The following are the notes taken by JMG of the meeting we had to discuss accptance tests for SAM. They are sketchy, but hopfully complete.

On looking them over, it seems that there are basicly two phases of tests: acceptance tests and warranty tests. The first phase should consist of these steps:

I. run noise tests
II. run data and update command load tests
III. run usability tests (complex instruments playing some piece).
IV. accept it.

The second phase should consist of opening up the box to the general user as much as possible and really get down to work using it. This will uncover more bugs than any other method. Problems encountered in this way will expose bugs in software as well as hardware. Hardware problems will still be covered by warranty, and in fact that's what warranty is all about.

By using this division of tests, we will be able to say to SC that we have a set of tests that will take a finite amount of time to run, and we will be in a position to pay them at that time. I see no reason to hold onto the money beyond such a time as we get the remainder of their contractual obligations, and we run the above tests.

Remaining problems to clear up with SC:
- get drawings
- theory of operation manual
- 5 days consultation/instruction at user site.
- device code conflict with mappiplexer
- [nice but maybe not in contract:
  - source code for exercizers MKEXER and JAXTST, plus
  - guided tour through them.]

1. Generator tests
   a. Noise tests:
      - Sum of cosine mode in generators, should not have glitch as described in Moore thesis.
      - rapidly changing envelopes, test for pops.
      - DACs: test on scope, and with analog spectrum analyzer.
      - digital FFT of signals read through DMA.
      - glissando: pops at points of update.
      - simulations on the 10, then A:B tests
   b. Usability tests:
      - Additive synthesis (delta frq)
      - complex instruments (FM, etc.)

2. Modifiers
   - verification of functions
   - noise tests as in generators
   - filter performance

3. Data load: If MKEXER really does what Pete says, then this has been tested.

4. Mode switching: pass a shitload of update ticks, revising run modes and updating parameters, see if it gets them all.

5. Power supplies (anybody remember what this was about?)

7. Validate spec. sheet. for op codes and fields.

Whereas it seems reasonable for us at this point to try to clarify
the methodology of our acceptance tests, allow me to propose
that the following outline should guide our efforts.

I. Formalize the categories of tests.
  A. Should cover each genera of algorithm in the machine.
    1. Generators:
        -dac, data write-read, sine functons, ramp and logic functions, fm.
    2. Modifiers:
        -logic testing, arithmetic, filtering, delay linkage.
    3. Delay lines.
    4. Command and data transfer from the six.
  B. A Final Test would bring out weaknesses.

II. Tests should go from simple tests to more musically useful (complex) tests.
  A. Generic tests should be very simple and clear.
  B. A Final Test would be designed to include any weaknesses discovered in the generic tests at a greater, but no less well defined, level of complexity.
    1. This test might also be referred to as a usability test, and would want to test the following:
        a. Command load test: a compound instrument playing in polyphony a piece of musically useful complexity.
        b. Other weaknesses test, as described above. The additional level of complexity to be added would be proportional to that which would be useful in a musical context for that algorithm. For example, if there were shown to be a bug in the rate term in generators running in parallel, the final test might include an ensemble of instruments which would do this in a fixed and predetermined number of permutations, such as, 8 instruments playing simultaneously, diddling this term.
    2. The weakness tests would themselves be generic and problems encountered there would not cause new genera to come into being, but would again cause additional simplifying and clarifying tests to be run, not more and more complex tests.

III. Exaustive tests of each function of each processing element
would be beyond the scope of these tests. Presumably,
problems found later in isolated categories can be handled by warranty.
True combinatorial tests would take forever in a machine of this complexity.

HOWEVER:

IV. Problems found in implementing the generic tests must be scrutinized
by simplification, and additional tests would need to be run in an attempt
to find a specific bug. Furthermore, assuming a problem is uncovered,
this specific bug would be included in the Final Test at a level of
complexity greater than in the generic tests, but at a specific level
of greater complexity, not an expanding one.

Here is a list of the tests that have been set up so far:
30-NOV-77 1331

FILNAM	EXT	PPN	
ZANFA	SAM	SAMDGL	two generators playing Stravinsky's Fanfare for a New Theater
GLSTST	SAM	SAMDGL	generator glissando
SINTST	SAM	SAMDGL	generic generator test
SOLFEG	SAM	SAMDGL	generic generator load test (the one that failed)
SOLF	SAM	SAMDGL	simplification of above
SOLSOP	SAM	SAMDGL	likewise
SOLALT	SAM	SAMDGL	"
SOLUTX	SAM	SAMDGL	"
SOLFEW	SAM	SAMDGL	"
NULTST	SAM	SAMDGL	two generators canceling eachother
CLPTST	SAM	SAMDGL	to see what would happen when the dacs overflow
FMTST	SAM	SAMDGL	generic generator test
GOSCVF	SAM	SAMDGL	variable frequency generator
ALLGEN	SAM	SAMDGL	256 generators simultaneously, not implemented
FLTTST	SAM	SAMDGL	analog filters test (yes, it's redundant!)
GENTST	SAM	SAMDGL	
GOSTST	SAM	SAMDGL	
SFMTST	SAM	SAMDGL	sin_fm test
GFMTST	SAM	SAMDGL	generator fm test
AMPTST	SAM	SAMDGL	simplification of generator fm test
GF1TST	SAM	SAMDGL	same
SLOGFM	SAM	SAMDGL	likewise
ONEAMP	SAM	SAMDGL	likewise
AMPLNG	SAM	SAMDGL	likewise
SLOAMP	SAM	SAMDGL	likewise
MOD	SAM	SAMDGL	generic modifier logic test
UNOISE	SAM	SAMDGL	same
TNOISE	SAM	SAMDGL	same
MIX	SAM	SAMDGL	same

Current state:
Basicly the generators have been tested, exept for data transfer to the six. The genera of tests are these:
Waveforms:
-sine wave generation
-pulse and saw waves
Fm:
-Sin_fm
-fm proper
Modes:
-dac
-a,b,c _running
Fields:
-go,gj,etc. singly and gp in combination with another generator.

Additional specific tests:
1. Generator data transfer from six (so far only commands have been stuffed).
2. exponential generator envelopes.
3. Five tests of modifiers:
logic: SIGNUM,
arithmetic: MIXING,
filtering: TWO_POLES,
uniform noise: TRIGGERED_UNIFORM_NOISE
and finally, of course delay linkage.
4. Delay lines.
5. The Final Test, as described above.