SYSTEMS CONCEPTS DIGITAL SYNTHESIZER SPECIFICATIONS

SYSTEMS CONCEPTS SAN FRANCISCO, CALIFORNIA

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

RRRR		osc. run?	env. run?	add to sum?
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 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: 8-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)

SYSTEMS CONCEPTS   524 SECOND STREET SAN FRANCISCO, CALIFORNIA 94107

SYSTEMS CONCEPTS

DIGITAL SIGNAL SYNTHESIZER

The Systems Concepts Digital Synthesizer is a computer-driven real-time device which creates signals such as represent the sounds of music and speech. It eliminates the former problems of analog synthesizers, such as drift, poor tracking between units, inaccuracy, and inflexibility. It adds the benefits of control from a general-purpose computer, with which sound can be composed, edited, and remembered or recalled in real time or at any slower rate, and it matches the computer in rapid flexibility.

Basic elements of the Digital Synthesizer are generators and modifiers. Generators are controlled sources of signals, and modifiers are controlled signal processors.

Each generator can provide any of the following waveforms: sine, sum of cosines (equal amplitude harmonics), square, sawtooth, or impulse train; performs frequency modulation if desired; and automatically can apply any of the following envelopes: linear rising or falling, exponential growth or decay, or asymptotic rise or fall.

Each modifier can do any one of the following: simulate a pole or pole pair (resonator), simulate a zero or pair of zeros, scale an input, mix two inputs, perform amplitude modulation or ring modulation (four-quadrant multiplication), or generate uniform noise. Basic nonlinear operations are also provided. Arbitrarily complex filtering (low pass, high pass, band pass, band stop) can be accomplished by cascading pole pairs and zeros.

TELEPHONE: 415-433-5400   TWX: 910—372—6062

System architecture permits sums to be formed of the outputs of any number of generators or modifiers. The output of any generator or modifier, or the sum of several such outputs, can then be used as the data input or modulation input to any modifier, or as the frequency modulation input to any generator.

An optional Delay Memory attachment permits any modifier to act as a delay unit or as an all-pass reverberator.

The number of generators and modifiers available at any instant depends on the sample rate at which the synthesizer is operated.

Sample Rate	Generators Available	Modifiers Available
50 KHz	96	48
30 KHz	160	80
20 KHz	240	120
18.75 KHz or less	256	128

Output can be sent either to four high-resolution digital-to-analog converters, or back to the controlling computer for further processing or storage. Complete test and diagnostic features are built in. Interfacing can be provided for any positive-logic TTL or DTL computer.

The Systems Concepts Digital Synthesizer

Introduction

This is a discussion of the Systems Concepts Digital Synthesizer from three viewpoints. First, the functional characteristics of the original design are reviewed. Then follows a survey of the facilities and techniques used in the engineering, manufacture, and checkout of the Synthesizer. Last, the performance of the finished unit is compared to the design goals and some comments are offered on the important issue of long-term reliability.

Functional Characteristics

From the start, the Synthesizer was designed with sufficient word lengths, computing power, and flexibility of interconnection to serve the needs of serious research and composition efforts. The following techniques of synthesis and processing are possible, and may in fact be in progress simultaneously: additive; subtractive; modulation; delay/reverberation; table look-up; DMA (direct memory access). Each of these techniques has proved its worth in software synthesis and should be supported by a comprehensive hardware synthesizer.

Additive Synthesis: The Synthesizer has 256 processing elements named generators. Each generator has an oscillator side and an envelope side; their instantaneous product is the output of the generator. The envelope side has the conventional angle, frequency, and sine look-up hardware, as well as the facility to bypass the sine function to produce sawtooth, square, and pulse-train wave forms. In addition, there is an fm input from other elements of the Synthesizer, and each generator also has a frequency sweep parameter built in to provide a linear rate of change in frequency. Each oscillator also can perform the Winham-Steiglitz sum-of-cosines algorithm to produce a band-limited pulse train. On the envelope side, each generator can create a linearly rising or falling envelope or one showing exponential growth or decay, upward or downward, around an arbitrary asymptote. A trigger feature permits a series of generators to have their envelope sides chained together, producing more complex envelope forms without the need for computer intervention. The product of oscillator and envelope is added into one of 64 locations in sum memory. Each generator can specify which location its output goes to; when several generators designate the same location, it performs the addition which is the basis of the additive synthesis technique. Word sizes in the generators differ according to function. The looked-up sine value, for instance, in accordance with perceptual tests, has twelve fraction bits, four exponent bits, and a sign bit. The frequency, by contrast, has 28 bits, so that it can be augmented slowly by the frequency-sweep term.

Subtractive Synthesis: There are 128 processing elements called modifiers. Among other possibilities, each modifier can be conditioned to act as a resonator (pole pair) or antiresonator (zero pair). More complex filtering operations can be implemented by cascading or paralleling modifiers. Each modifier has separate parameters indicating where in sum memory to find its input data, and where in sum memory to add its output data. A second input from sum memory can be used to sweep either of the filter coefficients. Possible excitation sources for filtering include one or more generators, perhaps in sum-of-cosines mode; or a modifier in uniform noise (similar to white noise) mode.

Modulation: Frequency modulation, as mentioned above, is available with every generator. Amplitude modulation and four-quadrant multiplication (corresponding to the analog process of ring modulation) are modes to which any modifier can be configured. Other modes include basic nonlinear operations such as minimum, maximum, sample and hold, threshold detect, and zero-crossing pulser. Since each of the 128 modifiers can be set to its own mode, depending on the needs of a particular piece the Synthesizer may be arranged to have as many as 128 resonators, for instance, or in another case as many as 128 ring modulators. Data paths in the modifiers are 20 bits wide, including the multiplier.

Delay and Reverberation: The Synthesizer has provision for 64K words (K=1,024), 20 bits wide, of <u>delay memory</u>. This is accessed by 32 <u>delay units</u>, which in turn communicate with the other elements of the Synthesizer through modifiers in delay mode. Each modifier in delay mode specifies which delay unit it deals with, and each delay unit specifies the area it uses in delay memory. Each combination of a modifier and a delay unit can perform not only the delay line function, but also various reverberation processes including the all-pass configuration of Schroeder. As with all parameters in the Synthesizer, those governing the reverberation characteristics can be altered by computer control during the progress of a piece.

Table Look-Up: Each of the 32 delay units can use delay memory for table look-up purposes instead of as a delay line. This feature can be used for stored waveforms or envelopes; and also to look up mathematical functions, such as square root, which may be needed for certain synthesis and signal-processing algorithms.

DMA: As it runs, the Synthesizer can take data streams, each comprising a 20-bit word per sample, direct from computer memory, and likewise can put data streams into memory. This can be used to advantage in several ways. A piece which exceeds the capacity of the Synthesizer can be run in several passes; the Synthesizer can merge signals it creates with sounds generated elsewhere; it can be used to process signals from another source, such as natural sounds, possibly returning the results to the computer for further use.

Speed: Based on an analysis of increasing performance versus increasing cost, the following was established as a design goal: In 780 nanoseconds, the Synthesizer should do the processing for four generators, two modifiers, and one delay unit.

Development

Engineering of the Synthesizer made heavy use of computer-aided design techniques developed on the Systems Concepts in-house computing facility. All schematic drawings for the system were done by machine, and machine-checked for consistency. For the printed-circuit logic boards, computer programs did the parts placement, resulting in assembly drawings; and then laid out the complete wiring of each card. Results were full-size artwork and a paper tape to run the numeric-controlled drill making the printed circuit boards. For those portions of the Synthesizer on Wire-Wrap panels, the computer system produced data files for fully automatic machine wrapping: this avoids a significant source of error and ensures the best possible workmanship. As another part of the design automation process, various items of metalwork--brackets, chassis, etc.--were designed with the aid of computer programs which created paper tapes to run the actual metalworking machines.

Checkout too made heavy use of the computer. A PDP-10 system was used to operate the Synthesizer, with the help of various assembly, editing, and debugging programs. Interactive routines were developed to send various strings of commands to the Synthesizer and to observe the results in real time in both analog and digital domains. Instrumentation used in checkout included a 16-channel logic analyzer and an audio-frequency spectrum analyzer, both of which proved to be of great value. One more tool of great importance in checkout was provided by the diagnostic hardware in the Synthesizer and the diagnostic software which employs it. More than 10% of the hardware in the Synthesizer is strictly for diagnostic purposes: it permits a computer program to check the calculations performed by the Synthesizer at each step of the processing, thereby pinpointing any failure. In support of this hardware, some 50,000 words of diagnostic software have been written.

Following checkout the unit, comprising approximately 2,500 integrated circuits, met all design goals.

Reliability and Maintainability

Long-term reliability and maintainability must be considered at all stages in the development of a system. In this Synthesizer, these issues have been addressed by techniques including conservative design, careful workmanship, extensive diagnostic features, and comprehensive hardware documentation. Based on our experience with these techniques in similar products, we expect the Synthesizer to compare favorably in terms of reliability with other devices of similar complexity. One measure of reliability is voltage margin: before shipment, margins on the 5-volt supplies exceeded 10.4 volts, with all elements of the Synthesizer being exercised at full speed. This gives substantial protection against failure due to environmental change and aging over the life of the unit.

A General-Purpose Digital Synthesizer

Peter R. Samson

Systems Concepts, Inc., San Francisco, California

The development of digital music synthesis has been handicapped by its severe computational requirements. A large digital synthesizer has been developed to meet these requirements and to perform synthesis of complex musical sounds in real time. Its architecture provides a large number of building blocks for digital synthesis and processing, and means for rapid and complete control of their interconnections.

0. INTRODUCTION

Over the past twenty years, digital music synthesis -- usually performed on large general-purpose computer systems with the aid of complex programs -- has given us not only a variety of musically significant compositions, but also several fundamental and elegant techniques for sound synthesis and a tool of unparalleled flexibility for psycho-acoustic research.

Digital synthesis offers numerous advantages over analog techniques. Every processing element can be controlled precisely, instantaneously, and repeatably. Digital processing, with its inherent accuracy and stability, can perform the tasks of current analog modules with substantially less noise and distortion, and introduces many new sonic resources, such as time-varying timbre and reverberation.

This work has been held back, however, by the sheer volume of computation required to digitally synthesize sounds of musical interest, and especially for complexes of such sounds. A time scale of 100 -- meaning 100 seconds of computational effort for each 1 second of sound -- is not uncommon. This is despite the use of powerful computers, and numerous simplifying assumptions and restrictions in the software. This time scale effectively precludes interactive experimentation and composition, and imposes a severe economic handicap on the otherwise attractive methods of digital synthesis.

It has recently become feasible to develop digital synthesis hardware, operated as a peripheral device in a computer system, which can regain the time factor of 100 or more. This article describes one such digital synthesizer, designed to meet the computational needs of composers, psychoacousticians, and musical researchers.

This synthesizer is viewed as general-purpose, not only because its design encompasses the currently favored synthesis techniques, but also because it offers basic computational building blocks which can be interconnected, under program control, to perform at high speed new synthesis techniques as they are developed.

Its architecture and performance specifications, in essentially their present form, were first presented informally at the Computer Music Conference at Michigan State University in 1974. The prototype unit was installed at the Stanford University Center for Computer Research in Music and Acoustics in October of 1977, and has since been in service there around the clock.

1. GENERAL DESCRIPTION

A block diagram of the synthesizer is shown in Fig. 1. There are six major functional blocks: the computer interface, the generators, the modifiers, the delay units, sum memory, and the digital-to-analog converters (DACs). The computer interface provides bidirectional communication with both the Input/Output Bus and the Memory Bus of the host computer; it also contains a first-in-first-out buffer (FIFO) to minimize delays in processing which might otherwise arise due to the comparatively slow speed of the computer memory. The generators and modifiers perform the actual computations for signal synthesis and processing, described in detail below. The delay units provide bulk storage for use as delay lines or stored function tables. Up to 16 DACs can be provided: these include appropriate deglitching and computer-selectable low-pass filters.

Associated with each processing element is a set of parameters. Each parameter can be categorized as either dynamic or static. The dynamic parameters are those whose values change every sample: the phase angle of an oscillator, for instance. The static parameters are not changed on a per-sample basis, though the term "static" is a relative one: they may be changed by computer command hundreds or thousand of times per second. Static parameters may be further subdivided into coefficients (numeric quantities comparable to knob settings), sum memory addresses (describing the interconnection of processing elements), and modes (which select the specific function performed by a processing element).

To produce the final output data comprising one sample of audio (for however many DACs are in use), certain calculations must be performed for each active generator, modifier, and delay unit. These calculations comprise one pass, which in the hardware is a series of 195-nanosecond ticks. The number of ticks in a pass can be programmed for a given composition, depending on the desired sample rate and the maximum number of processing elements to be active at one time.

2. SUM MEMORY

The interconnections between generators and modifiers are accomplished through the sum memory. Sum memory is divided into four quadrants, designated SA, SB, SC, and SD in Fig. 2. SA accumulates sums of generator outputs during a given pass (sample period); SB holds the totals of generator outputs from the previous pass; quadrant SC accumulates sums of modifier outputs during a pass; and SD holds the totals of modifier outputs from the previous pass. Each of the 256 generators has a parameter denoting which of 64 locations in quadrant SA the generator output is added into; and another parameter denoting which of 128 locations in sum memory (quadrants SB and SD) it takes its frequency modulation input from. Similarly, each of the 128 modifiers has a parameter indicating which of the 64 locations in quadrant SC its output is added into; and two other parameters indicating where in sum memory (quadrants SB and SD) its two inputs are to be found. This four-quadrant organization not only meets the severe bandwidth requirements of the sum memory (an average of five read or write accesses every 195 nanoseconds), but also provides a significant programming benefit: the signal flow from one processing element (or from a group of them added up) to another element does not put any restriction on the sequencing of the elements involved. For example, the output of generator number 2 can be used as the frequency modulation input to generator number 1, even though number 2 is performed by the hardware after number 1. Additional features are provided whereby the user can direct a modifier to take its input from quadrant SC, and to put its output into SC by replacement rather than addition. These features can be used to reduce the number of sum memory locations used in a configuration with cascaded modifiers, at the cost of requiring proper ordering of the modifiers involved.

3. GENERATORS

Most of the signal synthesis operations are performed by the generators. Fig. 3 shows the structure of each generator; to the user it appears as if the entire block is replicated 256 times, though this is in fact accomplished with one set of computational hardware time-multiplexed among 256 sets of data. Each generator has an oscillator portion, an envelope portion, and a multiplier which multiplies the oscillator result with the envelope result to produce the final output of the generator.

3.1. Oscillators

Each oscillator can produce any one of four standard waveforms -- sine, sawtooth, square, or pulse train -- or any one of a family of band-limited "buzz" waveforms. These last are sums of the first harmonics at equal amplitude, according to the sum-of-cosines relation noted by Winham and Steiglitz (1):

$$\cos kt + \cos 2kt + \ldots + \cos Nkt + 1/2 = \frac{1}{2}\,\frac{\sin [(2N+1)(kt/2)]}{\sin (kt/2)}$$

These waveforms are useful as harmonic-rich sources for subtractive synthesis. As N increases, the waveshape approaches that of the pulse train, but with no energy at frequencies beyond the Nth harmonic. By proper control of N, components above half the sampling rate can be avoided, thereby preventing the (usually) undesired nonharmonic aliases of those components. The sawtooth, square, and pulse train waves are intended primarily for use as control signals fed into further processing elements. Additionally, a binary scale factor M has effect on the sine and sum-of-cosines waves. Not only the choice of waveshape, but also the number of harmonics N and scale factor M can be specified independently for each of the 256 oscillators.

Basic to the oscillator are the frequency register J and the angle register K. In each sample period, for each active oscillator, the contents of J are added into K. The angle K is then taken as an unsigned fraction of a cycle (MSB = 180 degrees), and applied to the sine and "buzz" calculation sections. While only the high-order 13 bits of the angle are used to calculate the sine, a full 20 bits of angle is maintained and accumulated, giving frequency resolution to approximately 0.02 Hz (at 20 kHz sampling rate). This is necessary for proper control of beats and similar musical phenomena.

A special generator mode is provided in which angle accumulation is bypassed, and the sine is taken of the sum of J (normally the frequency) and the input from sum memory (normally used for FM). This permits a generator to be used for computation of the sine or cosine of a term in sum memory.

In addition to the frequency and angle, each oscillator has a sweep rate register W. Its contents are added to the frequency in J once each sample period, thereby linearly sweeping the frequency of the oscillator. Over an extended time period, such an effect is heard as a glissando. Over a shorter time period, so that it is not perceived explicitly, this effect contributes significantly to the interest and character of sounds made by additive synthesis. The frequency register is 28 bits long, with the sweep rate added into the low-order 10 bits. This provides increasing and decreasing sweep rates from about 1.5 Hz/sec to about 780,000 Hz/sec in magnitude (at 20 kHz sampling rate).

Also added into the angle register for each sample is the FM (frequency modulation) term from sum memory. The FM synthesis method of Chowning (2) can be implemented with any two generators, the output of one passing through any of 64 sum memory locations to the FM input of the other. Alternatively, though, that sum memory location can accumulate the sum of several generator outputs, or of one or more modifier outputs, for more complex FM synthesis techniques. Also several generators can take their FM input from the same sum memory location.

3.2. Envelopes

The envelope portion of each generator has a running term Q and an increment P. These are analogous to the angle and frequency parameters of the oscillator side, but more bits are provided in P to accommodate the case of an envelope changing steadily over a period of time substantially longer than the period of a typical waveform. No modulation or rate sweep provisions are included for envelopes, since these operations can be performed by modifiers.

Normally, the envelope increment is added into the running term at each sample period. Certain generator modes are provided, however, in which the addition is suppressed if it would result in overflow (going from a very large value to one near zero, or vice versa). In these modes the envelope "pins" or "sticks" when it reaches a maximum or minimum value.

The mode of the generator determines whether the running term is taken directly as a linear envelope, or its exponential is taken. The data chosen by this decision is then, according to the generator mode, either added to or subtracted from an asymptote, L -- a static parameter of the generator -- to produce the final instantaneous value of the envelope.

The oscillator value and the envelope value are multiplied to yield the value actually output by the generator. This result is added into sum memory at the location denoted by the sum address parameter for the generator.

3.3. Special Generator Modes

Any generator can be put into Pause mode or Wait mode. In either of these modes, the addition of the final result to sum memory is not performed and the oscillator angle is not updated. In Pause mode, updating of the envelope running term is also suppressed. Simple means are provided in the interface to enable all generators in Pause mode, or all those in Wait mode, to be put into normal running mode at the same instant. Also a trigger facility enables one generator to go from running to Pause mode when its envelope overflows or sticks, and simultaneously to signal another generator to go from Pause to running mode. By this means, with proper choice of asymptotes, a segmented envelope can be applied to a waveform without computer intervention; the cost is one generator per segment.

Finally, three generator modes are included which do not actually generate signals but are used to pass data between sum memory and computer memory, or from sum memory to the DACs. These are: Read Data mode -- computer memory to sum memory; Write Data mode -- sum memory to computer memory; and DAC mode -- sum memory to DAC. The source sum memory address is given by the generator FM address parameter; the destination sum memory address by the sum address parameter; and the DAC number by W, which in normal modes is the oscillator sweep rate. Computer memory addresses are provided by the computer interface.

4. MODIFIERS

The modifiers are the second major class of processing elements in the synthesizer. They are most often used to take signals generated by other elements and modify them, such as by filtering, or to combine them, such as by mixing or modulation. Modifiers can also be used as sources of signals, including pseudo-random data (white noise).

Like the generators, the modifiers are in fact implemented by time-multiplexing one set of computational data paths among many sets of data comprising the static and dynamic parameters of each modifier. In the case of the modifiers, this multiplexing can be up to 128-fold. The modifier data paths, shown in simplified form in Fig. 4, are sufficiently general to enable each modifier to perform any of a variety of algorithms, depending on its mode parameter.

Certain characteristics are shared by all modifier modes. A modifier can perform at most two multiplications; these take two signed 20-bit factors and produce a signed 20-bit product (either integer or fraction). Associated with each multiplication operation for a given modifier is a two-bit binary scale factor: for fractional products this imposes a further scaling by 1, 2, 4, or 8; for integer products by 1/4, 1/2, 1, or 2.

Each modifier can take at most two inputs from sum memory, and yields one output; the output either can be added into a sum memory location or can replace its contents. Each modifier can have up to two coefficients (static 20-bit numeric parameters, A and B) and up to two running terms (dynamic 20-bit numeric parameters, Y and Z). The word size of 20 bits was chosen after analysis of roundoff noise in cascaded filter applications.

4.1. Modifier Modes

Mixing: The two inputs are multiplied (fraction or integer) by the two coefficients, respectively; the scaled products are added to form the result.

Modulation: The two inputs are multiplied together, either as signed fractions (four-quadrant multiplication) or with one signed and the other unsigned (amplitude modulation). The product is then multiplied by a coefficient and scaled to give the result.

Two Poles: One input, two coefficients, and two running terms are used to implement the recurrence formula:

$$\mathrm{out}(nt) = \mathrm{in}(nt) + A\ \mathrm{out}((n-1)t) + B\ \mathrm{out}((n-2)t)$$

where in(nt) is the input for sample n and out(nt) the corresponding output, and A and B are the coefficients. Additionally, either coefficient may be ramped by the other modifier input to sweep the filter frequency. To accommodate sweep rates that are useful in practice, each coefficient has a 10-bit low-order extension. (In such uses, the coefficients are in fact 30-bit dynamic parameters.) The ramp term from sum memory is aligned so its low-order 10 bits are added into the extension and its high-order 10 bits are added into the low 10 bits of the 20-bit coefficient. The sign bit of the ramp term is extended into the high-order bits of the coefficient.

Two Zeros: The recurrence formula performed is:

$$out(nt) = in(nt) + A\ in((n-1)t) + B\ in((n-2)t).$$

Like Two Poles, this mode has the option of sweeping either coefficient.

One Pole, One Zero: These implement respectively the formulas:

$$out(nt) = A\ in(nt) + B\ out((n-1)t);\ \text{and}$$

$$out(nt) = A\ in((n-1)t) + B\ in((n-2)t).$$

Maximum, Minimum: The two inputs are multiplied by the two coefficients, respectively; the algebraically greater product in the case of Maximum, or the lesser one in the case of Minimum, is taken as the output. Rectification and clipping are particular cases of these operations.

Zero-Crossing Pulser: A running term is used to retain the previous value of the input, i.e. its value during the previous sample period. This is compared to the present input. If the previous sample value was not zero, and the sample has either become zero or changed sign, a non-zero output is produced; otherwise the output is zero.

Signum: Each input is multiplied by the corresponding coefficient and the products are compared. The output of the modifier is -1, 0, or +1 according to whether the first product is less than, equal to, or greater than the second product. Hard clipping is one case of this mode.

Latch: The data at the signal input is passed unchanged to the output so long as the control input signal is nonzero. When the control input signal is zero, the previous output is repeated. This is equivalent to the analog track-and-hold operation.

Threshold: If the control input is below the value specified by one coefficient, the output is zero. Otherwise the output is the product of the other input and the other coefficient.

Uniform Noise: A digital equivalent to white noise is generated by the linear congruential method discussed by Knuth (3). The spectral characteristics of the output are actually dependent on the coefficients and initial value of the running term; while these are usually chosen for white noise, they can be altered to introduce coloration. A related mode is Triggered Uniform Noise, which holds a given pseudo-random value for successive samples as long as the control input signal is zero.

Invoke Delay Unit: This mode permits use of the delay memory for table look-up, delay line, comb filter, or reverberator purposes. These operations are discussed further below.

While each of these modes provides a useful function in itself, they can also be taken as building blocks to be assembled into more complex functions. A number of modifiers in Two Poles and Two Zeros modes, for instance, can be cascaded or paralleled to perform filtering or equalization as needed. The particular modes that have been implemented were chosen to accomplish the most common operations using the smallest number of modifiers, and to provide a variety of basic linear and non-linear operations -- including stored functions and waveform lookup -- which can be combined to form complex or non-standard configurations. (Should a particular configuration become widely used, it may be made more convenient and more economical of resources by adding new modifier modes. This need be done only once, due to the time-multiplexed nature of the modifiers, to provide up to 128 of the new processing elements.)

5. DELAY UNITS

The third computational element in the synthesizer is that for the delay units. This can be time-multiplexed up to 32 ways. Each of the 32 resulting delay units has its own range of addresses in delay memory, which comprises up to 64K (65,536) 20-bit words of storage. The size and location of each delay unit's portion of delay memory are parameters of each delay unit and may be varied at will. It is permissible, and often useful, to have more than one delay unit using the same area of delay memory. Each active delay unit is connected to the other elements of the synthesizer by means of a modifier in Invoke Delay Unit mode. Which delay unit is coupled to a given modifier is indicated by a parameter of the modifier.

Each delay unit has a mode, which may be one of the following:

Inactive: Does not affect delay memory; returns indeterminate results.

Table Look-up: The input from the modifier is shifted and the result is used to address a location in the delay unit's area of delay memory. The amount of shift is a parameter of the delay unit. The word in the addressed location of delay memory is returned unchanged to the modifier. A related mode performs rounding of the shifted number as it is used to address delay memory.

Delay Line: The area of delay memory used by the given delay unit is treated as a delay line with one word per sample period. The input from the modifier is put into the beginning of the delay line, and the output from the end of the delay is returned to the modifier.

The modifier in Invoke Delay Line mode performs the following calculations:

$$q(mt) = A\ q((n-1)t) + q((n-p-1)t),$$

$$out(nt) = in(nt) + B\ q((n-p)t)$$

where p is the length of the delay line in samples. Depending on the values of a and b, these computations accomplish the following:

- straight delay $(A = B = 0)$;
- echo $(A \neq 0,\ B = 0)$;
- comb filter $(A = 0,\ B \neq 0)$;
- all-pass reverberation $(-A = B \neq 0)$.

This is similar to the algorithm of Schroeder (4) but produces the same result with two multiplications rather than three.

6. COMPUTER INTERFACE

Specific hardware in the synthesizer connects with the host computer to perform the following functions:

- Control and sensing of synthesizer status by computer;
- Transfer of sampled data between synthesizer and computer memory;
- Transfer of commands from computer memory to synthesizer and execution of the commands;
- Diagnostic operations.

To minimize the burden on the computer, most data transfers to or from the synthesizer are performed by direct memory access. The computer has only to indicate to the synthesizer, by control functions, the size and location in memory of a data area, for instance, and the synthesizer will read successive data words from this area as it needs them. The synthesizer can be conditioned to interrupt the computer when it has exhausted one data area and needs another. Such memory areas are of three types: per-sample data from the synthesizer, per-sample data to the synthesizer, and commands to the synthesizer.

6.1. Read Data and Write Data

Write data from the synthesizer comes from sum memory through a generator in Write Data mode; if more than one generator is in this mode, then data will be interleaved when written into memory. Per-sample data to the synthesizer is put into sum memory by means of a generator in Read Data mode. More than one generator can be in this mode if the data in memory is properly interleaved. With the Write Data and Read Data modes, the synthesizer can be used to advantage even for a piece that exceeds its capacity. For instance, 500 generators in parallel or 250 second-order filter sections in series can be performed in just two passes. The first pass would perform half the processing, writing its intermediate results into computer memory as it goes. The second pass would read the intermediate data from memory and perform the remaining processing. (Depending on the length of the piece and the amount of available memory, the computer may need to use a disk or other mass storage device as an extension of the memory.) Similarly, the synthesizer can process digitized sound from other sources, natural or synthetic; and material created by the synthesizer can be passed on without degradation to other digital processing or recording equipment.

6.2. Commands

While a great many uses of the synthesizer do not require the Read Data and Write Data features, essentially all make use of the command stream. Each command is 32 bits. Nearly all commands have the meaning "Set parameter X of generator (or modifier or delay unit) N to value V." The parameters to be changed can be modes, running terms, static computational values, or configuration parameters such as sum memory addresses. Also provided is the Linger command, which means "Wait until the end of sample period N, before executing more commands." The commands to be performed at a given instant will appear grouped together in the command stream, preceded by a Linger to denote when they should be performed. The synthesizer contains a 28-command buffer so that the commands of a group can be performed without delays due to computer memory contention or bandwidth limitations.

There are also commands to set parameters applying to the synthesizer as a whole, including the number of processing elements in use, the sample period, the breakpoint frequency of the analog low-pass filters following the DACs, and two 16-bit digital output buffers which can be used to control external apparatus in synchronism with the musical synthesis.

6.3. Diagnostic Functions

More than 10% of the 2,500 integrated circuits in the synthesizer are provided strictly for diagnostic purposes. They allow the host computer to set parameters in the synthesizer, step the synthesizer slowly through its computations, and read back the status of intermediate or final results at any point in the generators, modifiers, delay uits, or computer interface. To work in conjunction with this, a static diagnostic program was written, comprising approximately 70,000 36-bit words of assembly-language code for the PDP-10 computer. Additional dynamic diagnostic programs were also written, employing the Write Data feature, to check for various interactions when the synthesizer is run at full speed.

7. CONCLUSIONS

In the first fifteen months since its use began, this synthesizer has been employed productively by approximately 100 musicians and researchers, with a wide variety of musical styles and research objectives.

This experience having verified the architecture of the synthesizer, a second unit is under construction. It will be fully compatible with the first, with one significant enhancement: analog-to-digital conversion capability will be built in.

In the past, work in digital synthesis has centered on the development and understanding of each basic synthesis technique: additive, subtractive, modulation, waveshaping, reverberation, and so on; and the time factor worked against the creation of large or complex pieces. Now the means are at hand for musicians who can use all of these techniques, singly and in combination, to develop compositions of increased scope and richness.

8. REFERENCES

(1) G. Winham and K. Steiglitz, "Input Generators for Digital Sound Synthesis," J. Acoust. Soc. Am., vol. 47, p. 665 (1970).

(2) J. M. Chowning, "The Synthesis of Complex Audio Spectra by Means of Frequency Modulation," J. Audio Eng. Soc., vol. 21, p. 526 (1973).

(3) D. E. Knuth, The Art of Computer Programming, Vol. 2 (Addison-Wesley, Reading, Massachusetts, 1971), pp. 155-156.

(4) M. R. Schroeder, "Natural Sounding Artificial Reverberation," J. Audio Eng. Soc., vol. 10, p. 219 (1962).

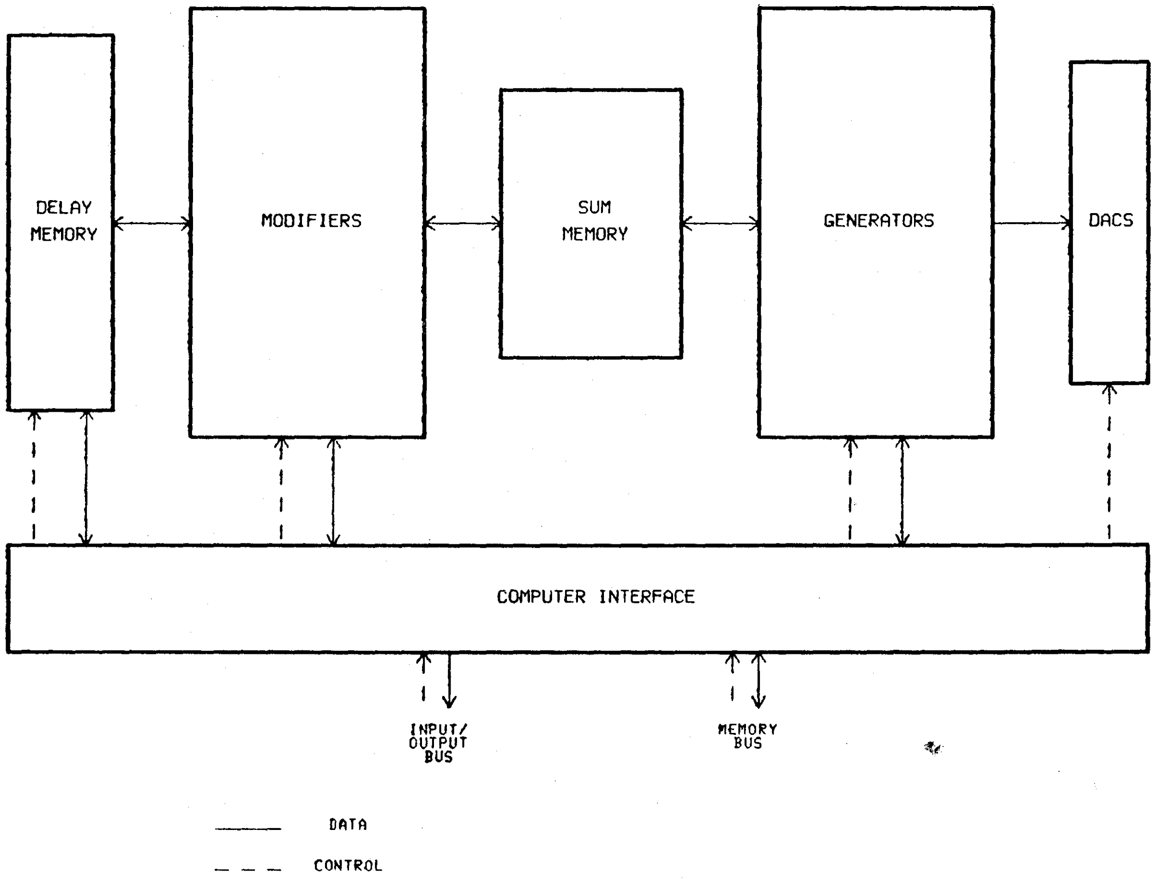

Fig. 1. Synthesizer Block Diagram

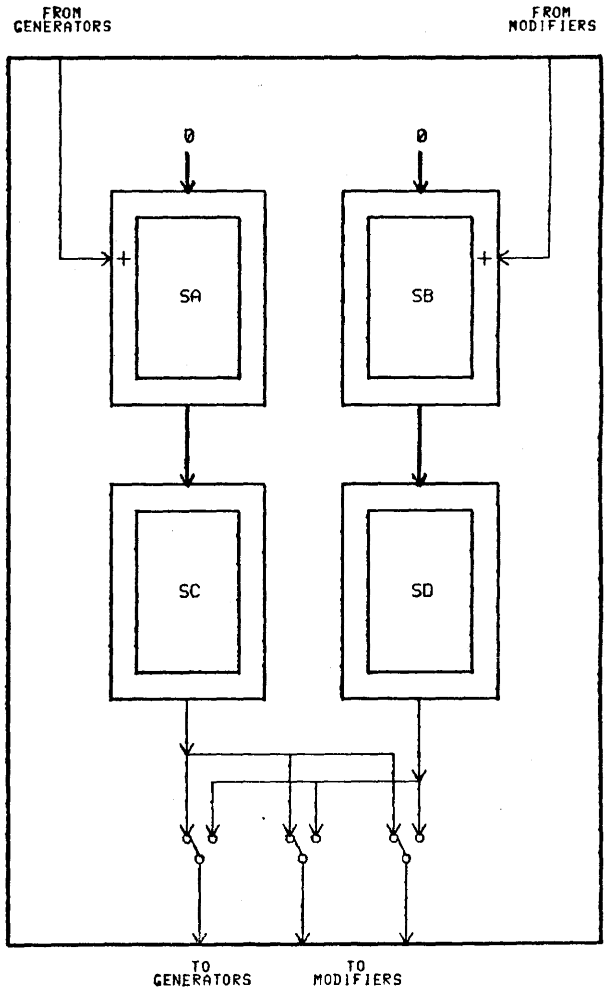

Fig. 2. Sum Memory

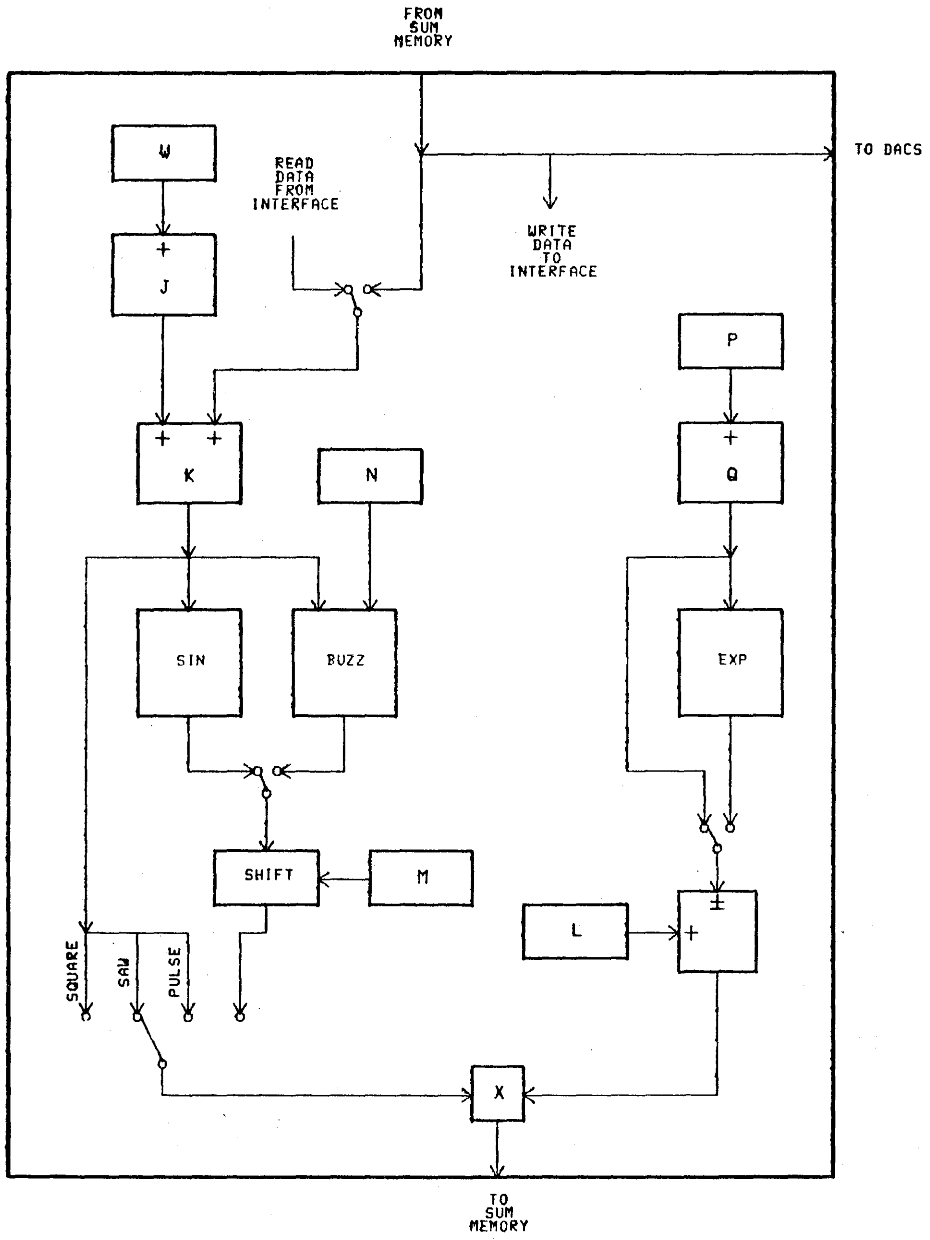

Fig. 3. Generator

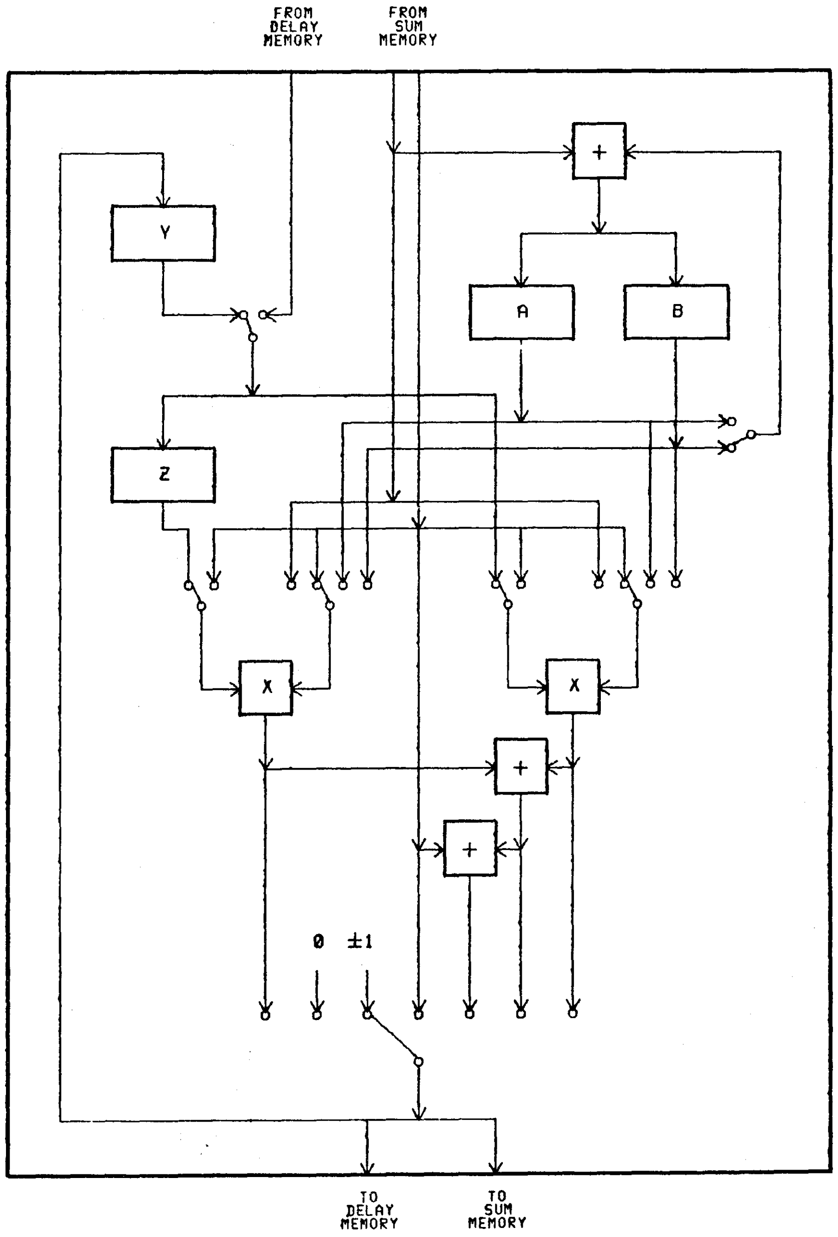

Fig. 4. Modifier

SYSTEMS CONCEPTS 524 SECOND STREET SAN FRANCISCO, CALIFORNIA 94107

SYSTEMS CONCEPTS

DIGITAL SYNTHESIZER

PROGRAMMING SPECIFICATION

<u>PRELIMINARY</u>

The information herein is preliminary in nature, and subject to change without notice. It is not to be taken as a commitment by Systems Concepts.

TELEPHONE: 415-433-5400 TWX: 910-372-6062

SYSTEMS CONCEPTS

Digital Synthesizer
Programming Specification Nov. 24, 1974

Generators and Modifiers

The device 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, 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 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.

Information is passed among generators and modifiers through a scratchpad area called sum memory. There are 64 sum memory locations which can be used to accumulate sums of generator outputs, and another 64 for sums of modifier outputs. Any generator or modifier can read any of the 128 sum memory locations.

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

The processing performed on a per-sample basis comprises one pass. A pass is a series of ticks, of two types: processing ticks and update ticks. Processing ticks perform the calculations corresponding to generators and modifiers, and update ticks permit loading of new parameters. Within a pass, all processing ticks are performed first, then all update ticks. A tick of either type takes 195 nsec. The number of processing ticks is nine more than the maximum of: the number of generators used; twice the number of modifiers used. The number of update ticks should be chosen according to the number of processing ticks to give the desired overall sample rate.

GENERIC INTERFACE

The computer interface is discussed here in terms of general 16- and 32-bit input and output operations. Implementation of these functions on a specific computer is covered elsewhere.

Summary

CONO	16 bits: sets overall status, diagnostic readback address
DATAO-A	32 bits: (only when not running) performs command
DATAO-B	32 bits: sets CA or WC for commands, read data, write data
CONI-A	16 bits: reads overall conditions
CONI-B	16 bits: reads cause of interrupt
DATAI	16 bits: (only when not running) diagnostic readback

```
CONO     : CC : R:  EE :  BB .  AAA    :      DDDDDD       :
```

```
CC: 00  no effect
    01  stop
    10  start
    11  cause one tick
R: 0  no effect
   1  reset tick counter to beginning of pass (if stopped)
EE: 00  no effect
    01  disable interrupts
    10  enable interrupts
    11  master reset
    (This controls an overall interrupt-enable bit, independent
    of the bits which enable or disable interrupt due to specific
    causes. It is ANDed with them, providing a global way to
    prevent interrupts.)
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
             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
    10110  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  indicate 16-bit write data
    11111  indicate 32-bit write data
DDDDDD: diagnostic readback address, specifies internal
    data to be read by DATAI.
```

Computer Interface

Information is passed to and from the computer in two ways: I/O instructions, and direct memory access. Both methods deal with data words which may be either 16 or 32 bits. 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 (one datum per sample) (to the device), and write data (one datum 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 and write data may each be either 16 or 32 bits, giving a choice between packed data and full precision (20 bits are significant in 32-bit mode). The device has buffering for 28 commands, 4 read data items, and 1 write data item.

The device 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 (more updates specified to be performed on a pass than update ticks provided). The other class of interrupt conditions relates to the three WCs. Separate indications are provided for each one being exhausted, and also for underrun conditions: a WC being exhausted AND more data needed (commands or read data) or available (write data).

```
CONI-A    :IR:IE:        xxxxxx        : R:  x:WU:RU:CU:WE:RE:CE:
```

IR:	interrupt desired (by 11AAA cause, WU, RU, CU, WE, RE, CE, regardless of CONO EE)
IE:	interrupt enabled (by CONO EE) (IR AND IE is interrupt request)
R:	running (not stopped)
WU:	write data underrun
RU:	read data underrun
CU:	command underrun
WE:	write data WC exhausted
RE:	read data WC exhausted
CE:	command WC exhausted

```
CONI-B    :I1:I2:I3:I4:I5: x:LC:           TTTTTTTTT            :
```

I1:	command overrun
I2:	modifier mixer overflow
I3:	modifier multiplier overflow
I4:	modifier add to sum overflow
I5:	generator add to sum overflow
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)

```
DATAO-B  :      xxxxxxxxx      :    UUU   :       A...A      :
```

UUU:	000	no effect
	001	set write data CA
	010	set read data CA
	011	set command CA
	100	(reserved)
	101	set write data WC
	110	set read data WC
	111	set command WC

A...A (20 bits): core address (if CA)
2's complement of word count (if 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
GMODE	(8 bits)	generator mode GMODE = RRREESS

Run Mode

			osc. run?	env. run?	add to sum?
RRR:	000	inactive	no	no	no
	001	pause	no	no	no
	010	running	yes	yes, sticky	yes
	011	running	yes	yes, free; triggers subseq.	yes
	100	wait	yes	no	no
	101	running	yes	yes, free; stops and triggers subseq.	yes
	110	read data from computer			yes
	111	write data to computer or DAC			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 001 to change to mode 010.
3) A MISC command can specify "clear all wait bits", which will cause any generator in run mode 100 to change to mode 010.
4) If the envelope side of a generator in run mode 101 overflows, that generator goes to run mode 100.
5) A generator in run mode 100 will go to run mode 101 if on the same pass the second preceding generator (the one whose generator number is two less) caused a trigger (was in run mode 011 or 101 and envelope overflowed).

<u>Envelope Mode</u>

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

<u>Oscillator Mode</u>

SS:	000	sum of cosines
	001	square
	010	sawtooth
	011	pulse train
	100	cos (K)
	101	cos (J + fm)

<u>Processing</u>

Calculations performed for a generator, as governed by the run 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 101, Temp0 is taken; otherwise GK. Call the 20-bit result Temp1E, and its high-order 12 bits Temp1.

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

5) If the run mode is 111, Temp1E is sent to the CPU as write data if GN is negative, else to the DAC addressed by GN.

6) Temp1 is multiplied by GN. Call the low-order 12 bits of the product Temp2.

7) If the oscillator mode is 100 or 101, pi/2 is taken; 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); then 2**(-GM) is subtracted. Call the result (12 bits) Temp4.

9) The result of the oscillator side (12 bits, call it Temp5) is then determined according to the oscillator mode.

```
SS: 000 Temp4
    001 -1/2 (on a scale from -1 to +1) if Temp1 is negative,
        else +1/2
    010 Temp1
    011 +1/2 if overflow occurred in step 1) or 4) above;
        else 0.
    100 Temp4
    101 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).

12) If the envelope mode is 10 or 11, 2**(-Temp6) is looked up; otherwise Temp6 is taken. Call the resulting 12 bits Temp7.

13) If the envelope mode is 00 or 10, 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 18 bits of the rounded product are added into the sum memory location designated by GSUM; except in run mode 110, the product is added to read data 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
MSUM	(7 bits) address in sum memory into which modifier result is added
MMODE	(9 bits) modifier mode MMODE = MMMMMAABB

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

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

MMMMM: function

00000:	inactive
00001:	mixing
00010:	latch
00011:	zero-crossing pulser
00100:	amplitude modulation
00101:	four-quadrant multiplication
00110:	minimum
00111:	maximum
01000:	two poles
01001:	two poles, M0 variable
01010:	two poles, M1 variable
01011:	(reserved)
01100:	two zeros
01101:	two zeros, M0 variable
01110:	two zeros, M1 variable
01111	(reserved)
10000	uniform noise (free run)
10001	uniform noise (when input nonzero)
10010, 10011	(reserved)
10100	use delay unit
10101-11111	(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 result that is added into the sum memory location addressed by MSUM. 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

00001: mixing. S := A*M0 + B*M1

00010: latch (sample and hold). S := L1; If B*M1 is not 0, L1 := A

00011: zero-crossing pulser. Temp0 := B*M0; Temp1 := L1*M1;
S := - epsilon if Temp0*Temp1 is negative, else S := 0;
L1 := Temp0

00100: amplitude modulation. S := L1*M1; L1 := A * ((B+1)/2)

00101: four-quadrant multiplication. S := L1*M1; L1 := A*B

00110: minimum. S := min (A*M0, B*M1)

00111: maximum. S := max (A*M0, B*M1)

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

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

01010: two poles, M1 variable. S := L0*M1 + L1*M0 + A;
L0 := L1; L1 := S; M1 := M1 + B

01011: (reserved)

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

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

01110 two zeros, M1 variable. S := L0*M1 + L1*M0 + A;
L0 := L1; L1 := A; M1 := M1 + B

01111 (reserved)

10000 uniform noise. S := L0 + L1*M0 (integer multiply, low-order 20 bits of product used; overflow ignored); L1 := S

10001 triggered uniform noise. S := L0 + L1*M0 (integer multiply, low-order 20 bits of product used; overflow ignored); if B*M1 is not 0, L1 := S

10010, 10011 (reserved)

10100 Invoke delay unit.
Unit # := RM (low-order 5 bits);
S := L0 + L1*M0; L0 := DM; Temp0 := A + DM*M1;
L1 := Temp0; DM := Temp0

10111...11111 (reserved)

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.

X base address (16 bits) and mode (4 bits). The base address is the lowest-numbered location used by this unit. The mode is interpreted as follows:

```
mode: 0000  delay line
      0001  (unused)
      0010  table look-up
      0011  table look-up, argument rounded
      0100...1111 (reserved)
```

Z unit length (16 bits) or binary scale factor (4 bits). In delay line mode, Z gives the total number of locations in the delay line, i.e. the number of samples delay the unit comprises. 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.

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

Processing

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 and then 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 are 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 (set to zero) a related parameter in the same generator or modifier.

```
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 DX; clear DX
             11  M1 left-adjusted, low bits from DX; clear DX

ML   :          (20) data          :1 1 1 0:N:   (7) mod #  :

         N: 0  L0
            1  L1

GQ   :          (20) data          :0 0 1:E:    (8) gen #   :

         E: 0  Q right-adjusted, sign extended
            1  Q left-adjusted, low bits from DX; clear DX

GP   :          (20) data          :0 1 1 0:    (8) gen #   :

GJ   :          (20) data          :0 1 0:E:    (8) gen #   :

         E: 0  J right-adjusted, sign extended
            1  J left-adjusted, low bits from DX; clear DX

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

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

GK   :          (20) data          :1 0 0 1:    (8) gen #   :
        :M:S:C:x:(9) MMODE :(7) MSUM:1 1 1 1 0:   (7) mod #   :
MMODE,
MSUM      M:  if 1, disable loading MMODE
          S:  if 1, disable loading MSUM
          C:  if 1, clear L0


        :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


        :M:F:C:x x:(8) 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 K


GO      :          (20) data          :1 0 1 1:   (8) gen #    :


        :          (20) data          :0 0 0 0 0:x x:R R:W:P:S:
MISC
          RR: 00  no effect
              01  load DX from data
              10  load TTL buffer 0 from data
              11  load TTL buffer 1 from data
          W:  if 1, clear all wait bits
          P:  if 1, clear all pause bits
          S:  if 1, stop


        :          (20) data          :0 0 0 1 0:x x:T T:x x x:
TIMER
          TT: 00  no effect
              10  process no further commands until pass counter equals data
              11  clear pass counter, then wait as for 01
              01  set pass counter from data


        :          (20) data          :0 0 0 1 1:x x:0:Q:x x x:
# TICKS
          Q: 0  set number of processing ticks per pass
             1  set total number of ticks (processing plus update) per pass


        :      (16) data      :(4)data:0 0 0 0 1:U U:(5)unit #:
DLY X, Y, Z
          UU:  00  X  16 bits base address, 4 bits mode; clear Y
               01  Y  16 bits index
               10  Z  16 bits delay unit size, or scale (low 4 bits of 16)
               11  (unused)
```

SYSTEMS CONCEPTS 520 THIRD STREET SAN FRANCISCO, CALIFORNIA 94107

SYSTEMS CONCEPTS

DIGITAL SYNTHESIZER

THEORY OF OPERATION

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

Exh. SC1

TABLE OF CONTENTS

TAKE CARE

1. HAZARDOUS VOLTAGES EXIST WITHIN THE POWER CONTROL ENCLOSURE AND AT THE TERMINALS OF THE POWER SUPPLIES. TAKE APPROPRIATE PRECAUTIONS WHEN WORKING NEARBY.

2. NEVER INSERT OR REMOVE PRINTED-CIRCUIT CARDS OR PDP-10 CABLES WHEN POWER IS ON. FIRST MAKE SURE THAT THE FANS ARE FULLY STOPPED.

3. TO TURN POWER ON OR OFF, USE ONLY THE LOCAL-OFF-REMOTE SWITCH; DO NOT USE THE CIRCUIT BREAKER OR OTHER POWER LINE CONTROL.

4. DO NOT INSERT, REMOVE, OR CHANGE CARDS OR CABLES UNNECESSARILY.

5. WHEN INSERTING OR REMOVING A CARD, APPLY PRESSURE EQUALLY TO BOTH OF ITS EJECTORS. DO NOT USE EXCESSIVE FORCE.

6. FOR PREVENTIVE MAINTENANCE, PERIODICALLY MAKE SURE THAT COOLING AIR IS ENTERING THE CABINET AT THE BOTTOM AND THAT ALL SEVEN FANS ARE RUNNING WHEN POWER IS ON.

Introduction

The Systems Concepts Digital Synthesizer is a large special-purpose digital processor that generates and modifies data streams that represent sounds and features of sounds like those found in music and speech. To accomplish its high computation rate, it operates several computational elements simultaneously and its data paths are extensively pipelined.

The synthesizer is a PDP-10 computer peripheral, with both I/O bus and direct memory access connections for a PDP-10 system. Its behavior and interface characteristics, as seen by the user, are described in the Systems Concepts Digital Synthesizer Programming Specification; an understanding of that document is assumed in the remainder of this manual. Familiarity is also assumed with the Systems Concepts Engineering Drawing Conventions.

An abbreviated notation is used in this manual for devices that are functionally equivalent to those in the 7400 series: for instance, 'H04 means 74H04 or equivalent. The terms .LT., .LE., .NE., .GE., .GT. mean respectively less than, less than or equal to, not equal to, greater than or equal to, greater than.

Drawings

The engineering drawings for the Synthesizer include schematic drawings and a parts-placement drawing for each type of printed-circuit card, and logic drawings for the wire-wrap panels and cable connection rack. For each type of card, the card drawing gives a set of generic names for the signals on any card of the type; and an accompanying table relates each generic name to the specific name of each signal on each card of the type. In the generic names, the term (M) is used for the base bit number in a four-bit slice; for example, the generic name FG(M) would correspond to FG0 on the most significant card of the type, to FG4 on the next card, and so on.

The following drawing numbers are used for the Synthesizer:

1110, 1110M, 1111.0, 1111M, 1112.0, 1112M, 1113.0, 1113M, 1114.0, 1114M, 1115.0, 1115M, 1116.0, 1116M, 1117.0, 1117M, 1118.0, 1118M, 1119.0, 1119M, 1120.0, 1120M, 1121.0, 1121M, 1122.0, 1122M, 1123.0, 1123M; 1160, 1161, 1162, 1163, 1164; 1171, 1172, 1173, 1175, 1176, 1177, 1179, 1180, 1181, 1182, 1184, 1185, 1187, 1188, 1190, 1191, 1193, 1194, 1195, 1197.

In this manual, a reference to drawing number 1191 (for example) is abbreviated to #1191.

Packaging

The Synthesizer is housed in a single free-standing cabinet. It contains the following, from top to bottom: five power supplies; three wire-wrap panels; two printed-circuit-card chassis, with backpanels designated YBACK (upper) and ZBACK (lower); seven cooling fans; a cable connection rack; and the power control enclosure.

The five power supplies and their uses are as follows:

Model No.	Voltage(s)	Rated Current	Use
LGS-EE-5	+5 V	110.0 A	YBACK, ZBACK
LXS-D-5	+5 V	27.5 A	Wire-wrap Panels
LXS-A-5	-5 V	4.0 A	Delay Memory, DEC Bus Interfaces
LXS-A-12	+12 V	2.7 A	Delay Memory
LXD-C-152	+,-15 V	2.5 A each	Analog Outputs

Each supply has an overvoltage protector.

The three wire-wrap panels hold ICs (integrated circuits), and a few discrete components, that comprise the once-only logic of the Synthesizer. The panels are numbered 1, 2, and 3, from the top.

The card chassis hold the printed-circuit cards of several types that are replicated in the system. A typical card has the hardware dealing with successive stages of a four-bit-wide slice of a data path. A data path wider than four bits is processed by a group of cards of the same type; for instance, five cards will be grouped together to process a 20-bit-wide section of the Synthesizer. Viewing the backpanels from their wiring side, card slots are numbered from left to right: Y1-Y39 on YBACK, Z1-Z39 on ZBACK (but some slots are not used). In a group of cards of the same type, the leftmost one processes the most significant bits of the data. The cards in the system are tabulated below.

Card Name	Dwg #	Short Name	Slots
Dual Analog Output	1110	ALOG	Y1-4, Z1-4
Miscellaneous-A	1118	MISCA	Y8-11
Sum Memory	1117	SUM	Y12-16
Miscellaneous-B	1119	MISCB	Y17-19
Modifier-A	1114	FILTA	Y20-24
Modifier-B	1115	FILTB	Y25-29
Delay Memory Data	1120	DMD	Y30-34
Delay Memory	1121	DMEM	Y35-38
Generator-C	1113	GENC	Z6-8
Multiplier	1116	MULT	Z9-13, Z22-26
Generator-B	1112	GENB	Z14-16
Generator-A	1111	GENA	Z17-21
36-bit Interface	1123	TENI	Z27-35
Generic Interface	1122	INTF	Z36-39

On a card, each IC is designated with a U number. Viewing a card in the orientation in which it is plugged in, U1 is at the top next to the edge connector and U2 is beneath it. Counting continues first downward by row, then outward by column. Positions without signal wiring are not counted.

The cable connection rack (#1160 through #1164) provides sockets for the I/O bus, memory bus, memory port multiplexer cable, and the outputs of the TTL output registers. The upper connector row is designated A and the lower B; slots are numbered 13 through 32, from left to right.

In the power control enclosure are the power control board, Local-Off-Remote switch, circuit breaker, and relays that control sequencing of the power supplies.

Controls and Indicators

Since the Synthesizer is designed for checkout by computer, there are very few manual controls and indicators. The only indicator is a red LED, on the power control board, which is lit when AC power is applied to that board. It is not visible unless the louvered cover of the power control enclosure has been removed. However, the sound of the fans is a clear indication that the power supplies are also on. On the power control panel are the main circuit breaker and, to turn the Synthesizer on and off, the Local-Off-Remote switch.

In location 3A33 is a package containing seven on-off switches. Switches 3-7 give the base device address of the Synthesizer on the I/O bus, and correspond respectively to IOS3-7 in the PDP-10 I/O structure. Switch 1 can be used to prevent the Synthesizer from writing into PDP-10 memory.

Signal Names

Data signals generally have names composed as follows:

a) One letter indicating the general area involved:

A	analog
C	clocking
D	diagnostics
F	modifiers ("filters")
G	generators
I	generic interface
P	phases of clock
R	delay units ("reverb")
S	sum memory
T	PDP-10 interface ("ten")
U	unused, provided for possible future additions

b) One or two letters arbitrarily chosen to distinguish busses in a general area;

c) A decimal number for bit position (0=most significant) in a bus, or for a decoded value of a field (0=all bits off).

A control signal is usually named by appending, to the name of the bus it controls, a letter to designate the signal's function. For instance, the clock to the FE register (bits FE0-19) is called FEC. Some of the more common functions are: C -- clock; E -- enable; G -- gate (of a latch); R -- reset; S -- select (ALU mode select or multiplexer input select). A control signal with several functions may be named instead for its derivation.

Names ending in -A, -B, etc., but otherwise alike, denote signals that are logically equivalent but physically distinct, as for loading purposes. Names on the drawings ending in -1, -2, etc. represent signals which are logically equivalent but generated on different printed circuit boards. In this manual, however, a notation such as BUS0-19 means the 20 bits BUS0 through BUS19. A number within a signal name, surrounded by letters, either denotes the time state of a quantity used at different stages of a pipeline or denotes a quadrant of sum memory.

XHI and XGND are forms of HI and GND brought onto printed-circuit cards from the backpanels through signal pins. The versions of HI for the wire-wrap panels and for XHI (named YHI for YBACK and ZHI for ZBACK) are generated by a resistor package shown on #1171.

Clocking

Clock generation for the Synthesizer, shown on #1171, has a 30,769,230-hertz crystal, whose output is divided by 6. A 10-pF capacitor in series with the crystal trims the oscillator to the specified frequency. Outputs of the frequency divider (three Schottky J-K flip-flops) go through delay lines and AND gates to form the basic clock pulses shown in Fig. 1. Each 195-nsec tick (also called a time state) has three equally-spaced clock pulses termed phases A, B, and C; there is also a pulse roughly halfway through the time state (between phases A and B) called phase H. The corresponding signals are CA, CB, CC, and CH. Each pulse is nominally 45 nsec wide. The trailing edge of phase C marks the end of each time state. #1171 also shows the formation of several special-purpose clocks which occur more than once per time state: SAC and CU, which occur on phases A, B, and C; and CHC, occurring on phases H and C.

The basic clock pulses are buffered for distribution throughout the Synthesizer by gating shown on #1172. The signals for the various phases take on the names PHA, PHB, PHC, and PHH. These are conditioned by the clock enable flip-flops, CEAB (for phases A, H, and B) and CEC (for phase C). Ungated clocks designated PHAU, PHBU, and PHCU are also created for the PDP-10 interface and delay memory control, which must run even when other activity in the Synthesizer is stopped.

The clock-enable state depends on the CRUN flip-flop, which is direct-set by the clock-start CONO-A; it is direct-cleared by master reset, the clock-stop CONO-A, or performance of the clock-stop command; and it is clocked off by occurrence of a 10AAA cause enabled by CONO-B. CRUN is ORed with the clock-one-tick CONO-A and the result is ANDed with terms indicating that the clock is not held for read or write data direct memory access (underrun conditions), to produce CRUNA; this is clocked into CEAB, which in turn is clocked into CEC.

Various clock counting functions are shown on #1173. The time state pipeline, on the MISCA cards, appears on sheet 2 of #1118. Fig. 2 shows the relationship of the principal signals involved. CTK0-9 counts the ticks of a pass. Its count is compared against two registers loaded by commands: CTP0-9, denoting total processing ticks, and CTT0-9, denoting total ticks per pass. The EPAS flip-flop is on for the last tick of a pass: it is direct-set by the reset-tick-counter CONO-A; held direct-cleared in the "all ticks update" state; clocked on when CTK0-9 equals CTT0-9; and clocked off at the end of the tick when it is on. Among its effects, EPAS conditions the CTK0-9 counter to parallel-enter zero at the end of the tick.

Generator and modifier calculations each require 9 steps of pipelining; the first is tick A for a given generator or modifier; then follow its ticks 0 through 7. Tick A is preparatory; during it various sum memory addresses are determined. Numeric processing does not start until tick 0. At any point in the generator data paths, data for successive generators is processed on successive ticks. In the modifier data paths, there are two ticks in a row for each modifier. For instance, if CP0B0-7 equals 2, it is tick 2 for for generator 0, tick 1 for generator 1, tick 0 for generator 2, tick A for generator 3; tick 2 for modifier 0, and tick 0 for modifier 1. The time-state pipeline consists of the CPAB0-7 counter and the CPnB0-7 shift registers. The CPnBm busses are used to address RAMs holding data for each generator or modifier, where n denotes the tick during which the RAM is referenced. CPAB0-7 during the processing ticks of a pass counts the same as CTK2-9; CP0B0-7 during processing tick n has the value CPAB0-7 had in tick n-1; CP1B0-7 has that value in tick n+1; and so on. During the interface ticks (also called update ticks) all the CPnBm terms are parallel-loaded with the generator number or modifier number of the next command to be performed (the interface address, IA0-7). CTRA is turned on at the beginning of a pass, when CTK0-9 is reset; it is turned off when CTK0-9 equals CTP0-9, to flag the end of the processing ticks. CTRA is on during a valid tick A for some generator and modifier. Similarly, CTR0 through CTR7 are on during valid processing ticks 0 through 7. CTRA and CTR0-7 are held off in "all ticks update" mode. Flip-flop ITR is on when "real" (i.e. processing) ticks are in progress anywhere in the pipeline; it is off during update ticks. The signal OT, meaning odd tick, is on for tick A of a pass (CPABn = 0), off for tick 0 (CP0Bn = 0), and so on. Similarly, OP means odd pass; it changes state at the same time that CTRA comes on.

Generator Data Paths

Fig. 3 is a block diagram of the generator data paths. Data processing for a generator, tick by tick, proceeds as follows.

Tick 0 -- Oscillator side: On phase A, the 'LS195A registers holding GOA0-7 on PC cards clock in CP0B0-7. This addresses the 256-bit RAMs holding GO0-19 (GENA) and GJI0-27 (MISCB, GENA) (called GJ in the Programming Specification). On phase B, the fm term from sum memory is clocked into 93H72s forming GRA0-19 (SUM). On phase C, GJI0-27 is latched in the 'LS157 multiplexers forming GJ0-27 (MISCB, GENA).

Tick 1 -- Oscillator side: During phase A, GO0-19 with sign extended is added to GJ0-27 by a fast adder, using 'S181 ALUs and 'S182 carry generators, to form GAZ0-27 (MISCB, GENA). The phase A clock writes this sum into GJI. At the same time, the ripple adder GA0-19 (GENA), using '283s, forms the sum of GJ0-19 and GRA0-19, which is clocked by phase A into the 'LS175s GWA0-19 (GENA). The 256-bit RAMs comprising GK0-19 (GENA) and GN0-10 (GENC) are addressed by CP1B0-7, buffered by 'LS04 inverters. On phase B, GK0-15 is latched into the 'LS157 multiplexers GXA0-19 (GENA). On phase C, the high-speed adder GB0-19 (GENA) (similar in structure to GAZ) develops the sum of GWA0-19 and GXA0-19. (Due to the polarities of the signals involved, the adder is actually configured to subtract the ones' complement of GXA, with a borrow, from GWA.) The phase C clock pulse writes GB0-19 into GK0-19, and clocks the 'LS298 register GXB0-12 (GENB, #1179) from either GXA0-12 or GWA0-12, according to the generator mode (the GXB12 selection is done by an 'H51 gate). Also on phase C, GN0-10 is clocked into the '175s GXC0-10 (GENC).

Tick 2 -- Oscillator side: GXB0-12 is combined with GXC0-10 in a modified Wallace tree (MULT, #1179) to form the low-order 13 bits of the expression GXB * (2 * GXC + 1) + GXC. (This has the effect of multiplying GXB by 2 * GXC + 1 if GXB and the product are both assumed to have an implied bit 13 equal to 1.) The 'LS298 register GX0-12 (GENB, #1179) is clocked on phase C from either the product or GXB0-12, depending on the generator mode. GWB0-12 ('LS175s on GENB) is clocked from GXB0-12 on phase C also.

Tick 3 -- Oscillator side: The sine of GX and the cosecant of GWB are looked up in ROM. The assumed low-order 1-bit in each case results in two simplifications: (1) GX0 and GWB0 need not be looked up, but are saved to govern whether the looked-up value will be negated; (2) GX1 and GWB1 need not be looked up, but merely cause ones'-complementing of GX2-12 or GWB2-12 respectively, if set. The ones'-complementing of GX2-12 is done by 'LS86s (GENB). For GWB2-12, 'H87s (GENB) are used in order to substitute all-ones (whose cosecant is approximately 1), except in sum-of-cosines mode. The sines and cosecants are stored in floating-point form in 512x4 PROMs on the MISCA cards. The low two bits of each address are decoded to select one of four banks of PROMs; the remaining nine bits address the PROMs through 'H04 buffers. The PROM outputs are clocked into registers on phase C as follows:

	Sine Exponent, Fraction	Cosecant Exponent, Fraction
PROM Output	GGE0-3, GGF0-11	GFE0-3, GFF0-11
Register	GYC0-3, GYD0-11	GYA0-3, GYB0-11

Exponent registers are on GENC, and fraction registers on GENB. Also on phase C, GWB0-11 is clocked into GWC0-11; and the RAM GM0-3 (MISCB, addressed by CP3B0-7 through 'LS04 inverters) is clocked into GYE0-3 (GENC).

Tick 3 -- Envelope side: Phase A clocks CP3B0-7 into the 'LS195As serving as address registers for the RAMs GP0-19 (GENA) and GQ0-23 (MISCB, GENA). During phase C, GQ0-23 is latched into the 'LS157s comprising GVA0-23 (MISCB, GENA); the end of phase C clocks GVA0-13 into GVB0-13 (#1179, MISCB).

Tick 4 -- Oscillator side: GYB0-11 is multiplied by GYD0-11 and the high-order 12 bits of the product are clocked on phase C into the 93H72s comprising GYP0-3 and GY4-11 (GENB) (GYQ3, clocked at the same time, is a late output from the Wallace tree which is added to GYP0-3 during tick 5 to correct the product). The exponents GYA0-3 and GYC0-3 are added together with scale factor GYE0-3 and the results clocked on phase C into the 'LS175 termed GYG0-3 (#1179). GWC0-11 is clocked on phase C into GWD0-11 (GENB).

Tick 4 -- Envelope side: GVB0-13 is treated as a negative exponent of 2, with a binary point between GVB3 and GBV4. The field GVB4-13 addresses the PROM GH0-11 (MISCA), the bits GVB4-12 addressing the PROMs directly and bit GVB13 in true and ones'-complement forms enabling one or the other bank of PROMs. Then GH0-11 is run through 'LS153 multiplexers (GENC), configured to shift right 0, 1, 2, or 3 places according to the value of GVB2-3. (If GVB0-1 = 11, the 'LS153s are disabled, producing zeros.) The result (GIM0-11) goes into 9309 multiplexers (GENC)that can shift right 0, 4, or 8 places (according to GVB0-1) or substitute GVB0-11 if in linear mode. This result, GIN0-11, is available in both polarities; on phase C, one or the other polarity is clocked into the GT register ('LS298s on GENC) according to the envelope mode.

Tick 5 -- Oscillator side: GY0-11, the fraction part of the floating point product, is shifted right 0 to 15 places by two banks of 'LS153s (GENB) according to the value of GYG0-3, the exponent part; the result is GIB0-11 (not named in the schematic drawing). A term GWE0-11 is derived by 'LS86s and '283s (GENB) as follows: GWE0-10 is GWD1-11, two's-complemented if GWD0 is 1; GWE11 is 0. Then GU0-11, the result of the envelope side (except for the sign, which is handled by control logic), is selected by 'LS153 multiplexers (GENB). It is either GIB0-11 (sine or sum-of-cosines mode), GWE0-11 (sawtooth mode), an overflow bit from the GA and GB adders (pulse-train mode), or 4000 octal (square-wave mode). On phase C this result is clocked into GZA0-11 (93H72s on GENB).

Tick 5 -- Envelope side: The RAM GL0-11 (GENC), addressed by CP5B0-7 through 'LS04 inverters, is added to GT0-11. A carry is injected in the low-order position if the ones'-complemented version of GIN was taken in tick 4, thereby accomplishing a two's-complement negate. The sum GE0-11 (GENC) is formed by a ripple-carry adder of '283s; it is clocked on phase C into GZB0-11 (93H72s on GENC).

Tick 6 -- The unsigned quantities GZA0-11 and GZB0-11 are multiplied (MULT, #1179) and the high-order 18 bits of the product are clocked on phase C into GZ0-17 (93H72s on #1179); again, one late bit, GZC3, is clocked at the same time to be added in later.

Tick 7 -- GZC3 is added to GZ0-3 by a '283 to form the correct high-order product bits GZS0-3. During clock phase CCB (see Fig. 1) the contents of the sum memory location to be augmented are latched into the 'LS157s GRB0-19 (SUM). On the FILTA cards, GZS0-3 and GZ4-11 are ones'-complemented by 'LS86s if the result should be negative, then added (sign extended) by '283s to GRB0-19, with a carry in if necessary to complete a two's complement. The sum, GF0-19, is returned to sum memory where it is written on phase C.

Modifications in certain generator run-modes: If the oscillator side is not running, write pulses are not given to the GJI and GK memories. If the envelope side is not running, write pulses are not given to the GQ memory. If a generator is not to add to sum memory, the sum memory write pulse is not given. If a generator is reading data from PDP-10 memory, the data read appear in GRB instead of the contents of the sum memory word being augmented. If a generator is feeding a DAC, GRA0-19 is clocked into AP0-19 ('175s on SUM) at the end of phase B, tick 1; and AP0-13 is clocked into the proper DAC hold register at the end of phase A, tick 2. If a generator is writing data into PDP-10 memory, GRA0-19 is clocked into IWB0-19 (SUM) on phase B of tick 1.

Command execution: During interface ticks, all CPnB0-7 hold the generator number from the command, clocked in on the previous phase C. Address registers (such as for GO and GQ) are clocked from CPnB0-7 on phase A. Data to be written comes direct from the generic interface into memories GO, GN, GM, GP, and GL. To write GJI or GQ, data from the interface is introduced by 'LS257 3-state multiplexers instead of the GO or GP RAMs (which are disabled), and the GAZ or GC adder is put in the mode where it passes the "A" input through to the output. To write GK: GWA is held reset during interface ticks; during phase B, the GK memory is disabled. 'LS257s are enabled to place interface data on the GK lines; GK is latched into GXA at the end of phase B. All commands cause memories to be written on phase C.

Generator Control

The generator run mode, GRMD0-3, is stored in RAMs on a MISCB card, addressed by GRMDA0-7 on #1175. GRMDAn (93H72s) is clocked from CPABn on phase H and from CP4Bn on phase C. The mode for a generator is read on the second half of the generator's tick A, and clocked on phase C into GRMD0B0-3 ('S175 on #1175). The mode bits go through a pipeline of 'LS174s through GRMD5Bn; on phase H of tick 5 these bits are written back into GRMDn. Along the way the mode may have been altered by IRP (clear all pause bits), IRW (clear all wait bits), GCOD (envelop overflow), or GT (trigger from previous generator). The GRMDn write pulse, GRMDW, also occurs by command (IIM50).

The other mode bits, corresponding to bits I7-12 of the command data, are shown on #1177. GUS0-1 select the waveform; GXS selects the output of GX; GXBS selects the signal input to GX; GTS selects whether the envelope is added or subtracted from the asymptote; and GINVBE chooses between linear and exponential envelope modes. Each RAM is addressed by the time state in which it is used.

Straightforward gating (#1176, #1177), based on the mode bits and the time state, creates the enables, write pulses, clocks, and selects required for the processing described in "Generator Data Paths" above.

Modifier Data Paths

Fig. 4 is a block diagram of data paths for the modifiers. The constituents, and the boards they appear on, are as follows:

FL0-19:	256-word RAM holding the L0 and L1 terms, written from the FLI bus (FILTA)
FE0-19:	93H72s, clocked from FL0-19 (FILTA)
FEE0-19:	93H72s, clocked from FL0-19 (FILTA)
FN0-19:	'LS670s, written from FE0-19; addressing accomplishes a 3-stage delay advanced every other tick (FILTB)
FG0-19:	'S175s, clocked from sum memory (FILTA)
FV0-19:	93H72s, clocked from sum memory (FILTA)
FW0-19:	'LS157s, latched from sum memory (FILTA)
FUI0-29:	256-word RAM holding the M0 and M1 terms, written from FA adder; 3-stated with 'LS258 multiplexérs from generic interface (MISCB, FILTA)
FU0-29:	'LS157s, latched from FUI0-29 (MISCB, FILTA)
FA0-29:	fast adder ('S181s and 'S182s); adds FU0-29 and FG0-19, sign extended (MISCB, FILTA)
FQI0-19:	'LS175s clocked from FUI0-19 (FILTA)
FP0-19:	'S257s 3-stated with 'S258s (not named on schematic); inputs are: FEE0-19; R0-19 (from delay memory); FG0-19; sum memory (FILTA)
FQ0-19	9309 multiplexers (not named on schematic); inputs are: FQI0-19; FV0-19; 0, -FV0, FV1-18; 0 (FILTA)
FXA0-19:	93H72s clocked from FP0-19 (FILTA)
FXB0-19:	93H72s clocked from FQ0-19 (FILTA)

FXP, FXH, FXJ:	intermediate stages of FX multiplier (see below) (FILTB)
FX0-19:	93H72s, final product of FXA and FXB (FILTB)
FK0-19:	'LS175s clocked from FX0-19 (FILTB)
FT0-19:	'LS157s, can provide: 0, FK0-19, or FN0-19 (FILTB)
FB0-19:	ripple-carry adder of '283s; adds FT0-19 and FX0-19 (FILTB)
FJ0-19:	'LS175s clocked from FB0-19 (FILTB)

comparators indicating FJ .GT. FK, FJ = FK, FJ .LT. FK (FILTB, #1182)

FR0-19:	'LS258s 3-stated together; inputs are: FK0-19; FJ0-19; FF0-19; FD0-19 (FILTB)
FC0-19:	fast adder ('S181s, 'S182) adding FR0-19 and FW0-19 (FILTB)
FY0-19:	'LS157s; inputs are: FJ0-19; FC0-19 (FILTB)
FD0-19:	'LS175s clocked from FY0-19 (FILTB)
FZ0-19:	9309s; inputs are: FJ0-19; FC0-19; FK0-19; FW0-19 (FILTB)
FF0-19:	'LS175s clocked from FZ0-19 (FILTB)
FLI0-19:	'LS257s 3-stated together; inputs are: FD0-19, FF0-19, I0-19 (FILTB); FE0-19, R0-19 (from delay memory) (FILTA)

The processing steps are listed in order below for each of the modifier modes. The notation "1A", for instance, means phase A of a modifier's tick 1. The FX multiplier is discussed in a separate section below.

Two Poles, Two Zeros (possibly M0 or M1 variable)

```
0H:  L1 to FE, FEE
0B:  M1 to FU if M1 variable
0C:  FEE to FXA; L0 to FEE; M1 to FXB; sum(MRM) to FG
1H:  FE to L0
1B:  M0 to FU if M0 variable
1C:  FEE to FXA; M0 to FXB; FA to M0 if M0 variable, or
       FA to M1 if M1 variable
4C:  FX (L1 * M1) to FK
5C:  FX (L0 * M0) + FK to FJ
6A:  sum(MIN) to FW
6C:  FJ + FW to FD; FW to FF
7B:  sum(MSUM) to FW
7C:  FD to L1 if two poles, or FF to L1 if two zeros;
       FW + FD to sum(MSUM)
```

Mixing, Integer Mixing

```
0B:  if MIN is in modifier-this-pass quadrant, sum(MIN)
       to FG
0C:  sum(MRM) to FXA; M1 to FXB
1A:  if MIN is not in modifier-this-pass quadrant, sum(MIN)
       to FG
1C:  FG to FXA; M0 to FXB
4C:  FX (B * M1) to FK
5C:  FX (A * M0) + FK to FJ
6C:  FJ to FD
7B:  sum(MSUM) to FW
7C:  FW + FD to sum(MSUM)
```

Amplitude Modulation, Four-Quadrant Multiplication

```
          0H:  L1 to FEE
          0B:  if MIN is in modifier-this-pass quadrant, sum(MIN)
                 to FG
          0C:  FEE to FXA; M1 to FXB; sum(MRM) to FV
          1A:  if MIN is not in modifier-this-pass quadrant, sum(MIN)
                 to FG
          1C:  FG to FXA; if four-quadrant multiplication, FV to
                 FXB; if amplitude modulation, 0 to FXB0, -FV0 to
                 FXB1, FV1-18 to FXB2-19
          4C:  FX (L1 * M1) to FK
          5C:  FX (B * A) to FJ
          6C:  FK to FF; FJ to FD
          7B:  sum(MSUM) to FW
          7C:  FD to L1; FW + FF to sum(MSUM)
```

Minimum, Maximum

```
          0B:  if MIN is in modifier-this-pass quadrant, sum(MIN) to
                 FG
          0C:  sum(MRM) to FXA; M1 to FXB
          1A:  if MIN  is not in modifier-this-pass quadrant, sum(MIN)
                 to FG
          1C:  FG to FXA; M0 to FXB
          4C:  FX (B * M1) to FK
          5C:  FX (A * M0) to FJ
          6C:  FK to FF; FJ to FD
          7B:  sum(MSUM) to FW
          7C:  FD or FF (depending on mode and comparison FJ:FK) +
                 FW to sum(MSUM)
```

Zero-Crossing Pulser

```
          0H:  L1 to FEE
          0C:  FEE to FXA; M1 to FXB; sum(MRM) to FG
          1C:  FG to FXA;M0 to FXB
          4C:  FX (L1 * M1) to FK
          5C:  FX (B * M0) to FJ
          6C:  all-ones to FD; FJ to FF
          7B:  sum(MSUM) to FW
          7C:  FF to L1; FW + FD + (0 if FK is not 0 and if  either
                 FJ is 0 or FK * FJ is negative; else 1) to sum(MSUM)
```

Invoke Delay Unit

```
          0H:  L0 to FE
          0C:  R (delay memory) to L0, FXA; FE to FN; M1 to FXB
          1H:  L1 to FEE
          1C:  FEE to FXA; M0 to FXB
          4C:  FX (R * M1) to FK
          5C:  FX (L1 * M0) + FN (L0) to FJ
          6A:  sum(MIN) to FW
          6C:  FW + FK to FF, delay memory; FJ to FD
          7B:  sum(MSUM) to FW
          7C:  FF to L1; FW + FD to sum(MSUM)
```

Latch

```
          0H:  L1 to FE
          0C:  FE to FN; sum(MRM) to FXA; M1 to FXB
          1C:  0 to FXB
          4C:  FX (B * M1) to FK
          5C:  FX (0) + FN (L1) to FJ
          6A:  sum(MIN) to FW
          6C:  FW to FF; FJ to FD
          7B:  sum(MSUM) to FW
          7C:  FF to L1 if FK is not 0; FW + FD to sum(MSUM)
```

Uniform Noise, Triggered Uniform Noise

```
        0H:  L0 to FE
        0C:  FE to FN; sum(MRM) to FXA; M1 to FXB
        1H:  L1 to FEE
        1C:  FEE to FXA; M0 to FXB
        4C:  FX (B * M1) to FK
        5C:  FX (L1 * M0) + FN (L0) to FJ
        6C:  FJ to FD
        7B:  sum(MSUM) to FW
        7C:  FD to L1 if FK is not 0, or not in triggered mode;
               FW + FD to sum(MSUM)
```

One Pole

```
        0H:  L1 to FEE
        0C:  FEE to FXA; L0 to FEE; M1 to FXB; sum(MRM) to FV
        1C:  FEE to FXA; FV to FXB
        4C:  FX (L1 * M1) to FK
        5C:  FX (L0 * B) + FK to FJ
        6C:  FJ to FD
        7B:  sum(MSUM) to FW
        7C:  FW + FD to sum(MSUM)
```

One Zero

```
        0H:  L1 to FE, FEE
        0C:  FEE to FXA; L0 to FEE; M1 to FXB
        1H:  FE to L0
        1C:  FEE to FXA; M0 to FXB
        4C:  FX (L1 * M1) to FK
        5C:  FX (L0 * M0) + FK to FJ
        6A:  sum(MIN) to FW
        6C:  FJ to FD; FW to FF
        7B:  sum(MSUM) to FW
        7C:  FF to L1; FW + FD to sum(MSUM)
```

Signum

```
0B:  if MIN is in modifier-this-pass quadrant, sum(MIN)
       to FG
0C:  sum(MRM) to FXA; M1 to FXB
1A:  if MIN is not in modifier-this-pass quadrant, sum(MIN)
       to FG
1C:  FG to FXA; M0 to FXB
4C:  FX (B * M1) to FK
5C:  FX (A * M0) to FJ
6C:  0 to FF; all-ones to FD if FJ .LT. FK
7B:  sum(MSUM) to FW
7C:  if FJ .GE. FK, FF to FR; if FJ .LT. FK, FD to FR;
       FW + FR + (1 if FJ .GT. FK, else 0) to sum(MSUM)
```

Threshold

```
0H:  L0 to FE
0B:  if MIN is in modifier-this-pass quadrant, sum(MIN) to
       FG
0C:  FE to FN; sum(MRM) to FXA; M1 to FXB
1A:  if MIN is not in modifier-this-pass quadrant, sum(MIN)
       to FG
1C:  FG to FXA; M0 to FXB
4C:  FX (B * M1) to FK
5C:  FX (A * M0) + FN (L0) to FJ
6C:  if FJ .GE. 0, FK to FF; if FJ .LT. 0, 0 to FF
7B:  sum(MSUM) to FW
7C:  FW + FF to sum(MSUM)
```

Inactive

```
6C:  0 to FF
7B:  sum(MSUM) to FW
7C:  FW + FF to sum(MSUM)
```

Modifier Control

The modifier mode FMD0-4 is stored in 256x1 RAMs (MISCB, #1180), all addressed by CPABn. These mode bits are clocked through a pipeline FMDnBm (a '175 and a '174 on #1180). These are decoded by PROMs (on #1180 and #1181), addressed by FMD0Bn for ticks 0 and 1 and by FMD6Bn for ticks 6 and 7. PROM outputs and their meanings are as follows:

FXM0P:	multiply in fraction mode
F1M1V:	2 poles or 2 zeros, M1 variable
F1M0V:	2 poles or 2 zeros, M0 variable
FGCA:	clock FG on phase B of even tick or phase A of odd tick
FGCB:	clock FG on phase C of even tick
FLA7P:	off if addressing L0 in first half of tick 1
FLWCA:	write FL in second half of tick 0
FLWHA:	write FL in first half of tick 1
FEECC:	clock FEE on phase C of tick 0
FEECH:	clock FEE on phase H of tick 1
FPRSA:	select delay memory to FXA on tick 0
FPGSA:	select FG to FXA on tick 1
FPSGE0:	select FG or sum memory to FXA on tick 0
FPSGE1:	select FG or sum memory to FXA on tick 1
FQS0A:	select FV or FV shifted (A.M.) to FXB on tick 1
FQS1A:	select 0 or FV shifted to FXB on tick 1
FRKS:	select FK to FR on tick 6
FYJS:	select FJ to FD on tick 6
FYEA:	clear FF on tick 6
FRFSA:	select FF to FR on tick 7 if FK .GT. FJ
FRFSB:	select FF to FR on tick 7 if FK .LE. FJ
FRFSC:	select FF to FR on tick 7
FZS0:	select FC or FW to FF on tick 6
FZS1:	select FW or FK to FF on tick 6
FMDCP:	signum mode
FFRA:	clear FF on tick 6 if FJ is negative
FZCP:	zero-crossing pulser mode
FLWCB:	write FL on phase C of tick 7
FLWCC:	write FL on phase C of tick 7 if FK .NE. 0

The FX multiplier scaling bits FXM1P and FXM2P come from RAMs on #1181. Two pairs of RAMs are 3-stated together, being enabled on alternate passes for the two successive multiplies of one modifier. Both enables are asserted when writing data into the RAMs. The bits are pipelined on #1182 to correspond to the data pipeline on the FILTB cards as previously described.

The PROM outputs, and in some cases mode bits taken directly, are combined with clock pulses as needed (see #1180, #1181) to implement the processing described in "Modifier Data Paths".

Multipliers

There are four multipliers in the Synthesizer: GX, GY, GZ, and FX. Each is implemented in the form of a Wallace tree of four-bit slices. Because of the time required by a large tree, in the larger multipliers a pipeline is employed: partial products are formed, on one tick and added together on the next tick.

Partial products are formed with 8875A and 8875B ICs and added by '283 adders. In a few cases carries are added together by 'H183s. Various portions of the four trees are allocated among the ten MULT cards, with a few remaining portions on the wire-wrap panels.

GX: this multiplier yields the low-order 13 bits of the product. The low-order 12 bits are generated by the Wallace tree, and the high-order bit by XORing the proper bits of the operands and carries out of the tree. As noted above, GX is modified to perform a function slightly different from simple multiplication.

GY, GZ: these are straightforward unsigned multipliers. The high-order part of the product is taken, but the low-order part of the tree is present to compute the proper carries into the high bits.

FX: this is the largest multiplier. It multiplies two 20-bit two's-complement numbers for a 39-bit two's-complement product. The signed result is generated in the Wallace-tree structure with the aid of three special types of PROM. One type takes in two four-bit numbers and produces the high-order four bits of their product, assuming that one of the operands is signed, in two's-complement form; the second type is similar but assumes both operands are signed. These are used in place of 8875As when the high-order four bits of either multiplier operand are involved. Corresponding PROMs for the low product bits are normal 8875Bs since the low bits are the same whether the multiply is signed or unsigned. The third special PROM type is used for sign extension, being added into a four-bit slice of the product, with its inputs coming from the high-order output bits of all signed-multiply PROMs in less significant slices. The 20-bit result FX0-19 can be selected from eight different positions in the 39-bit product. This is determined by mode bits FXM0-2, which control three successive stages of 'LS298s: FXG and FXH; FXJ; FX. The FXG and FXH partial products are added by '283s named FXI. Successive ticks in FX perform the following: 0 -- operands are clocked into FXA and FXB; 1 -- partial products are clocked into FXG and FXH, selecting between integer and fraction multiplication; 2 -- FXI is clocked into FXJ, selecting zero or two units of shift; 3 -- FXJ is clocked into FX, selecting zero or one unit of shift.

Sum Memory

Sum memory is composed of 80 16x4 RAMs, organized in four quadrants named S0, S1, S2, and S3. Each quadrant is 64 words by 20 bits. Generator outputs are added in S0 and S1; modifier outputs in S2 and S3. On one pass, S0 will be "this pass" and S1 "last pass"; on the next pass, the functions will be exchanged. S2 and S3 alternate similarly. During a single tick, a quadrant may have as many as three separate read accesses or one read-pause-write access. These are interleaved as shown in Fig. 5. Two classes of modifier modes are distinguished: "mod-mix" modes which use the "A" operand early in their processing, and "pole-0" modes which use it later.

Sum memory activity is based on the clock phases PHA, PHB, and PHC. While a reference is occurring in each quadrant during a 65-nsec clock phase, the six-bit address in each quadrant is being generated for the next phase. On each phase, the addresses are clocked into 'S175s and 'S174s (center of #1184) producing S0A0-5, S1A1-5, S2A1-5, and S3A1-5. The inputs, SnAmI, are created on the four MISCA cards (left of sheet 2, #1184) by 'S153 multiplexers. The multiplexer selects, SnAS0-1, are based only on clock phases, OP, and OT. They are generated by gating on #1184 and by 'S51s on two SUM cards. Data inputs to the sum address multiplexers are as follows: FRM2-7, RAMs (on a MISCA) addressed by CP0Bn, holding modifier "B" addresses; GFM1-6, 93H72s on #1184, clocked every phase C from GFMI1-6, RAMs (on another MISCA) addressed by CPABn; FIN2-7, '175s on #1185 clocked every phase C from RAMs addressed on alternate ticks by CPABn and CP4Bn; and SUMA0-5, which is GSUM0-5 for quadrants 0 and 1 and FSUM1-6 for quadrants 2 and 3. GSUMn and FSUMn come from 'S161 counters on #1185; during processing ticks these counters are parallel-loaded on each clock phase from GSUMIn and FSUMIn, RAMs on MISCA cards addressed by CP6Bn through 'H04 inverters. Of the six address bits for a quadrant, the low-order four directly address the sum memory RAMs in the quadrant and the high-order two bits are decoded by 'S51s on various SUM cards to form the enables SnEm, where n is the quadrant and m denotes one of four banks of RAMs which are 3-stated together.

The high-order bits FRM0-1, GFM0, and FIN0-1 come from the RAMs as do the low-order bits, but instead of addressing sum memory they are used to control multiplexers which route sum memory outputs to the generators and modifiers. There are three sets of multiplexers, all on the SUM cards: SA0-19 (not labelled on the drawing), the generator fm input, formed by pairs of 'S257s; SB0-19 (not labelled), the generator sum term, formed by an 'S257 from quadrants 0 and 1 and an 'S258 from IRB0-29 in the generic interface for DMA read data; and SC0-19, the modifier input, two 'S258s from sum memory.

The signals that control these multiplexers are generated on #1184. SA23E, true when fm is coming from the modifier side of sum memory, is simply a buffered version of GFM0; SA1S to select quadrant 1 (as opposed to 0) and SA3S to select quadrant 3 (as opposed to 2) are just copies of OP. SBSE enables SB0-19 from sum memory, and SBIE enables it from the interface; these are opposite sides of a flip-flop ('S175 in 3E8) clocked from CHRP which indicates that the upcoming generator is in read-data mode. SB1S, selecting quadrant 1 rather than 0, is -OP buffered. The multiplexer controls for SC0-19 involve OP and two signals, CSS0-1, coming from #1172, which sequence through the various phases of even and odd ticks. The CSSn address an 'S153 multiplexer to select SCSP0-1 from the high-order bits of FIN, FRM, or FSUM. The SCSPn are combined in gating and clocked into flip-flops on every clock phase to form SC01E (enable from generator side), SC23E (enable from modifier side), and SC3S (quadrant 3 as opposed to 2). SC1S, selecting quadrant 1 as opposed to 0, is OP buffered.

Resetting sum memory is governed by flip-flop SR on #1184. Its D input is SRI (#1173), arranged so that SR will set after the first interface tick of a pass and will clear by processing tick 6. SR is ANDed with OPD (OP delayed -- see Fig.2) and its complement to give SR0 (reset even quadrants) and SR1 (reset odd quadrants). These are ANDed with CU (clock on phases A, B, and C) by 'S51s on SUM cards, to assert chip enables and write pulses for all RAMs in the appropriate quadrants. While SR is asserted, the GSUMn and FSUMn 'S161 counters on #1185 are conditioned to count on every clock phase, disabling the parallel entry. When they have counted through 16 states (less than 6 ticks), sum memory has been reset.

Normal writing into sum memory is controlled by the write grant signals SnWG (#1184). For the modifier quadrants these are the AND of OT, CTR7, and OP or -OP. The generator write grants do not involve OT but include a generator mode bit which governs adding to sum memory.

Delay Memory

The DMEM cards (#1121) use 4096-bit dynamic MOS RAMs. Each card has a 21x4 array of RAMs comprising 16K 20-bit words plus parity. Addresses, write enable, column strobe, and chip enable are buffered by '128s and series-terminated with 33-ohm resistors. There are six address lines, time-multiplexed to give row address and column address in sequence. Input data, RT0-19 and RTP (parity), are buffered in 'LS174s clocked on the high-going transition of RMCOL. Output data goes through 'LS365 buffers, enabled by RMDSn, which gate data from the proper DMEM card onto the 3-state bus RR0-20. Parity is generated (RTP) and checked (RR20) with 9348s on INTF cards (#1122). The row strobes RS0-3 determine which row of chips on the DMEM card actually perform a given cycle. RSn on board m is the AND of the board select RMBSm and the row select RMRSn.

Control signals for the DMEM cards are generated on #1188. There are three types of cycles: normal (i.e. delay unit), refresh, and PDP-10 access, associated with the control signals RMNY, RMRY, and RMTY respectively. Each such signal is true during the four ticks of a cycle of the proper type. An 'LS195A shift register, clocked on phase C, counts the four ticks of a cycle: RMC0 is true in the first tick; RMC0 and RMC1 in the second; RMC0-2 for the third; RMC0-3 all true for the fourth. Another 'LS195A, clocked on the leading edge of phase C, provides the timing signals which after gating are used on the DMEM cards: RMRAS (row strobe), RMCAS (column strobe), and RMWPP (write pulse). An 'H74 clocks in RMRAS on phase A to create the RMCOL signal. Priority arbitration for the next cycle is done by gating at the input of the 'LS175 which generates RMTY, RMNY, and RMRY. A PDP-10 access has the highest priority and its request line goes right to RMTY. If there is no PDP-10 request, a request for a normal cycle (RQP) on a processing tick (CTRA) turns on RMNY. Failing both those conditions, a refresh request (RMRQ) sets RMRY.

The cycle-type flags are only clocked on ticks in which RMI is true, indicating a cycle is about to end or none is in progress. This signal also resets the 'LS195As. The normal cycle request RQP is an 'LS109 conditioned by clock enables to permit only one normal cycle per gated clock tick. (Most of the DMEM control logic runs on ungated clocks, since refresh and PDP-10 cycles must be permitted and normal cycles completed once begun, even if the clock is stopped.) RQP is held off by RQO, which is overflow from the counter RQ0-4 ('LS161s). The RQn counter is reset at the beginning of each pass and counts the 32 delay units. Its trickle enable, RQET, is asserted during the third tick of a normal cycle. The refresh request flip-flop RMRQ ('LS109) is set by RMRT, trickle carry out of a 7-bit counter ('LS161s), which counts out and requests a refresh cycle approximately every 24 microseconds. The RMRQ flip-flop provides one level of buffering for RMRT so that a second refresh request can be timed out while one is pending. A refresh address counter RMR0-5 ('LS161s) is advanced at the end of each refresh cycle. It runs through all 64 states to ensure that all row and column addresses in the dynamic RAMs are refreshed in turn. The 16-bit delay memory address RV0-15 is treated as follows: RV0-3 are latched in an 'LS157 as RMA0-3. Then RMA0-1 are decoded by an 'LS139 to form the board selects RMDSn and RMBSn (through an 'LS158 to make them row strobe pulses, all on for refresh cycles). RMA2-3 are decoded by another 'LS139 section to form the row selects RMRSn. A set of 'LS153 multiplexers form the six address bits RMA4-9 sent to the RAMS directly; these select either RV4-9 for the row address, RV10-15 for the column address, or RMR0-5 for both in refresh cycles.

Delay Memory Data

The DMD cards (#1120) contain the delay unit data paths other than the delay memory itself. The general organization is shown in Fig. 6. There are seven 32-word memories (one word per delay unit): RA0-19, RB0-19, RC0-19, RD0-19, RX0-19, RY0-19, RZ0-19. The RA, RB, RC, and RD memories are used for interfacing to the modifiers. On even passes, modifiers write into RB and read from RD while delay memory is being read into RC and written from RA. On odd passes RA and RB exchange functions as do RC and RD. The memories are addressed through '157 multiplexers (#1187) selected by OP. The address of the memory sending data to the modifiers is FRM3-7 (discussed above); for the memory receiving data from the modifiers, RG0-4, which is FRM3-7 delayed by 'LS670s; for the memory sending data to delay memory, RQ0-4, the delay unit counter; for the memory reading data from delay memory, RK0-4 (an 'LS174 on #1188), a delayed version of RQ0-4. RX, RY, and RZ are addressed by RS0-4 (RS0-3 are buffered by 'H04s on the DMD cards) which is generated by multiplexers (a 9309 and an 'LS158) on #1187. The multiplexer inputs are: RQ0-4, the delay unit counter; and IC7-11, command bits from the generic interface, which is selected during update ticks. RX0-15 has the base address in delay memory; RY0-15 (in delay line mode) has the index into delay memory; RZ0-15 has the limit in delay line mode, RZ12-15 has the scale factor in table look-up mode; RX16-19 has the mode; RX16-19 and RY16-19 are unused.

For PDP-10 cycles, data to be written is selected at the RT multiplexers by RTS, and the address is selected at the RV multiplexers by RVS; both selects are simply copies of RMTY (#1187). At the end of the cycle, RLC clocks the RR bus into RL0-19, which can be read by the PDP-10 interface.

For normal cycles in delay-line mode, the information to be written is taken from RA or RB by the RAB0-19 multiplexers ('LS157s on DMD) and goes through RT. The address is generated by '283 adders which add the outputs of the RX and RY RAMs. At the same time, RY is incremented by 1 ('283s) and the result clocked into RW0-15 ('LS175s). The unincremented RY is also being compared to RZ by '85s. On phase C of the third tick, RY is written (by RYW) from the RU0-15 multiplexers ('LS158s, outputs not labelled): either the incremented value (RW) or zero will be written, depending on RUE which will be asserted (to enable RW) unless RY = RZ. The RUE gating is on #1187. On phase C of the fourth tick, RC or RD is written (RCW or RDW, gated from RCDW on #1188) from RR0-19.

For normal cycles in table look-up mode, the base address RX is taken as before, but the RY RAMs are disabled (RYEA and RYEB held false) and the 'LS253 RY multiplexers are enabled (RYEC true) (gating from the mode on #1187). The RO and RY multiplexers are selected by the scale factor RZ12-15 to apply 0 to 15 units of shift to RAB. A low-order bit, RY16, is generated on #1187; if it is 1 and the rounding mode is specified, RVIC16 is asserted to inject a carry into the '283s adding RX and RY. Writing of RC or RD is as above.

The RI0-19 lines, from 'LS157 multiplexers on the FILTB cards, carry both data from the modifiers to be written in RA or RB and data from the interface to be written in RX, RY, or RZ. The selection is governed by RIS (#1181), which is a copy of ITR. RI0-19 appear directly on the data inputs to RX and RZ and, through the RU multiplexers, to RY. RI is the input to the RE0-19 register, 'LS175s on DMD clocked on phase C, which in turn is written into RA or RB (by RAW or RBW, gated on #1187 with FPRD, to indicate that the current modifier is in delay mode).

Generic Interface

The principal feature of the generic interface, shown in block diagram form in Fig.7, is the 32x32 FIFO, designated IF0-31, comprised of 64-bit RAMs on the INTF cards. This holds up to 28 commands and 4 read-data items. Commands come out of IF into the '175 registers I0-19 and IC0-11; read data go to the 'LS175s forming IRB0-19 (on the SUM cards). Input to the FIFO is from the 16-bit register IM0-15 ('LS175s on INTF). The FIFO can be addressed from four different counters: IFIC0-5 for command input; IFOC0-5 for command output; IFIQ0-2 for data input; IFOQ0-2 for data output. Each counter is advanced at the conclusion of the relevant type of cycle. The high-order bits do not actually address the FIFO but are used to distinguish full and empty conditions: if all input and output counter bits agree, the buffer is empty; if all but the high-order bits agree, the buffer is full. The data counters go from 0 to 3; the command counters from 4 to 31 decimal. All are formed of 'LS161s on #1190. On a given tick, the FIFO can do either a 32-bit output cycle or a 16-bit input cycle, according to the following priority scheme:

- A -- On processing ticks:
  - A1 -- If possible, FIFO to IRB
  - A2 -- Else if possible, IM to FIFO (left half, then right half)
- B -- On update ticks:
  - B1 -- If possible, FIFO to I and IC
  - B2 -- Else if possible, IM to FIFO (left half, then right half)

Here "if possible" means if the source has data and the destination has room to accept data of that type.

Flip-flop IMBF ('109 on #1190) is set when the PDP-10 interface puts information into IM, and cleared on a right-half IM-to-FIFO cycle. Similarly, IRBF is set by a FIFO-to-IRB cycle and cleared when the generator calculator takes the data from IRB.

The priority scheme is implemented in a straightforward though lengthy manner with a comparator, gates, and one PROM. The comparator develops IFCNF, "IF commands not full". The PROM generates IFQNF ("IF data not full") and IFQAV ("IF data space available"); the difference is that in packed data mode (determined by TP7), there must be two data slots free in the FIFO before initiating a data read. In such a case IFQAV is more cautious than IFQNF. The combinational logic develops IIL, which calls for a B1 cycle, and IRBCA, which calls for A1. Final outputs are IFAS0-1, which are the selects on the FIFO address multiplexers IFA0-4 (a '157 and a 9309 on #1190).

IFAS0	IFAS1	Address	Type of cycle
0	0	IFIQ1-2	data to FIFO
0	1	IFOQ1-2	FIFO to IRB
1	0	IFIC1-5	command to FIFO
1	1	IFOC1-5	FIFO to I, IC

The type of information in IM is encoded by IMT0-2, which are set up by the PDP-10 interface.

IMT0-2	Data in IM
000	packed data right half
001	packed data left half
010	unpacked data right half
011	unpacked data left half
100	(invalid)
101	(invalid)
110	command right half
111	command left half

For writing packed data into the FIFO, an 'LS157 on DMD introduces 0 in the low-order bits if IMT1 is false.

Command decoding is shown on #1191. IILD (a '109 on #1190), true when a command is present in I and IC, and -ITR are used to enable 'LS138 decoders whose outputs (of the form IIMn) identify the various commands. Various commands have effect within the generic interface: IIM02 and IIM03 are gated (#1190) to form ITAC and ITBC, which clock TTL buffers A and B, respectively -- 'LS157s on INTF -- whose outputs are followed by 7437 buffer gates. The register IX0-15 ('298s on INTF) is clocked by IXC (from an 'H51 on #1190) either due to IIM01 or on the tick after an IIM1, IIM2, or IIM6 with IC3 true; in the latter case, zeros are loaded into IX, corresponding to the "clear DX" function of certain commands. IIM21 loads IP0-19 ('LS163 counters, IP0-15 on INTF cards and IP16-19 shown in #1190), the pass counter. The counter is enabled by EPAS.

The commands to clear all wait bits and to clear all pause bits use flip-flops IRW and IRP ('H74s on #1190). Each is set by PHBM (phase B of an update tick when performing a "miscellaneous" command, generated on #1191) if the appropriate command bit is set in IC. They are clocked off at the end of the next pass's real (processing) ticks.

For the case of a linger command, I0-19 is always being compared to IP0-19 ('283s performing subtraction followed by gates to indicate a zero result). ISOK (meaning linger satisfied) is generated (on #1190) if the pass count is at least equal to the command data and does not exceed it by 4,096 or more; ISU (underrun) is true if ISOK is true and the numbers are not exactly equal. The IIM23 version of linger clears the pass counter first (signal IPR on #1190); repeated clearing is inhibited by the IPRD flip-flop which remains set until the linger is finished. This condition contributes to the term IILF (#1190) which indicates that a new command can be loaded into I and IC. The intermediate term IIMW (#1190) is true when a linger command is present and the pass counter is valid (not being reset).

The interface address, IA0-7, is the generator or modifier number of the next command to be performed. If it is a generator number, it contains all 8 bits; a modifier number occupies the left 7 bits with the rightmost bit distinguishing L0 from L1 or M0 from M1. The address must be available the tick before the command is executed, so that it can be loaded into the clock pipeline. Therefore the IA multiplexers ('LS257s and 'LS258s on #1190) have four possible sources: two (generator or modifier number) from IF, to load the address into the CPnBm at the same time the command is loaded into I and IC; and two from IC, similarly, for a command previously loaded into I and IC but not yet performed. The enables and selects IAEA, IAEB, IASA, and IASB perform this function.

Data being written to computer memory is put in IWB0-19 ('LS298s on SUM cards) by the generator calculator, which at that time sets IWBF ('109 on #1190). The PDP-10 interface clears IWBF at the end of a DMA write cycle.

PDP-10 Interface

Data paths for the PDP-10 interface are on the nine TENI cards. The low-order six cards, termed "full", have both data and memory address logic; the high-order three cards, termed "partial", have data but not address logic. A block diagram is given in Fig. 8.

The DEC-level I/O and memory busses are received and converted to TTL by 75110 differential line receivers, biased to VREF (about -1.5 volts). The memory data, being in pulse form, is "caught" by 9314 latches to form TD0-35. Parity of each four-bit slice is generated on the TENI card by 'LS86s; the results are then combined to form TDEP by a 9348 on #1193.

The busses are driven by 75110s followed by 2N4258 transistors. In keeping with PDP-10 practice, the I/O bus lines are driven to ground, and the memory lines toward -5 volts with a 100-ohm parallel termination to ground. Data to memory bus bits 4-23 comes direct from IWB0-19; zeros are written for the other bits. The parity bit, IWBP, is formed by 9348s on #1184. The I/O bus bits 16-35 are driven from the lines TTT16-35, coming from 'LS153 multiplexers that select one of four sources: D0-19, the diagnostic bus; RL0-19, the delay memory output register; various bits for CONI-A; and the TCC register (slightly scrambled), for CONI-B. I/O bus bits 0-15 are not driven. The received I/O bus appears on the output of 'LS157 multiplexers as TR0-35 (the other multiplexer input is currently unused); this is clocked, in the case of DATAO, into the 'LS175s TH0-35. Data passes from the PDP-10 interface to the generic interface (IM register) through 'LS257s on the INTF card. These select between left half (bits 4-19) and right half (bits 20-35) of either TH (DATAO data) or TD (data read from PDP-10 memory). TH0-35 also go to the delay units for PDP-10 references to delay memory.

On the full TENI cards is a 16x24 memory called TCI0-23. It is addressed by TAA0-3 (a '157 on #1195). Six words of this memory are actually used, as follows:

Address	Contents
1	write data CA
2	read data CA
3	command CA
5	write data WC
6	read data WC
7	command WC

TCI0-23 is clocked into two different registers, both comprised of 'LS175s on the TENI cards: TA14-35, which is the data actually put on the address lines of the PDP-10 memory bus; and an unlabelled register whose output is incremented by 1 in a set of '283s to form TC0-23. Associated with each WC is a flip-flop ('LS109) on #1195: TWE for write data, TRE for read data, TCE for commands. These are the "exhausted" flags for the three types of DMA; they are directly set by the master reset function.

When a DATAO-B is performed, TH0-3 (the high-order four bits of the DATAO data) are selected onto TAA0-3, and TH12-35 are selected for the data input to the TCI RAMs (selection is by 'LS158s on TENI). The select signal, TAWW, comes from an 'H30 on #1195 and indicates not only that a DATAO is occurring but also that no memory cycle is in progress. It is used to turn off the proper "exhausted" flag, as selected by TAAS1-3, a phase C clock pulse decoded from TAA0-3 by an 'LS138.

TAW, the write pulse to the TCI RAMs, occurs on phase C ('H00 on #1195).

Addressing of a memory cycle proceeds through four successive ticks, indicated by one of the following in succession being true: TMG, TMGD, TMGDD, TMG3D. On the first tick, the appropriate CA word is addressed in TCI; phase C clocks it into TA to drive the PDP-10 memory address lines, and also into the other register to be incremented. On phase C of the second tick, the incremented CA is written back into TCI. On the third tick, the relevant WC is clocked into the unnamed register; on the fourth tick the incremented WC is written back into TCI and, if a carry comes out of the incrementer, the proper "exhausted" flag is set.

Control of the actual memory cycle is shown on #1193. Three 'LS10 gates create cycle requests TRY, TWY, and TCY (read data, write data, and commands) according to the "exhausted" flags and buffer-register or FIFO full or available flags. When a memory cycle is not in progress any request forms TMQ, which is clocked into an 'LS174 (on #1195) and comes out as TMY. This is turn becomes TMG ("memory go") if no cycle or DATAO is in progress. TMG is clocked into TMB ("memory busy") to indicate a cycle in progress. At the same time, TWQ is clocked into TMW if a write cycle is requested. TMB asserts TMRQ, the memory port multiplexer request. The acknowledgement TACKN asserts TAE, enabling the driving of the memory address lines, and also, after a 50-nsec delay (SNG82), putting up TREQCYC to indicate a memory cycle request. The memory responds with TMAA (address acknowledge) which resets the TD latches if in a read cycle, and is latched as TMA which turns off TMRQ. TREQCYC was turned off by TMAA and is held off by the absence of TMRQ. On a write cycle, the fall of TMAA initiates an 85-nsec delay (SNG82) during which TWE is on to enable driving the data bits of the memory bus. When the delay expires, TMDS sets TMD which is resynchronized as TMDD. TMDD causes TMF which clears TMB to end the cycle.

A read cycle proceeds as above through the address acknowledge pulse. Then TDR clears the pulse-catcher TD0-35 and TMA then enables TD for input (TDE). The read restart pulse TRDRS eventually clocks on TMD, turning off TDE. If there is no parity error, -TDEP and TMDD then finish the cycle. If a parity error occurs, the memory control stays in a busy state with TMPE on until the problem is acknowledged by a CONO-B to reset it. Such a CONO sets TMC ("memory continue") which permits the cycle to finish, asserting TMER to reset the error flip-flop. To detect nonexistent memory, a counter ('LS163s) counts about 50 microsec after the start of a cycle and then asserts TMXO, which, if the memory is still busy, sets TMXE. This state is also gotten past by a CONO-B which sets TMC.

The I/O bus interface begins with the device address comparator, a 9324 on #1194, which asserts TS if the high-order five IOS bits (TRS3-7) agree with five bits set by switches TSS3-7. TS and either DATAI or CONI create TTE which enables the bus drivers on the TENI cards. TS is ANDed with DATAO-CLR to give the clock TDOC. The low-order IOS bits, TRS8-9, are used to decode TSA (device code A selected) and TSB (for device code B), and are clocked by TDOC into a '175 to emerge as TMDOR and -TMDOA. CONO-A, CONI-B and DATAOs are synchronized in an 'LS175. In the case of a DATAO, the TMDO flip-flop is set and remains set until the function of the particular DATAO has been performed. Its K term comes from an 'LS20 whose four inputs correspond to completion of the four DATAOs:

IWBG	DATAO-C
RTD	DATAO-D
TMDOW	DATAO-A
TAWW	DATAO-B

CONO-A, indicated by the unsynchronized clock TCOCA, sets TIA0-2 and TIB0-2 ('LS174s on #1195) according to the CONO data; these are the PIA numbers. It also clocks the diagnostic address from TR25-31 into IDA0-5 and IDS (a '174 and a '175 on #1194). The synchronized version TCOCADA clocks IRH ('LS109) which is on in "all ticks update" mode.

CONO-B uses bits TR32-35 to address two 9334 addressable latches, writing bit TR31 into the addressed cell. The latched outputs, TP0-15, are the BB.AAA conditions of the programming specification.

The master reset signals MR and TR have three causes: IOB RESET, called TMR; power-on reset, POR; and the reset bit of CONO-A.

The TCC register is read by CONI-B. This is formed mostly of 'LS175s on INTF cards; but bit TCC5, the lost cause bit, is an 'LS109 on #1195. The register is clocked from TJ0-4 and CTK1-8. The CTK lines are the current time state; the TJ lines, generated on #1191, represent in order the I1-5 bits of CONI-B. TCC is clocked by TCCC each phase C until a bit appears in it which is masked on by an 01AAA CONO-B bit (TP9-13). Such a condition asserts TCCF (#1195), which suppresses TCCC. If on a subsequent tick a TJn masked on by TP9-13 occurs, ITJ (#1191) will be true and TCC5 will be turned on. TCC5 is cleared by TCIBR which comes from CONI-B (#1194).

The interrupt request lines TIOBPI1-7 come from a 9334 addressable latch on #1195. Twice per tick a bit in the latch is strobed in; on the first half of the tick from TIA (interrupt request, channel A), and on the second half from TIB; address is TIA0-2 or TIB0-2 respectively. The interrupt requests are formed from the various interrupt conditions and the mask bits, TP6 and TP8-14. All requests are reset by the master reset function, and by any DATAO, CONO-A or CONO-B.

Analog Outputs

Each ALOG card contains two independent analog output channels. All channels share the same clock (phase A) and data inputs (AP0-13), differing only in their SEL signals. These are the enables AE0-15 decoded from GO16-19 by 9301s on #1175, including the GD term indicating a generator in DAC mode.

A 14-bit register, shown on the ALOG schematic (#1110) as D0-13 ('LS174s), holds the current DAC input. The SAMP flip-flop ('LS109) is true when the sampling switch is in the sample state and false when it is in the hold state. A counter ('LS109 and 'LS163) times the duration of the hold state.

The DAC has a built-in op amp which converts the current switch outputs to a low-impedance voltage source. R35, combined with internal feedback and offset resistors, adjusts the DAC output range to + or - 1.5V. This is applied to a sample-and-hold circuit whose output, SIG, drives the filter chain or final output amplifier. The sampled DAC value is stored in C14, which is buffered by U10 to drive SIG. Q5 acts as a switch which, when closed, allows C14 to charge to the DAC voltage through R3 and R39.

About 260 nsec before the DAC register is updated, the SAMP signal goes high (false), cutting off Q1 and allowing Q2 to turn on. This turns on Q3 which applies -15V to the gate of Q5 through CR3 and CR4, thereby cutting off Q5. Approximately 6 microsec later, after the DAC output has settled to its new value, SAMP falls, allowing +15V to be applied to the cathode of CR4, permitting Q5 to turn on. Some of the switching voltage is transferred to C14 because of the capacitance of Q5. C13 couples a pulse of opposite polarity and approximately the same amplitude into C14 to minimize this effect.

An 'LS175 holds FE (filter enable) and FA0-1 (which determine filter frequency). FE drives a DG154 FET switch which selects filtered or unfiltered output; FA0-1 operate other switches which program the UAF31 active filters. The filter configuration is 6-pole Butterworth, with breakpoints as follows:

FA0-1	freq
0	4.5 kHz
1	9.0 kHz
2	13.5 kHz
3	18.0 kHz

The FE and FA 'LS175 is loaded at the same time, with the same data for all channels, by AFC (#1175) which derives from IIM03.

Diagnostic Readback

The high-order 6 bits of the diagnostic address, IDA0-5, are decoded by 'LS138s on #1197, to form diagnostic enables DE00-63 (numbered in octal). Each enable goes to a group of up to five 'LS257s (or 'LS258s) whose outputs form the 3-state diagnostic bus D0-19. The low-order address bit IDS is buffered to perform the selection on all the diagnostic multiplexers. Contents of the several addresses are tabulated below.

Left Column: Diagnostic Address (in octal)
Right Columns: 20-bit Diagnostic Word (bit numbers in decimal), X=undefined bit

```
0 TTT0-19
1 TC4-23
2 X X X X X X TIOBPI0-1 X X TIOBPI2-3 X X TIOBPI4-5 TTT12-13 TIOBPI6 X
3 X X X X X X X X X X X X X X X X TC0-3
4 TD16-35
5 TH16-35
6 X X X X TD0-15
7 X X X X TH0-15
10 TA4-23
12 X X X X X X X X X X X X X X X X TA0-3
13 X X X X IWBP X X X TMRQ TMPXCLR X X TREQCYC TMW X X X X X X
14 IX0-15 X X X X
15 TCC0-4 TCC6-16 X X X X
16 II16-19 IC0-11 X X X X
17 IM0-15 X X X X
20 IP0-15 X X X X
21 II0-15 X X X X
22 ITB0-15 X X X X
23 ITA0-15 X X X X
24 GT0-11 X X X X X X X X
25 GL0-11 X X X X X X X X
26 GZA0-11 X X X X X X X X
27 GZB0-11 X X X X X X X X
```

```
30 GXC0-10 X X X X X X X X X
31 GYA0-3 GYC0-3 GYE0-3 X X X X X X X X
32 X X X X GY4-11 FRM4-7 GSUM2-5
33 X GX1-11 FRM0-3 X X GSUM0-1
34 GYB0-11 CP4B0 CP5B0 CP6B0 CP7B0 CP4B2 CP5B2 CP6B2 CP7B2
35 GYD0-11 CP4B1 CP5B1 CP6B1 CP7B1 CP4B3 CP5B3 CP6B3 CP7B3
36 GWD0-11 FSUM3-6 GFMI3-6
37 GU0-11 X FSUM0-2 X GFMI0-2
40 GXB0-11 CP4B4 CP5B4 CP6B4 CP7B4 CP4B6 CP5B6 CP6B6 CP7B6
41 GWB0-11 CP4B5 CP5B5 CP6B5 CP7B5 CP4B7 CP5B7 CP6B7 CP7B7
42 GVB2-12 X X X X X X X X X
43 GJ0-7 GVA0-3 X X X X X X X X
44 GAZ0-7 GC0-3 X X X X X X X X
46 GM0-3 GRMD0-3 FMD0-3 X X X X X X X X
47 X X FU0-9 X X X X X X X X
50 FE0-19
51 FEE0-19
52 FXA0-19
53 FXB0-19
54 FW0-19
55 FG0-19
56 FU10-29
57 FQI0-19
60 FV0-19
61 -GRMD0B0-3 -GRMD1B0-3 -GRMD3B0-3 -GRMD5B0-3 GRMD6B0-1 GRMD7B0-1
62 FN0-19
```

```
63 TP0-15 X X X X
64 FF0-19
65 FD0-19
66 FJ0-19
67 FK0-19
70 FXJ0-19
71 FX0-19
72 FXH0-19
73 FXH20-23 FXG6-7 FXG10-11 FXG14-15 FXG18-19 FXG21-23 FXH24 FXG3 X X X
74 AP0-19
76 S0B0-19
77 S1B0-19
100 S2B0-19
101 S3B0-19
102 GRA0-19
103 IWB0-19
104 GRB0-19
105 -IRB0-19
106 GWA0-19
107 GB0-19
110 GXA0-19
111 GK0-19
112 GVA4-23
113 GJ8-27
114 GAZ8-27
115 GC4-23
```

```
116 GP0-19
117 GO0-19
120 S0A0-5 S1A0-5 SCS0-1 SA23E SR OPD GFM0-2
121 S2A0-5 S3A0-5 X X X X GFM3-6
122 GXB12 GWB12 -CSUM0-5 IA0-3 X CPP CTR1F CTR4F CTRO CTRA ITR OT
123 FIN0-7 IA4-7 CTR0-7
124 CPAB0-7 X X CTK0-9
125 CTP0-9 CTT0-9
126 FXM1P FXM2P FMD4 FMD6B0 IRP IRW IMBF IWBF IP16-19 IFIQ1-2 IFIC0-5
127 FMD6B1-4 IIL IILD -IFAS1 -IFAS0 X X IFIQ0 IFOQ0-2 IFOC0-5
130 GZ0-17 GZC3 GZCN
131 GWBO GWCO GEC GUF GY0-3 GVB0-1 GVB13 GX0 GJ0D GRA0D FLIDFEA X X X X GYQ3
132 RMRY RMNY RMTY RMC0 RP0-1 RMC2 RG0-4 RPE RQO FPRD RQ0-4
133 RMRAS RMCAS RMWPP RMC1 RP2-3 RMC3 RK0-4 RMRT RMRQ RMR0-5
134 IDA0-3 GXS GXBS GUS0-1 X X X X GRMDA0-3 TIA0-2 RY20
135 IDA4-5 X IDS GX12 X X X X X GINVBE GTS GRMDA4-7 TIB0-2 X
136 IMT0-2 TMW TMB TMC TMD TMDD TZA X TCC5 X TMDO TMDOA TMDOR TDOCD TMGD TMGDD TMG3D TMXO
137 X X X X -TMY TAAI0-1 X TDOCDD TCOCAD TCOCADD X IRH TCE TRE TWE GTT GTA GCOD GCOG
140 RE0-19
141 RT0-19
142 R0-19
143 RV0-15 X X X X
144 RX0-19
145 -RW0-19
146 RY0-15 X X X X
147 RZ0-19
```

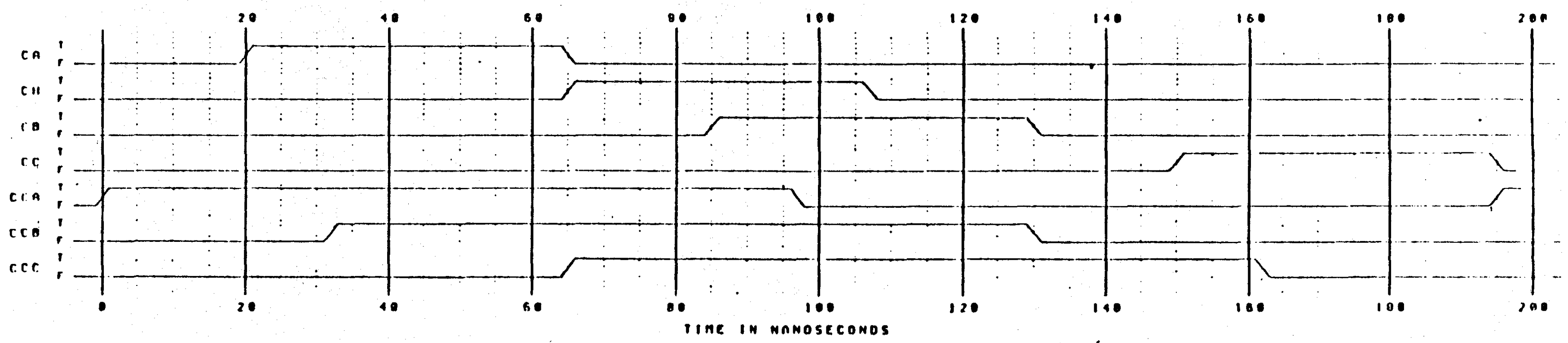

Figure 1. Clock Phases

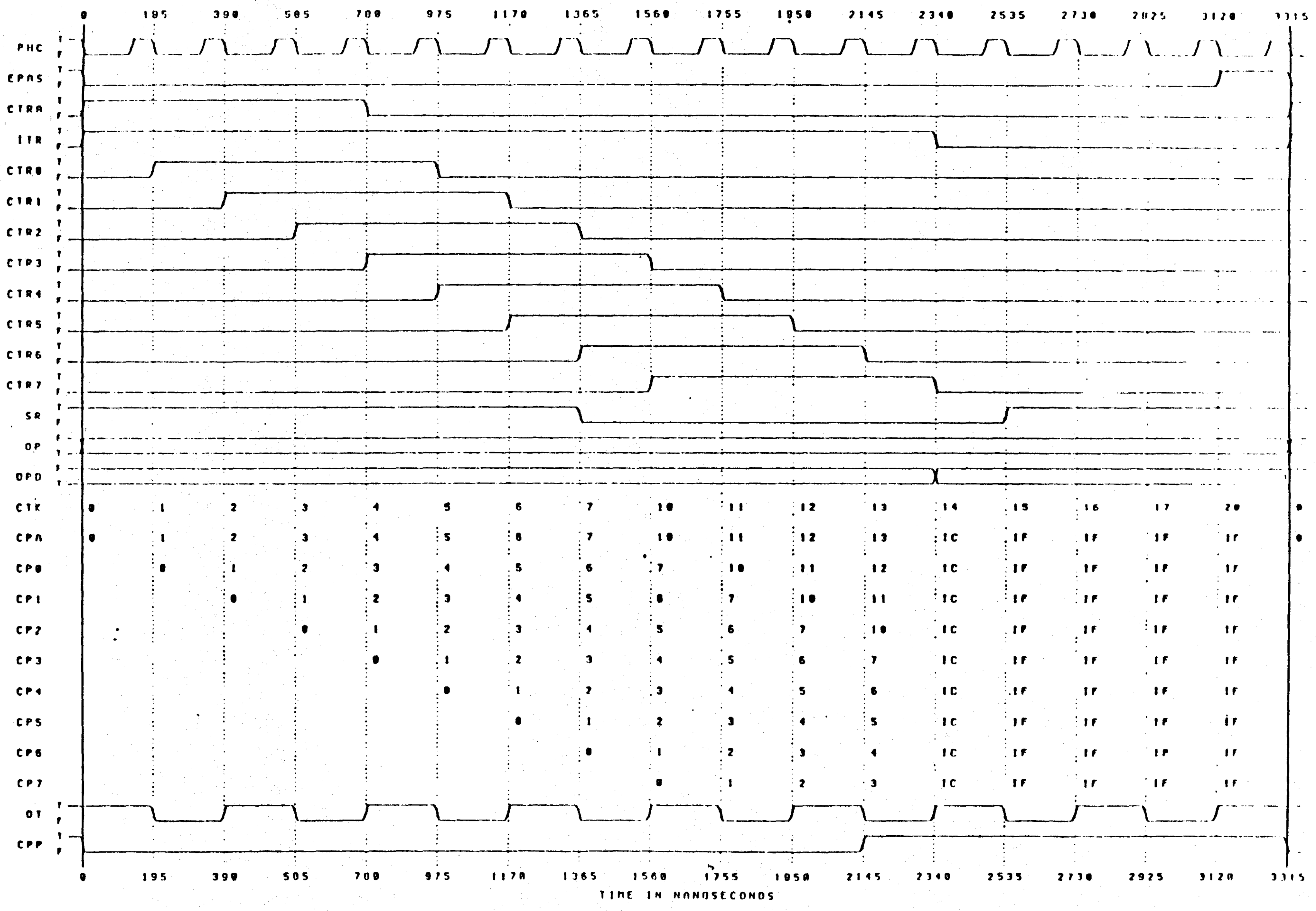

Figure 2. Clock Counting

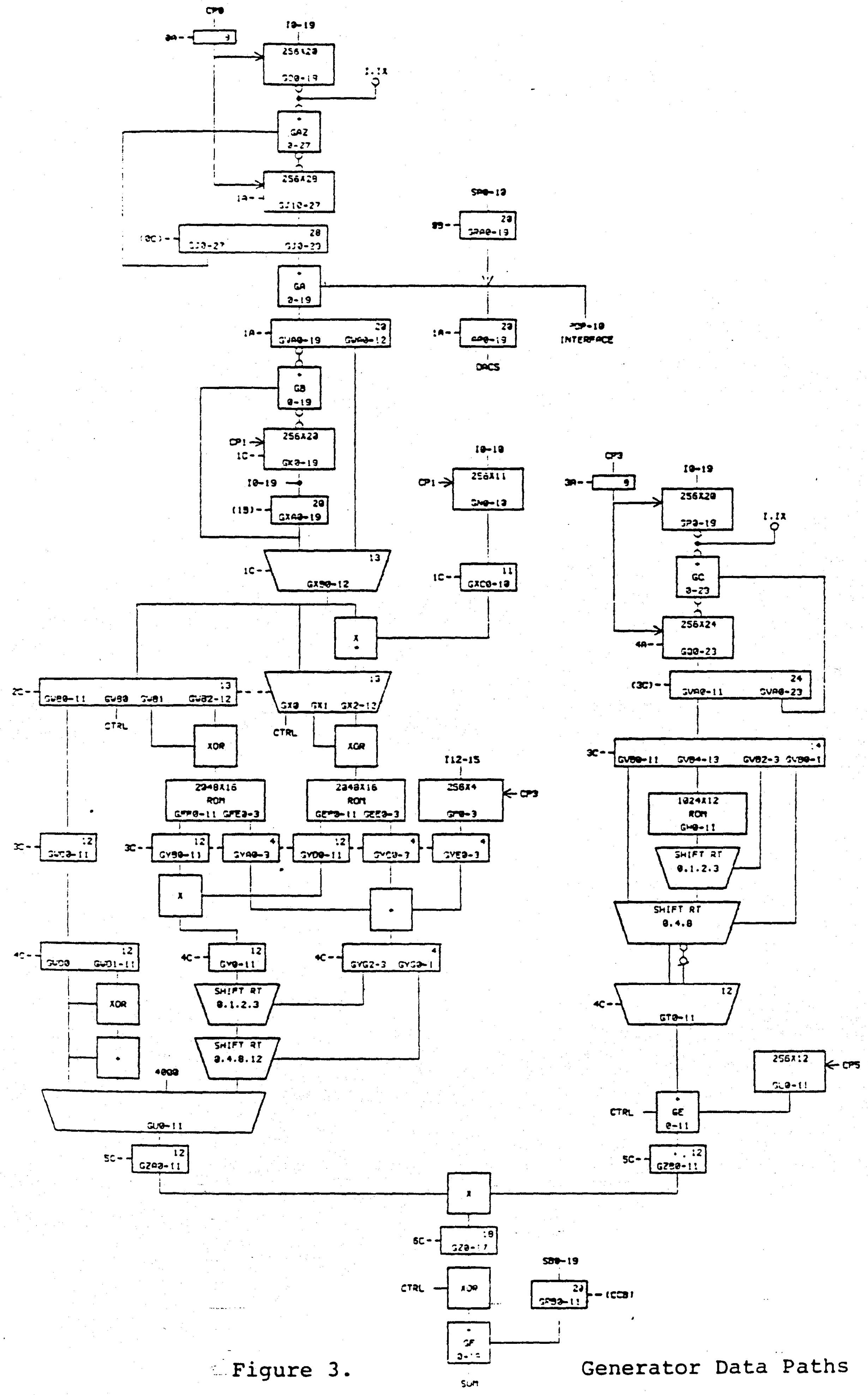
Figure 3. Generator Data Paths

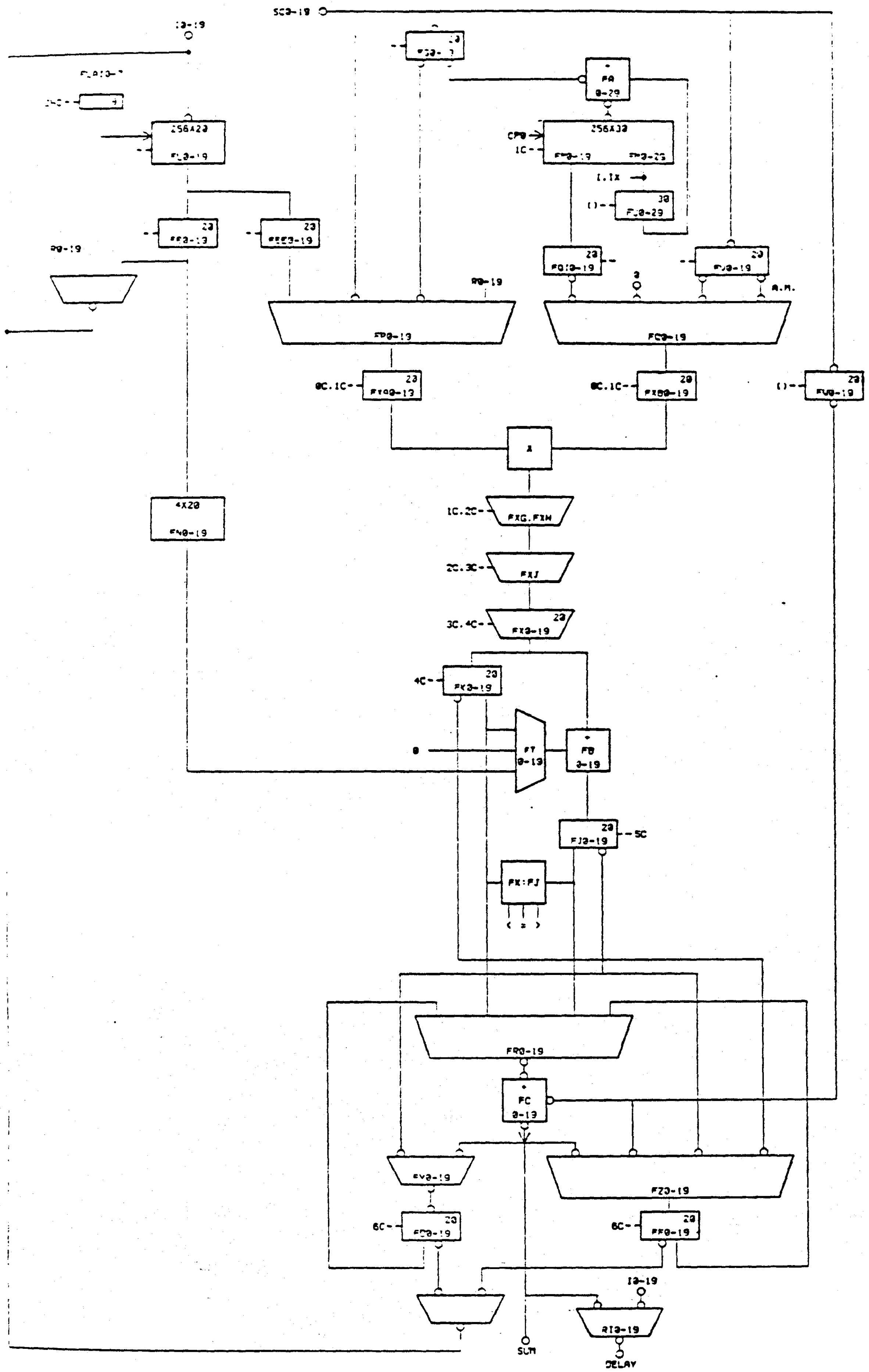

Figure 4. Modifier Data Paths

	EVEN TICK			ODD TICK		
	phase A	phase B	phase C	phase A	phase B	phase C
S0	pole-0 read IN	generator read FM	modifier read RM	mod-mix read IN	generator read FM	
S1	generator read - pause - write SUM			generator read - pause - write SUM		
S2	pole-0 read IN	generator read FM	modifier read RM	mod-mix read IN	generator read FM	
S3	pole-0 read IN	mod-mix read IN	modifier read RM		modifier read SUM	modifier write SUM

EVEN PASS SHOWN (OP false)

EXCHANGE S0 WITH S1 AND S2 WITH S3

FOR ODD PASS (OP true)

Figure 5. Sum Memory References

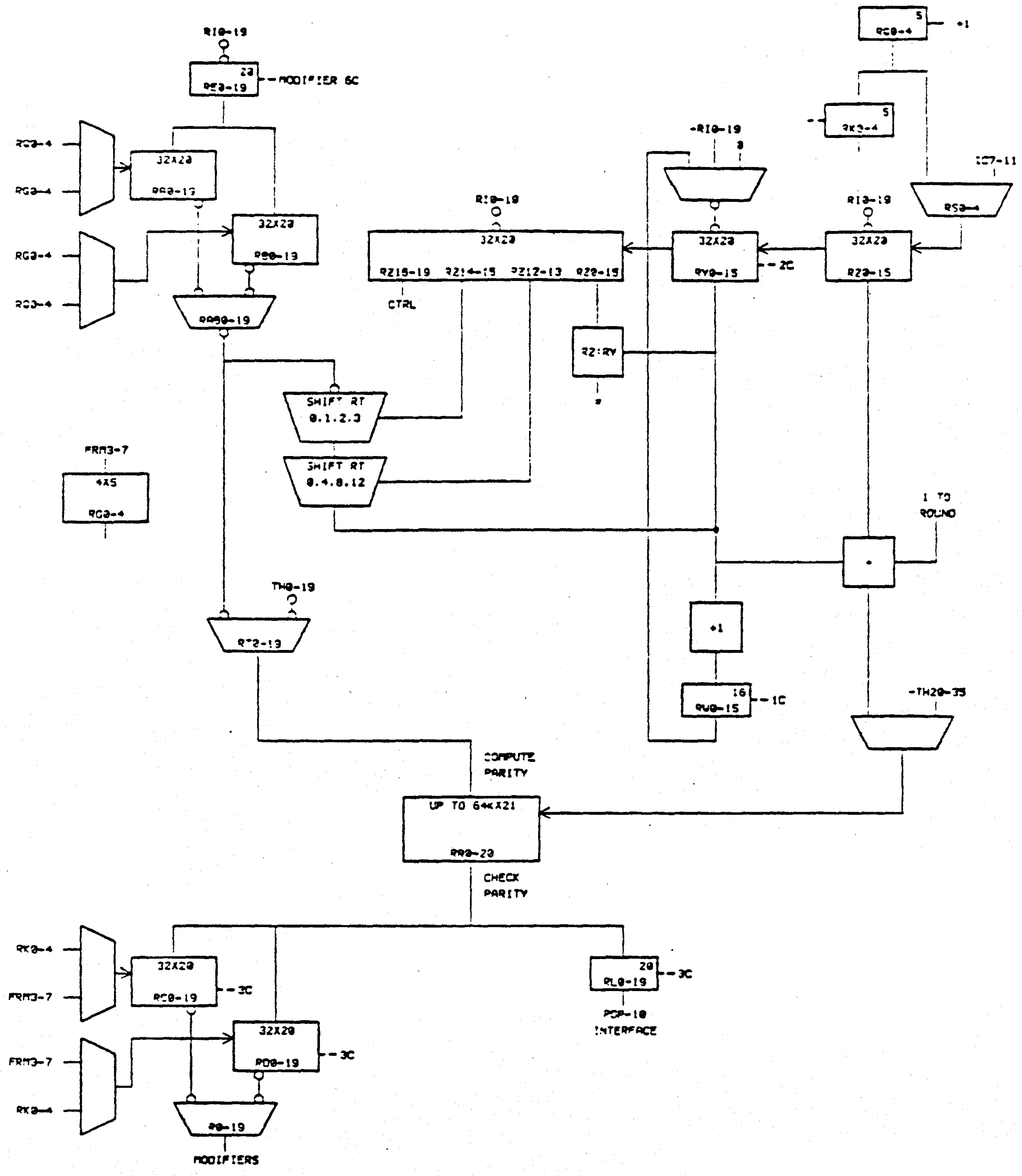

Figure 6. Delay Unit Data Paths

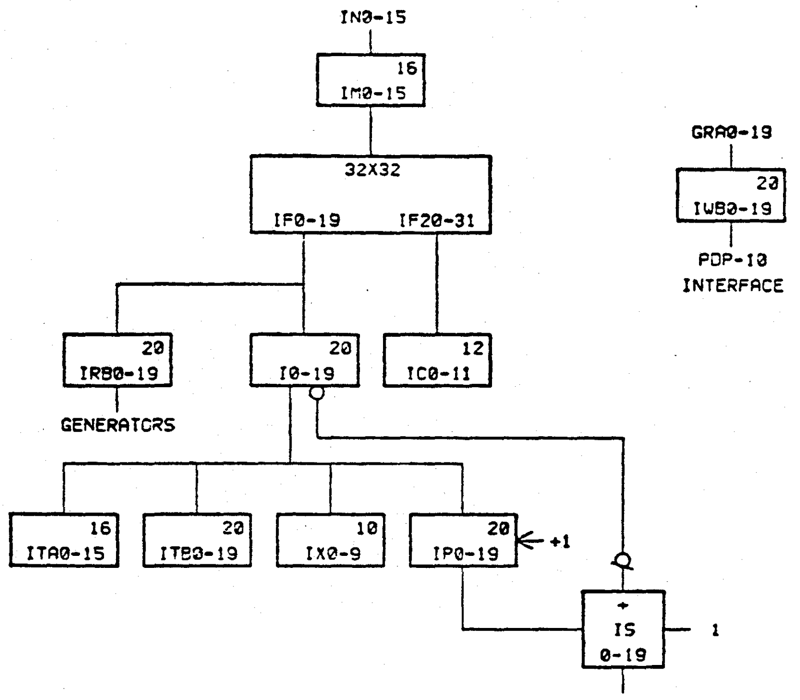

Figure 7. Generic Interface Data Paths

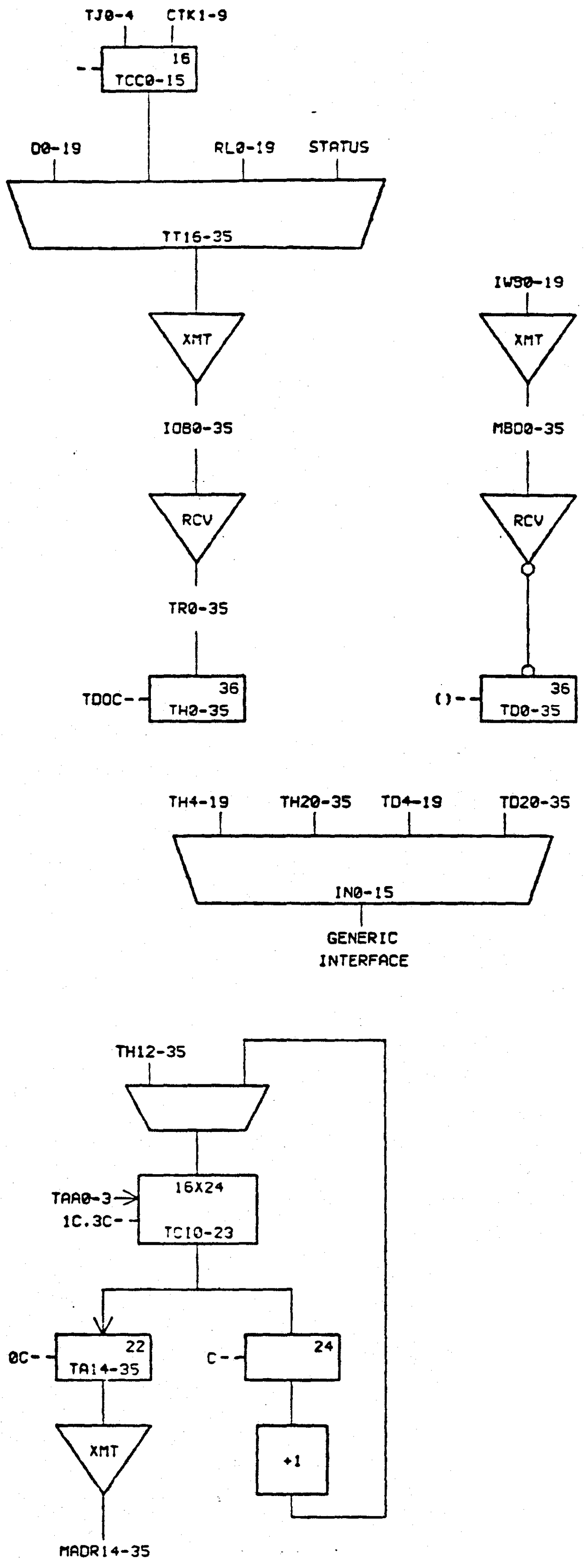

Figure 8. PDP-10 Interface Data Paths