SYSTEMS CONCEPTS 520 THIRD STREET SAN FRANCISCO, CALIFORNIA 94107

SYSTEMS CONCEPTS

DIGITAL SYNTHESIZER

SPECIFICATIONS

TELEPHONE: 415-442-1500 TWX: 910-372-6062

SYSTEMS CONCEPTS DIGITAL SYNTHESIZER PROGRAMMING SPECIFICATION

INTRODUCTION 780714

Generators and Modifiers

The synthesizer has two kinds of processing elements: generators and modifiers. An additional type of element, termed a delay unit, is optional.

Generators produce sine, square, and sawtooth waves, pulse trains, and equal-amplitude sum-of-cosines (band-limited pulse trains); apply linear and exponential envelopes; perform frequency modulation; can automatically sweep frequency linearly; read data from computer memory; and write data into computer memory or to digital-to-analog converters. Up to 256 generators can be active at one time.

Modifiers simulate a resonance or antiresonance; perform amplitude modulation, four-quadrant multiplication, mixing, clipping, and memory (sample and hold) functions; can generate uniform noise; and pass data to and from the optional delay units. Up to 128 modifiers can be active at the same time.

Delay units have two uses: as delay lines for signals; and to hold precomputed tables, such as time-domain waveforms. Up to 32 delay units can be active at the same time.

Passes and Ticks; Sum Memory

The processing performed on a per-sample basis comprises one pass. A pass is a series of ticks, of three types: processing ticks, overhead ticks, and update ticks. Processing ticks perform the calculations corresponding to generators and modifiers, and update ticks permit performance of commands to load new parameters. Within a pass, all processing ticks are performed first, then all overhead ticks, then all update ticks. A tick of any type takes 195 nsec. The number of processing ticks per pass is the maximum of: the number of generators used; twice the number of modifiers used. For delay units, divide the number of processing ticks minus six by four to get the number of delay memory cycles possible per pass. The number of delay units that can be used is this number less however many delay memory cycles the computer may make during the processing ticks. There are eight overhead ticks per pass. The number of update ticks per pass should be chosen according to the number of processing and overhead ticks to give the desired overall sample rate.

Information is passed among generators and modifiers through a scratchpad area called sum memory, which is divided into four 64-word quadrants. In one quadrant, sums are accumulated of generator outputs during a given pass; another quadrant holds the accumulated generator sums from the previous pass. The other two quadrants act likewise for modifier outputs. Any generator or modifier can read data from either previous-pass quadrant, and any modifier can read from the current-pass modifier quadrant also.

Computer Interface

Information is passed to and from the computer in two ways: I/O instructions, and direct memory access. With the delay memory option, a low-bandwidth bidirectional 20-bit path permits read- and write-accesses by the computer.

Computer I/O instructions perform general control, status sensing, and diagnostic functions. The direct memory access path is provided for data transfer in real time. There are three types of such data transfer: commands (to the device), read data (per sample) (to the device), and write data (per sample) (from the device). Each of these three has its own word count (WC) and core address (CA) registers in the device; they are set up by I/O instructions. Commands are always 32 bits; read data may be either 16 or 32 bits, giving a choice between packed data and full precision (the left 20 bits are significant in 32-bit mode; in 16-bit mode, the left 16-bit data item precedes the right one); write data is the left 20 of 32 bits. The device has buffering for 28 commands, 4 read-data items, and 1 write-data item.

The synthesizer can be conditioned to interrupt the computer in various circumstances. One class of them can be termed data errors: arithmetic overflow during processing, and command overrun. Command overrun occurs when a Linger command is performed which specifies a pass at least 1, but no more than 4096, before the current pass. The other class of interrupt conditions relates to direct memory access. Separate indications are provided for read data, write data, and command WCs being exhausted, and also for underrun conditions. Command underrun occurs when on an update tick there is no command to be performed (normally when there is no update activity due, a Linger command is being performed). The read data and write data underrun states occur when the device must stop its clock momentarily to wait for memory access; this means the device is not operating in real time.

PDP-10 INTERFACE

The computer interface specifications are discussed here in terms of the implementation for the PDP-10 computer. Direct memory access refers to 32-bit data and commands right-justified in 36-bit words. The synthesizer uses a group of four contiguous device codes (beginning with one which is divisible by four), referred to below as A, B, C, and D. Codes A, B, and C are used by the basic synthesizer; code D is used for the Delay Memory option. Two priority interrupt channels are employed; channel B for command word count exhausted, and channel A for all other interrupt causes.

Summary

CONO-A	18 bits:	sets overall status, diagnostic readback address
CONO-B	18 bits:	sets miscellaneous status
DATAO-A	32 bits:	sends command to be performed
DATAO-B	36 bits:	sets CA (core address) or WC (word count) for commands, read data, write data
DATAO-C	20 bits:	(only when running) for diagnostic purposes, sets write-buffer data from bits 4-23
DATAO-D	36 bits:	writes bits 0-19 into Delay Memory location designated by the ones' complement of bits 20-35. Data overwritten is saved to be read by DATAI-D.
CONI-A	20 bits:	reads overall conditions
CONI-B	16 bits:	reads cause of interrupt
DATAI-A	20 bits:	(only when not running) diagnostic readback
DATAI-D	20 bits:	reads Delay Memory data saved when overwritten by most recent DATAO-D.
CONI-D	:	reads state of TZA flag into bit 25. TZA is cleared by DATAO-D and set shortly thereafter when the overwritten data is available to be read by DATAI-D. Between the DATAO-D and the setting of TZA no DATAO should be given to the synthesizer.

```
          18 19 20 21 22 23 24 25                         31 32 33    35
         ------------------------------------------------------------
CONO-A   :  CC : T: A: B:  NN :        DDDDDDD       : R:   PIA   :
         ------------------------------------------------------------
```

```
CC: 00  no effect
    01  stop clock
    10  start clock
    11  cause one tick
T: 0  no effect
   1  reset tick counter to beginning of pass (if stopped,
      and processing ticks permitted)
A: 0  set interrupt channel A from PIA
   1  no effect
B: 0  set interrupt channel B from PIA
   1  no effect
NN: 00  no effect
    01  permit processing ticks
    10  inhibit processing ticks (all ticks update)
            Note:  To ensure that all ticks update,
            after this CONO is given the clock must
            be run at least eight ticks.
    11  (reserved)
DDDDDDD: diagnostic readback address, specifies internal
    data to be read by DATAI-A.
R: 0  no effect
   1  reset (also caused by the PDP-10 I/O Bus Reset)
          Principal effects:  stops clock; inhibits
          processing ticks (all ticks update); resets ME,
          PE, NX errors; disables stop and interrupt on
          AAA causes, CE, WE, and RE; indicates 16-bit
          read data; sets WC exhausted for commands, read
          data, and write data; marks empty the buffers
          for commands, read data and write data; sets PIA
          channels A and B to 0.  Does not reset the tick
          counter, pass counter, or CONI-B information.
          18                          28 29 30 31 32 33    35
          ------------------------------------------------------
CONO-B    :         xxx xxx xxx xx         :  ZZ :  BB .  AAA    :
          ------------------------------------------------------
```

```
ZZ: 00  no effect
    01  reset ME error
    10  reset PE, NX errors
    11  reset ME, PE, NX errors
        (for error descriptions see CONI-A below)
BB:  (decoded with AAA)
    00AAA  disable stop on cause AAA
    10AAA  enable stop on cause AAA
    01AAA  disable interrupt on cause AAA
    11AAA  enable interrupt on cause AAA
        AAA: 001  command overrun: Linger command being
                  performed specifies pass number less
                  than current pass count (but difference
                  less than 4,096 passes).
             010  modifier mixer overflow
             011  modifier multiplier overflow
             100  modifier add to sum overflow
             101  generator add to sum overflow
    00110  disable interrupt on write data WC exhausted
    10110  enable interrupt on write data WC exhausted
    01110  disable interrupt on read data WC exhausted
    11110  enable interrupt on read data WC exhausted
    01000  disable interrupt on command WC exhausted
    11000  enable interrupt on command WC exhausted
    00111  indicate 16-bit read data
    10111  indicate 32-bit read data
    01111  (reserved)
    11111  (reserved)
```

```
          16 17 18 19 20 21 22 23 24 25 26 27 28 29 30    32 33    35
         -------------------------------------------------------------
CONI-A   :AR:BR:IR:CE:WE:RE:ME:PE:NX: R:NH:CU:WU:RU: PIA-A  : PIA-B  :
         -------------------------------------------------------------
```

```
AR: interrupt desired on channel A (regardless of PIA)
BR: interrupt desired on channel B (regardless of PIA)
IR:  interrupt desired (by 11AAA cause, ME, PE, NX, WE, RE,
     CE, regardless of PIA).  The actual interrupt request
     will not occur before the interrupt-desired indication.
     The clock must be running for an interrupt request to
     be presented.
ME:  parity error detected in delay memory
PE:  parity error during direct memory access
NX:  non-existent memory addressed by direct memory
        access (PE and NX errors suppress further memory
        access and DATAO-A functions until reset by reset
        or CONO-B)
R:   clock running (not stopped)
NH:  not held (like R but also off while clock stopped
        for memory access)
WU:  set by write data underrun; cleared by this CONI
RU:  set by read data underrun; cleared by this CONI
CU:  set by command underrun; cleared by this CONI
WE:  write data WC exhausted
RE:  read data WC exhausted
CE:  command WC exhausted
PIA-A: Priority Interrupt Assignment, channel A
PIA-B: Priority Interrupt Assignment, channel B
```

```
          20 21 22 23 24 25 26 27                    35
         ----------------------------------------------
CONI-B   :I1:I2:I3:I4:I5: x:LC:        TTTTTTTTT     :
         ----------------------------------------------
```

I1: command overrun
I2: modifier mixer overflow: T...T = (2 * modifier #) + 7
I3: modifier multiplier overflow: T...T = (2 * modifier #) + 5 or 6
I4: modifier add to sum overflow: T...T = (2 * modifier #) + 9
I5: generator add to sum overflow: T...T = generator # +9
Note: I1...I5 only come on if the associated condition occurs and interrupt is enabled on it (11AAA). If I1...I5 are all off TTTTTTTTT is indeterminate. I1...I5 and LC are cleared by this CONI.
LC: (lost cause) After the interrupt cause encoded in this word occurred, but before this word was read by the computer, another of these interrupt causes occurred.
TTTTTTTTT: tick number when cause occurred (nine bits needed to allow for pipelining)

```
          0        3 4            11 12                  35
         ----------------------------------------------------
DATAO-B  :   UUUU    :   xx xxx xxx   :        A...A        :
         ----------------------------------------------------
```

UUUU:	0000	no effect
	0001	set write data CA
	0010	set read data CA
	0011	set command CA
	0101	set write data WC
	0110	set read data WC
	0111	set command WC
	others:	(reserved)

A...A (24 bits): core address (if CA)
two's complement of word count (if WC)
Note: A WC becomes not exhausted as soon as it is written into, thereby permitting memory cycles, so a CA should be written before the corresponding WC.

GENERATORS

Parameters

Associated with each generator are the following quantities:

GO	(20 bits) alpha -- oscillator frequency sweep rate
GJ	(28 bits) omega -- oscillator frequency
GK	(20 bits) theta -- oscillator angle
GN	(11 bits) number of cosines to be summed
GM	(4 bits) binary scale of cosine or sum of cosines
GP	(20 bits) delta -- decay rate
GQ	(24 bits) phi -- decay exponent
GL	(12 bits) asymptote
GSUM	(6 bits) sum memory address into which output is added
GFM	(7 bits) sum memory address from which frequency modulation data is taken

```
GFM  (7 bits) sum memory address from which frequency modulation
        data is taken
        GFM = QAAAAAA
        Q: 0  generator-last-pass quadrant
           1  modifier-last-pass quadrant
        AAAAAA:  sum address within quadrant

GMODE  (10 bits) generator mode
        GMODE = RRRREESSSS
```

Run Mode

		osc. run?	env. run?	add to sum?
RRRR:0000	inactive	no	no	no
0001	pause	no	no	no
1111	running A	yes	yes, sticky	yes
1110	running B	yes	yes, free; triggers subseq. on overflow	yes
1001	wait	yes	no	no
1101	running C	yes	yes, free; stops and triggers subseq. on overflow	yes
0111	read data from computer	no	yes	yes
0011	write data to computer	no	no	no
0010	write data to DAC (address in GO)	no	no	no

The envelope side of the generator can be sticky, which means that rather than overflow it will stay at the last value it attained before it would have overflowed; or it can be free, in which case it wraps around.

Transitions between run modes can be accomplished in various ways.

1) A command can output a new GMODE.
2) A MISC command can specify "clear all pause bits", which will cause any generator in run mode 0001 to change to mode 1111.
3) A MISC command can specify "clear all wait bits", which will cause any generator in run mode 1001 to change to mode 1111.
4) If the envelope side of a generator in run mode 1101 overflows, that generator goes to run mode 1001.
5) A generator in run mode 1001 will go to run mode 1101 if on the same pass the preceding generator (the one whose generator number is one less) caused a trigger (was in run mode 1110 or 1101 and envelope overflowed).

Envelope Mode

EE:	00	L - Q
	01	L + Q
	10	L - 2**(-Q)
	11	L + 2**(-Q)

Oscillator Mode

SSSS:	0100	sum of cosines
	0001	sawtooth
	0010	square
	0011	pulse train
	0000	sin (K)
	1000	sin (J + fm)

Processing

Calculations performed for a generator, governed by its mode, proceed as detailed below.

1) The word in sum memory addressed by GFM is read (20 bits); the sum is formed of it and the high-order 20 bits of GJ (call the result Temp0).

2) If the oscillator side is running, GO, right-adjusted with sign extended, is added into GJ.

3) If the oscillator mode is 1000, Temp0 is taken; otherwise GK. Call the 20-bit result Temp1E, and its high-order 13 bits Temp1.

4) If the oscillator side is running, Temp0 is added into GK.

5) If the run mode is 0011, the word in sum memory addressed by GFM is sent to the CPU as the next write-data item; if the run mode is 0010, it is sent to the DAC addressed by the low-order 4 bits of GO.

6) In oscillator modes other than 0000 and 1000, Temp1 is multiplied by GN. Call the low-order 12 bits of the product, with two bits equal to 01 appended to the right, the 14-bit result Temp2. In oscillator modes 0000 and 1000, Temp2 is the high-order 13 bits of Temp1E, with a bit equal to 1 appended to the right.

7) If the oscillator mode is 0000 or 1000, pi/2 is taken (the binary number 010...0); otherwise Temp1. Call the result Temp3.

8) In floating point, the product csc (Temp3) * sin (Temp2) is formed; then converted to fixed point with a scale factor of 2**(-GM). Call the result (13 bits) Temp4.

9) The result of the oscillator side (13 bits, call it Temp5) is then determined according to the oscillator mode.

```
SSSS: 0100 Temp4
      0001 Temp1 (but 0 when Temp1 is 1000000000000)
      0010 -1/2 (on a scale from -1 to +1) if Temp1 is negative,
           else +1/2
      0011 +1/2 if overflow occurred in step 1) or 4) above;
           else 0.
      0000 Temp4
      1000 Temp4
```

10) The high-order 12 bits of GQ are taken (call this Temp6).

11) If the envelope side is running, GP right-adjusted, sign extended, is added into GQ (overflow dealt with according to the run mode). (The overflow condition is GQ changing sign such that the high-order bit of the resultant GQ equals the sign bit of GP.)

12) If the envelope mode is 10 or 11, 2**(-Temp6) is looked up; otherwise Temp6 is taken. Call the resulting 12 bits Temp7. Scaling is such that if Temp6 is 0, then 2**(-Temp6) is 111 111 111 101 binary; if Temp6 is 000 100 000 000 binary, then 2**(-Temp6) is 011 111 111 110.

13) If the envelope mode is 01 or 11, Temp7 is added to GL; else it is subtracted from GL. This creates Temp8, the result of the envelope side.

14) Temp5 is multiplied by Temp8. If the run mode specifies adding into sum memory, the high-order 19 bits of the rounded product, right-adjusted with sign extended, are added into the sum memory location designated by GSUM; except that in run mode 0111, the product is added to the next read-data item from the CPU and the sum replaces the contents of the sum memory location addressed.

MODIFIERS

Parameters

Each modifier has the following numeric parameters.

M0 (30 bits) coefficient

M1 (30 bits) other coefficient

L0 (20 bits) running term

L1 (20 bits) other running term

MIN (8 bits) address in sum memory where modifier reads "A" data
MRM (8 bits) address in sum memory where modifier reads "B" data
MIN, MRM = QQAAAAAA

```
    QQ: 00  generator-last-pass quadrant
        01  modifier-last-pass quadrant
        10  modifier-this-pass quadrant
        11  (reserved)
```

AAAAAA: sum address within quadrant

MSUM (7 bits) result address in sum memory
MSUM = RAAAAAA

```
    R: 0  add to sum
       1  replace sum
```

AAAAAA: sum address in modifier-this-pass quadrant

MMODE (9 bits) modifier mode
MMODE = MMMMMAABB

AA: scale of second multiplication
BB: scale of first multiplication

For fraction multiplications:

00:	x 1
01:	x 2
10:	x 4
11:	x 8

For integer multiplications:

00:	x 1/4
01:	x 1/2
10:	x 1
11:	x 2

A multiplication involving parameter M1 will be the first multiplication; one involving M0 will be the second.

MMMMM: function

00000:	inactive
00010:	uniform noise
00011:	triggered uniform noise
00100:	latch
00110:	threshold
00111:	invoke delay unit
01000:	two poles
01001:	two poles, M0 variable
01011:	two poles, M1 variable
01100:	two zeros
01101:	two zeros, M0 variable
01111:	two zeros, M1 variable
10000:	integer mixing
10001:	one pole
10100:	mixing
10110:	one zero
11000:	four-quadrant multiplication
11001:	amplitude modulation
11010:	maximum
11011:	minimum
11100:	signum
11101:	zero-crossing pulser
others:	(reserved)

Processing

Computations performed by a modifier depend entirely on its mode. In the descriptions below, A is the 20-bit sum memory word addressed by MIN; B is the word addressed by MRM; when M0 or M1 is used, its high-order 20 bits are taken, but when a quantity is added to M0 or M1 it is added right-justified, with sign extended; S is the 20-bit result that is added into the sum memory location addressed by MSUM. DM is the 20-bit word read from or sent to a delay unit. Multiplications are 20 bits x 20 bits, signed, and the product (unless otherwise noted) is the high-order 20 bits, rounded.

```
MMMMM
-----

00000:  inactive.  S := 0

10000:  integer mixing.  S := A*M0 + B*M1 (integer multiply, low-order
        20 bits of product used; overflow ignored)

10100:  mixing.  S := A*M0 + B*M1

00100:  latch (sample and hold).  S := L1;  If B*M1 is not 0, L1 := A

11100:  signum.  If A*M0 is less than B*M1, then S := -1 (integer);
                 if A*M0 equals B*M1, then S := 0;
                 if A*M0 is greater than B*M1, then S := 1 (integer)

11101:  zero-crossing pulser.  Temp0 := B*M0; Temp1 := L1*M1;
        if Temp1 is not 0 and either Temp0 is 0 or Temp0*Temp1 is
        negative then S := -epsilon, else S := 0; L1 := Temp0
        (The term -epsilon is a binary number with all bits set.)

11011:  minimum.  S := min (A*M0, B*M1)

11010:  maximum.  S := max (A*M0, B*M1)

11001:  amplitude modulation.  S := L1*M1;  L1 := A * ((B+1)/2)
        (The term ((B+1)/2) interprets B as a signed two's-complement
        fraction ranging in value from -1 to +1-epsilon.)

11000:  four-quadrant multiplication.  S := L1*M1; L1 := A*B
```

10001: one pole. S := L1*M1 + B*L0; L1 := S

10110: one zero. S := L1*M1 + L0*M0; L0 := L1; L1 := A

01000: two poles. S := L1*M1 + L0*M0 + A; L0 := L1; L1 := S

01001: two poles, M0 variable. S := L1*M1 + L0*M0 + A; L0 := L1; L1 := S; M0 := M0 + B

01011: two poles, M1 variable. S := L1*M1 + L0*M0 + A; L0 := L1; L1 := S; M1 := M1 + B

01100: two zeros. S := L1*M1 + L0*M0 + A; L0 := L1; L1 := A

01101: two zeros, M0 variable. S := L1*M1 + L0*M0 + A; L0 := L1; L1 := A; M0 := M0 + B

01111: two zeros, M1 variable. S := L1*M1 + L0*M0 + A; L0 := L1; L1 := A; M1 := M1 + B

00010: uniform noise. S := L0 + L1*M0 (integer multiply, low-order 20 bits of product used; overflow ignored); L1 := S

00011: triggered uniform noise. S := L0 + L1*M0 (integer multiply, low-order 20 bits of product used; overflow ignored); if B*M1 (integer multiply, low-order 20 bits of product used; overflow ignored) is not 0, L1 := S

00110: threshold. If A*M0 + L0 is less than 0, then S := 0; if A*M0 + L0 is equal to or greater than 0, then S := B*M1

00111: invoke delay unit.
Unit # := MRM (low-order 5 bits);
S := L0 + L1*M0; L0 := DM; Temp0 := A + DM*M1;
L1 := Temp0; DM := Temp0

Timing Considerations

The following relationships apply to references to the modifier-this-pass quadrant of sum memory.

1) Modifier number M writes into sum memory (read-add-write or replace) on tick number 2*M + 7.

2) Modifier number M reads word B on tick number 2*M.

3) Modifier number M reads word A on tick number 2*M in the following modes: integer mixing; mixing; signum; minimum; maximum; amplitude modulation; four-quadrant multiplication; threshold.

4) Modifier number M reads word A on tick number 2*M + 6 in the following modes: latch; one zero; two poles, two zeros (all six modes); invoke delay unit.

DELAY UNITS

A common pool of addressable memory, which may comprise up to 65,536 20-bit words, is available for use by the delay units. By programming, each active delay unit is assigned its own contiguous area of the memory.

Quantities

Each delay unit has the following numeric parameters.

P mode (4 bits). The mode is interpreted as follows:

```
          mode: 0000  inactive
                1000  delay line
                1010  table look-up
                1011  table look-up, argument rounded
                others: (reserved)
```

Z unit length (16 bits) or binary scale factor (4 bits).
In delay line mode, Z gives 1 less than the total number of locations in delay memory used by this delay unit, i.e. the index of the last delay memory address for this unit. In table look-up modes, the low-order four bits of Z specify the number of binary places that the argument is shifted to the right before it is used to address the memory; if rounding is specified, the address after shifting is incremented by 1 if the most-significant bit shifted out was a 1.

Y index (16 bits). In delay line mode, this is the running index on the memory area for the unit.

X base address (16 bits). The base address is the lowest-numbered delay memory location used by this unit.

Processing

In inactive mode, delay memory is not modified and the unit returns indeterminate results. Delay units not accommodated due to the number of ticks in a pass act as if in the inactive mode. If the number of processing ticks is 4*n + m where m is 1, 2, or 3, delay unit number n should be put in the inactive mode.

In delay line mode, a 20-bit data word is received from the modifier that calls for the delay unit, and another 20-bit word is sent to it. The word received is put into the next slot in the delay line. It will be retrieved and sent back to the modifier Z+3 passes later.

In table look-up mode, the 20-bit data word received from the modifier is shifted to the right Z bits, bringing in zeros, and the right 16 bits of the result are used to address the memory area assigned to the unit. The 20-bit word in the addressed memory location is returned to the modifier three passes later.

COMMANDS

All commands have 32 bits. Generally the left 20 bits are data, the next 4 or 5 bits identify the kind of parameter, and the last 8 or 7 bits address the generator or modifier affected. If more than one data field is packed in the 20 bits, disable bits will be provided to facilitate loading a subset of the fields. In a few cases, a bit is also provided in the data area to clear (put to zero) a related parameter in the same generator or modifier.

```
        4                       23 24           28 29 30 31 32 33 34 35
    -----------------------------------------------------------------
        :         (20) data         : 0  0  0  0  0:  RR : x  x: W: P: S:
MISC-----------------------------------------------------------------

        RR: 00  no effect
            01  load DX from data
            10  load TTL buffer A from left 16 bits of data
            11  load TTL buffer B from left 16 bits of data;
                  set analog output filters from right 4 bits of data:
                    01xx  Mode 0
                    00nn  Mode 1, frequency f0, f1, f2, or f3 according
                        to nn
                    1xxx  no change
        W:  if 1, clear all wait bits
        P:  if 1, clear all pause bits
        S:  if 1, stop clock
        4               19 20   23 24           28 29 30 31             35
    -----------------------------------------------------------------
        :     (16) data     :(4)data: 0  0  0  0  1: U  U:  (5) unit #  :
    -----------------------------------------------------------------
DLY X, Y, Z
        UU:  00  X     16 bits base address; clear Y
             01  Y     16 bits one's complement of index
             10  Z,P   16 bits delay unit size minus 1, or scale (low
                             4 bits of 16); 4 bits mode
             11  (unused)
        4                       23 24           28 29 30 31 32 33     35
    -----------------------------------------------------------------
        :         (20) data         : 0  0  0  1  0: x  x: T  T: x  x  x:
    -----------------------------------------------------------------
TIMER
        TT: 00  no effect
            10  Linger: process no further commands until pass counter
                    equals data
            11  clear pass counter, then Linger as for 10
            01  set pass counter from data
```

```
     4                          23 24          28 29 30 31 32 33    35
    ------------------------------------------------------------------
   : xxx xxx xxx x : (10) data  : 0  0  0  1  1: x  x: 0: Q: x  x  x:
    ------------------------------------------------------------------
# TICKS
        Q: 0  designate highest-numbered processing tick per pass
                    (should not exceed 255)
           1  designate next-to-highest-numbered tick (processing
              plus overhead plus update) per pass
     4                          23 24    26 27 28                   35
    ------------------------------------------------------------------
GQ  :         (20) data         : 0  0  1: E:      (8)   gen #      :
    ------------------------------------------------------------------

        E: 0  Q right-adjusted, sign extended
           1  Q left-adjusted, low bits from left of DX; clear DX
     4                          23 24    26 27 28                   35
    ------------------------------------------------------------------
GJ  :         (20) data         : 0  1  0: E:      (8)   gen #      :
    ------------------------------------------------------------------

        E: 0  J right-adjusted, sign extended
           1  J left-adjusted, low bits from left of DX; clear DX
     4                          23 24       27 28                   35
    ------------------------------------------------------------------
GP  :         (20) data         : 0  1  1  0:      (8)   gen #      :
    ------------------------------------------------------------------

     4 5 6   8 9      19 20   23 24       27 28                     35
    ------------------------------------------------------------------
GN, :N:M:x x x: (11) GN :(4) GM : 0  1  1  1:     (8)    gen #      :
GM  ------------------------------------------------------------------

        N:  if 1, disable loading GN
        M:  if 1, disable loading GM
     4 5 6          17 18       23 24       27 28                   35
    ------------------------------------------------------------------
GL, :L:S:  (12) GL   : (6) GSUM : 1  0  0  0:      (8)   gen #      :
GSUM------------------------------------------------------------------

        L:  if 1, disable loading GL
        S:  if 1, disable loading GSUM
     4                          23 24       27 28                   35
    ------------------------------------------------------------------
GK  :         (20) data         : 1  0  0  1:      (8)   gen #      :
    ------------------------------------------------------------------
```

```
           4 5 6 7          16 17   23 24        27 28                 35
          ---------------------------------------------------------------
          :M:F:C:  (10) GMODE :(7) GFM: 1  0  1  0:      (8)   gen #     :
          ---------------------------------------------------------------
GMODE,
GFM         M:  if 1, disable loading GMODE
            F:  if 1, disable loading GFM
            C:  if 1, clear GK
           4                        23 24        27 28                 35
          ---------------------------------------------------------------
GO        :          (20) data        : 1  0  1  1:      (8)   gen #     :
          ---------------------------------------------------------------

           4                        23 24     26 27 28 29              35
          ---------------------------------------------------------------
MM        :          (20) data        : 1  1  0: V  V:      (7)   mod #  :
          ---------------------------------------------------------------

            VV: 00  M0 right-adjusted, sign extended
                01  M1 right-adjusted, sign extended
                10  M0 left-adjusted, low bits from left of DX; clear DX
                11  M1 left-adjusted, low bits from left of DX; clear DX
           4                        23 24        27 28 29              35
          ---------------------------------------------------------------
ML        :          (20) data        : 1  1  1  0: N:      (7)   mod #  :
          ---------------------------------------------------------------

            N: 0  L0
               1  L1
           4 5 6 7 8        16 17   23 24           28 29              35
          ---------------------------------------------------------------
          :M:S:C:H: (9) MMODE :(7)MSUM: 1  1  1  1  0:      (7)   mod #  :
          ---------------------------------------------------------------
MMODE,
MSUM        M:  if 1, disable loading MMMMM bits of MMODE
            S:  if 1, disable loading MSUM
            C:  if 1, clear L0
            H:  if 1, disable loading AABB bits of MMODE
           4 5 6 7 8      15 16     23 24           28 29              35
          ---------------------------------------------------------------
          :R:I:C:x: (8) MRM : (8) MIN : 1  1  1  1  1:      (7)   mod #  :
          ---------------------------------------------------------------
MRM,
MIN         R:  if 1, disable loading MRM
            I:  if 1, disable loading MIN
            C:  if 1, clear L1
```

SYSTEMS CONCEPTS DIGITAL SYNTHESIZER ANALOG OUTPUT SPECIFICATION

The signal path for one analog output involves the following sections:

Channel selection logic (addressing)
Digital hold register
Digital to analog converter
Sample-and-hold
Program-controlled low-pass filter
Buffer amplifier.

Each section is specified at 25 degrees C as follows.

Channel selection logic: 4 bits (1 of 16)

Digital hold register: 14 bits

Digital to analog converter: 14 bits
Linearity: 0.005%

Sample-and-hold: full power bandwidth 0 to 40 kHz

Filter: two modes
Mode 0: 1-pole RC at 200 kHz
Mode 1: 6-pole Butterworth, 4 programmable frequencies subject to the relationships f0=A, f1=A+B, f2=A+C, f3=A+B+C; full power bandwidth 0 to 18.5 kHz max.

Buffer amplifier: output +/- 5 V max., unbalanced
Output current: 4 mA max.
Short circuit protection: to ground only
Full power bandwidth: 0 to 18.5 kHz for 10 V swing
Output source impedance: 100 ohms
Output connector: BNC jack

The following are overall figures with Mode 0 filtering:

Gain error: 2.5%

Offset error: 20 mV

Noise at sampling rate and its harmonics: 10 mV max. (RMS)

Other noise 10 Hz to 50 kHz: 1 mV max. (RMS)