THE MAXC MICROPROCESSOR

MAXC 8.2

July 26, 1972

by

Butler Lampson
Ed Fiala
Ed McCreight
Chuck Thacker

Xerox Palo Alto Research Center
3180 Porter Drive
Palo Alto, CA 94304

1.0 Overview

The MAXC microprocessor is interded to be a reasonably general purpose processor, customized to some extent for PDP-10 emulation. It is used as a central processor and disc controller in the MAXC system. Physically, the processor occupies 24 card positions in two Augat card cages (19" x 8.7"), and the disc control occupies 8 card positions in a third cage. Figure 1 is a logical block diagram of the processor. It is organized around a 36-bit bus, on which all transfers between subsections of the machine occur. Data transfers to and from this bus and all other functions in the machine are under control of a 72-bit microinstruction word. A machine may be configured with either 1024 or 2048 words of instruction memory.

Two fields in every microinstruction specify a bus source, which loads data onto the bus, and a bus destination which reads, and usually stores, the data. Sometimes a single value of the source or destination field may specify additional operations, or several different source or destination values may specify the same bus operations. These peculiarities are specified in the appropriate section of this manual. The sources and destinations are listed and their properties summarized in Appendix D. In general any source may be sent to any destination, with the following exception: a slow source may not be sent to a slow destination.

Slow sources are:

- a local memory
- NOT F
- the ALU;
- KSTAT and KUNIT in the disk interface

Slow destinations are:

- a local memory
- Y if the next instruction contains PQ RCY [Y]
- Q if the next instruction certains QODD or QEVEN

There are also two function fields F1 and F2 which invoke various actions supplementary to the source-destination scheme. These actions are specified where appropriate throughout the manual and summarized in Appendix C.

The machine is synchronous, with a cycle time of 150 ns. The technology with which the processor is implemented is 74H TTL; IC's are mounted on wire-wrap cards, and the back panels are also wire-wrapped. An exception is the 1024 x 19-bit memory card which is used for the instruction, dispatch, map, and scratch memories; this card is a printed circuit. All cables exit the processor from the rear edges of the cards. No special

mechanical provisions are required for cabling. The processor is cooled by a fan unit which mounts immediately below the processor card cage, and powered by a power supply mounted on the bottom of the cabinet.

The external interfaces to the processor are shown dashed in Figure 1, and consist of the following:

1. 8 disc unit cables, which connect the disc control portion of the processor to 8 2314 or 3330 type disc files;

2. 2 memory port cables, which connect to two ports of the MAXC memory system. This memory is a 512K (expandable to 1024K) x 40 bit (+8 error correction and detection bits) dynamic MOS system. Access time and cycle time are 800 ns. One port is used for disk transfers, the second for CPU transfers.

3. One interprocessor communication cable (labeled "TO NOVA"). This interface has two functions.
   a. It carries interprocessor communication strobes between all processors of the MAXC system. All normal communication between processors occurs through memory, and these strobes serve to indicate the presence of messages in mailbox locations known to all processors.
   b. It is connected to a controlling minicomputer (Data General Nova), which has the task of monitoring the system for errors and abnormal conditions. This interface is used for debugging microcode in the processor under control of a debugger in the Nova. The control memory of the microprocessor is loaded via this interface at start up, during debugging, and when errors occur during normal operation.

1.1 Notation

All numbers in this document are in decimal unless followed by a B, in which case they are octal. Thus, 10 = 12B. Arithmetic is 2s complement.

Names for fields in the microinstruction are in Appendix A. Registers, memories and data paths are named L, R, P, Q, X, AC, Y, E (bus), S (scratchpad), D (dispatch), MAP, I (instruction memory), NPC, STACK, IMA (instruction memory address), MAR, MDR, MDRL (low 4 bits of the 40 bit memory word), EALUBC (bus and ALU branch conditions), F(flag register), ALU (output of arithmetic-logic unit), G, H, J, K (F register bits).

Bits in registers (and on data paths like B and AIU) are referenced by integers in brackets following the register name, counting from the left as though the register (or path) were 36 bits wide. Numbering registers in this way is compatible with PDP-10 documentation (it would otherwise be better to number from the right). Thus B[0] is the sign bit of the bus, Y[27] is the sign bit of the 9-bit Y register, and B[9-12] is the AC field of a PDP-10 instruction on the bus. For 40-bit registers like MDR, the extra 4 bits are MDR[36-39].

If A is a number with a bits and B a number with b bits, then (A,B) is a number with a+b bits and

(A,B)[(36-b)-35]=B
(A,B)[(36-a-b)-(35-b)]=A

Destination names always appear as NAME¬ and they are the only names in this manual which are written with a final ¬. If a register is both a source and a destination, these are always called NAME (the source) and NAME¬ (the destination). Also, some operations can be initiated by either primary or secondary functions, and these are given the same name in F1 and F2. When a field in the microinstruction is used to address a memory M, the field is called MA (e.g., LA, RA, SA). Sources, destinations, and functions pertaining to the disk control section of the microprocessor have names beginning with "K".

The word "illegal" means "must be avoided by the programmer, since the result is not well-defined by the implementation of the processor." The hardware does not check for illegal operations.

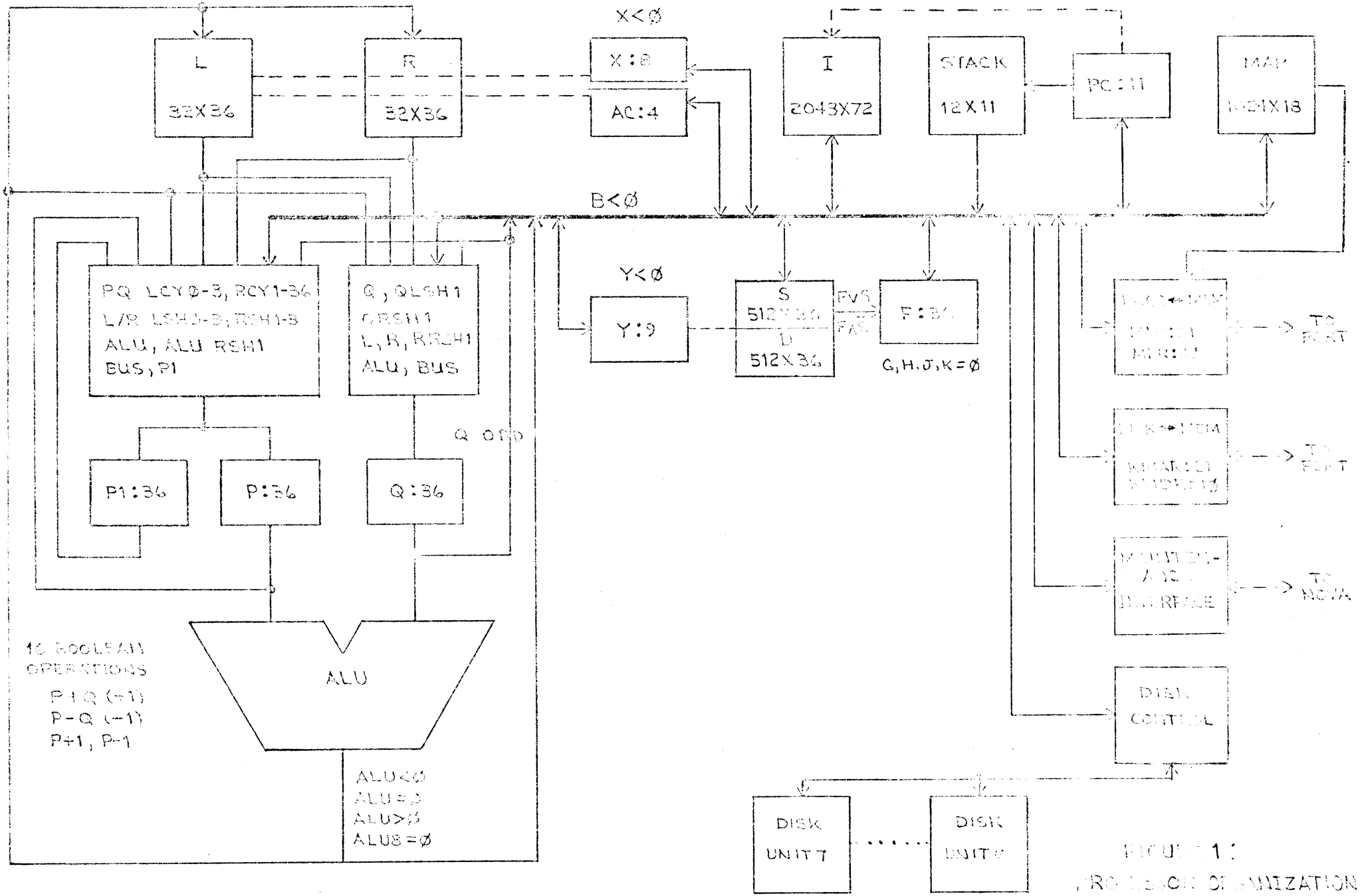

FIGURE 1:
PROCESSOR ORGANIZATION

2.0 Control

The control section of the processor consists of a 12-bit program counter (NPC) in which the most significant bit if currently unused), a 12-bit x 12-level subroutine stack, gating to produce an instruction memory address, and the instruction memory.

The processor has single instruction lookahead, i.e., the fetch of an instruction occurs during the execution of the previous instruction. All instructions require one cycle for execution. An idle cycle (during which an instruction is fetched from the instruction memory, but no instruction is executed) occurs only after a read or write of the instruction memory. Execution of an instruction can be delayed one or more cycles by the memory interface; see section 5.

Three fields of the microinstruction are used for control. These are an eleven-bit branch address field (BA), a five-bit field (BC) which specifies one of 32 conditions to be tested to determine whether a branch is to be done, and a two-bit field which specifies the type of branch (BT). The BT field is interpreted as follows:

TYPE	EFFECT
0	CALL (BA field) IF (condition)
1	GOTO (BA field) IF (condition)
2	RETURN IF (condition)
3	DGOTO (BA field) IF (condition)

If BT = DGOTO and the branch condition is true, no interrupt can occur after this instruction (see section 2.1).

The condition selected by BC (see Appendix B) is tested, and if true, the branch specified by BT occurs. The branch conditions which test the values of the ALU output and the bus refer to the values computed by the previous instruction (unless F1 = FRZBALUBC and INT=0 in that instruction, in which case they have the same result that they would have had in that instruction). Those which test bits in registers refer to the value at the beginning of the current instruction. Note that the complement of every branch condition is also a branch condition.

Table 1a specifies how the next instruction and the next program counter (NPC) value are determined by the current instruction and the interrupt system. Note that a deferred branch (DGOTO) allows the next instruction in sequence to be executed before sending control to the location specified by BA. The effect of a DGOTO can therefore be cancelled by a GOTO or RETURN in the next instruction, and a CALL in the next instruction will push the address supplied by the DGOTO. The effect of DGOTO B[25-35] is provided by F2=LOADPC.

The 12-level subroutine stack holds return links for subroutine calls and interrupts. The ways in which the stack can be affected by the current instruction are specified in Table 1b. The STACK← destination first pushes B[1-11], then B[12-23], onto the stack; normally this is combined with F2=NPC← to provide a 3-level dispatch. Note that STACK← sets G=1 when B[Ø]=Ø but leaves G unchanged when B[Ø]=1. This is the reason why the first push is of 11 bits and the second 12 bits. It is illegal to do a RETURN in the instruction following one which does STACK←. No explicit PUSH operation is provided, since the same effect can be obtained by

```
NPC←, B[25-35]←argument to be pushed;
CALL [.+1];
```

The stack can be read onto the bus (right justified); it is illegal to do this in an instruction which has a CALL, RETURN or POP.

Action of Current Instruction	Address of Next Instruction (IMA)	Next Value of NPC
No branch	NPC	NPC + 1
No branch & interrupt	INTADR	NPC
GOTO	BA	BA + 1
GOTO & interrupt	INTADR	BA
DGOTO	NPC	BA
DGOTO & interrupt	--not allowed; interrupt is delayed--	
CALL	BA	BA + 1
CALL & Interrupt	INTADR	BA
RETURN	STACK	STACK + 1
RETURN & interrupt	INTADR	STACK

Note: F2=LOADPC makes the next value of NPC be B[25-35] regardless of what is said above.

Table 1a. Instruction Sequencing

Action of Current Instruction	Effect on Stack
CALL	PUSH NPC
RETURN	POP
F1 or F2=POP*	POP
BD=STACK¬*	PUSH B[1-11], then PUSH B[12-23]

*Illegal in the same instruction with CALL or RETURN

Table 1b: Stack Actions

2.1 Interrupts

An interrupt system is provided to allow high speed devices such as the disks to be serviced. The elements of the interrupt system are:

1. A flag, INT, which determines whether the processor is in normal mode or in interrupt mode.

2. Duplicate copies of some processor registers; see below for details.

3. A 16-bit ARM register, one bit per interrupt channel. This register may be a bus data sink or source (selected by functions). An interrupt request for which the corresponding ARM bit is 0 is ignored.

4. A single interrupt enable flag in the F register (see 2.2).

The first 16 microinstructions are reserved for an interrupt transfer vector. When interrupt n occurs, the instruction in location n ($0 \leq n \leq 17B$) is executed and INT is set. The interrupt instruction is simply sandwiched into the normal flow of control, so that when it is in execution, NPC contains the address of the instruction which the program would have executed during that cycle if the interrupt had not occurred. The interrupt instruction must contain an unconditional CALL to save NPC on the stack and send control to the start of the interrupt routine. The last instruction of the interrupt routine should be a RETURN which includes the IRET function. This function clears INT and restores the state to its pre-interrupt value. See below for a description of the timing.

The scheme just described works only if everything currently known about the sequencing of the main program is contained in the NPC value. Since this is not the case immediately after the execution of an instruction which loads NPC with anything except IMA + 1, an interrupt is not permitted to occur after such an instruction, but must wait for a more opportune moment. Only instructions containing F2=LOADPC or a successful DGCTC have this problem, and the processor automatically inhibits an interrupt from occurring in the cycle after these instructions.

It is the programmer's responsibility to inhibit interrupts in other cases where that is necessary by setting F2=INHINT. This must be done

1. If BD = RMW- or F1 = RMWREF or RMWREFDXR, since an interrupt cannot be allowed during the RM phase of a RMW memory reference. The processor automatically inhibits interrupts after every instruction of the RM phase

except the first, so the programmer need provide F2 = INHINT only on the instruction which starts the reference.

2. If BD = WRITE¬ or F1 = WREF or WREFDXK and MDR does not yet contain the data which is to be written (see section 5). If another instruction is executed before MDR is loaded, the programmer must have F2 = INHINT on that instruction also. It is not necessary to INHINT on an instruction containing WRESTART, but if by the end of the instruction after the WRESTART, MDR is not loaded, then that instruction must INHINT. It is not necessary to have F2 = INHINT on the instruction which loads MDR, since an interrupt after that instruction causes no trouble.

When interrupts are inhibited, any pending interrupt is simply delayed. No pending interrupt request is lost.

However, if the IREQ level which requests an interrupt is removed before the interrupt occurs, it will be forgotten (see the end of this section for a note on the timing). Note that a loop consisting entirely of instructions with successful DGETC's, LOADPC's or INHINT's will lock out interrupts indefinitely.

Because micro-interrupt routines are used for data transfers to and from the disk packs, it is important to avoid time-consuming state saving and restoring by micro-interrupt routines. With a single disk unit in operation, each additional micro-instruction in the interrupt routine reduces throughput by 1%. Consequently, considerable extra hardware has been put in to automate state saving and restoring during interrupts.

During non-interrupt instruction execution, duplicate registers for P, X, Y and EALUBC are loaded whenever the primary registers are loaded. During an interrupt, however, these duplicate registers remain frozen at their former values. The primary X, Y, and EALUBC registers are loaded from the duplicates by the IRET function. The first instruction of the interrupt routine is expected to save NPC on the stack by calling the interrupt routine, and to save Q in one of the register banks, say at SAVEDQ; this is why duplicates for Q and NPC are not provided. The final instruction of the interrupt routine must, to restore all these things, include:

IRET, RETURN, Q¬SAVEDQ, P¬P1

A duplicate register for KUNIT is also provided, but this is handled in a different way, discussed in section 7. Also note that AC and F are not duplicated (because interrupt routines only change F intentionally and don't use AC).

The interrupt system accepts 16 levels called interrupt requests (IREQi, i = 0 to 15). Interrupt i will occur after the execution of the current instruction if:

1. INT = 0 (i.e., no interrupt is in progress) or F1 = PREIRET in the previous instruction. Note that this implies that if PREIRET is not used, at least one non-interrupt instruction is executed after each interrupt routine is done, before the next one is started. To avoid this, the next to last instruction of the interrupt routine should specify F1 = PREIRET. The next instruction after one which has PREIRET must have IRET.

2. IENABLE (a flag register bit) = 1. When the interrupt system is disabled all interrupts have to wait.

3. No READ-MODIFY-WRITE is in its RM phase (i.e., has started to read but not started to write). The instruction which issues the RMW does not automatically inhibit interrupts, and the programmer must do so for that instruction.

4. The current instruction does not have F2 = INHINT or NPC→ or a successful DGOTO.

5. i is the largest number for which ARMi AND IREQi = 1, i.e. interrupts with bigger numbers have higher priority.

Changes in the value of IREQ, ARM or IENABLE do not affect the interrupt system until the second following instruction. Thus if instruction i clears IENABLE, an interrupt may occur (if the other conditions are satisfied) after i or after i+1, but will not occur after i+2.

2.2 Flag Register

The 36-bit flag register F serves as a repository for various flags in the processor and provides a number of general-purpose single bit flags which can be conveniently manipulated. Some bits of F are set or cleared by assorted events in the processor; these are mentioned in connection with the description of the relevant event and summarized in Appendix E. In addition, there are operations which work on all the bits of F:

NOT F	reads NOT F onto the bus
SETF [s]	sets the bits of F which are 1 in S [s]
SETFC[s, cond]	does SETF[s] if the branch condition is true (there will also be a branch if the condition is true)
CLEARF[s]	clears the bits of F which are 1 in S[s]
CLEARFC[s, cond]	does CLEARF[s] if the branch condition is true (there will also be a branch if the condition is true)
SETFB[s,cond]	does SETF[s] if the branch condition is true, CLEARF [s] if it is false (there will also be a branch if the condition is true).
SETSF[s]	sets bits of F selected by S[s][32-35] (i.e., K, J, H, and G) if (F AND S[s] AND -20B) #0.

All of these are specified by functions except NOT F, which is a bus source. SETSF[s] is also a F2. G, H, J and K are bits of F which can be tested by branch conditions; they can also be set in a variety of ways (see Appendix E).

For i = 0, 1, . . ., 35, if

1) the instruction contains SETF, CLEARF or SETFB, or if it contains SETFC or CLEARFC and the branch condition is true, or i ≥ 32 and it contains SETSF;
2) bit i of the word read from S is 1;
3) bit i of the flag register (F[i]) is being set or cleared independently by some other part of the processor;

then the new value of F[i] is the OR of the value it would have gotten from (1) and (2) above, and the value it would have gotten from (3) above.

3.0 Arithmetic/Logic Section

The arithmetic/logic section of the processor is shown in the left half of Figure 1. It consists of two register banks I and R with 32 registers per bank, two working registers P and Q, multiplexing for inputs to P and Q, and a 36-bit arithmetic/logic unit (ALU).

3.1 Register Banks

The two register banks are addressable from two five-bit fields LA and RA in the microinstruction, or from the low order five bits of the 8-bit X register, or from the 4-bit AC register. The source of a register bank address is determined by the appropriate A field as follows:

1) A = 0 or 1: take the address from X
2) A = 2 or 3: take the address from AC
3) A = 4: address register 4, but never write into it (see below)
4) A > 4: address register A

The above rules imply that registers 0-3 can only be referenced from X or AC, and register 4 can be stored into only when addressed via X or AC. For the left bank, if LA = 0 (2) and X [32-35]=0 (AC=0), the instruction will read the value 0 regardless of the contents of the register addressed and will not write into the register bank. This kludge is provided so that PDP-10 indexing and self-instructions can be emulated conveniently. RA=1 (3) is the same as RA=0 (2).

The X register can be loaded from
B[28-35]
B[14-17] (PDP-10 index field)
B[6-11] (PDP-10 byte pointer size field)

The AC register can be loaded from
B[32-35]
B[9-12] (PDP-10 AC field)

Both registers may be incremented and decremented with functions and may be read onto the bus (right justified). X[30-35] may also be read onto the bus left-justified (i.e., into B[0-5]); this puts it in the PDP-10 byte pointer position field. Two branch conditions exist to test the sign of X. The value of X (but not AC) is preserved across an interrupt.

In each instruction it is possible to read from or write into (but not both) the left register bank, and independently to do the same with the right register bank. The decision on whether to read or write is made as follows. If the register bank is

addressed by PS or QS, it is read. Otherwise, it is written unless the microinstruction addresses register 4, in which case nothing is done. Note that LA=0 or 2 may override this for the left bank if register 0 is addressed by X[32-35] or AC.

3.2 P and Q Registers; Cycle and Mask

The multiplexers on the inputs to P and Q are under control of two fields PS and QS in the microinstruction. The possible inputs for the working registers selected by these fields are given in Tables 2 and 3. P and Q are always loaded with the data specified by these tables with two exceptions; P is not loaded if F1 = LDPALUH AND ALU0=0; P0 is not loaded if F2=ASHOVF.

When P is loaded from anything except B, P1 or ALU RSH 1 it is possible to mask the input with 2**n - 1, i.e., keep the rightmost n bits of input and zero the rest. This action is selected by one of four functions:

Function	N
SAMASK	NOT SA
BAMASK	NOT BA
AMASK	AF (limits N to < 40B)
XMASK	X register

where the mask length n = MAX(36, N mod 64). If F1 is not one of these four, no masking takes place.

Note that the mask and PS features allow an arbitrary field to be extracted from P (or Q, using RCYQQ or NOTALU, using RCYNOTALUQ) and put into P right justified. The field can be specified either by the instruction (using one of SA, BA and AF) or by the X(length) and Y(right cycle required) registers.

F2=ASHOVF, in addition to inhibiting the loading of P0, sets the flag register bit OVF to 1 if P0≠P1; the intended use is to set OVF if a left shift would have changed the sign of P. There are branch conditions (QODD, QEVEN) to test the bottom bit of Q at the start of the instruction. They are illegal if Q was loaded from a slow source in the last instruction.

In normal mode (INT = 0), both P and P1 are loaded when loading of P is specified by the instruction. In the interrupt routines (INT=1), the loading of P1 is inhibited. P1 thus preserves the contents of P across the interrupt routine. The last instruction of the interrupt routine should therefore have PS = P1 as well as IRET.

3.3 Arithmetic and Logic Operations

The ALU can compute all 16 Boolean functions of P and Q as well as a number of arithmetic functions. Its operation is controlled by a 5-bit field in the instruction called AF. The values of AF which produce the various ALU functions are specified in Table 4.

The arithmetic functions (AF ≥ 20) are affected by the value of CARRYIN, which is 0 unless one of the function fields selects 1 (F1 or F2=CARRY1) or J (F1=CJ&SJC).

In addition to the 36-bit result specified by Table 4, the ALU provides three additional bits for the arithmetic functions starred in Table 4.

ALUC0 is the carry out of bit 0 from the twos-complement add specified in parentheses in Table 4.

ALUC1 is the carry out of bit 1

OVERFLOW is ALUC0 ≠ ALUC1. It is 0 if the 36-bit twos-complement result correctly represents the specified function, 1 if the result is wrong by $\pm 2^{35}$

The function SETOVPC01 sets flag register bits PC0 and PC1 to the values of ALUC0 and ALUC1 respectively and ors OVERFLOW into flag register bit OVF. The function CJ&SJC sets J to ALUC0. The function SETHOVF sets H to ALUC0*ALUC1.

The value of the 36-bit ALU output relative to 0 is stored in BALUBC and may be tested by a branch condition in the next instruction. ALU8 (for PDP-10 floating point normalization) and B0 are also stored in BALUBC and may be tested. This information is automatically preserved across interrupts. If INT=0 and F1=FRZBALUBC, BALUBC is frozen at its previous value rather than being updated to reflect the results of the current instruction.

3.4 Communication with the Bus

The arithmetic/logic section communicates with the rest of the processor via the bus (aside from flag bits and branch conditions). As mentioned above, X and AC can be loaded from or read onto the bus, and P or Q can be loaded from the bus. Loading of P and Q is controlled by PS and QS as described above and does not require the destination field. Note that P and Q are always loaded so it is the programmer's responsibility to have PS select P and QS Q when he does not wish the values to change. In addition, Q and the ALU result may be read onto the bus by specifying them as sources, and there is a function READALU to or the ALU result with the bus value specified by the source field. Note that the ALU is a slow bus source.

PS (octal)	P Input	Notes (see next page)
0-46	PQ RCY[0-46]	1
47	B	cannot be masked
50	P1	cannot be masked
51	ALU	
52	ALU ARSHC 1 (P0¬ALUC0)	cannot be masked
53	L LSH[3]	4
54	L LSH[2]	4
55	L LSH[1]	4
56	L	
57	L RSH[1]	4
60	L RSH[2]	4
61	L RSH[3]	4
62	R LSH[3]	4
63	R LSH[2]	4
64	R LSH[1]	4
65	R	
66	R RSH[1]	4
67	R RSH[2]	4
70	R RSH[3]	4
71	PQ LCY[3]	1
72	PQ LCY[2]	1
73	PQ LCY[1]	1
74	unused	
75	PQ RCY[44-Y]	1,2,3: Illegal if Y was loaded on the previous instruction, or if INT=1. BEWARE.
76	PQ RCY[Y]	1,2: Illegal if Y was loaded from a slow source on the previous instruction.
77	unused	

Table 2: P Input Selection

Notes:

1. PQ is a 72-bit number which can have one of the following values:

Condition	left 36 bits	Right 36 bits
F1 or F2=RCYQQ	Q	Q
F1 or F2=RCYØQ	Ø	Q
F1=RCYNCTALUQ	NCT ALU	Q (must have AF<2ØB)
otherwise	P	Q

The resulting P input is the leftmost 36 bits of the cycled 72-bit number.

2. Also sets H to (Y > 44B). If Y > 44B, then let C = (Y IF PS = PQ RCY Y ELSE 44B-Y MOD 1ØØB IF PS = PQ RCY44B-Y). These functions cause the P input selection to be accomplished as if Y and 77B or (44B-Y) AND 77B were in the PS field to begin with. Note that if Y can contain a value causing either L or R to be read into P it will not be possible to write into L or R during the same microinstruction. BEWARE.

3. Note that RCY44B-Y is not the same as LCY Y, since it is also necessary to exchange P and Q.

4. Zeros are shifted into the vacated bit positions.

Table 2: P Input Selection (continued)

QS	Q Input	Notes
_0	L	
_1	Q LSH 1	Q35←(ALU0≠G) IF F1=Q35ALUG ELSE Q0 IF * ELSE 0
_2	ALU	
_3	Q RSH 1	Q0←F35 IF F2=ASHOVF ELSE Q35 IF * ELSE 0
_4	R	
_5	Q	
_6	R RSH 1	Q0←ALU35 IF PS=ALU RSH1 ELSE R35 IF * ELSE 0
_7	B	Q is a slow sink if the next instruction has BC=QODD or QEVEN

* F1 = RCYQQ or F2 = RCYQQ or F1 = RCYNOTALUQ

Table 3: Q Input Selection

AF	Result	AF	Result (add 1 if CARRYIN = 1)
20	NOT P	0	P - 1
21	NOT P AND Q	1	x
22	NOT (P OR Q)	2	x
23	0	3	2P
24	NOT P OR Q	4	P AND NOT Q - 1
25	Q	5	x
26	P = Q (bitwise)	6*	P - Q - 1 (P + NOT Q)
27	P AND Q	7	(P AND NOT Q) + P
30	NOT (P AND Q)	10	P AND Q - 1
31	P ≠ Q (bitwise)	11*	P + Q (P + Q)
32	NOT Q	12	x
33	P AND NOT Q	13	(P AND Q) + P
34	1 (all bits)	14	-1 (twos complement)
35	P OR Q	15	x
36	P OR NOT Q	16	x
37	P	17	P

* carry and overflow outputs are valid

Table 4: ALU Functions

4. Local Memories

The processor physically contains three 1024 word memories with 18 bits/word (plus parity). These are logically arranged as two 512 word x 36-bit memories called the scratchpad (S) and the dispatch memory (D), and an 18-bit memory called the MAP. Since they are physically parts of the same memory, S and D cannot both be referenced in the same instruction. All three memories can be addressed from the 9-bit Y register, can read data onto the bus, and can store data from the bus. They are all slow sources and sinks.

There are functions to increment and decrement Y and to increment it by 4, and branch conditions to test its sign. Y can be read onto the bus (right justified) and can be loaded from a number of places:

```
  -Y
  E[27-35]
  B[18-26]  (page number)
  E[0-8]  (PDP-10 opcode or floating-point exponent)
  B[0-5]  (PDP-10 byte pointer position field)
_ 400B + B[33-35]*20B     (converts a disk unit number on the
                          bus into the address of a 16-word
                          table for each unit in the upper
                          half of S)
_ (B[18], B[28-35])  (shift count)
```

Y is a slow sink if the next instruction contains PS = PQ RCY Y. The value of Y is preserved across an interrupt.

4.1 Scratch Pad Memory (S, SM)

Unlike the other local memories, this one can be addressed from the instruction as well as from Y. The 8-bit SA field is used for this purpose. If it is < 20B, it is or'ed with Y to produce the S address; otherwise SA is the address. This means that only locations 20B-377B can be referenced directly from the instruction without using Y.

In addition to being read onto the bus, the data from S may independently be sent to F, where they perform various useful functions (see section 2.2). In addition to the usual source and destination values to put S onto the bus or load it from the bus, there are also functions READS and LOADS to do those things. The READS function or's S with whatever is put on the bus by the source field. Note that D and S cannot be referenced in the same instruction.

4.2 Dispatch Memory (D, DM)

This memory is physically the top 512 words of a 1024 word memory of which S is the bottom 512 words. As a result, it behaves exactly like a second copy of S which is selected instead of S when D is the source or destination or when F2=USED. Thus it can be addressed from SA and is sent to F just like S. The READS and LOADS functions apply to it, but make no sense since D can only be selected by source or destination.

D is intended to be used to hold three 11-bit microcode addresses and a flag for each of the 512 PDP-10 opcodes, but nothing in the processor hardware constrains it in this way.

4.3 Map Memory (MAP, MP)

Since this memory has 1024 18-bit words, it needs a 10-bit address. The Y register is used for the bottom 9 bits. The top bit, which in the intended use selects the user map (1) or monitor map (0), is taken from the current user mode (CUM) bit of F. In order to facilitate the selection of user or monitor map according to the Tenex rules, an instruction in which the function is one of the following provides the indicated value as the top bit of the MAP address, and also sets CUM to that value (XCTi are F register bits):

Function	Value of top bit of MAP address
IREF	CUM OR XCT0
RREF or RMWREF	CUM OR XCT1
BIREF	CUM OR XCT2
WREF	CUM OR XCT3

The functions also set H to the XCT bit which they reference. Note that the REF functions also set the G flag (see Appendix E), modify MAR, and start memory references (see section 5). They do not use the bus. Note also that MAPVAL sets CUM to UM.

To facilitate clearing the MAP, which must be done every time the system switches users, there is a destination MAP4L which initializes four registers simultaneously from E[18-35]; the four are the register addressed by (CUM, Y) AND 1774B and the three following ones. This destination also increments Y by 4.

4.4 Instruction Memory (I, IM)

The instruction memory I may be read and written by the following kludge.

To read;
    B ← address, LOADPC;
    P ← I, DGOTO[.+1];
    next instruction must have a successful CALL, GOTO or RETURN

Note that I can be read out only into P; it goes over the bus, but so slowly that it cannot be sent to any other destination.

To write:
    B ← address, LOADPC
    I ← bus, DGOTO[.+1];
    next instruction must have a successful CALL, GOTO or RETURN

If the instruction which references I has F2=INHINT, I[0-35] is referenced; otherwise, I[36-71] is referenced. During the cycle after the reference to I the instruction being executed is the one which was referenced, but some special logic prevents this instruction from doing anything.

5.0 Processor Memory Interface

The memory interface used by normal (non-interrupt) microprograms consists of a 40-bit data register (MDR), a 21-bit address register (MAR), and circuitry to implement the request-response protocol of the main memory system. The memory interface allows the processor to make read, write, and read-modify-write references of several types, some of which include access checking based on the access permission bits received from the MAP memory. The memory interface suspends activity in the processor under conditions in which a microinstruction would yield erroneous results if allowed to execute. The period during which the memory interface is active and in which the processor will be suspended by microinstructions which reference the memory interface is discussed in section 5.2.

The memory interface uses five bus destinations, three of which have side effects other than simply loading registers:

READ¬: MAR[15-35] ¬ B[15-35], start memory read

WRITE¬: MAR[15-35] ¬ B[15-35], start memory write
The program must load MDR with the data to be stored within two instructions after this destination is invoked. The micro-instruction containing WRITE¬ must contain F2=INHINT if the MDR has not been loaded. INHINT must also be set in the instruction after WRITE¬ if MDR is not loaded until two instructions after WRITE¬. However, the instruction which loads MDR does not have to have INHINT since an interrupt after that instruction causes no problems.

RMW¬: MAR[15-35] ¬ B[15-35], start RMW
The program must explicitly disable interrupts by INHINT only during the irstruction which uses this destination. Once the irterface has begun the RMW, interrupts will be automatically disabled until the program has initiated the store portion of the RMW. The store portion must be begun within three microseconds after the RMW¬ or a memory error will occur (section 5.2)

MDR¬: MDR[0-35] ¬ B[0-35]
If a memory write is at a point at which the memory expects MDR to be stable, the processor is suspended until the reference is completed.

MAPVA¬: MAR[15-35] ¬ B[15-35], G¬(((S IF F1 = ACFS ELSE ALU) AND 777760)=0), Y ¬ B[18-26], CCM¬UM, X¬B[28-35]

The interface uses the following three bus sources:

MDR: B[0-35] ← MDR[0-35]
MDRL: B[32-35] ← MDR[36-39]
The processor is suspended on read or RMW if the memory has not yet supplied data.
MAR: B[15-35] ← MAR[15-35]

The interface uses the following functions:

MDRL←: MDR[36-39] ← B[32-35]
The comment under MDR← applies.

WRESTART: This function starts the store portion of an RMW reference. Interrupts are inhibited between the execution of an instruction containing INHINT, RMW← and the execution of the instruction after the one containing WRESTART. The remarks on loading MDR and inhibiting interrupts which apply to the WRITE← destination also apply here, except that the instruction containing WRESTART does not have to use INHINT.

RREFDXK, WREFDXK, RMWREFDXK, XREF, IREF, PIREF, RREF, WREF, RMWREF:
These functions load MAP[15-26] from the low order bits of the map memory via its direct outputs, and conditionally start the specified type of reference. The conditions under which the processor is halted when these functions are executed and the rules about loading MDR and inhibiting interrupts are identical to those which apply on a normal reference of the same type (WRITE← for WREF and WREFDXK, RMW← for RMWREF and RMWREFDXK, READ← for the others). In addition, these functions check the legality of the reference against the access permit bits from the map memory. A reference is legal iff:

RREFDXK	MAP[18] = 0
RREF	MAP[18] = 0
IREF	MAP[18] = 0
PIREF	MAP[18] = 0
WREFDXK	MAP[19] = 0
WREF	MAP[19] = 0
RMWREFDXK	MAP[18] = MAP[19] = 0
RMWREF	MAP[18] = MAP[19] = 0
XREF	MAP[20] = 0

If the specified reference type is legal, and if G = 0, the interface is started. If the access is illegal or if G = 1, the interface is not started.

and G is set. MAPVA→ leaves G = 1 iff an AC reference is detected.

To allow the processor to be debugged in single step mode, the memory interface has two additional features. In single step mode, the instructions which normally start RMW references start READs, and WRESTART starts a WRITE. When WRESTART is issued or WRITE is issued and F2 = INHINT, the actual store is deferred until an instruction is executed with INHINT=0.

5.1 Disk Memory Interface

The disk memory interface is similar to the processor interface, but is considerably simpler, since it deals only with physical addresses and has a more limited command repertoire. The disk memory interface should be used only by interrupt routines, and is provided principally to avoid saving the state of a memory interface during interrupts, rather than to increase memory bandwidth.

The interface contains a 40-bit (plus parity) data register, KMDR, and a 21-bit address register KMAR. Several of the interface control operations transfer data directly from the 40-bit disk data register KDATA.

The interface has timing and register loading considerations similar to those in the processor interface; however, the comments concerning the inhibiting of interrupts do not apply, since this interface is used only by interrupt routines.

The disk memory interface uses the following functions, with actions as specified:

KWDATA: KDATA[0-35] → B[0-35] → KMDR[0-35],
KDATA[36-39] → KMDR[36-39]

KRDATA: KMDR[0-35] → B[0-35] → KDATA[0-35]
KMDR[36-39] → KDATA[36-39]

KMDRL→: KMDR[36-39] → B[32-35]

KWRESTART: Start the store portion of an RMW reference.

If a bus source is specified in an instruction which uses KRDATA or KWDATA, the data from the specified source will be merged on the bus.

The following bus sources are used by the disk memory interface:

KMDR: B[0-35] → KMDR[0-35]

KMAR: B[15-35] 993

KMDRL: B[32-35] → KMDR[36-39] 995

and the following bus destinations: 997

KWRITE→: KMAR[15-35] → B[15-35], start write; 999

KREAD→: KMAR[15-35] → B[15-35], start read; 1001

KRMW→: KMAR[15-35] → B[15-35], start RMW; 1003

KMDR→: KMDR[Ø-35] → B[ℓ-35]; 1005

6.0 Maintenance Interface

The maintenance interface has two independent functions. The first is to facilitate 16-bit data transfer between the NOVA and any of 256 external devices, several of which are used by the microprocessor; the second is to process interrupts from the MAXC system used for interprocessor communication and error reporting.

6.1 NOVA Portion

At the NOVA, the maintenance interface consists of two sections, one for data transfers and one for interrupt handling. The data transfer portion of the interface consists of an 8-bit external device address register AD, and gating to allow bidirectional data transfers. The address register is loaded from the low order 8 bits of AC with DOB AC, MAINT. This address is sent to all external devices, and causes them to place data on the 16-bit bus if they are input devices, or prepare to receive data if they are output devices. Due to timing constraints, a unique external device address is associated with an input device or an output device, but not both. To output 16 bits from AC to the device addressed by AD, DOA AC, MAINT should be executed. Similarly, DIA AC, MAINT inputs 16 bits from the (input) device addressed by AD. Doing input from a device designed to accept output results in Ø, and doing output to an input device has no effect. Since all I/O activity occurs within the span of one NOVA instruction, the normal BUSY and DONE logic associated with NOVA I/O devices is not present, and the START and CLEAR functions have no effect. It is possible to send a single pulse to the device addressed by AD by executing NIOP MAINT. Interpretation of this signal varies with the device.

The second portion of the maintenance interface receives two communication signals from the remainder of the system, and intercepts two error signals, FATAL ERROR (FER) and NON-FATAL ERROR (NFER). The latter two signals are generated by various portions of the system when errors are detected. NFER is currently used only to detect corrected single-bit failures in the memory system. It has no effect except to cause a Nova interrupt.

The fatal error signal is generated when an uncorrectable error occurs at the memory or at the processor. All devices in the system sample this signal, and halt when they detect it. The NOVA must therefore take action to restart the system when this interrupt occurs.

The four sources of interrupts, FER, NFER, CCMA, and CCMB, are merged to cause a single NOVA interrupt. This interrupt may be masked off in the normal way with MSKO, using bit 6. The single interrupt is connected to the DONE flag for MAINT, so that the state of these interrupts may be tested while they are masked out (however, the functions which normally set and clear DONE have no effect). The four interrupts may be enabled and disabled

separately by executing DOC AC, MAINT with a four-bit mask in AC. The bits are:

12	FER	
13	NFER	
14	COMA	(MAXC processor to NOVA signal)
15	COMB	(unused)

One's in AC disable the interrupt. After a given interrupt is disabled, it may occur once more providing it was pending at the time it was disabled.

When DIC AC, MAINT is executed, a four-bit mask is read into AC, with one's corresponding to the source(s) of the interrupt (the top 12 bits contain garbage). These flags remain set until explicitly cleared with NIOC MAINT.

The correct sequence of events in servicing the single maintenance interface interrupt is:

1. Read interrupt flags with DIC AC, MAINT.
2. Disable maintenance interrupts and clear the flags with DOC AC, MAINT (AC=17).
3. Service the interrupts as determined by the flagword.
4. Re-enable the maintenance interrupts with DOC AC, MAINT (AC=Ø).
5. Re-enable NOVA interrupt (INTEN) and return.

In servicing the FER and NFER interrupts it is necessary to poll devices capable of causing these interrupts to determine the source. This is described in detail by the documentation for each device.

6.2 Processor Section

The processor section of the system maintenance interface consists of a number of registers which may be loaded from the Nova, allowing it to control the operations of the processor. These registers are (for exact format, see Appendix F):

a) A 64-bit register, PIR, which holds a single microinstruction (not including the branch address field) which can be executed under control of the Nova.
b) A 36-bit bus data register, BR, which can be gated onto the processor bus under control of the Nova.
c) A multiplexer capable of returning 64 bits of data to the Nova. 36 bits are used for the processor bus, the

remainder return status conditions. The status bits returned are the state of the RUN flip flop, the state of the two memory interfaces, and the parity error flags. When any parity error occurs, FER (fatal error) is set throughout the system, causing all processors (including the one which caused the error) with the exception of the Nova, to halt. The Nova is interrupted, and will be expected to deal with the error and restart the processor. The parity error flags are reset by ERRESET.

d) A 16-bit control register, CR, which may be loaded from the Nova.

The bits of the control register are latched at the processor except for the ones starred in the table below. Starred bits generate signals which last for the duration of the EOA.

EIC	Enable instruction controlled changes
EB	Enable changes in BALUBC
EIMA	Enable changes in IMA
EPC	Enable changes in PC
	(The four bits above enable various flavors of clock in the processor.)
SS	(single step) If set, the RUN flipflop is cleared one cycle after it is set.
SETRUN*	Sets RUN. Run is cleared by SS and by various error conditions.
ERRESET*	Resets error conditions (parity, etc.) in the processor.
EFM	(execute from memory) If set, microinstructions are executed from the instruction memory. If clear, microinstructions are executed from PIR.
REGTOB	Causes the contents of the bus register to be placed on the processor bus.
INTON	E processor interrupts.
STROBE	Nova-to-MAXC processor strobe.

7.0 Disk Control

The disk interface consists of three bus destinations (KUNIT←, KSET←, and KCSET←), two bus sources (KUNIT and KSTAT), three functions (KRDATA, KWDATA and KNEWCOMM), and some interrupt machinery. The letter 'K' has been chosen to preface all disk register names.

The disk controller hardware divides into two parts. The first part, called the common controller, provides services to all disk units. The KUNIT register, interrupt control, write oscillator, bus interfacing, and memories which implement the KDATA registers are all part of the common controller.

The second part, called the unit controller, is replicated for each disk unit. Incorporated in the disk unit controller are registers which respond to KSET←, KCSET←, and KSTAT, logic to control the transfer of data bytes to and from the common controller, generate interrupt requests and detect error conditions, and control the sequencing of commands to the disk unit, and a phase-locked loop for disk data recovery. The design provides for one common controller interfacing with (up to) eight unit controllers, each of which in turn interfaces with one Century Data Systems 213 disk unit.

With each disk unit controller are associated five logical registers: a disk command register, a controller command register, a disk and controller status register, one input data register, and one output data register. These registers are logically connected to the above-mentioned bus sources and bus destinations if and only if the KUNIT register points to the designated unit controller.

During ordinary processing, the contents of the KUNIT register may be changed by using KUNIT← as the bus destination. However during the processing of an interrupt KUNIT is temporarily forced to point to the highest-priority disk unit which is requesting the highest priority interrupt. During this period, the pushed-down KUNIT register can be changed by using KUNIT← as the bus destination; however this change will not be reflected in the KUNIT bus source until after interrupt processing is complete. In practice, one would probably not want to use KUNIT← as the bus destination during interrupt processing. However, reading the KUNIT bus source during an interrupt routine is the only way of finding out with what unit the interrupt is to be associated.

The following paragraphs describe the effect of KCSET←, KSET←, KSTAT, KWDATA, KRDATA, and KNEWCOMM upon the unit selected by KUNIT. No other units are affected.

The KCSET¬ destination modifies the command register of the unit controller according to various bus bits (see Table 5). This permits the processor to alter the unit's processor interrupt mask, to reset interrupt conditions, and to reset error conditions.

The KSET¬ destination performs the action specified above for KCSET¬, and in addition loads the disk command register from the bus. These data are latched by the disk command register and presented to the disk unit for a prescribed time interval.

The KNEWCOMM function (same as READS) is interpreted only in conjunction with the KSET¬ bus destination. It causes the unit controller to reset the command it is currently presenting to the disk unit before latching up the new command being issued by KSET¬. KNEWCOMM is required when setting the head register, resetting the head register, setting the cylinder register, and starting seeks. It should not be used at other times for fear of head select glitches and erase turn-off blasts.

The KSTAT source puts status bits from the disk unit and controller onto the bus. (See Table 6.)

The KWDATA function buffers data from B[0-35] and KMDR[35-39] for eventual writing on its disk unit. The bus data may be read into P or Q in the same microinstruction for checksum computation.

The KRDATA function places KDATA[0-35] onto the bus, loads KMDR[0-35] from the bus, and loads KMDR[36-39] directly from KDATA[36-39]. The bus data may be read into P or Q for checksum computation in the same microinstruction.

The details of the controller-disk file interface and a number of tedious programming details are discussed in Appendix G.

Bus Bit Position	Meaning
0	Enable/disable sector interrupts on channel 5B
1*	Load cylinder register from B[15-23]
2*	Load head register from B[18-23]
3	Interpret B[15-23] as a command and execute it
4	Enable/disable word interrupts or channel 13B (reading)
5	Enable/disable word interrupts on channel 12B (writing
6	Enable/disable word interrupts or channel 11B (dispatch)
7	Reset sector condition
8	Reset processor data late
9	Reset controller data late
10	Reset sector overflow
11*	Deselect/select this unit
12-14	---Unused---
15-23*	Disk drive bus, interpreted according to B[1-3]
24-35	---Unused---

* Interpreted only for KSET¬. Not interpreted by KCSET¬.

Table 5. KSET¬ and KCSET¬ Bus Interpretation

Bus Bit Position	Meaning
0	Index condition (comes up with sector condition. Stays up for one sector)
1	Unit unsafe (operator must take action)
2	Unit offline (illegal unit or operator must take action)
3	Unit not ready (= seeking if other stuff OK)
4	Seek has failed (very rare--restore and try again but probably a hardware failure).
5	Unit is read only (This is controlled by a manual switch, but the hardware will look at this switch only when the unit is deselected. This means that the software will have to deselect the unit before the effect of the operator throwing the switch will be received by the unit).
6	Controller not ready (set until previous command has been received by disk unit--about two usec)
7	Sector condition (sector interrupt request is held until it is dismissed, but the "sector condition" becomes true concurrent with the sector interrupt request and false at the second word time afterwards).
8*	Processor data late (microinterrupt serviced too late)
9*	Controller data late (hardware problems)
10*	Sector overflow (still reading, writing, erasing, or word-interrupting at sector pulse. Reading writing and erasing are turned off and no future word interrupts will be requested).
11	Unit deselected.
12-35	---Unused---

* Requires reset by KSET¬ or KCSET¬. Reading, writing, erasing, and word interrupting are prevented by any of these errors.

Note: All errors prevent writing inside the file.

Table 6: KSTAT Bus Data

Appendix A: Summary of Microinstruction Bits

Field	Size	Position	Meaning
EA	11	0-10	Branch address
ET	2	11-12	Branch type: GOTO, CALL, RETURN, DGOTO
BC	5	13-17	Branch condition (BC0 inverts the meaning)
LA	5	18-22	Left bank address: 0/1 = use X, 2/3 = use AC
RA	5	23-27	Right bank address: 0/1 = use X, 2/3 = use AC
PS	6	28-33	Select input to P
QS	3	34-36	Select input to Q
AF	5	37-41	ALU function
BS	5	42-46	Bus source
BD	5	47-51	Bus destination
F1	6	52-57	Function
F2	4	58-61	Second Function
SA	8	62-69	Scratchpad address: <20B = use SA CR Y
BRKP	1	71	Breakpoint
TRIG	1	72	Scope trigger
Total	72		

Appendix B: Summary of Branch Conditions

BC (octal)	Meaning	EC (octal)	Meaning	Reference
0	Always	20	Never	
1	Q odd	21	Q even	3.2
2	*ALU8 = 0	22	ALU8 ≠ 0	3.3
3	K = 1	23	K = 0	Appendix E
4	*ALU < 0	24	ALU ≥ 0	3.3
5	H = 1	25	H = 0	Appendix E
6	X ≥ 0	26	X < 0	
7	---	27	---	
10	ALU = 0	30	ALU ≠ 0	3.3
11	G = 1	31	G = 0	Appendix E
12	*B < 0	32	B ≥ 0	3.3
13	---	33	---	
14	*ALU ≤ 0	34	ALU > 0	3.3
15	J = 1	35	J = 0	Appendix E
16	Y ≥ 0	36	Y < 0	
17	---	37	---	

*Preserved across interrupts in BAIUBC

Appendix C: Summary of Primary and Secondary Functions

F1 (Octal)	NAME	MEANING
00	---	No action
01	IREF	MAPREF (XCT0, RP), E←XCT0
02	BIREF	MAPREF (XCT2, RP), E←XCT2
03	RREF	MAPREF(XCT1, RP), E←XCT1
04	RREFDXK	MAPREF(0,RP)
05	RMWREF*	MAPREF(XCT1, RP AND WP), E←XCT1
06	RMWREFDXK*	MAPREF(0, RP AND WP)
07	WREF**	MAPREF(XCT3, WP), E←XCT3
10	WREFDXK**	MAPREF(0, WP)
11	XREF	MAPREF(0, XP)
12	---	
13	LOADMDR	MDR[36-39]←E[32-35]
14	WRESTART***	Start the write cycle on a RMW.
15	---	
16	KMDR←	KMDR[36-39]←E[32-35]
17	KWRESTART	Start the write cycle on a KRMW.
20	KRDATA	B[0-35]←KDATA[0-35], KMDR[0-35]←E[0-35], KMDR[36-39]←KDATA[36-39]
21	KWDATA	B[0-35]←KMDR[0-35], KDATA[0-35]←E[0-35], KDATA[36-39]←KMDR[36-39]
22	SIGNOVA	Request NOVA interrupt
23	INCY	Y←Y+1
24	DECY	Y←Y-1
25	NEGY	Y←-Y
26	YKPTR←	Y←400E + E[33-35] * 20B
27	---	
30	INCX	X←X+1
31	DECX	X←X-1

* Must be accompanied by F2=INHINT

** Must be accompanied by F2=INHINT if MDR is not loaded by the end of the instruction.

*** The instruction after WRESTART must be accompanied by F2 = INHINT if MDR is not loaded until two instructions after WRESTART.

Note: MAPREF (umbit, permission) is CUM←CUM CR umbit, start memory reference if G = Ø and permission = 1, and G ← G CR (permission = Ø).

Appendix C: Functions (Continued)

F1 (Octal)	NAME	MEANING
32	INCAC	AC←AC+1
33	DECAC	AC←AC-1
34	SETF	F←F OR S
35	SETFC	F←F OR S IF EC is true
36	CLEARF	F←F AND NOT S
37	CLEARFC	F←F AND NOT S IF EC is true
40	SETFB	F←(F OR S IF EC ELSE F AND NOT S)
41	SETSF	Bits of F selected by S[32-35] are set to (F AND S AND -2**B)#0 (F[32-35] are K, J, H, and G)
42	CARRY1	Supplies input carry = 1 to ALU
43	CJ&SJC	Supplies input carry = J to ALU and sets J to ALUC0
44	SETHCVF	Sets H to ALUC0 ≠ ALUC1
45	SETOVPC01	PC0←ALUC0, PC1←ALUC1, CVF←(ALUC0 # ALUC1) OR CVF
46	RCY0Q	Change cycler input from FQ to 0Q
47	RCYNOTALUQ	Change cycler input from FD to [NOT ALU, Q)
50	RCYQQ	Change cycler input from FD to QQ. Change Q←R RSH1 to Q←R RCY 1. Change Q RSH1 & Q LSH1 into Q RCY1 & Q LCY1.
51	LDPALUH	Don't load P if ALU0=H
52	Q35ALUG	Modifies Q LSH1. See table 3.
53	READALU	Or ALU result with bus value specified by source
54	XMASK	Sets P mask to 2**[X AND 77B] - 1
55	SAMASK	Sets P mask to 2**[(NOT SA) AND 77B] - 1
56	BAMASK	Sets P mask to 2**[(NOT BA) AND 77B] - 1
57	AMASK	Set P mask to 2**AF - 1

60	READS	Or S with bus value specified by source.	1575 1576
	KNEWCOMM	Reset disk command lines if destination=KSET¬.	1577 1578
61	LOADS	Write bus into S	1580
62	ARM¬	ARM¬B[20-35]	1582
63	ARM	B[20-35]¬ARM, B[1-4]¬INTNO, B[0]¬INT	1585

Appendix C: Functions (Continued)

F1 (Octal)	NAME	MEANING
64	PREIRET	Promises return from interrupt after next instruction
65	IRET	Return from interrupt
66	FRZBALUBC	Prevent latched bus and ALU branch conditions from changing at the end of this instruction. Ineffective if INT = 1.
67	POP	Pop the stack. Must not accompany CALL or RETURN.

Appendix C (continued): Secondary Functions

F2 (Octal)	NAME	MEANING
_0	INHINT	Prevent an interrupt after this instruction.
_1	NPC¬	NPC¬B[24-35] and prevent an interrupt after this instruction.
_2		
_3		
_4	USED	Set the high bit of the SD memory address so that the address is in D rather than S.
_5	READS* KNEWCOMM*	Or S with data on bus. Reset disk command lines if destination=KSET¬.
_6	LOADS*	Load S from data on bus.
_7	WRESTART*	Start the write cycle on a RNW.
10	---	No Action
11	SETSF*	Bits of F selected by S[32-35] are set to (F AND S AND -20B)≠0 (F[32-35] are K, J, B, and C).
12	RCY0Q*	Change cycler input to 0Q.
13	CARRY1*	Supplies input carry = 1 to AIU.
14	ASHOVF	OVF¬ (P0≠P1) OR OVF, disable loading of P[L], Q0¬P35 on Q FSH 1.
15	RCYQQ*	Change cycler input to QQ. Change Q¬R RSH1 to Q¬R RCY1. Change Q FSH1 & Q LSH1 into Q RCY1 & Q ICY1.
16	POP*	Pop the stack.
17	ACFS	C¬(777760 AND S = 0). Overrides the usual setting of C by MAPVA¬.

*Also provided as a primary function.

Appendix D: Summary of Bus Sources and Destinations

NO. (Octal)	SOURCE	DESTINATION	MEANING
0	NULL	B←	None, Bus value is 0
1	X	X←	8-bit, X-register
2	Y	Y←	9-bit, Y-register
3	AC	AC←	4-bit, AC register
4	*MAP	*MAP←	18-bit, Map memory
5	*D	*D←	36-bit, Dispatch memory
6	*S	*S←	36-bit, Scratch pad memory
7	**I	*I←	Instruction memory, bits 0-35 if F2=INHINT, so 36-71 otherwise
10	MDR	MDR←	Processor memory data register
11	MDRL		Extra 4 bits of memory data
11		READ←	MAR←E and start read
12	MAR		Memory address register
12		RMW←	MAR←E and start read-modify-write
13	---	WRITE←	MAR←B and start write
14	*KMDR	*KMDR←	Disk memory data register
15	*KMDRL		
15		*KREAD←	
16	*KMAR		Disk memory address register
16		KWRITE←	
17	---	KRMW←	
20	*KUNIT	KUNIT←	Disk unit in F[33-35]
21	*KSTAT		Put disk status on bus
21		KSET←	Both controller and file
22	---	KCSET←	Controller only
23	*NOT F		
23		ISPLIT←	X←B[14-17], G←(E[13]=0)
24	Q		
24		FSPLIT←	Y←B[0-8]
25	*ALU		
25		BSPLIT←	X←B[6-11], Y←B[0-5]

*Slow

**Very slow. I can only be sent to P register.

Appendix D: Bus Sources and Destinations (Continued)

No. (Octal)	SOURCE	DESTINATION	MEANING
26	STACK		B[25-35]← top entry of stack. Illegal if combined with CALL or STACK←
26		STACK←	Push stack twice, leaving B[12-23] on top and B[1-11] next to the top. G←G or (B[0]=0)
27	NPC		11-bit program counter
27		MAPVA←	Y←B[18-26], MAR[27-35]←B[27-35], G←(((S IF F2 = ACFS ELSE ALU) AND 777760B)=0), CUM←UM, X←B[28-35]
30	---	*MAP4←	T←((CUM,Y) AND 1774B, Y←Y+4, MAP[T]←MAP[T+1]←MAP[T+2]←MAP[T+3]←B[18-35]
31	XTOP		B[0-5]←X[30-35]
31		XSPLIT←	Y←B[0-8], AC←B[9-12], X←B[14-17], G←H←(B[13]=0)
32	---	YSHIFT←	Y[27]←B[18]. Y[28-35]←B[28-35]

*Slow
**Very slow. IM can only be sent to the P register.

Appendix E: Summary of Flag Register Bits

BIT	NAME	SET/USED
0	OVF	Turned on by SETCVFC01 if CVERFICW, by ASHOVF if P0≠P1
1	PC0	Set to ALUC0 by SETCVFC01
2	PC1	Set to ALUC1 by SETCVFC01
3-4		No special uses
5	UM	Used to set CUM by MAPVA¬
6-13		No special uses
14-17	XCT0-XCT3	Used to set MAP address and CUM by some REF destinations
18-26		No special uses
27	CUM	Current user mode. See 4.3
28		No special use
29	IENABLE	Interrupt enable
30		No special use
31	NOVA	Set by Nova to signal processor
32-35		On SETSF, the flags selected by ones in S[32-35] are set to (F AND S AND (-20B))≠0
32	K	Used by K=0 branch condition
33	J	If F1 = JCCARRYBC set to ALUC0 and used as CARRYIN Used by J=0 branch condition.
34	H	Set by XSPLIT¬ to (B[13]=0), by some REFS to the selected XCT bit, by SETFCV to ALUC0≠ALUC1, by PS = PQ RCY Y or PQ RCY 44-Y to (Y > 44B). Used by LOADPALUH and H=0 branch condition.
35	G	Set by XSPLIT¬ and ISPLIT¬ to (B[13]=0), by STACK¬ to G OR (B[0]=0), by MAPVA¬ AND NOT ACFS to (ALU AND 777776=0), by ACFS to (S AND 777760B)=0, by REFs to G OR

(map violation). Used by Q35ALUG and G=Ø 1838
branch condition.

Appendix F: Maintenance interface devices used in the processor

SMI device address (I=input to Nova) (O=Output to Nova)	Name	Bit name and significance
210 O	CR	Control register 0:EIC 4:SS 8:REGTOB 1:EB 5:SETRUN 9:INT 2:EIMA 6:ERRESET 10-15:UNUSED 3:EPC 7:EPM
213 O	BR0	Bus register; gated to processor bus when REGTOB=1. Nova bits 0-15 correspond to processor bits 0-15.
212 O	BR1	Bus register; Nova bits 0-15 are processor bus bits 16-31.
213 O	BR2	Bus register; Nova bits 8-11 are processor bits 32-35.
217 O	PIR0	Pseudo-instruction register Nova Bit: Processor instruction field 0-1 BT[00-01] Branch type 2-6 BC[00-04] Branch condition 7-11 LA[31-35] Left bank address 12-15 RA[31-34] Right bank address
216 O	PIR1	0 RA[35] Right bank address 1-6 PSEL[00-05] P multiplexer select 7-9 QSEL[00-02] Q multiplexer select 10-14 ALUF[00-04] ALU function 15 SRC[00] Bus source
215 O	PIR2	0-3 SRC[01-04] Bus source 4-8 DEST[00-04] Bus destination 9-14 FCN[00-05] Primary functions 15 FCN2[00] Secondary functions
214 O	PIR3	0-2 FCN2[01-03] Secondary functions 3-10 SA[28-35] Scratchpad address 11 SCOPE TRIGGER For debugging
200 I	RUN	Bit 8 of this register = 0 indicates RUN=1 in the processor. All other bits are undefined.

203 I	BØ	Processor bus bits 0-15 are input to Nova bits 0-15.	1898 1899
202	B1	Processor bus bits 15-31 are input to Nova bits 0-15.	1901 1902
201	B3	Processor bus bits 32-35 are input to Nova bits 8-11.	1904 1905

Appendix G. More Than You Really Wanted to Know About Disk Control

Part I: The Disk Drive to Controller Interface

The disk drive (Century Data Systems model 213 or 215) communicates with its unit controller over a MAXC cable. Disk commands, disk status, and data bits travel endlessly back and forth over this cable. Signal paths consist of twisted pairs, one grounded at both ends, the other driven with an open collector TTL gate at one end and resistively terminated at both ends. The signal paths are low true or low active. Thirteen signal paths are reserved for commands from the unit controller to the disk drive. What follows is a modified excerpt from the CDS 215 Interface Specification. Note that this section does not describe disc control from the viewpoint of microprograms. It discusses the signals to which the unit controller must interface. Microprogramming considerations are in part II of this appendix.

Module Select
: Selects the disk drive attached to the control unit and enables it to accept signals presented over the bus and tag lines and to generate signals on the status lines.

Drive Bus (Ø-8)
: Nine lines to transmit address and control information as determined by one of three tag lines:

		TAG LINES	
Line Name	Control	Set Cyl	Set Head
Drive Bus Ø		Cyl 256	
Drive Bus 1	Wr Gate	Cyl 128	
Drive Bus 2	Rd Gate	Cyl 64	
Drive Bus 3	Seek St	Cyl 32	
Drive Bus 4	Rst Hd Reg	Cyl 16	Hd Add 16
Drive Bus 5	Erase Gate	Cyl 8	Hd Add 8
Drive Bus 6	Sel Hd	Cyl 4	Hd Add 4
Drive Bus 7	Rtn ØØØ	Cyl 2	Hd Add 2
Drive Bus 8	Hd Adv	Cyl 1	Hd Add 1

Set Cylinder Tag
: Indicates that the cylinder number on the bus lines is stable and loads it into the cylinder register. This function does not initiate a seek operation.

Set Head Tag
Indicates that the head address is stable on Bus lines 4 through 8 and loads it into the head register. This function must be preceded by a reset head function, since the unit internally OR's the new head address with the previous one.

Control Tag
Indicates that bus data is stable and contains control information. The signals on each of the nine bus lines are defined as follows:

Bus 0 (No Function)

Bus 1 (Write Gate)
Specifies that data on the Write Data line from the unit controller is to be written on the currently selected cylinder of the Disk Drive.

Bus 2 (Read)
Specifies that the data on the selected cylinder and head be transmitted over the Read Data line to the unit controller.

Bus 3 (Seek Start)
Provides a pulse which starts a seek operation. The seek operation causes the head carriage mechanism to move from its present address to a new address. This function normally follows a "set cylinder" operation.

Bus 4 (Reset Head Register)
Provides a pulse signal to clear the Head Address Register (Head 00 condition).

Bus 5 (Erase Gate)
Enables the selected Head to Straddle Erase recorded data. To ensure a complete Straddle Erase of the guard bands of a data record, the Erase Gate must remain active for 20 us ± 10% after the Write Gate is inactive.

Bus 6 (Select Head)
Select the head addressed by the Head Address Register.

Bus 7 (Restore)
Initiates arm motion to Cylinder 000. The Disk Drive will generate a "seek complete" signal when done.

Bus 8 (Head Advance)
Provides a pulse to increment the Head Address Register.

In addition, two other signal paths (Sequence Pick In and Controlled Ground) are used to turn the disk unit rotation on and off remotely, and one non-standard signal path (Termination

Power) provides +5vDC, 0.9 amps, to a resistive termination block located at the disk unit.

There are ten signal paths in the cable reserved for status information from the disk unit to the controller:

Module Selected
Indicates the disk unit is selected and not unsafe. The Module Selected signal occurs within 500 nsec. from the leading edge of Module Select.

Gated Attention
Indicates that either a power-on sequence, a seek command, or a restore is completed. This signal is reset by the read gate.

Drive Ready
Indicates that a selected seek command has been successfully completed and that the Disk Drive is ready to read or write.

On-Line
Indicates the heads are extended and the Disk Drive is ready to be operated by the control unit.

Sector
A pulse on this line indicates the beginning of a sector. Pulse width is 80 us ± 20%. Jitter should be less than 10 microsec.

Index
Is set on one and only one sector of a revolution. The pulse width is 80 us ±20%. Index is delayed from sector by approximately 120 microsec.

Drive Unsafe
A signal on this line indicates the selected Disk Drive is unsafe. Within the Disk Drive, safety circuits are provided to protect the recorded information.

The following conditions inside the Disk Drive generate the unsafe signal.

(1) DCUNSAFE - Any dc power supply output low.

(2) HDUNSAFE - A head unsafe condition is: Controller initiates a Select Head but no head becomes selected or more than one head becomes selected or no select head is initiated from the controller but a head becomes selected.

(3) RDY/ . (ERGATE + WRTGATE) - Disk Drive not ready for operation but Write or Erase Gates raised by controller.

(4) RDGATE . (ERGATE + WRTGATE) - Read gate and write gate or erase gate raised by controller.

(5) IWON/ . IEON - Write current off and erase current on for longer than 60 microseconds.

(6) IEON/ . IWON - Erase current off and write current on.

(7) ERGATE . IEON/ - Erase gate up and erase current off.

(8) IEON . ERGATE/ - Erase gate down and erase current on.

(9) WRTGATE . IWON/ - Write gate up and write current off.

(10) IWON . WRTGATE/ - Write gate down and write current on.

(11) SEEKUNSAFE - Drive oscillator low, Heads extended and not up to speed, or SEEKERROR during forward motion of FIRSTSEEK or RESTORE.

(12) AIR FILTER SYSTEM FAULT.

Seek Incomplete
Indicates that the Drive has been directed to a non-existent cylinder or has failed to generate Seek Ready within 1 sec. of a "Seek Start."

End of Cylinder
Indicates that the head address register in the disk drive has advanced from head address 19 to 20 in response to a Head Advance command.

Write Current Sense

This signal indicates that normal write current is present. Write current sense is active within 10 us from the leading edge of Write Gate.

The "Gated Attention," "End of Cylinder," and "Write Current Sense" signals are not present in the disk status word presented to the MAXC processor because they are irrelevant to MAXC's mode of operation or because they are redundant. In addition, three other signal paths (Sequence Power, Sequence Pick Out, and Heads Extended) are used to sense the rotational status of the disk unit for AC and DC power sequencing purposes.

There are two signal paths (implemented with coaxial cable) which carry serial data to and from the unit controller. During write operations, negative-going edges on the write data line trigger a complementing flip-flop in the disk unit, whose state is written by the selected read-write head on the disk. Similarly, during a read operation, a flux reversal sensed by the selected read-write head (corresponding to a change in the state of the complementing flip-flop during writing) fires a one-shot in the disk unit which sends an 80 nanosecond negative-going pulse onto the read data line.

The following table summarizes many of the important parameters of the Century Data Systems 213/215 disk drive:

Maximum Head Positioning Time	55 ms
Maximum Track-to-Track Positioning Time	10 ms
Maximum Rotational Latency	25 ms
Recording Method	Double Frequency, bit serial
Recording Surfaces Available per Drive	20
Number or Recording Heads per Drive	20 (00-23 B octal); one per disk surface
Type of Head	Straddle Erase
Disk Rotational Speed	2400 RPM +2%
Number of Cylinders per Disk Pack	406 (000-625 B octal)
Track-to-Track Spacing	0.005 inches nominal
Minimum Recommended Time per Bit	360 ns
Maximum Start Time	90 seconds to ready
Maximum Stop Time	11 seconds

In general, the drive bus lines should be stable for at least 200 ns before activation of any tag lines. Active tag lines should remain active for a period of at least 800 ns, and pulsed tag lines (for all commands except read, write, or erase) should remain active for at most 10 us. Following the de-activation of all tag lines, the drive bus lines should remain stable for at least 200 ns; then all bus drive lires should be de-activated for at least 400 ns.

Further information on the disk unit is available from the Century Data Systems (Anaheim, California) Interface Specification, Model 215 Disk System, and from the Model 215 Disk Drive Maintenance Manual (which also includes circuit diagrams).

Part II: Programming Considerations

Many programming considerations relate to errors, and these are in Table 5.

1. Selecting the unit (= loading KUNIT register) is accomplished automatically by the hardware prior to a disk microinterrupt and no other units can be referenced during the interrupt routine. The function YKPTR← is provided especially for disk microprograms. It is planned that the interrupt instruction will include

YKPTR←, X←KUNIT;

This will select the right bank checksum register and the 16-word scratch memory array peculiar to the unit causing the interrupt. Non-interrupt programs select a disk unit by explicitly loading KUNIT.

2. One microinstruction must elapse after explicitly selecting KUNIT← as a destination or after the start of a disk interrupt routine before doing a KSET← or KCSET←.

3. KSET← commands must be separated by more than two microseconds. During the interim the "controller not ready" bit returned by KSTAT will be one. This time delay permits the commands to be presented to the disk according to the unusually slow specifications of CDC's disk units.

4. To perform a seek it is necessary to load the cylinder register using one KSET←, wait for controller ready, and then start the seek.

5. The "index sector" indication will remain true (or false) for the entire duration of a sector.

6. KSET← commands with KNEWCOMM must merge a scratch pad register onto the bus because KNEWCOMM is a different name for the READS function. KNEWCOMM should be used on a KSET← given in the following circumstances:
   clearing the head register;
   setting the head register;
   selecting the head;
   setting the cylinder register;
   starting a seek;
but not when changing the state of read, write, and erase (because this will glitch the select head line).

7. There is one bit counter per unit. A sector pulse resets the bit counter to 0 and generates a sector interrupt request. After

that time word interrupts, if enabled, will occur at 40-bit intervals (measured from the sector pulse). This means that skew of the header record is an integral number of word times from the sector pulse. Bit clocking is discussed below.

8. There is only one bit-clocking mechanism per disk unit. When a unit is not reading, bit timing is defined by the write oscillator. When a unit transitions from not reading to reading, bits are not clocked until the synch pattern is recognized. Thereafter bit timing is controlled by the data recorded on the disk. The first bit clocked is the first data bit. When reading is later stopped, bits will be clocked by the write oscillator again.

9. The synch pattern is a sequence of eight consecutive one's (4 ones in top 36 bits and 4 ones in the tag bits). The controller must read at least 20 microsec of all-zeroes preamble prior to the synch pattern to ensure proper correction for the worst case differences in frequency and phase between the write oscillator and the read data. The first data word should immediately follow the synch pattern.

10. Reading should be done only over valid preamble and data. If a read is started over an erased area or other wrong-frequency pattern, then the phase-locked loop may be so badly confused that it cannot converge to the correct frequency. If a bad spot occurs during a read, several bit times elapse before the locking circuit loses synch. There are actually two phase-locked loops: one for coarse locking, the other for fine locking. The coarse locking loop is on when the unit is not reading. As soon as a read is started, the fine-locking loop is turned on. The fine-locking loop may not converge if the frequency error is too great, and this is the reason why reading should be started only over valid preamble.

11. To start a read or write at the current arm position, it is necessary to go through the following painful sequence of disk commands:

A. Clear the head register.

B. Wait for "controller not ready" to be 0. (about 2 us).

C. Set the head register to the desired value.

D. Wait for controller ready (about 2 us).

E. Select the head.

F. Wait 3 us before writing (and erasing) at the selected head. 10 us may elapse after head selection before reliable read data appears.

G. 5 us of garbage may be written before valid data is written.

The implications of B, D, and F are that it is impractical to do A, C, E and start reading or writing without timing separation between them, and since the timing requirements on interrupt routines are so stringent, it will probably be necessary to begin operations at a sector as follows:

A. During sector interrupt, clear the head register. Maybe the 2 usec wait can be overlapped so that the head register can be set during the sector interrupt routine also.

B. If necessary, wait during the first word interrupt. Then select the head.

C. Skip the second word interrupt.

D. Start writing or reading no sooner than the third word interrupt.

12. Write gate and erase gate should be turned on in the same microinstruction and not in the same KSET→ that turns read gate off (or "unit unsafe" occurs).

13. Every record written on the disk should be followed by at least nine bits of valid data so that a word interrupt will be triggered for the final word of the record.

14. An erase turn-off "blast" may endanger data 20 us behind the write head. None of our contacts at CDS are convincing. The purpose of erase is to narrow the data so that adjacent tracks are noise protected by an erased guard band. To insure this erase should be kept on for 20 us after write is turned off. Erase may not be continued for longer than 60 us or the hardware generates an "unsafe" error. Head select should remain stable for at least 1 us after erase is turned off.

15. Disk word interrupts may be dismissed by either KRDATA or KWDATA, regardless of whether reading, writing, or nothing is being done on the unit itself.

16. When a write is started, there are two hardware buffer words of interest. One of these is presented for writing onto the disk, the other for access by KWDATA. Since the first word of preamble written should be all zeroes, both buffer words should be zeroed during the two word interrupt routines prior to the one which turns on the write gate.