog voltage or current, capable of driving the external audio line on the 3600 cart connector. It is essentially a 10-bit monolithic DAC, followed by a current buffer. It is necessary for the output to be able to swing 2 volts into 6.8k ohms, sourcing 225 uA.

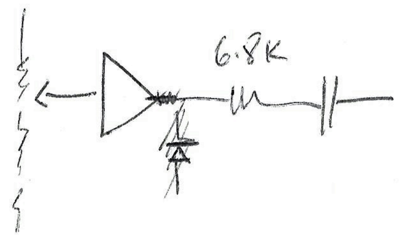
Linearity would be nice, but we'll settle for monotonicity. It also will be necessary to design carefully, to avoid hassles with the gobs of 4.5 MHz that will be getting shoved up the output pin.

2.6 MECHANICAL AND ELECTRICAL SPECIFICATIONS

The Minnie will be contained in a 24-pin .3" DIP plastic package. It will also be possible to use it in a chip-on-board environment, if desired.

2.6.1 PINOUT (obsolete)

Pin 1.	PCK2	(1.8 MHz clock from the 3600)
2.	DB7	(External 8-bit bidirectional data bus, MSB)
3.	DB6	
4.	DB5	
5.	DB4	
6.	DB3	
7.	DB2	
8.	DB1	
9.	DB0	(External data bus LSB)
10.	A0	(External address bus LSB)
11.	A1	
12.	VSS	(Gnd)
13.	A2	
14.	A3	
15.	A4	
16.	A10	
17.	A11	
18.	A12	
19.	A13	
20.	A14	
21.	A15	(external address bus MSB)
22.	AUD	(Analog audio output pin)
23.	RWF	(Read/write)
24.	VDD	(+5)



Area Speed

$200 \times \frac{1}{4} \times \frac{1}{3} \to \underline{\underline{17\text{mw}}}$

2.6.2 ELECTRICAL SPECIFICATIONS

The Minnie requires a single 5 volt supply, regulated to within +/-10%, at a supply current of not more than 50 mA. Its maximum airspeed over level ground will be 23 furlongs per fortnight, with storage conditions not to exceed 100% humidity (non-condensing) at 10 atmospheres (10,000 uBar), and it had damn well better not dissipate more than 500 mW or we're all gonna die. Absolute maximum input voltage is 7 volts, above which catastrophic discoloration of the epoxy package can be reasonably expected.