APPLICATION NOTE

An Intelligent, Fast Disk Controller Using the Am29116

By Paul Chu, Brad Kitson, and Otis Tabler
Advanced Micro Devices

September 1982

Until recently, advances in high-performance disk systems were limited mainly by the state of the art in Read/Write circuits and head. Today, track densities and transfer rates are becoming so high that the design of the controller is becoming a bottleneck. The need for high bandwidth is accompanied by demands for more powerful command sets and the transfer of many operating system software tasks into the controller firmware.

To implement intelligent high-bandwidth controllers, flexible and very fast VLSI building blocks are needed. This article shows how two such building blocks, the Am29116 Bipolar Microprocessor and the Am9520 Burst Error Processor, can be combined to form a disk controller with over 20MHz bandwidth, and incorporate such features as detection and correction of burst errors up to 11 bits long, I/O request queue sorting, sector caching, device transparency, logical record I/O, and associative (content-addressed) reading and writing of logical records.

The Am29116 performs 10 million instructions per second within a 16-bit parallel architecture and 32 x 16 register file. Its 16-bit barrel shifter allows an operand to be masked and rotated from 1 to 15 places and then optionally compared with a second operand within a single instruction cycle. Within a single cycle, it is also possible to rotate an operand and merge it with a second operand under a mask.

Other important features of the Am29116 includes its generation of forward and reverse CRCs; its ability to prioritize event and status bits under mask; and its ability to set, reset, and test arbitrary bits. The Am29116 is the largest and most complex such bipolar device produced. Fabricated using AMD's proprietary ion-implemented oxide-isolated (IMOX^{TM}) process, it contains emitter-coupled logic (ECL) circuitry scaled to VLSI proportions. Although ECL is used internally, all input and output buffers are fully TTL-compatible.

The Am9520's features, which make it a cornerstone of this design, include the ability to generate check bits and detect and correct single and burst errors for four different modified Fire code polynomials--including the popular 48-bit polynomial and the exceptionally powerful 56-bit polynomial used in this design. High throughput of the Am9520 is achieved by using an 8-bit parallel network of exclusive OR gates that accomplishes the equivalent, in a single clock, of eight clockings of a linear feedback shift register. In less than 200 microseconds, the correct high speed mode of the Am9520, which is used in this design, permits correction of a maximum-length error burst (11 bits) anywhere within a 256-byte sector using the microcode logic shown and the 56-bit polynomial. The Am9520 performs the correct high-speed function by simultaneously dividing the data input by all of the factors (except the first) of the polynomial. Location and correction of the error burst is fast because the periods of the factors are short compared with the period of the composite polynomial.

IMOX is a trademark of Advanced Micro Devices, Inc.

Am29116 Organization

The Am29116 includes a 32 x 16 RAM with latched outputs, a 16-bit accumulator, a 16-bit data input latch, a 16-bit barrel shifter, a three-input arithmetic/logic unit, a 16-bit priority encoder, a status register, a condition-code generator/multiplexer, 16 tristate output buffers and a 16-bit instruction latch and decoder (Figure 1).

The single-port RAM has output latches that are transparent when the clock input CP is HIGH and latched when CP is LOW. Data is written into the RAM while the clock is low if the $\overline{\text{IEN}}$ input is also LOW and if the instruction being executed selects the RAM as destination. Data is written into the low-order 8 bits of the addressed word for byte instructions and into all 16 bits for word instructions. Separate read and write RAM addresses may be used by supplying a multiplexer on instruction inputs I4-I0 using CP as the select signal.

The accumulator, which is edge-triggered, accepts data on the LOW-to-HIGH transition of CP if $\overline{\text{IEN}}$ is also LOW and if the instruction being executed selects it as the

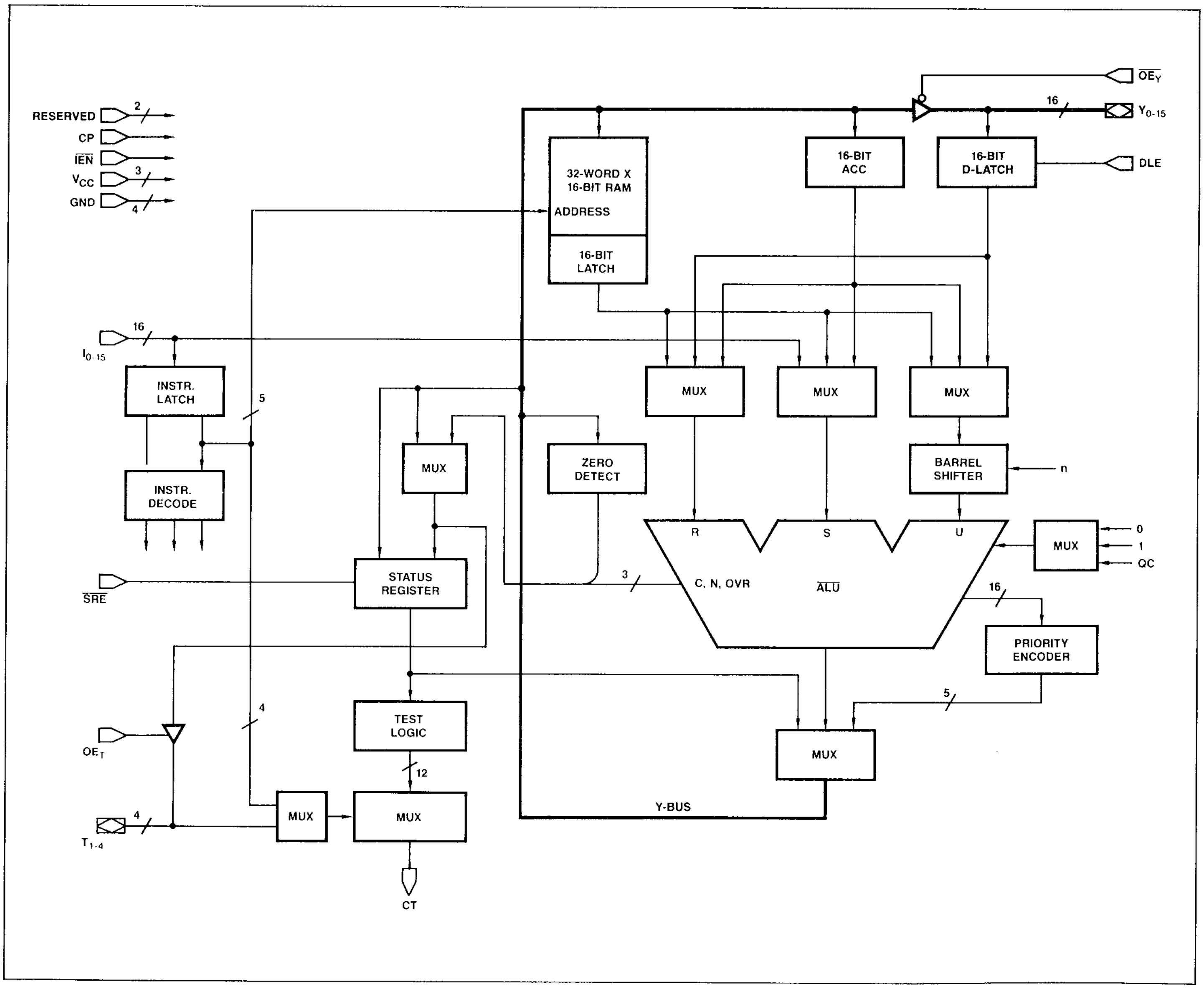


Figure 1. Am29116 Organization

destination. As with RAM locations, byte instructions modify only the lower half of the accumulator while word instructions modify the full register.

The data input latch (D-latch) holds the data input to the ALU on the bidirectional Y bus. The latch is transparent when the DLE input is HIGH and latched when the DLE input is LOW. The sources of the ALU operation are the RAM, the accumulator, the D-latch and the instruction inputs during IMMEDIATE instructions.

The ALU, which can operate on one, two, or three operands depending upon the instruction being executed, contains full carry lookahead across all 16 bits. All ALU operations can be performed in either word or byte mode. Status outputs Carry (C), Negative (N), and Overflow (OVR) are generated at the byte level for byte-mode operations and at the word level for word-mode operations. A fourth flag, Zero (Z), is generated outside the ALU and also operates in either byte or word mode. The Stored Carry (QC) bit of the status register may be selected (along with 0 and 1) as the ALU carry input to support multi-precision arithmetic operations. This is used by the correct high speed microcode of the disk controller, which employs coefficients as large as 2,647,216.

The priority encoder produces a binary-weighted code to indicate the location of the highest-order non-masked one at its input. If none of the masked bits is HIGH, the output of the priority encoder is zero. If bit i is the most significant HIGH bit then the output of the priority encoder is equal to s - i + 1 where s is the position of the sign bit and is equal to 15 in word mode and 7 in byte mode. To understand why s - i + 1 is used in place of s - i (the usual priority encoding), consider the following example (Figure 2). The eight Attention Drive signals are presented on the time-

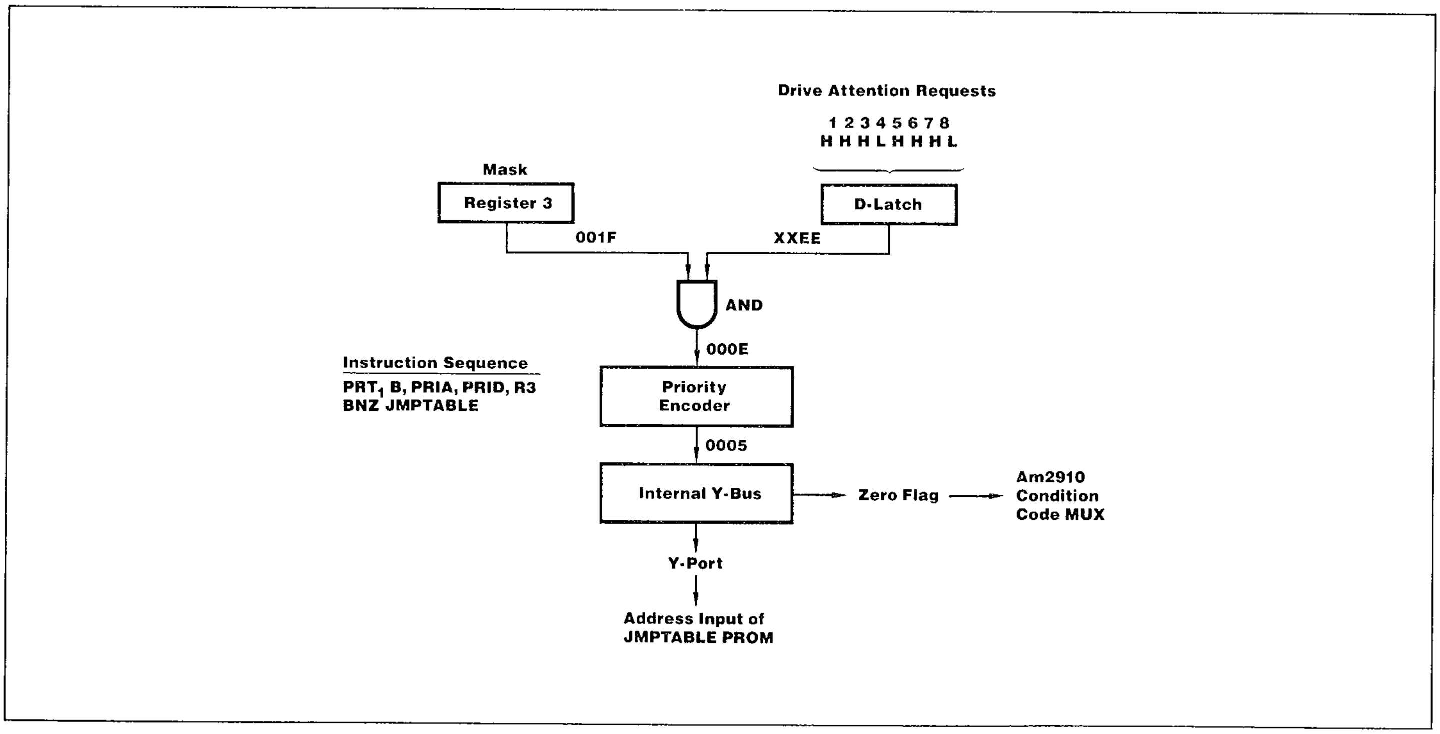


Figure 2. Using the Am29116 Prioritize Instruction

multiplexed drive command/data bus and are read through the Y-bus and data input latch of the Am29116. If the controller has already serviced all attention requests from drives 1-3 and wishes to service the highest priority attention request (if any) from drives 4-8, it executes a Prioritize instruction in byte mode using hexadecimal 001F as the mask, followed by a Branch if Not Zero into a jump table indexed by the priority encoder output.

The 8-bit status register and the condition-code generator/multiplexer contain the information and logic necessary to develop 12 condition-code test signals. The multiplexer selects one test signal and places it on the condition-code test (CT) output for use by the microprogram sequencer. The multiplexer is addressed in two ways. In the first, which is used here to maximize throughput, the T-bus is used in input-only mode to specify the multiplexer select position directly. In the second, the CT output is selected through a test instruction.

The output enable Y-bus ($\overline{\text{OEY}}$) input enables the 16 tristate output buffers when it is LOW. When $\overline{\text{OEY}}$ is HIGH, the output buffers are read in the high-impedance state (allowing read/write and status data to reach the D-latch from the controller's 16-bit system data bus).

The 16-bit instruction latch is normally transparent to allow decoding of the 16 instruction inputs I15-0 into internal control signals for the Am29116 and the execution of the instruction within a single clock cycle. The only exceptions to this rule are the immediate-operand instructions, which execute in two clock cycles rather than one. These are captured in the instruction latch during the first clock and executed during the second. It is during the second clock that the immediate operand, which resides in the I15-0 field of the next microinstruction, is fetched and execution is completed. Immediate instructions are used extensively in the disk controller microcode whenever masks and special arithmetic constants are needed. (The Am29116 allows the addition or subtraction of 2^N, and the use of 2^N and its complement as a mask, for any N between 0 and 15 within a single clock, so that for these 16 common numbers and 32 common masks, an immediate instruction is not required).

Am29116 Instructions

The 16-bit instructions of the Am29116 can be grouped into eleven types which correspond in a natural way with the Am29116's internal instruction decoding logic: single operand, two operand, single bit shift, rotate and merge, bit oriented, rotate by n bits, rotate and compare, prioritize, cyclic redudancy check, status, and no-op. The microprogrammer needs to be familiar with these groupings (and certain subgroupings) because the System 29 AMDASM DEF file provides mnemonics that correspond to them. For example, the AMDASM SRC file line

```
        SOR            W,INC,SORY,R1
```

increments the full 16-bit contents of Am29116 RAM location 1 by one and places it onto the Y-bus and

```
        TOR1           B,SUBR,TORAR,R2
```

subtracts the low-order byte of the

accumulator from the low-order byte of RAM location 2 while leaving the high-order byte of location 2 unchanged.

Table 1 summarizes the basic operations that Am29116 instructions can perform within a single cycle. (Two cycles are used if one operand is immediate data.) Note that for a typical line of this table, there are several Am29116 mnemonic operation codes, depending upon the choice of operand source(s) and destination.

TABLE 1. SINGLE-CLOCK Am29116 OPERATIONS

Add
Add with Carry
Add 2^N
And
Complement
Accumulate forward CRC
Accumulate reverse CRC
Exclusive Nor
Exclusive Or
Increment
Load 2^N
Load 2^N Complemented
Move
Nand
Negate (2's complement)
Nor
Or
Prioritize under mask
Reset bit N
Reset status bit
Rotate N bits
Rotate N bits and compare under mask
Rotate N bits and merge according to mask
Set bit N
Set status bit
Single bit shift
Subtract
Subtract with Carry
Subtract 2^N
Test bit N
Test status bit

Many of the operations prove particularly useful when implementing intelligent disk controllers. For example, the packing of ASCII characters (which occupy 8-bit bytes in main memory yet need only occupy 7-bit contiguous frames in the disk record) is accomplished efficiently and at high speed by Rotate and Merge instructions as shown in Figure 3. The microinstructions shown on the arrows perform the bit mapping indicated by them. In this example, 8 ASCII bytes requiring 64 bits of main storage are packed into 56 bits (8 7-bit contiguous frames) prior to being written to disk. In general, the ability of the Am29116 to rotate a 16-bit operand by N bits and merge it with a second 16-bit operand under mask within a single cycle makes the manipulation of arbitrary-length, arbitrarily-aligned data fields efficient and simple. Other operations that are especially valuable in this application are provided by the CRC Forward instruction (used to generate or check the integrity of header records), the instructions which add and subtract 2^N, load 2^N and its complement, reset, set, and test bit N, and (as indicated above) the masked Prioritize instruction. If the intelligence incorporated into the controller includes associative retrieval based upon recognition of an arbitrary bit string within the data record, the instructions which rotate by N bits and then (within the same cycle) compare under mask are almost indispensable.

Functions of An Intelligent Controller

The interface signal names, polarities, and functions used in this article are similar to those used in the current ANSI standard for rigid disks. However, the methods and

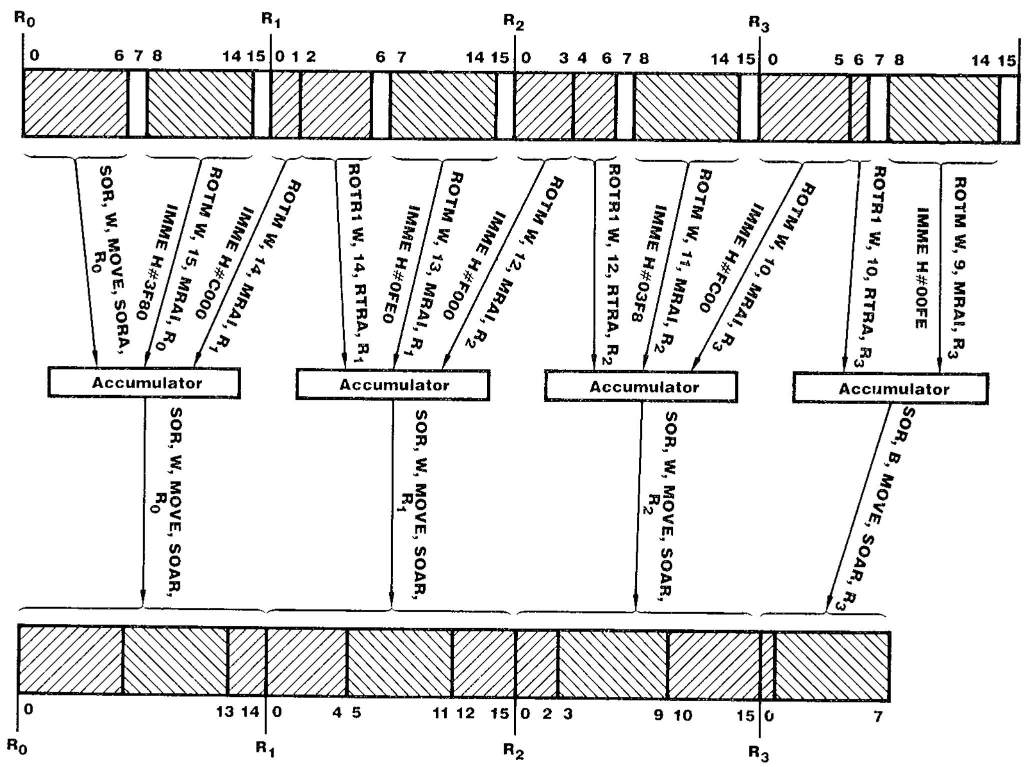


Figure 3. Packing ASCII Fields by Means of the Rotate and Merge Instruction

functions discussed can be used for most current rigid or flexible disk drives. With minimal external logic, the Am29116 and Am9520 perform all the functions needed to format, read, and write disks at over 20 MegaBits per second. These include generating and checking header CRCs, performing header-sector acquisitions, enabling and disabling drive read/write circuits at the appropriate times, managing data flow through a high-speed buffer memory, generating check bits during writes, and detecting and correcting single and burst errors up to 11 bits long during reads.

Except for seeks, retries, and formatting, all of the above have been microcoded. The microcode fits within 256 words (one-fourth of the microprogram memory used in the design), and it is appropriate here to describe some additional intelligent functions that can be microprogrammed:

Maintaining I/O Request Queues. To maximize throughput, the controller orders its read and write request queues by sector, head, drive, and cylinder. (Cylinders appear last in the order of sorting because a seek on one drive may be overlapped with a read or write on another.) The Am29116 maintains the read/write request queue in its 4096 x 16 high-speed buffer memory.

Selective sorting of the read/write request queue is performed by the controller. Each new request is assigned a "bump count" of 0 when the controller receives it. The request is then placed into the queue at the position determined by the following:

(1) Behind all requests whose bump counts equal N ("Queue 1")
(2) Inserted in sorted order into the remaining queue of requests whose bump counts are less than N ("Queue 2") as follows:

(a)	By type (read after write)
(b)	By sector number
(c)	By head number
(d)	By drive number
(e)	Finally, by cylinder number

(3) Before each new request is queued, Queue 2 is scanned head-to-tail. Each request encountered during the scan that has a bump count of N is removed from Queue 2 and placed at the end of Queue 1.
(4) After each new request is queued, the bump count is increased by 1 for each Queue 2 member that has been bumped by it (i.e., now follows it).

It should be noted that the choice of N is application-dependent, since increasing N increases throughput but also lengthens response time for some read/write requests.

2. Avoiding Redundant Reads. The Am29116 also maintains copies of the last eight sectors read from or written to disk. Before each read request is entered into the queue, the Am29116 compares it with

a list of buffer memory-resident sector images. If a match is found, the contents of the sector images are used to satisfy the read request and no enqueueing is performed.

3. <u>Performing Logical Record I/O and Maintaining Device Transparency</u>. The Am29116 translates I/O requests by logical record number into physical select, seek, and I/O operations by drive, track, head, and sector numbers. The CPU software need not concern itself with the characterisitics of the particular drives attached, and it is minimally affected by deletions and additions of drives of varying types.

4. <u>Performing Associative Logical Record I/O</u>. The Am29116 reads, writes, or returns the logical record numbers of logical records that contain specified fields. The CPU software merely specifies the type of operation to be performed and the length, relative position within the logical record, and value of the content-addressing field.

5. <u>Performing Data Compression and Expansion</u>. Much of the information routinely stored on disk as 8-bit bytes is character data. While it is convenient to manipulate these data in the central processor in 8-bit EBCDIC notation, they can usually be stored much more efficiently on disk as either 6-bit BCD (or FIELDATA) bytes or 7-bit ASCII bytes. The usefulness of compressing information in this manner depends entirely upon the database. For example, most accounting and management information system data do not involve lower-case alphabetics and can be recorded in 6-bit BCD (or FIELDATA), giving approximately a 25% reduction in disk storage occupied and a 33% increase in storage effectiveness. Most word processing data involve lower-case alphabetics but can be recorded in 7-bit ASCII, giving approximately a 12.5% reduction in disk storage occupied and a 14.3% increase in storage effectiveness. The recording of data compressed in this manner is accomplished by a translation from EBCDIC to BCD/FIELDATA or ASCII followed by packing and an unformatted write operation. Compressed data are read by an unformatted read operation followed by unpacking and a translation from BCD/FIELDATA or ASCII to ECBDIC. The translations are performed two bytes at a time by the two 2048 x 8 Am27S291 PROMs illustrated in Figure 8. The three microcode bits labelled XLAT2 -XLAT0 select one of eight code translations; four are used by the BCD/FIELDATA and ASCII compression algorithms and four are spares.

Many other types of application-dependent data compression can be performed directly by the controller. The following IBM VM/370 CMS commands perform various types of compression depending upon the old file type:

(1) COPY ,old file name. ,old file type. ,old file mode. ,new file name. ,new file type. ,new file mode. (REP PACK)

(2) COPY ,old file name. ,old file type. ,old file mode. ,new file name. ,new file type. ,new file mode. (REP UNPACK)

All the functions of COPY (PACK) and COPY (UNPACK) can be performed by the Am29116 and Am9520-based controller. The controller allows packed files to be read and written as if they were unpacked, just as it allows 6-bit BCD/FIELDATA and 7-bit ASCII files to be read and written as if they were 8-bit EBCDIC files.

System Organization

Figure 4 is an overall block diagram of the disk controller. The interface to the drives includes separate bit-serial data paths for read data and write data, and byte-parallel paths for commands, disk addresses, and disk status as described in the current ANSI standard. The Am2910 microsequencer and 1K x 8 Am27S35 registered microprogram memory drive the 80-bit control bus that directs the actions of the other components. Data flows serially and asynchronously at over 20 MegaBits per second between the drives and the time-division multiplexed serial input/serial output ports of the 16 x 16 FIFO array. Data flows synchronously in 16-bit parallel form between the FIFO array and the 4K x 16 Am9147 buffer memory. In this design, it is assumed that support of disk transfer rates of close to 30Mbit/sec. is desirable. This is why the burst error processor, which can handle data up to 20Mbit/sec, is placed in parallel with the first-in-first-out memory array and the Am9147 RAM buffer*. During

*AMD now offers the Am9520-1, a 30Mbit/sec part which will simplify the microcode shown in the application note.

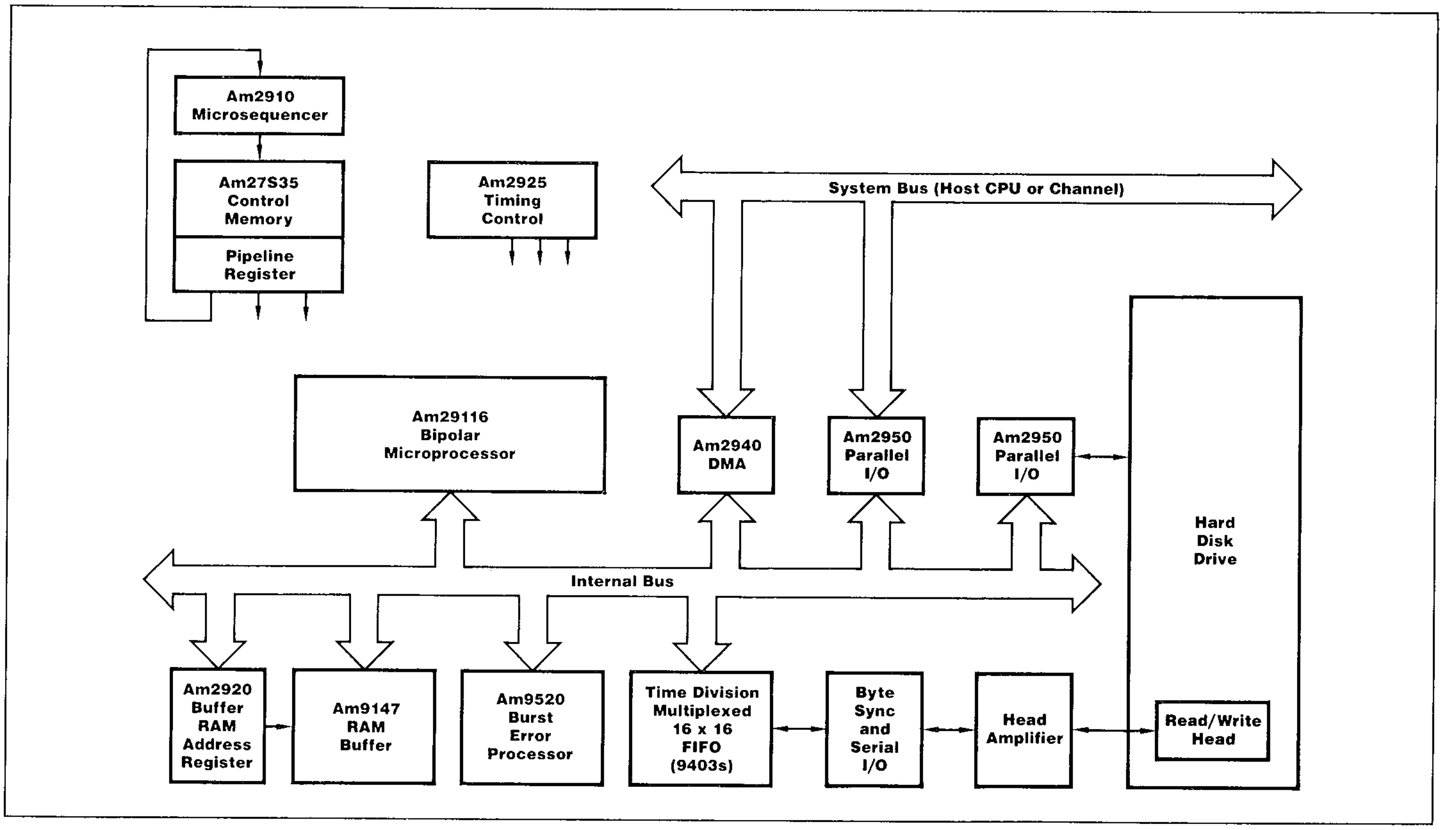


Figure 4. Block Diagram of the Am29116/Am9520 Disk Controller

disk reads, the Am29116 maintains two pointers: a write pointer for transferring data from the FIFO array to the buffer memory at a rate close to 30MHz, and a read pointer for concurrently transferring data from the buffer memory to the burst error processor at a rate equivalent to 15MHz. During disk writes, in which the timing of the checksum calculation is more critical, the transfers are not overlapped. If the data transfer rate needed is 20Mbit/sec or less in an alternative design, the burst error processor can be placed in line with the FIFO array. Table 2 lists the interface signals between the controller and up to eight drives.

TABLE 2. CONTROLLER/DRIVE INTERFACE SIGNALS

SYMBOL	PROSE SIGNAL NAME	SIGNAL SOURCE
$\overline{\text{ADMC}}$	Address Mark Control	Controller
$\overline{\text{ATTN}}$	Attention	Selected Drive
$\overline{\text{BACK}}$	Bus Acknowledge	Selected Drive
$\overline{\text{BOUT}}$	Bus Direction Out	Controller
$\overline{\text{BUSY}}$	Busy	Selected Drive
$\overline{\text{CBPA}}$	Control Bus Parity	Controller or Selected Drive
$\overline{\text{CBDA}}_{0-7}$	Control Bus Data (multiplexed with SADR_{0-7})	Controller or Selected Drive
$\overline{\text{CREQ}}$	Command Request	Controller
$\overline{\text{INDX}}$	Index	Selected Drive
$\overline{\text{PENB}}$	Port Enable	Controller
$\overline{\text{PREQ}}$	Parameter Request	Controller
$\overline{\text{RDCM}}$	Read/Reference Clock -	Selected Drive
$\overline{\text{RDCP}}$	Read/Reference Clock +	Selected Drive

TABLE 2 CONTROLLER/DRIVE INTERFACE SIGNALS (Cont.)

SYMBOL	PROSE SIGNAL NAME	SIGNAL SOURCE
$\overline{\text{RDDM}}$	Read Data -	Selected Drive
$\overline{\text{RDDP}}$	Read Data +	Selected Drive
$\overline{\text{RDGA}}$	Read Gate	Controller
$\overline{\text{SADR}}_{0-7}$	Select/Attention Drive$_{0-7}$ (multiplexed with $\overline{\text{CBDA}}_{0-7}$)	Controller or Selected Drive
$\overline{\text{SAMD}}$	Sector/Address Mark Detected	Selected Drive
$\overline{\text{SAST}}$	Select/Attention Strobe	Controller
$\overline{\text{WRCM}}$	Write Clock -	Controller
$\overline{\text{WRCP}}$	Write Clock +	Controller
$\overline{\text{WRDM}}$	Write Data -	Controller
$\overline{\text{WRDP}}$	Write Data +	Controller
$\overline{\text{WRGA}}$	Write Gate	Controller

The host CPU and memory interface is through either DMA or a host data channel, depending upon the host machine and application. Although the interface is not shown in detail, it can readily be implemented using the Am2940 DMA Address Generator and the Am2950 Parallel I/O Data Port.

Figure 5 depicts the byte-sync logic and timing logic for the FIFO buffer. It has been assumed here that the encoding scheme used by the drives is one that employs all-zero preambles (e.g., Modified Frequency Modulation). If 3PM or any other non-zero-preamble scheme is used, the byte-sync logic shown must be appropriately redesigned. Redesign of the byte-sync logic will also be necessary for drives that suppress transmission of part or all of the preamble.

Byte sync is achieved by three binary counters, which present and maintain a low output as soon as at least K zeroes followed by binary 11111110 (hexadecimal FE) have been encountered. The value of K may be "programmed" by means of the D, C, B, A

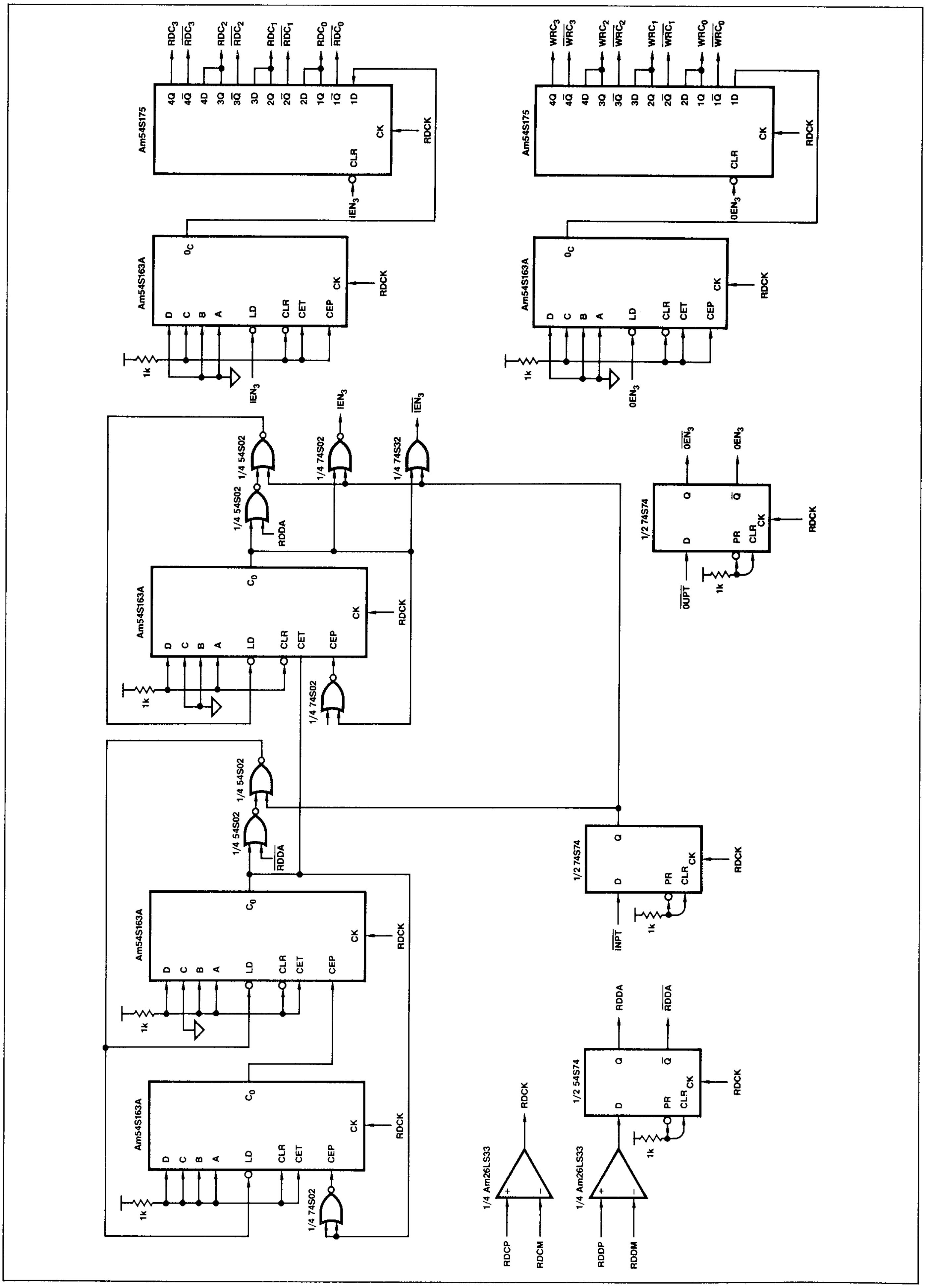


Figure 5. Byte-Sync and Timing Logic

inputs to U1 and U2. These inputs are shown tied to hexadecimal F7. Since $FF_{16} - F7_{16} = 08_{16} = 08_{10}$, K = 8 for this instance. Higher values of K may render the detector unduly sensitive to phase locktime jitter and should be avoided. The bits first encountered during a sync burst are the least likely to be sampled correctly, since the drive's clock/data separator is still acquiring phase lock with the sync byte train. The optimal choice for K depends upon the acquisition rate and other characteristics of the clock/data separator.

Figure 6 depicts the serial-to-parallel and parallel-to-serial conversion interface using an array of 9403As operated in

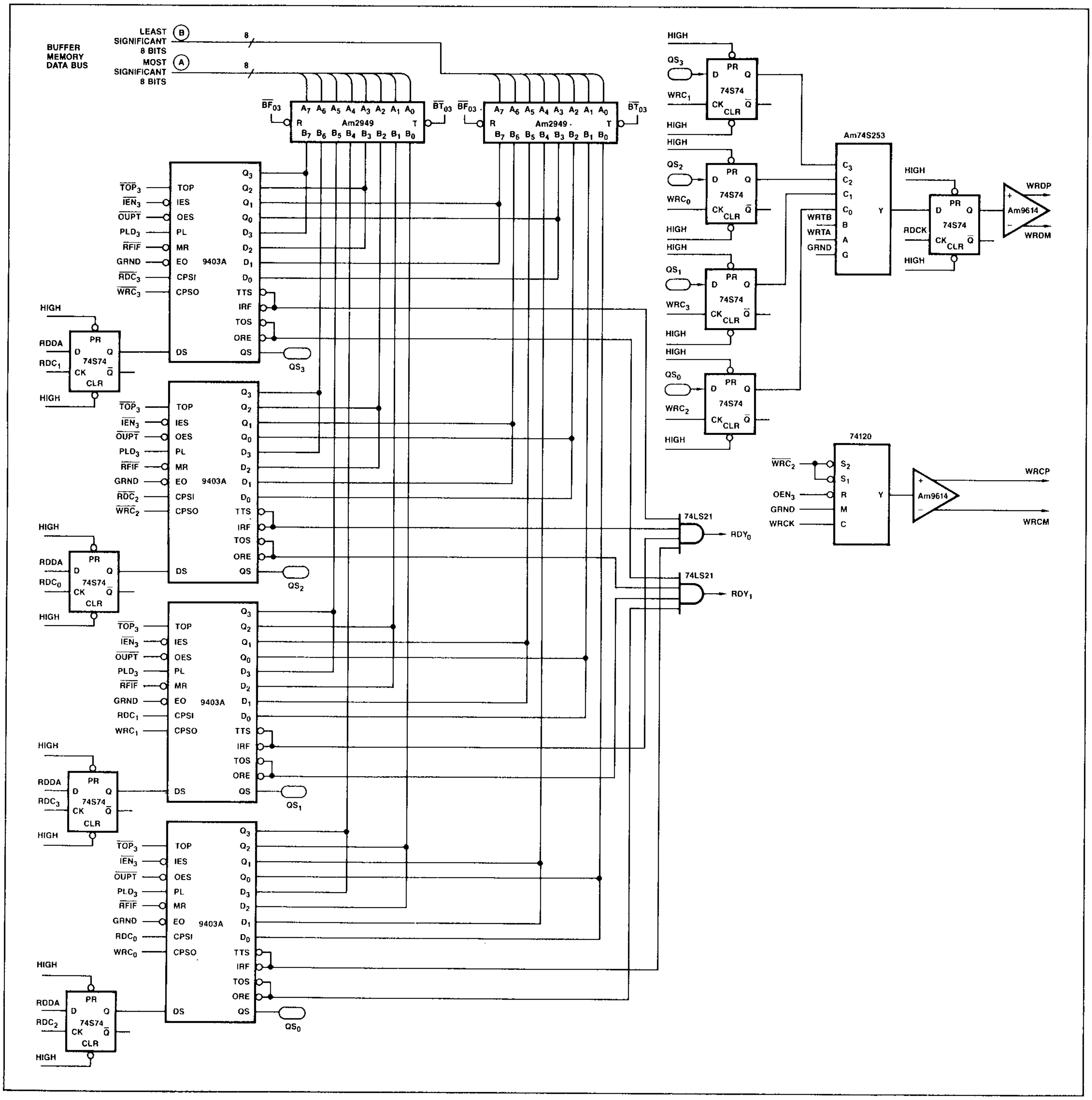


Figure 6. Serial: Parallel Interfacing

parallel at an aggregate rate of 30Mbit/sec per second. The FIFOs themselves are individually operated at 7.5Mbit/sec per second, and the 30Mb aggregate data rate is achieved by an alternate clocking scheme (Figure 7). This same scheme is used for both read and write clocking and that the FIFO serial input and output clocks, $\overline{\text{CPSI}}$ and $\overline{\text{CPSO}}$, are falling-edge active. Pipelining is used to satisfy the setup time requirements of the FIFO serial inputs, DS. The FIFO serial outputs QS are also pipelined. However, the FIFO parallel inputs and outputs, D3-D0 and Q3-Q0, are fast enough to communicate with the buffer memory bus without pipelining.

The major elements of the remaining portion of the data path are the Am29116, the Am9520 and 4096 words of Am9147-55 buffer memory (Figure 8). These elements interface through an internal 16-bit data bus. The Am29116 is connected to this bus through two Am2949 bidirectional bus transceivers. During data compression operations, the read and write data are actually routed through two sets of Am27S291 translation PROMs. The Am29116 also generates and maintains the buffer memory addresses. The buffer memory comprises sixteen Am9147-55 4096 x 1 RAMs. It contains images of the last eight sectors read from or written to disk, the I/O request queues, and additional

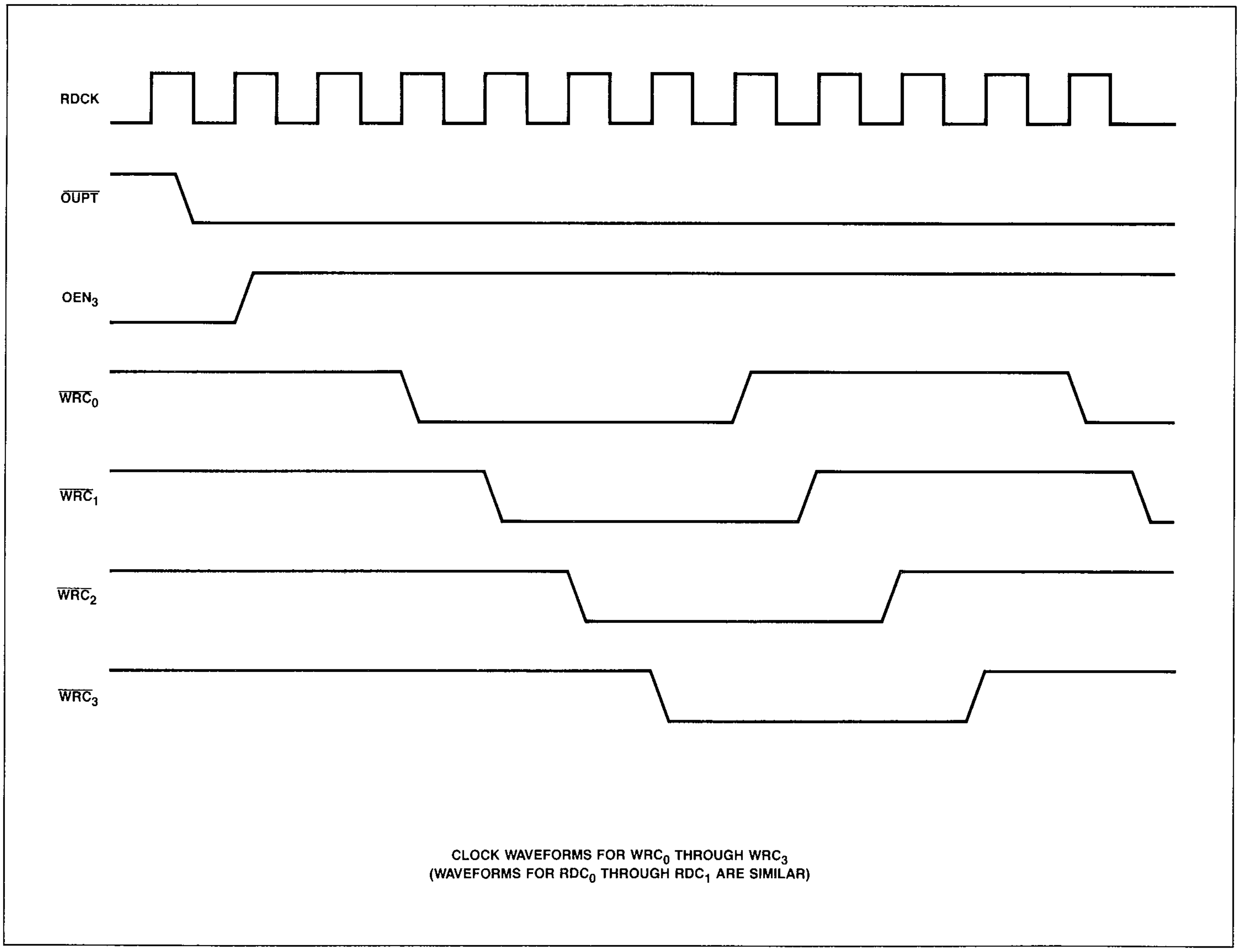

RDCK
$\overline{\text{OUPT}}$
OEN_3
$\overline{\text{WRC}}_0$
$\overline{\text{WRC}}_1$
$\overline{\text{WRC}}_2$
$\overline{\text{WRC}}_3$
CLOCK WAVEFORMS FOR WRC_0 THROUGH WRC_3
(WAVEFORMS FOR RDC_0 THROUGH RDC_1 ARE SIMILAR)

Figure 7. FIFO Alternate Clocking

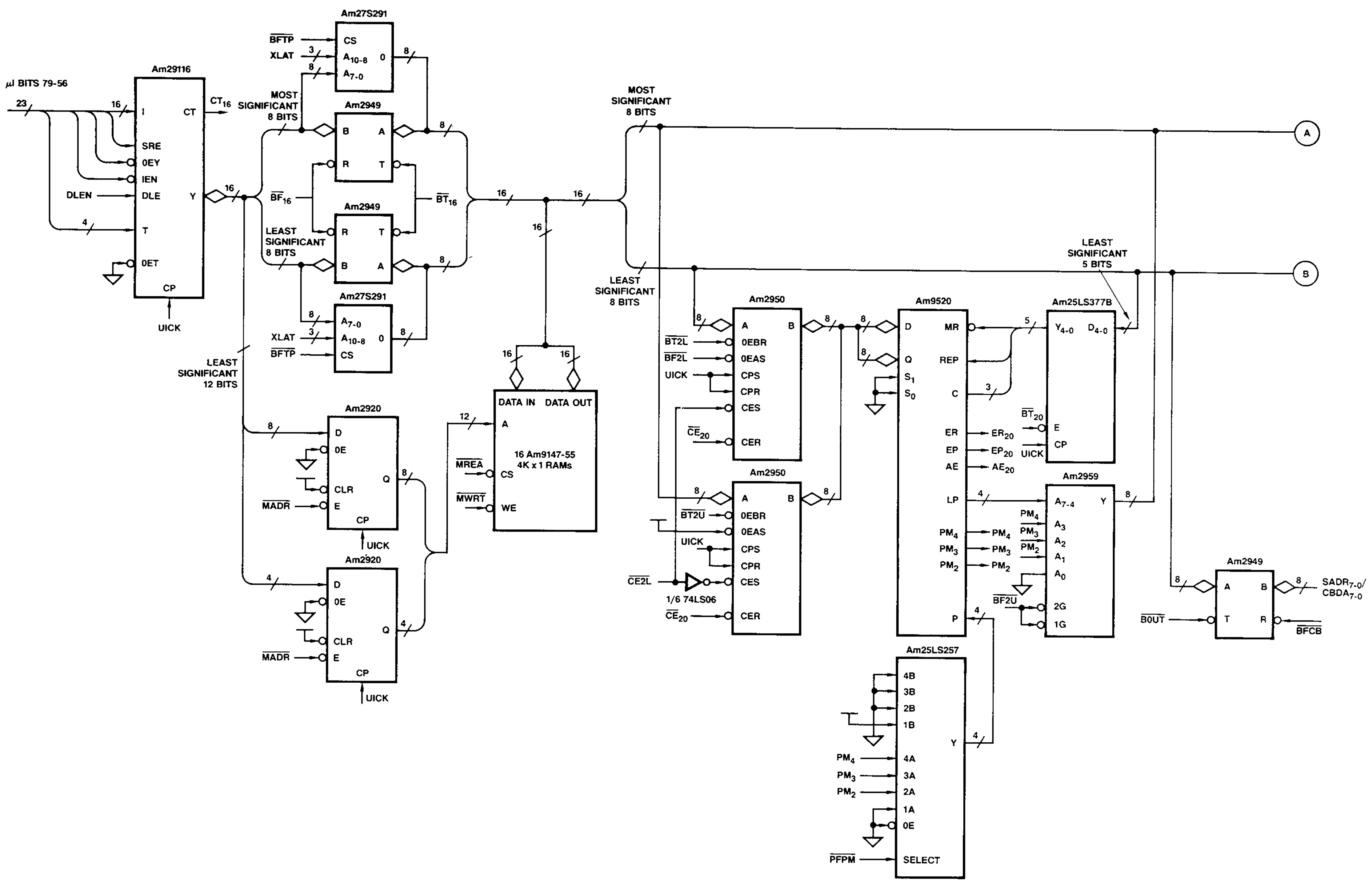


Figure 8. Processors and Buffer Memory

housekeeping tables. The 8-bit data input and output lines of the Am9520 are connected to the 16-bit internal data bus through a low and high byte bidirectional I/O port using two Am2950s. The instruction (C2-0) and read error pattern (REP) inputs of the Am9520 are generated by the Am29116 and are strobed into the command register under microprogram control. The Am9520 status signals--located error pattern (LP3-0) and pattern match (PM4-2)--are communicated to the Am29116 through the Am2959 buffer during high-speed error correction. In addition, the ANSI Control Bus Data ($CBDA_{7-0}$) and the Select/Attention Drive ($\overline{SADR}_{7-0}$) signals to and from the selected drive are multiplexed and connected to the least significant byte of the internal data bus through an Am2949 bidirectional bus transceiver.

The Am2910 microprogram sequencer generates the next address to 1K words of control memory (Figure 9). The control memory is 80 bits wide and is configured using ten Am27S35 1024 x 8 registered PROMs. The test condition ($\overline{CC}$) input to the Am2910 comes from one of sixteen sources (including a forced HIGH and a forced LOW) selected through multiplexers by five microinstruction bits. Except for the Am29116 CT status output, all of the test conditions are synchronized by the microinstruction clock (MICK) because they are from such asynchronous sources as the disk drives and the FIFO array.

Microinstruction Format

The format of the 80-bit microinstruction is outlined in Figure 10. The intent here is not to create a minimum-width, shared-field control word but to demonstrate microcoding the controller in a straightforward manner. Table 3 details the definition for each of the fields. A microinstruction word and field definition (DEF) file incorporating these is available to System 29 users.

Sample microcode has been written (and a source (SRC) file is available to System 29 users) for uncompressed sector read and write operations. The header and data segment format is shown in Figure 11. The code includes header and sector acquisition, error checking of the header (via CRC), and error checking and correction of the data segments (via the Am9520 and its 56-bit modified Fire code polynomial) (Figure 12).

The sector input/output microroutine (SECTIO) performs input or output of a single 256-byte sector. Seek and retry operations are the responsibility of the calling microprogram.

At entry to SECTIO, R0 contains 0 to request a sector read, or +1 to request a sector write. R1 contains the I/O head number in its upper byte. The I/O track number is split between the lower byte of R1 and the upper byte of R2, while the lower byte of R2 contains the I/O sector number. R3 contains the buffer memory start address.

SECTIO first checks to see whether (R0) = +1 and, if so, uses the Am9520 to calculate the 56-bit modified Fire Code check bits that are to be appended during write. The check bits are stored in buffer memory immediately following the data.

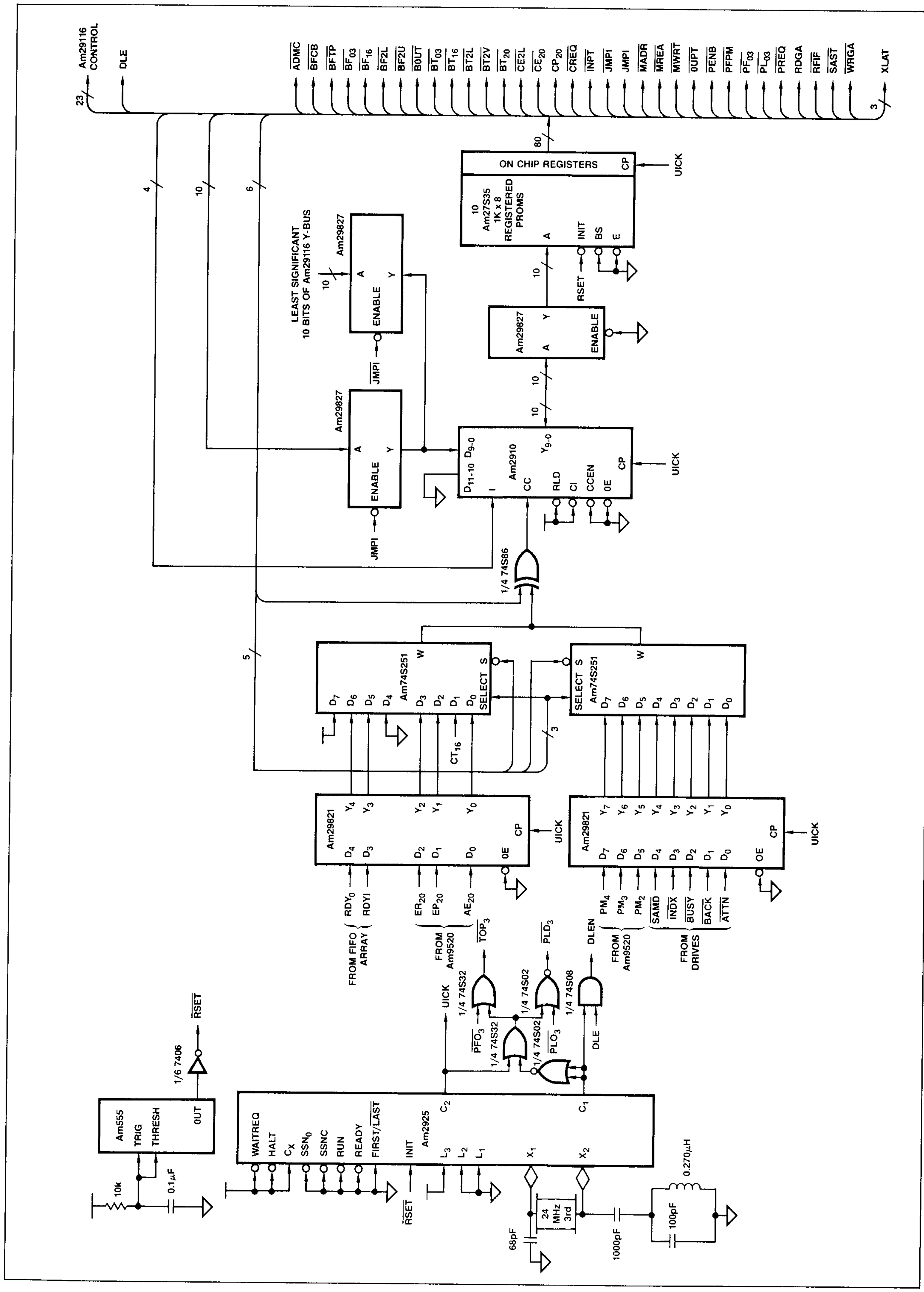


Figure 9. Microinstruction Sequencing

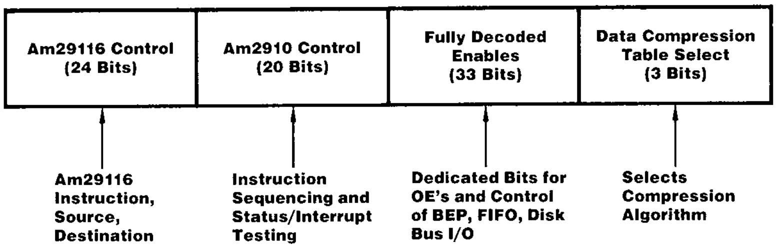

Am29116 Control (24 Bits)
Am2910 Control (20 Bits)
Fully Decoded Enables (33 Bits)
Data Compression Table Select (3 Bits)
Am29116 Instruction, Source, Destination
Instruction Sequencing and Status/Interrupt Testing
Dedicated Bits for OE's and Control of BEP, FIFO, Disk Bus I/O
Selects Compression Algorithm

Figure 10. Microinstruction Format

TABLE 3. MICROINSTRUCTION FIELDS

MICROINSTRUCTION FIELD			MNEMONIC
BITS	WIDTH (BITS)		
79-64	16	I15-I0	Am29116 Instruction
63-60	4	T4-T1	Am29116 Conditional Test Select
59	1	SRE	Am29116 Status Register Enable
58	1	OEY	Am29116 Output Enable Y-Bus
57	1	IEN	Am29116 Instruction Enable
56	1	DLE	Am29116 Data Latch Enable
55-52	4	I3-I0	Am2910 Instruction
51-42	10	D9-D0	Am2910 Direct Input
41-36	6	-	Test Multiplexer Condition and True/False Select
35	1	$\overline{\text{ADMC}}$	Address Mark Control (Table 1)
34	1	$\overline{\text{BFCB}}$	(Enable Memory) Bus From (Disk Drive) Control Bus
33	1	$\overline{\text{BFTP}}$	(Enable Memory) Bus From Translate PROM
32	1	$\overline{\text{BF03}}$	(Enable Memory) Bus From 9403A FIFO Array
31	1	$\overline{\text{BF16}}$	(Enable Memory) Bus From Am29116 Y-Bus
30	1	$\overline{\text{BF2L}}$	(Enable Memory) Bus Lower Byte From Am9520 Q-Bus
29	1	$\overline{\text{BF2U}}$	(Enable Memory) Bus Upper Byte From Am9520 Q-Bus
28	1	$\overline{\text{BOUT}}$	Bus Direction OUT (Table 1)
27	1	$\overline{\text{BT03}}$	(Enable Memory) Bus To 9403A FIFO Array
26	1	$\overline{\text{BT16}}$	(Enable Memory) Bus To Am29116 Y-Bus
25	1	$\overline{\text{BT2L}}$	(Enable Memory) Bus Lower Byte To Am9520 D-Bus
24	1	$\overline{\text{BT2U}}$	(Enable Memory) Bus Upper Byte To Am9520 D-Bus
23	1	$\overline{\text{BT20}}$	(Enable Memory) Bus To Am9520 REP, P3-P0, & C2-C0
22	1	$\overline{\text{CE2L}}$	Clock Enable Am9520 To Lower-Byte Bus Interface Register
21	1	$\overline{\text{CE20}}$	Clock Enable Memory Bus To Am9520 Interface Registers
20	1	$\overline{\text{CP20}}$	Clock Pulse For Am9520 (Microcoded Waveform)

TABLE 3. MICROINSTRUCTION FIELDS (Cont.)

MICROINSTRUCTION BITS	FIELD WIDTH (BITS)	MNEMONIC	DESCRIPTION
19	1	$\overline{\text{CREQ}}$	Command Request (Table 1)
18	1	$\overline{\text{INPT}}$	(Enable Serial Data) Input To 9403A FIFO Array
17-16	2	$\overline{\text{JMPI}}$	(Enable) Jump Indirect Am29116 Y-Bus (Double-Rail)
15	1	$\overline{\text{MADR}}$	(Enable Loading Of Buffer) Memory Address Register
14	1	$\overline{\text{MREA}}$	(Enable Buffer) Memory Read
13	1	$\overline{\text{MWRT}}$	(Enable) Memory Write Operation
12	1	$\overline{\text{OUPT}}$	(Enable Serial Data) Output From 9403A FIFO Array
11	1	$\overline{\text{PENB}}$	Parameter Enable (Table 1)
10	1	$\overline{\text{PFPM}}$	(Enable Setting Of Am9520) P Bits From Am9520 PM Bits
09	1	$\overline{\text{PF03}}$	(Enable) Parallel Fetch From 9403A FIFO Array
08	1	$\overline{\text{PL03}}$	(Enable) Parallel Load Of 9403A FIFO Array
07	1	$\overline{\text{PREQ}}$	Parameter Request (Table 1)
06	1	$\overline{\text{RDGA}}$	Read Gate (Table 1)
05	1	$\overline{\text{RFIF}}$	Reset 9403A FIFO Array
04	1	$\overline{\text{SAST}}$	Select/Attention Strobe (Table 1)
03	1	$\overline{\text{WRGA}}$	Write Gate (Table 1)
02-0	3	XLAT	Translate Table Select For Data Compression PROM

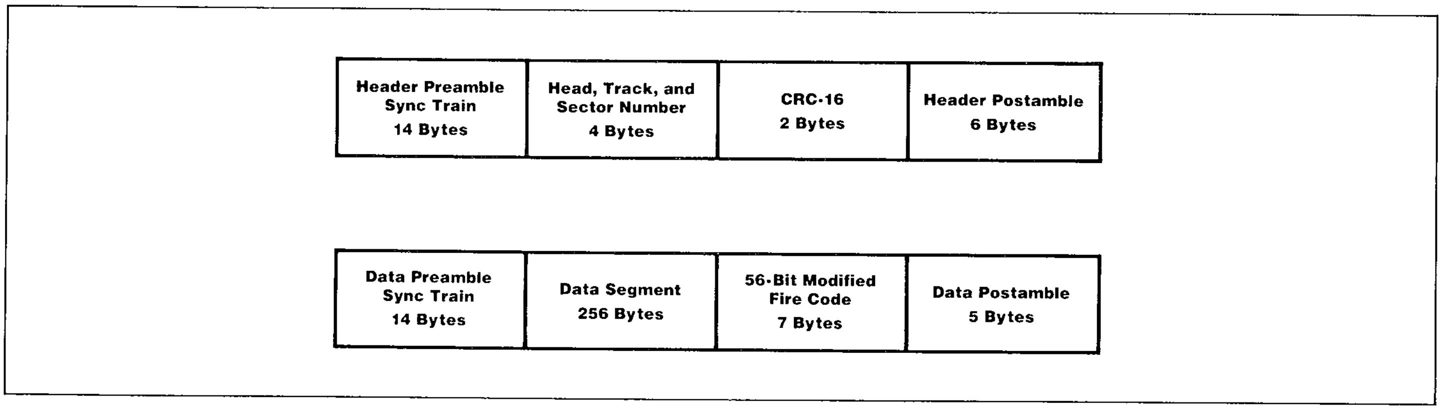
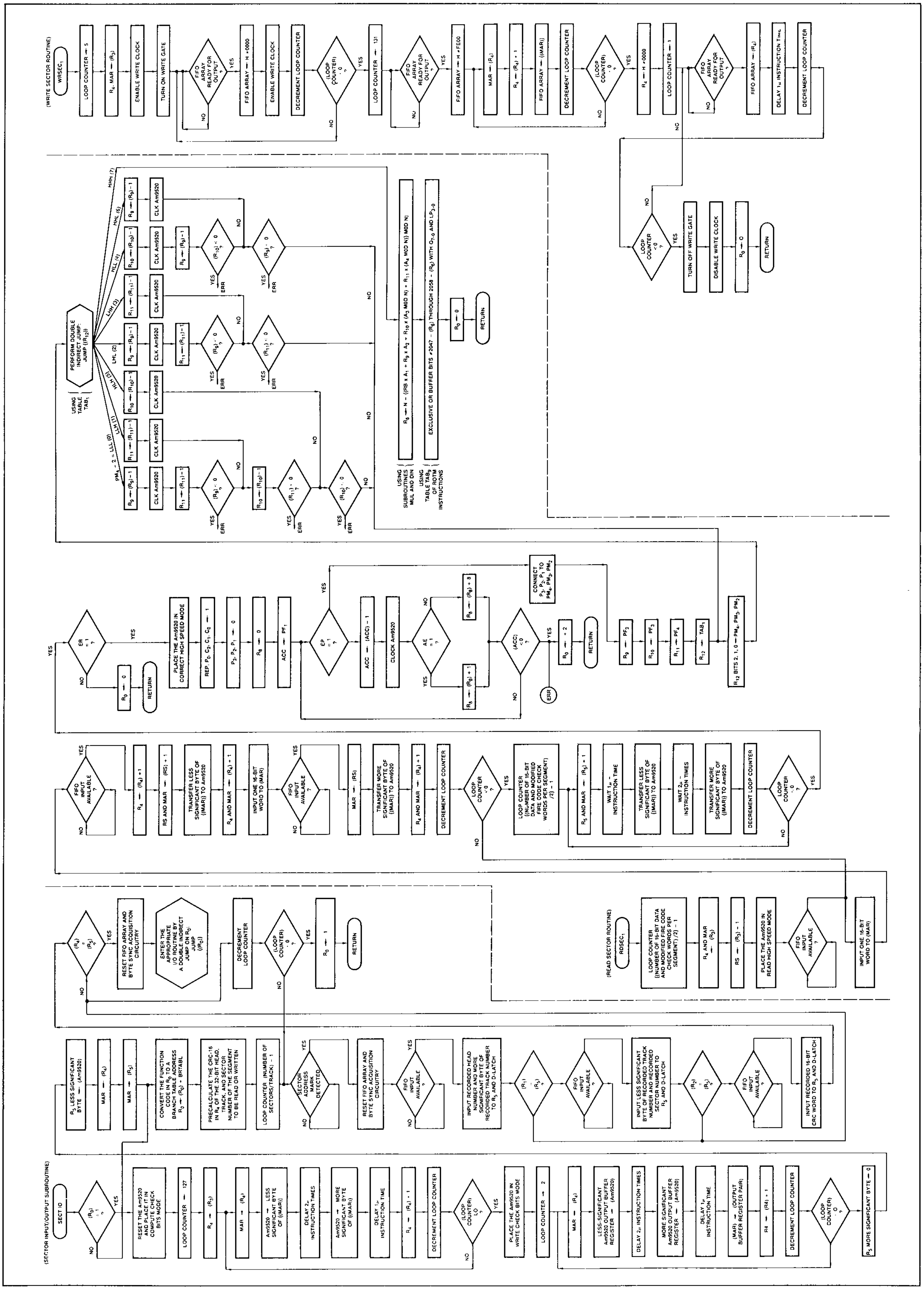

Header Preamble Sync Train 14 Bytes
Head, Track, and Sector Number 4 Bytes
CRC-16 2 Bytes
Header Postamble 6 Bytes
Data Preamble Sync Train 14 Bytes
Data Segment 256 Bytes
56-Bit Modified Fire Code 7 Bytes
Data Postamble 5 Bytes

Figure 11. Header and Data Segments

SECTIO then (both for reads and for writes) uses the Am29116 to calculate the CRC of the header contained in R1 and R2. This CRC is saved in R4.

A search is then made of the entire track for a header whose head, track, sector, and CRC fields match the contents of R1, R2, and R4. If the search fails, R0 is loaded with +1 (defective or missing header) and control is returned to the calling microprogram. If the search is successful, control is passed (via (R0) and table BRTABL) to either the read sector (RDSEC1) or the write sector (WRSEC1) microcode module.



(WRITE SECTOR ROUTINE)
WRSEC1
LOOP COUNTER ← 5
R4, MAR ← (R3)
ENABLE WRITE CLOCK
TURN ON WRITE GATE
FIFO ARRAY READY FOR OUTPUT ?
FIFO ARRAY ← H #0000
DECREMENT LOOP COUNTER
(LOOP COUNTER) = 0 ?
LOOP COUNTER ← 131
FIFO ARRAY ← H #FE00
MAR ← (R4)
R4 ← (R4) + 1
FIFO ARRAY ← ((MAR))
R4 ← H #0000
LOOP COUNTER ← 1
DELAY 1u INSTRUCTION TIME
LOOP COUNTER < 0 ?
TURN OFF WRITE GATE
DISABLE WRITE CLOCK
R0 ← 0
RETURN
USING TABLE TAB1
PERFORM DOUBLE INDIRECT JUMP: JUMP ((R12))
PM4 − 2 = LLL (0)
LLH (1)
HLH (5)
LHL (2)
LHH (3)
HLL (4)
HHL (6)
HHH (7)
R9 ← (R9) − 1
R11 ← (R11) − 1
R10 ← (R10) − 1
CLK Am9520
(R9) < 0 ?
(R10) < 0 ?
(R11) < 0 ?
ERR
USING SUBROUTINES MUL AND DIV
R8 ← N − ((R8 x A1 + R9 x A2 + R10 x (A3 MOD N) + R11 x (A4 MOD N)) MOD N)
USING TABLE TAB2 OF ROTM INSTRUCTIONS
EXCLUSIVE OR BUFFER BITS #2047 − (R8) THROUGH 2058 − (R8) WITH Q7-0 AND LP3-0
ER = 1 ?
PLACE THE Am9520 IN CORRECT HIGH SPEED MODE
REP, P0, C2, C1, C0 ← 1
P3, P2, P1 ← 0
R8 ← 0
ACC ← PF1
EP = 1 ?
ACC ← (ACC) − 1
CLOCK Am9520
AE = 1 ?
R8 ← (R8) + 1
R8 ← (R8) + 8
(ACC) < 0 ?
R0 ← + 2
CONNECT P3, P2, P1 TO PM4, PM3, PM2
R9 ← PF2
R10 ← PF3
R11 ← PF4
R12 ← TAB1
R12 BITS 2, 1, 0 ← PM4, PM3, PM2
FIFO INPUT AVAILABLE ?
R4 ← (R4) + 1
RS AND MAR ← (RS) + 1
TRANSFER LESS SIGNIFICANT BYTE OF ((MAR)) TO Am9520
R4 AND MAR ← (R4) + 1
INPUT ONE 16-BIT WORD TO (MAR)
MAR ← (RS)
TRANSFER MORE SIGNIFICANT BYTE OF ((MAR)) TO Am9520
R4 AND MAR ← (R4) + 1
LOOP COUNTER ← (NUMBER OF 16-BIT DATA AND MODIFIED FIRE CODE CHECK WORDS PER SEGMENT) /2) − 1
R5 AND MAR ← (R5) + 1
WAIT 1u INSTRUCTION TIME
WAIT 2u INSTRUCTION TIMES
(R4) = (R5) ?
RESET FIFO ARRAY AND BYTE SYNC ACQUISITION CIRCUITRY
ENTER THE APPROPRIATE I/O ROUTINE BY A DOUBLE INDIRECT JUMP ON R0: JUMP ((R0))
R0 ← 1
(READ SECTOR ROUTINE)
RDSEC1
RS ← (R5) − 1
R4 AND MAR ← (R5)
PLACE THE Am9520 IN READ HIGH SPEED MODE
(SECTOR INPUT/OUTPUT SUBROUTINE)
SECT IO
(R0) = 1 ?
RESET THE Am9520 AND PLACE IN COMPUTE CHECK BITS MODE
LOOP COUNTER ← 127
R4 ← (R3)
MAR ← (R4)
Am9520 ← LESS SIGNIFICANT BYTE OF ((MAR))
DELAY 2u INSTRUCTION TIMES
Am9520 ← MORE SIGNIFICANT BYTE OF ((MAR))
DELAY 1u INSTRUCTION TIME
R4 ← (R4) + 1
(LOOP COUNTER) < 0 ?
PLACE THE Am9520 IN WRITE CHECK BITS MODE
LOOP COUNTER ← 2
LESS-SIGNIFICANT Am9520 OUTPUT BUFFER REGISTER ← (Am9520)
MORE SIGNIFICANT Am9520 OUTPUT BUFFER REGISTER ← (Am9520)
(MAR) ← (OUTPUT BUFFER REGISTER PAIR)
R5 MORE SIGNIFICANT BYTE ← 0
R5 LESS SIGNIFICANT BYTE ← (Am9520)
MAR ← (R5)
CONVERT THE FUNCTION CODE IN R0 TO A BRANCH TABLE ADDRESS R0 ← (R0) + BRTABL
PRECALCULATE THE CRC-16 IN R5 OF THE 32-BIT HEAD, TRACK, AND SECTOR NUMBER OF THE SEGMENT TO BE READ OR WRITTEN
LOOP COUNTER (NUMBER OF SECTORS/TRACK) − 1
SECTOR ADDRESS MARK DETECTED ?
INPUT RECORDED HEAD NUMBER AND MORE SIGNIFICANT BYTE OF RECORDED TRACK NUMBER TO R3 AND D-LATCH
(R1) = (R3) ?
INPUT LESS SIGNIFICANT BYTE OF RECORDED TRACK NUMBER AND RECORDED SECTOR NUMBER TO R3 AND D-LATCH
(R2) = (R3) ?
INPUT RECORDED 16-BIT CRC WORD TO R3 AND D-LATCH
YES
NO

Figure 12. Microcode Logic

Read Sector Microcode Module (RDSEC1)

This module transfers synchronized information, a 16-bit word at a time, from the 9403A FIFO array to buffer memory and from buffer memory to the Am9520 operating in Read High-Speed mode. Since the current Am9520 data sheet only guarantees operation at 20MHz, some form of buffering must be used between the 30MHz disk and the Am9520. This is accomplished by using R4 as a memory buffer pointer for transfer in from the 9403A's and R5 as a pointer for transfer out to the Am9520. For simplicity in the microcode loop, R5 increments at half (rather than two-thirds) the rate at which R4 increments.

At the end of the read loop, R5 has advanced halfway through the data read in and a second loop is executed to process the remaining half of the data through the Am9520.

When all the data have been processed by the Am9520 Read High-Speed operation, the Am9520 error (ER) flag is tested to determine whether an error was detected. If ER is low (no error), R0 is loaded with 0 (operation completed successfully) and control is returned to the calling program.

If ER is high, error correction is performed using the Am9520's correct high speed mode. This uses the Chinese Remainder Theorem method to calculate the error location (as a bit displacement from the end of the data segment) and error pattern (a 12-bit mask). The error is corrected by exclusive or-ing the error pattern with the 12-bit data field beginning at the error location. The capabilities of the Am9520 and the properties of the 56-bit modified Fire Code polynomial make this correction technique extremely fast. Less than 200 microseconds are required for a worst-case error location and correction using the microcode shown.

The location of an error burst is calculated by:

$$L = N \times K - (M_1 \times A_1 + M_2 \times A_2 + M_3 \times A_3 + M_4 \times A_4)$$

where:

L is the difference in position between the last bit transferred and the beginning of the burst error.

N is the composite period of the 56-bit polynomial and is equal to 585,442.

K is the smallest integer such that L is positive.

A_1, A_2, A_3, and A_4 are Chinese Remainder Theorem coefficients:

$A_1 = 452,387$
$A_2 = 2,521,904$
$A_3 = 578,864$
$A_4 = 2,647,216$

M_1, M_2, M_3, and M_4 are factor match clock counts that are accumulated by the microcode while clocking the Am9520 in Correct High-Speed mode. For burst errors of length not exceeding 11 bits, it can be shown that M_1 will never exceed 22 (the period of the first factor of the 56-bit polynomial); M_2 will never exceed 13 (period of the second factor); M_3 will never exceed 89 (period of the third factor); and M_4 will never exceed 23 (period of the fourth factor).

Consequently, the maximum number of Am9520 clock cycles needed to locate an 11-bit (or shorter) error burst is the sum of the first period and the maximum of the remaining three periods:

$$22 + \text{MAX}\ (13,\ 89,\ 23) = 22 + 89 = 111$$

It should be noted that the above number of Am9520 clock cycles is far less than the composite period, 585,442, which is the upper limit for correct normal operations and is representative of how long a less sophisticated part would require to locate and correct the error burst.

To perform error location and correction, the Am9520 is placed in correct high-speed mode and its clock enable P0 is set high for factor match clock count M_1 accumulation. R8 is initialized to 0 to serve as the M_1 counter. PF_1 (the maximum permissible value for M_1, which will be exceeded only for multiple bursts or bursts longer than 11 bits) is loaded into the accumulator (ACC). The EP output is tested. If EP is low, alignment exception (AE) is tested while the ACC is decremented and the Am9520 is clocked. If AE is high, the burst error is not on a byte boundary and R8 is incremented by 1. If AE is low, R8 is incremented by 8. The ACC is now tested. If positive, PF1 is not exceeded and a loop back to the EP test is performed. If negative, an uncorrectable error exists; R0 is set to +2; and control is returned to the calling microprogram. If EP is high, the M_1 calculation is complete; the error pattern is available; and M_2 through M_4 can now be accumulated.

The inherent parallelism of the Am9520 is then exploited by concurrently accumulating M_2 through M_4. This reduces the number of Am9520 clocks required from the sum of the three periods (125) to their maximum (89). R9 through R11 serve as the counters for M_2 through M_4. The microprogram flow of control reflects the completeness or incompleteness of each factor match by looping through a jump table indexed by the Am9520 Pattern Match (PM_2 through PM_4) outputs, and by selectively disabling the P_1 through P_3 clock enables with the same PM_2 through PM_4 outputs. This yields eight possible paths (Figure 12), in each of which the appropriate combination of R9 through R11 can be operated upon and tested to see if it exceeds period factor limits (i.e., a multiple-burst error or an error burst longer than 11 bits has been encountered).

Once M_1 through M_4 have been obtained, the expression:

$$(M_1 \times A_1 + M_2 \times A_2 + M_3 \times A_3 + M_4 \times A_4)$$

is evaluated by calling a specialized multiply subroutine (MUL) four times. This subroutine utilizes the special nature both of the period factor values and of the Chinese Remainder Theorem coefficients to maximize throughput. A specially optimized divide subroutine (DIV) is then called to calculate:

$$(M_1 \times A_1 + M_2 \times A_2 + M_3 \times A_3 + M_4 \times A_4) / N$$

leaving a remainder of (-L + N). One additional subtract obtains L*.

The word-boundary address of the error burst in buffer memory is extracted from L using the Am29116 Rotate and Merge instruction. A 16-way branch on the low-order 4 bits of L is used to enter a table (TAB2) of Rotate and Merge instructions.

These align the error pattern (using a single ROTM instruction if the error burst does not cross a word boundary and two instructions if it does). The error burst is then exclusive OR-ed with the aligned error pattern; R0 is loaded with 0 (operation completed successfully); and control is returned to the calling microprogram.

Write Sector Microcode Module (WRSEC1)

This module transfers information one 16-bit word at a time to the 9403A FIFO array. The information transferred comprises a data preamble (13 all-zero bytes), data sync byte (hexadecimal FE), 256 data bytes, 7 check bytes, and a data postamble (5 all-zero bytes). Both the data bytes and the check bytes are located in buffer memory, beginning at word (R3). (Calculation of the check bytes has already occurred at the beginning of SECTIO).

R0 is loaded with 0 (operation completed successfully) and control is returned to the calling program.

Conclusion

The high-speed and parallel architecture of the Am29116 and Am9520-based controller allows handling of high data transfer rate disk drives and complex data manipulation and management. The availability of cost-effective microprogrammable building blocks in the Am2900 Family has led to systems with increasingly distributed control. This allows functions to be performed at system locations that optimize overall cost/performance.

Significant improvements in host computer system performance can be realized by down-loading many time-consuming operating system tasks into the controller firmware. This allows mainstream processing of the application programs to proceed with minimal I/O overhead. System response is enhanced and main storage usage, software requirements and system overhead are reduced.

* The method used here to obtain the error location is not the only one possible. One alternative is to subtract some form of the Chinese Remainder Theorem coefficients iteratively instead of multiplying and dividing. With each subtraction L would be tested. If negative, N would be added to L. This approach still exploits the parallel nature of the Am9520.

```
 ;
 ; This .DEF file (DISKCTLR.DEF) was created by editing CONTROLR.DEF;
 ; by adding DEF and EQU statements, deleting some others, and by
 ; changing the basic microword format.  The bulk of the effort required
 ; to create such a file was considerably reduced by beginning from the
 ; "master" file (CONTROLR.DEF) rather than typing a new file from scratch.
 ;
 ; This particular .DEF file was created for a specific Am29116-Am9520
 ; disk controller, described in the AMD application note:
 ; "A High-Performance Intelligent Disk Controller," by Otis Tabler and
 ; Brad Kitson, to be released by AMD in early 1982.  The source file
 ; is DISKCTLR.SRC.
 ;
 ; The major difference between this DEF file and the CONTROLR.DEF file is
 ; the approach to the microprogramming.  This file makes heavy use of
 ; DEF statement overlays while the other uses the comma-positional
 ; notation.  The choice is a matter of preference. THE Am29116 MNEMONICS
 ; AND INSTRUCTION LAYOUT ARE IDENTICAL IN THESE FILES.
 ;
 ;
 ; This file may also be used as a master file which the user can edit to
 ; suit his/her application.
 ;
 ; Anyone finding an error in this file is requested to send a marked listing
 ; or portion thereof to: AMD APPLICATIONS     or  AMD CUSTOMER EDUCATION CENTER
 ;                        PO BOX 453  MS#70        PO BOX 453  MS#71
 ;                        SUNNYVALE, CA 94086      490-A LAKESIDE DRIVE
 ;                                                 SUNNYVALE, CA 94086
 ;
 ; Advanced Micro Devices reserves the right to make changes in its product
 ; without notice in order to improve design or performance characteristics.
 ; The company assumes no responsibility for the use of any circuits or
 ; programs described herein.
 ;
 ;
 ; Am29116 Mnemonics Copyright (c) 1982 Advanced Micro Devices, Inc.
 ;
 ;
 ;
 ;
WORD    80
 ;
 ; **************************************************
 ; GENERAL MNEMONICS
 ; **************************************************
 ;
 ; BYTE - WORD MODE SELECT [M] <---------- referenced by DEF statements
 ;
B:              EQU             1B#0    ; BYTE MODE
W:              EQU             1B#1    ; WORD MODE
 ;
 ;
 ; **************************************************
 ; N SELECT [N]
 ;
N0:             EQU             H#0     ; 0
N1:             EQU             H#1     ;
N2:             EQU             H#2     ;
N3:             EQU             H#3     ;
N4:             EQU             H#4     ;
N5:             EQU             H#5     ;
N6:             EQU             H#6     ;
N7:             EQU             H#7     ;
N8:             EQU             H#8     ;
N9:             EQU             H#9     ;
NA:             EQU             H#A     ;
NB:             EQU             H#B     ;
NC:             EQU             H#C     ;
ND:             EQU             H#D     ;
NE:             EQU             H#E     ;
NF:             EQU             H#F     ;
```

```
;
; *****************************************************
; 32 RAM REGISTERS [R]
;
R0:             EQU             5D#0     ; 00000
R1:             EQU             5D#1     ;
R2:             EQU             5D#2     ;
R3:             EQU             5D#3     ;
R4:             EQU             5D#4     ;
R5:             EQU             5D#5     ;
R6:             EQU             5D#6     ;
R7:             EQU             5D#7     ;
R8:             EQU             5D#8     ;
R9:             EQU             5D#9     ;
R10:            EQU             5D#10    ;
R11:            EQU             5D#11    ;
R12:            EQU             5D#12    ;
R13:            EQU             5D#13    ;
R14:            EQU             5D#14    ;
R15:            EQU             5D#15    ;
R16:            EQU             5D#16    ;
R17:            EQU             5D#17    ;
R18:            EQU             5D#18    ;
R19:            EQU             5D#19    ;
R20:            EQU             5D#20    ;
R21:            EQU             5D#21    ;
R22:            EQU             5D#22    ;
R23:            EQU             5D#23    ;
R24:            EQU             5D#24    ;
R25:            EQU             5D#25    ;
R26:            EQU             5D#26    ;
R27:            EQU             5D#27    ;
R28:            EQU             5D#28    ;
R29:            EQU             5D#29    ;
R30:            EQU             5D#30    ;
R31:            EQU             5D#31    ;
;
;
;
; ******************************
; SINGLE OPERAND INSTRUCTIONS
; ******************************
;
; OPCODES [1]
;
MOVE:           EQU             H#C      ; 1100 MOVE
COMP:           EQU             H#D      ; 1101 COMP
INC:            EQU             H#E      ; 1110 INC      INCREMENT
NEG:            EQU             H#F      ; 1111 NEG      INCREMENT COMP
;
; SOURCE-DESTINATION SELECT [2]
;
SORA:           EQU             H#0      ; RAM    ACC
SORY:           EQU             H#2      ; RAM    Y BUS
SORS:           EQU             H#3      ; RAM    STATUS
SOAR:           EQU             H#4      ; ACC    RAM
SODR:           EQU             H#6      ; D      RAM
SOIR:           EQU             H#7      ; I      RAM
SOZR:           EQU             H#8      ; 0      RAM
SOZER:          EQU             H#9      ; D(OE)  RAM
SOSER:          EQU             H#A      ; D(SE)  RAM
SORR:           EQU             H#B      ; RAM    RAM
;
; ************************************************************
SOR: DEF 1V,  B#10,4V%D#,     4V%D#,          5V%D#,64X    ; SINGLE OPERAND RAM
;            \              \      \               \
;               MODE,QUAD,OPCODE,SOURCE-DEST,REGISTER
;                 [M]        [1]      [2]         [R]    <---- refer to proper EQU groups
; ************************************************************
```

```
;
; SOURCE (R/S) [3]
;
SOA:            EQU             H#4      ; ACC
SOD:            EQU             H#6      ; D
SOI:            EQU             H#7      ; I
SOZ:            EQU             H#8      ; 0
SOZE:           EQU             H#9      ; D(OE)
SOSE:           EQU             H#A      ; D(SE)
;
; DESTINATION [4]
;
NRY:            EQU             D#0      ; Y BUS
NRA:            EQU             D#1      ; ACC
NRS:            EQU             D#4      ; STATUS
NRAS:           EQU             D#5      ; ACC,STATUS
;
; ***************************************************************
SONR: DEF 1V,  B#11,4V%D#,    4V%D#,    5V%D#,64X        ; SINGLE OPERAND NON-RAM
;
;               MODE,QUAD,OPCODE,SOURCE,DESTINATION
;               [M]         [1]    [3]      [4]
; ***************************************************************
;
;
;
; ***********************************
; TWO OPERAND INSTRUCTIONS
; ***********************************
;
; OPCODES [5]
;
SUBR:           EQU             H#0      ; S minus R
SUBRC:          EQU             H#1      ; S minus R with carry
SUBS:           EQU             H#2      ; R minus S
SUBSC:          EQU             H#3      ; R minus S with carry
ADD:            EQU             H#4      ; R plus S
ADDC:           EQU             H#5      ; R plus S with carry
AND:            EQU             H#6      ; R . S
NAND:           EQU             H#7      ; R . S
EXOR:           EQU             H#8      ; R   S
NOR:            EQU             H#9      ; R + S
OR:             EQU             H#A      ; R + S
EXNOR:          EQU             H#B      ; R   S
;
;
; SOURCE-DESTINATION [6]                 ; R     S       DEST
;
TORAA:          EQU             H#0      ; RAM   ACC     ACC
TORIA:          EQU             H#2      ; RAM   I       ACC
TODRA:          EQU             H#3      ; D     RAM     ACC
TORAY:          EQU             H#8      ; RAM   ACC     Y BUS
TORIY:          EQU             H#A      ; RAM   I       Y BUS
TODRY:          EQU             H#B      ; D     RAM     Y BUS
TORAR:          EQU             H#C      ; RAM   ACC     RAM
TORIR:          EQU             H#E      ; RAM   I       RAM
TODRR:          EQU             H#F      ; D     RAM     RAM
;
; ***********************************************************
TOR1: DEF 1V,  B#00,4V%D#,         4V%D#,    5V%D#,64X   ; TWO OPERAND RAM (1)
;
;               MODE,QUAD,SOURCE-DEST,OPCODE,REGISTER
;               [M]             [6]      [5]     [R]
; ***********************************************************
```

```
;
;
; SOURCE-DESTINATION [7]                              R      S        DEST
;
TODAR:           EQU            H#1     ; D      ACC      RAM
TOAIR:           EQU            H#2     ; ACC    I        RAM
TODIR:           EQU            H#5     ; D      I        RAM
;
; **************************************************************
TOR2: DEF 1V,   B#10,4V%D#,          4V%D#,     5V%D#,64X    ; TWO OPERAND RAM (2)
;
;                MODE,QUAD,SOURCE-DEST,OPCODE,REGISTER
;                [M]            [7]       [5]     [R]
; **************************************************************
;
; SOURCE    [8]                                    R      S
;
TODA:            EQU            H#1     ; D      ACC
TOAI:            EQU            H#2     ; ACC    I
TODI:            EQU            H#5     ; D      I
;
; *************************************************************
TONR: DEF 1V,    B#11,4V%D#,     4V%D#,     5V%D#,64X        ; TWO OPERAND NON-RAM
;
;                MODE, QUAD,SOURCE,OPCODE,DESTINATION
;                [M]           [8]     [5]      [4]
; *************************************************************
; ***************************************************
; SHIFT INSTRUCTIONS
; ***************************************************
;
; DIRECTION AND INPUT [9]
;
SHUPZ:           EQU            H#0      ; UP 0
SHUP1:           EQU            H#1      ; UP 1
SHUPL:           EQU            H#2      ; UP QLINK
SHDNZ:           EQU            H#4      ; DOWN 0
SHDN1:           EQU            H#5      ; DOWN 1
SHDNL:           EQU            H#6      ; DOWN QLINK
SHDNC:           EQU            H#7      ; DOWN QC
SHDNOV:          EQU            H#8      ; DOWN QN QOVR
;
;
; SOURCE [10]
;
SHRR:            EQU            H#6      ; RAM    RAM
SHDR:            EQU            H#7      ; D      RAM
;
;
; *******************************************************
SHFTR: DEF 1V,  B#10,4V%D#,          4V%D#,     5V%D#,64X   ; SHIFT RAM
;
;               MODE,QUAD,SOURCE,DIRECT-INPT,REGISTER
;               [M]         [10]        [9]       [R]
; *******************************************************
;
;
; SOURCE [11]
;
SHA:             EQU            H#6      ; ACC
SHD:             EQU            H#7      ; D
;
;
; *******************************************************
SHFTNR: DEF 1V,  B#11,4V%D#,     4V%D#,     5V%D#,64X       ; SHIFT NON-RAM
;
;               MODE,QUAD,SOURCE,DIRECT-INP,DESTINATION
;               [M]          [11]    [9]          [4](NRY; NRA ONLY)
; *******************************************************
```

```
;
; ****************************************************
;ROTATE INSTRUCTIONS
; ****************************************************
;
; SOURCE-DESTINATION [12]
;
RTRA:           EQU             H#C      ; RAM   ACC
RTRY:           EQU             H#E      ; RAM   Y BUS
RTRR:           EQU             H#F      ; RAM   RAM
;
;
; *****************************************************
ROTR1: DEF  1V,  B#00,4V%D#,4V%D#,        5V%D#,64X      ; ROTATE RAM (1)
;
;               MODE,QUAD,N,SOURCE-DEST,REGISTER
;               [M]      [N]     [12]      [R]
; *****************************************************
;
; SOURCE-DESTINATION [13]
;
RTAR:           EQU             H#0      ; ACC   RAM
RTDR:           EQU             H#1      ; D     RAM
;
;
; *****************************************************
ROTR2: DEF  1V,  B#01,4V%D#,4V%D#,        5V%D#,64X      ; ROTATE RAM (2)
;
;               MODE,QUAD,N,SOURCE-DEST,REGISTER
;               [M]      [N]    [13]       [R]
; *****************************************************
;
; SOURCE DESTINATION [14]
;
RTDY:           EQU             D#24     ; D     Y BUS
RTDA:           EQU             D#25     ; D     ACC
RTAY:           EQU             D#28     ; ACC   Y BUS
RTAA:           EQU             D#29     ; ACC   ACC
;
;
; *****************************************************
ROTNR: DEF  1V,  B#11,4V%D#,H#C,      5V%D#,64X  ; ROTATE NON-RAM
;
;               MODE,QUAD,N,FIXED CODE,DESTINATION
;               [M]      [N]              [14]
; *****************************************************
; ******************************************************
; BIT ORIENTED INSTRUCTIONS
; ******************************************************
;
; OPCODES [15]
;
SETNR:          EQU             H#D      ; SET RAM, BIT N
RSTNR:          EQU             H#E      ; RESET RAM, BIT N
TSTNR:          EQU             H#F      ; TEST RAM, BIT N
;
;
; **********************************************************
BOR1: DEF 1V,  B#11,4V%D#,4V%D#,    5V%D#,64X     ; BIT ORIENTED RAM (1)
;
;               MODE,QUAD,N,OPCODE,REGISTER
;               [M]       [N]  [15]     [R]
; **********************************************************
;
;
; OPCODES [16]
;
LD2NR:          EQU             H#C      ; 2^N --- RAM
LDC2NR:         EQU             H#D      ; 2^N --- RAM
A2NR:           EQU             H#E      ; RAM + 2^N - RAM
S2NR:           EQU             H#F      ; RAM - 2^N - RAM
;
;
; **********************************************************
BOR2: DEF 1V,  B#10,4V%D#,4V%D#,    5V%D#,64X     ; BIT ORIENTED RAM (2)
;
;               MODE,QUAD,N,OPCODE,REGISTER
;               [M]       [N]  [16]     [R]
; **********************************************************
;
; OPCODES [17]
;
TSTNA:          EQU             D#0      ; TEST ACC, BIT N
RSTNA:          EQU             D#1      ; RESET ACC, BIT N
SETNA:          EQU             D#2      ; SET ACC, BIT N
A2NA:           EQU             D#4      ; ACC + 2^N -- ACC
S2NA:           EQU             D#5      ; ACC - 2^N --ACC
LD2NA:          EQU             H#6      ; 2^N -- ACC
LDC2NA:         EQU             D#7      ; 2^N -- ACC
TSTND:          EQU             D#16     ; TEST D, BIT N
RSTND:          EQU             D#17     ; RESET D, BIT N
SETND:          EQU             D#18     ; SET D, BIT N
A2NDY:          EQU             D#20     ; D + 2^N -- Y BUS
S2NDY:          EQU             D#21     ; D - 2^N -- Y BUS
LD2NY:          EQU             D#22     ; 2^N -- Y BUS
LDC2NY:         EQU             D#23     ; 2^N -- Y BUS
;
;
; ***********************************************************
BONR: DEF 1V,  B#11,4V%D#,B#1100,     5V%D#,64X    ; BIT ORIENTED NON-RAM
;
;               MODE,QUAD,N,FIXED CODE,OPCODE
;               [M]       [N]            [17]
; ***********************************************************
; ****************************************************
; ROTATE AND MERGE
; ****************************************************
;
; SOURCE-DEST SELECT [U,S,MASK-DEST] [18]
;
;                                        ROT   NON-ROT  MASK-DEST
MDAI:           EQU             H#7     ; D     ACC      I
MDAR:           EQU             H#8     ; D     ACC      RAM
MDRI:           EQU             H#9     ; D     RAM      I
MDRA:           EQU             H#A     ; D     RAM      ACC
MARI:           EQU             H#C     ; ACC   RAM      I
MRAI:           EQU             H#E     ; RAM   ACC      I
;
;
; ************************************************************
ROTM: DEF 1V,  B#01,4V%D#,4V%D#,        5V%D#,64X        ;ROTATE AND MERGE
;
;               MODE,QUAD,N,SOURCE-DEST,REGISTER
;               [M]      [N]    [18]       [R]
; ************************************************************
;
; ****************************************************
; ROTATE AND COMPARE
; ****************************************************
;
; ROT.SRC(U)-NON ROT.SRC(S)/DEST-MASK(S)[19]
;
CDAI:           EQU             H#2     ; D     ACC      I
CDRI:           EQU             H#3     ; D     RAM      I
CDRA:           EQU             H#4     ; D     RAM      ACC
CRAI:           EQU             H#5     ; RAM   ACC      I
;
;
; ***********************************************
ROTC: DEF 1V,  B#01,4V%D#,4V%D#,              5V%D#,64X   ; ROTATE AND COMPARE
;
;               MODE,QUAD,N,SOURCE-DEST-MASK,REGISTER
;               [M]      [N]       [19]        [R]
; ***********************************************
; ****************************************************
; PRIORITIZE
; ****************************************************
;
; SOURCE [20]
;
PRT1A:          EQU             H#7     ; ACC
PR1D:           EQU             H#9     ; D
;
;
; DESTINATION [21]
;
PR1A:           EQU             H#8     ; ACC
PR1Y:           EQU             H#A     ; Y BUS
PR1R:           EQU             H#B     ; RAM
;
; ************************************************
PRT1: DEF 1V,  B#10,4V%D#,             4V%D#,     5V%D#,64X         ; RAM ADDR MASK(S)
;
;               MODE,QUAD,DESTINATION,SOURCE,REG-MASK
;               [M]             [21]      [20]    [R]
; ************************************************
;
;
; DESTINATION [23]
;
PR2A:           EQU             H#0     ; ACC
PR2Y:           EQU             H#2     ; Y BUS
;
; MASK (S) [22]
;
PRA:            EQU             H#8     ; ACC
PRZ:            EQU             H#A     ; 0
PRI:            EQU             H#B     ; I
;
;
; ************************************************
PRT2: DEF 1V,  B#10,4V%D#,  4V%D#,  5V%D#,64X     ; PRIORITIZE RAM
;
;               MODE,QUAD,MASK,DEST,REG-SOURCE
;               [M]       [22] [23]    [R]
; ************************************************
; SOURCE (R) [24]
;
PR3R:           EQU             H#3     ; RAM
PR3A:           EQU             H#4     ; ACC
PR3D:           EQU             H#6     ; D
;
;
; ************************************************
PRT3: DEF 1V,  B#10,4V%D#,  4V%D#,    5V%D#,64X  ; PRIORITIZE RAM
;
;               MODE,QUAD,MASK,SOURCE,REG-DEST
;               [M]       [22]  [24]    [R]
; ************************************************
;
;
; SOURCE (R) [25]
;
PRTA:           EQU             H#4     ; ACC
PRTD:           EQU             H#6     ; D
;
;
; **********************************************
PRTNR: DEF  1V,  B#11,4V%D#,  4V%D#,    5V%D#,64X        ; PRIORITIZE NON-RAM
;
;                  MODE,QUAD,MASK,SOURCE,DESTINATION
;                  [M]       [22]  [25]    [4](NRY,NRA ONLY)
; **********************************************
;
; **********************************************
; CYCLIC REDUNDANCY CHECK
; **********************************************
;
; *******************************************
CRCF: DEF B#11001100011,5V%D#,64X        ; FORWARD
; *******************************************
;
; *******************************************
CRCR: DEF B#11001101001,5V%D#,64X        ; REVERSE
; *******************************************
;
; ********************************************
;
; NOOP
;
; ********************************************
NOOP: DEF H#7140,64X    ; NO OPERATION
; ********************************************
```

```
;
; ************************************************
; STATUS
; ************************************************
;
; OPCODE [26]
;
SONZC:          EQU             5D#3     ; SET OVR,N,C,Z
SL:             EQU             5D#5     ; SET LINK
SF1:            EQU             5D#6     ; SET FLAG 1
SF2:            EQU             5D#9     ; SET FLAG 2
SF3:            EQU             5D#10    ; SET FLAG 3
;
;
; **************************************************
SETST: DEF B#011,H#BA,5V%D#,64X  ; SET STATUS
;
;                           OPCODE
;                            [26]
; **************************************************
;
; OPCODE [27]
;
RONCZ:          EQU             D#3      ; RESET OVR,N,C,Z
RL:             EQU             D#5      ; RESET LINK
RF1:            EQU             D#6      ; RESET FLAG 1
RF2:            EQU             D#9      ; RESET FLAG 2
RF3:            EQU             D#10     ; RESET FLAG 3
;
;****************************************************
RSTST: DEF B#011,H#AA,5V%D#,64X         ; RESET STATUS
;
;                            OPCODE
;                             [27]
; ****************************************************
;
; ****************************************************
SVSTR: DEF  1V,  B#10,H#7A, 5V%D#,64X   ; SAVE STATUS-RAM
;
;                  MODE,QUAD,FIXED,RAM ADDRESS/DEST
;                    [M]                  [R]
; ****************************************************
;
;
;****************************************************
SVSTNR: DEF  1V, B#11,H#7A, 5V%D#,64X    ; SAVE STATUS NON-RAM
;
;                  MODE,QUAD,FIXED,DESTINATION
;                   [M]            [4](NRY,NRA ONLY)
; ****************************************************
```

```
;
; **************************************************
; TEST STATUS
; **************************************************
;
; OPCODE (CT)
;
TNOZ:           EQU             D#0      ; TEST (N OVR) + Z
TNO:            EQU             D#2      ; TEST N OVR
TZ:             EQU             D#4      ; TEST Z
TOVR:           EQU             D#6      ; TEST OVR
TLOW:           EQU             D#8      ; TEST LOW
TC:             EQU             D#10     ; TEST C
TZC:            EQU             D#12     ; TEST Z + C
TN:             EQU             D#14     ; TEST N
TL:             EQU             D#16     ; TEST LINK
TF1:            EQU             D#18     ; TEST FLAG 1
TF2:            EQU             D#20     ; TEST FLAG 2
TF3:            EQU             D#22     ; TEST FLAG 3
;
;
; *****************************************************
TEST: DEF B#011,H#9A,5V%D#,64X    ;  TEST STATUS
;
;                FIXED,     OPCODE
;                           [CT]
; *****************************************************
;
; added DEF and EQU statements
;*******************************
;
;       IMMEDIATE OPERAND
;
IMME:   DEF     16V%D#, 64X
;
;       CT MULTIPLEXER CONTROL
;
CT:     DEF     16X, 4V%D#, 60X
NOZ:    EQU     H#0
NO:     EQU     H#1
Z:      EQU     H#2
OVR:    EQU     H#3
LOW:    EQU     H#4
C:      EQU     H#5
ZC:     EQU     H#6
N:      EQU     H#7
L:      EQU     H#8
F1:     EQU     H#9
F2:     EQU     H#A
F3:     EQU     H#B
;
;       STATUS REGISTER ENABLE
;
SRE:    DEF     20X, B#1, 59X
NOSRE:  DEF     20X, B#0, 59X
;
;       OUTPUT ENABLE Y
;
OEY:    DEF     21X, B#0, 58X
NOOEY:  DEF     21X, B#1, 58X
;
;       INSTRUCTION ENABLE
;
IEN:    DEF     22X, B#0, 57X
NOIEN:  DEF     22X, B#1, 57X
;
;       D-I-LATCH ENABLE
;
DLE:    DEF     23X, B#1, 56X
NODLE:  DEF     23X, B#0, 56X
```

```
;
;
;
;---------------------------------------------------
;     Am2910 COMMANDS AND BRANCH ADDRESSES
; note use of DEF statements - overlay in SRC file
;---------------------------------------------------
JZ:      DEF      24X, H#0, 10V$D#1023, 42X
CJS:     DEF      24X, H#1, 10V$D#1023, 42X
JS:      DEF      24X, H#1, 10V$D#1023, 6Q#36, 36X        ;  UNCONDITIONAL JUMP TO SUBR.
JMAP:    DEF      24X, H#2, 10V$D#1023, 42X
CJP:     DEF      24X, H#3, 10V$D#1023, 42X
JP:      DEF      24X, H#3, 10V$D#1023, 6Q#36, 36X        ;  UNCONDITIONAL JUMP
PUSH:    DEF      24X, H#4, 10V$D#1023, 42X
JSRP:    DEF      24X, H#5, 10V$D#1023, 42X
CJV:     DEF      24X, H#6, 10V$D#1023, 42X
JRP:     DEF      24X, H#7, 10V$D#1023, 42X
RFCT:    DEF      24X, H#8, 10V$D#1023, 42X
RPCT:    DEF      24X, H#9, 10V$D#1023, 42X
CRTN:    DEF      24X, H#A, 10V$D#1023, 42X
RTN:     DEF      24X, H#A, 10V$D#1023, 6Q#36, 36X        ;  UNCONDITIONAL RETURN
CJPP:    DEF      24X, H#B, 10V$D#1023, 42X
LDCT:    DEF      24X, H#C, 10V$D#1023, 42X
LOOP:    DEF      24X, H#D, 10V$D#1023, 42X
CONT:    DEF      24X, H#E, 10V$D#1023, 42X
TWB:     DEF      24X, H#F, 10V$D#1023, 42X
;
;                             |
;                             |
; NOTE: For proper assembly, a "$" must be used in any field which
; will be used to accept a symbolic address in the SRC file.
;
;
;
;
;
;
;     Am2910 CONDITION CODE SELECTIONS
;
IF:      DEF      38X, 5V%D#, B#0, 36X
IFNOT:   DEF      38X, 5V%D#, B#1, 36X
;
;
AE20:    EQU      5Q#10:          ; AM9520 ALIGNMENT ERROR FLAG
CT16:    EQU      5Q#11:          ; AM29116 CONDITIONAL TEST FLAG
EP20:    EQU      5Q#12:          ; AM9520 ERROR PATTERN FLAG
ER20:    EQU      5Q#13:          ; AM9520 ERROR DETECTED FLAG
FAIL:    EQU      5Q#14:          ; UNCONDITIONAL FAILURE OF "TEST"
RDYI:    EQU      5Q#15:          ; NOT READY INPUT (DATA UNAVAILABLE FROM FIFOS)
RDYO:    EQU      5Q#16:          ; NOT READY OUTPUT (FIFOS FULL)
SUCC:    EQU      5Q#17:          ; UNCONDITIONAL SUCCESS OF "TEST"
ATTN:    EQU      5Q#20:          ; ATTENTION
BACK:    EQU      5Q#21:          ; BUS ACKNOWLEDGE
BUSY:    EQU      5Q#22:          ; BUSY
INDX:    EQU      5Q#23:          ; INDEX
SAMD:    EQU      5Q#24:          ; SECTOR / ADDRESS MARK DETECTED
PM2:     EQU      5Q#25:          ; AM9520 PATTERN MATCH 2 FLAG
PM3:     EQU      5Q#26:          ; AM9520 PATTERN MATCH 3 FLAG
PM4:     EQU      5Q#27:          ; AM9520 PATTERN MATCH 4 FLAG
```

```
;
;       MISCELLANEOUS CONTROL SIGNALS
;
ADMC:   DEF     44X, B#0, 35X   ; ADDRESS MARK CONTROL
BFCB:   DEF     45X, B#0, 34X   ; MEMORY BUS FROM DRIVE CONTROL BUS
BFTP:   DEF     46X, B#0, 33X   ; MEMORY BUS FROM TRANSLATE PROM
BF03:   DEF     47X, B#0, 32X   ; MEMORY BUS FROM 9403AS
BF16:   DEF     48X, B#0, 31X   ; MEMORY BUS FROM AM29116
BF2L:   DEF     49X, B#0, 30X   ; MEMORY BUS FROM AM9520 - LOWER BYTE
BF2U:   DEF     50X, B#0, 29X   ; MEMORY BUS FROM AM9520 - UPPER BYTE
BOUT:   DEF     51X, B#0, 28X   ; (DISK) BUS DIRECTION OUT (FROM CONTROLLER)
BT03:   DEF     52X, B#0, 27X   ; MEMORY BUS TO 9403AS
BT16:   DEF     53X, B#0, 26X   ; MEMORY BUS TO AM29116
BT2L:   DEF     54X, B#0, 25X   ; MEMORY BUS TO AM9520 - LOWER BYTE
BT2U:   DEF     55X, B#0, 24X   ; MEMORY BUS TO AM9520 - UPPER BYTE
BT20:   DEF     56X, B#0, 23X   ; MEMORY BUS TO AM9520 - CONTROL INFORMATION
CE2L:   DEF     57X, B#0, 22X   ; CLOCK ENABLE AM9520 TO LOWER-BYTE BUS INT.
CE20:   DEF     58X, B#0, 21X   ; CLOCK ENABLE MEMORY BUS TO AM9520 TRANSFER
CP20:   DEF     59X, B#0, 20X   ; CLOCK PULSE (ACTUAL WAVEFORM) FOR AM9520
CREQ:   DEF     60X, B#0, 19X   ; COMMAND REQUEST
INPT:   DEF     61X, B#0, 18X   ; INPUT SERIAL DATA TO 9403AS
JMPI:   DEF     62X, B#01, 16X  ; JUMP INDIRECT AM29116 REGISTER
NOJMPI: DEF     62X, B#10, 16X  ; NO INDIRECT JUMP
MADR:   DEF     64X, B#0, 15X   ; MEMORY ACCESS
MREA:   DEF     65X, B#0, 14X   ; MEMORY ADDRESS
MWRT:   DEF     66X, B#0, 13X   ; MEMORY WRITE
OUPT:   DEF     67X, B#0, 12X   ; OUTPUT SERIAL DATA FROM 9403AS
PENB:   DEF     68X, B#0, 11X   ; PARAMETER ENABLE
PFPM:   DEF     69X, B#0, 10X   ; SET 9520 P BITS FROM 9520 PM BITS
PF03:   DEF     70X, B#0, 9X    ; PARALLEL FETCH FROM 9403AS
PL03:   DEF     71X, B#0, 8X    ; PARALLEL LOAD INTO 9403AS
PREQ:   DEF     72X, B#0, 7X    ; PARAMETER REQUEST
RDGA:   DEF     73X, B#0, 6X    ; READ GATE
RFIF:   DEF     74X, B#0, 5X    ; RESET FIFO
SAST:   DEF     75X, B#0, 4X    ; SELECT / ATTENTION STROBE
WRGA:   DEF     76X, B#0, 3X    ; WRITE GATE
;
ASCEBC: EQU     Q#0             ; ASCII TO EBCDIC SUBSET PREFIX
BCDEBC: EQU     Q#1             ; BCD TO EBCDIC SUBSET PREFIX
EBCASC: EQU     Q#2             ; EBCDIC SUBSET TO ASCII PREFIX
EBCBCD: EQU     Q#3             ; EBCDIC SUBSET TO BCD PREFIX
;
XLAT:   DEF     77X, 3V%D#      ; TRANSLATE PREFIX
;
;
        END

TOTAL PHASE 1 ERRORS =    0
```

```
;
; CREATED 9/81 TABLER-KITSON
;
;
;
; This SRC file was created for the AMD application note:
; "A High-Performance Intelligent Disk Controller"
; by Otis Tabler and Brad Kitson.
;
; Mnemonics and word format are defined in DISKCTLR.DEF
;
; Advanced Micro Devices reserves the right to make changes in its
; product without notice in order to improve design or performance
; characteristics.  The company assumes no responsibility for the
; use of any circuits or programs described herein.
;
; Am29116 Mnemonics Copyright (c) 1982 Advanced Micro Devices
;
;
;
;
;
;    SECTOR READ / WRITE SUBROUTINE
;*******************************************
;
;       INPUTS:
;
;               FUNCTION CODE IN R0:
;
;                       0 TO READ SECTOR
;
;                       +1 TO WRITE SECTOR
;
;               HEAD NUMBER IN MSB OF R1
;
;               MSB OF TRACK NUMBER IN LSB OF R1
;
;               LSB OF TRACK NUMBER IN MSB OF R2
;
;               SECTOR NUMBER IN LSB OF R2
;
;               START ADDRESS OF RAM SECTOR BUFFER IN R3
;
```

```
       ;
       ;       OUTPUT:
       ;
       ;               R0 CONTAINS:
       ;
       ;                       0       IF THE FUNCTION SPECIFIED WAS COMPLETED
       ;                               EITHER WITHOUT ERROR OR WITH A
       ;                               SUCCESSFULLY CORRECTED READ ERROR
       ;
       ;                       +1      IF THE SECTOR'S HEADER IS BAD
       ;
       ;                       +2      IF AN UNCORRECTABLE ERROR WAS DETECTED IN
       ;                               READING THE SECTOR'S DATA SEGMENT
       ;
       ;  ADDITIONAL MNEMONICS
       ; *********************
       ;
       ;
C001 CRCMSK: EQU     16H#C001   ; CRCF POLYNOMIAL MASK
0010 CRCNIT: EQU     16         ; CRCF NUMBER OF ITERATIONS (D#16 <-- default bas
     NSPASS: EQU     64         ; NUMBER OF SECTOR PASSES (SET THIS EQUAL
0040 ;                            TO THE NUMBER OF SECTORS PER TRACK.)
     RDITCT: EQU     65         ; READ ITERATION COUNT, EQUAL TO THE NUMBER
     ;                          ; OF 16-BIT WORDS (DATA PLUS MODIFIED FIRE
0041 ;                          ; CODE) PER SECTOR, DIVIDED BY TWO, MINUS 1.
0016 PF1:    EQU     22         ; PERIOD FACTOR ONE
000D PF2:    EQU     13         ; PERIOD FACTOR TWO
0059 PF3:    EQU     89         ; PERIOD FACTOR THREE
     PF4:    EQU     23         ; PERIOD FACTOR FOUR
0017 ;
E723 A1LSW:  EQU     H#E723     ; A1 CONSTANT(LEAST SIG. WORD)
0006 A1MSW:  EQU     6          ; A1 CONS.(MOST SIG. WORD)
BFA8 A2LSW:  EQU     H#BFA8     ; A2 CONS.(LEAST SIG. WORD)
D530 A3LSW:  EQU     H#D530     ; A3' CONS.(LEAST SIG. WORD)
     A4LSW:  EQU     H#A928     ; A4' CONS.(LEAST SIG. WORD)
A928 ;
7100 KL128:  EQU     H#7100     ; K(LEAST SIG. WORD) SHIFTED UP
                                ; BY SEVEN PLACES
0477 KM128:  EQU     H#0477     ; K(MOST SIG. WORD) SHIFTED UP
                                ; BY SEVEN PLACES.
EEE2 KLSW:   EQU     H#EEE2     ; K(LEAST SIG. WORD)
     KMSW:   EQU     8          ; K(MOST SIG. WORD)
0008 ;
```

```
      ;
      ;
      ;       IF THE FUNCTION CODE IN R0 EQUALS +1 (WRITE SECTOR), PRECALCULATE
      ;       THE MODIFIED FIRE CODE'S PARTIAL CHECKSUM FOR THE FIRST HALF OF
      ;       THE DATA SEGMENT TO BE WRITTEN.
      ;
0000 SECTIO:  BOR2    W,0,S2NR,R0
      /       &NODLE  &NOIEN  &NOOEY  &NOSRE ;<------  note use of overlayed
      /       &CT     Z       &NOJMPI        ;         DEF statements
      /       &IFNOT  CT16    &CJP    CFCODE ;         signified by "&"
      ;
      ;
      ;       RESET THE AM9520 AND THEN PLACE IT IN COMPUTE CHECK BITS MODE.
      ;       INITIALIZE COUNTER FOR CHECK BITS PRECALCULATION LOOP.
      ;
0001          SONR    W,MOVE,SOI,NRY
      /       &NODLE  &NOIEN  &OEY    &NOSRE
      /       &NOJMPI
      /       &CONT
      ;
0002          IMME    H#0000
      /       &NODLE  &NOIEN  &OEY    &NOSRE
      /       &BT20   &NOJMPI
      /       &CONT
      ;
0003          SONR    W,MOVE,SOI,NRY
      /       &NODLE  &NOIEN  &OEY    &NOSRE
      /       &NOJMPI
      /       &CONT
      ;
0004          IMME    H#0010
      /       &NODLE  &NOIEN  &OEY    &NOSRE
      /       &BT20   &NOJMPI
      /       &LDCT   127
      ;
      ;
      ;       (R3) TO R4
      ;
0005          SOR     W,MOVE,SORY,R3
      /       &DLE    &NOIEN  &OEY    &NOSRE
      /       &NOJMPI
      /       &CONT
      ;
0006          SOR     W,MOVE,SODR,R4
      /       &NODLE  &IEN    &NOOEY  &NOSRE
      /       &NOJMPI
      /       &CONT
      ;
      ;       BEGIN CHECK BITS PRECALCULATION LOOP.
      ;       (R4) TO THE MAR.
      ;
0007 PCPREL:  SOR     W,MOVE,SORY,R4
      /       &NODLE  &NOIEN  &OEY    &NOSRE
      /       &MADR   &NOJMPI
      /       &CONT
      ;
      ;
      ;       CLOCK THE LESS SIGNIFICANT BYTE OF ((MAR)) INTO THE AM9520.
      ;
              NOOP
0008  /       &NODLE  &NOIEN  &NOOEY  &NOSRE
      /       &BT2L   &NOJMPI
      /       &CONT
      ;
              NOOP
0009  /       &NODLE  &NOIEN  &NOOEY  &NOSRE
      /       &CP20   &NOJMPI
      /       &CONT
      ;
      ;
      ;       NOOP FOR TIMING PURPOSES
      ;
              NOOP
000A  /       &NODLE  &NOIEN  &NOOEY  &NOSRE
      /       &NOJMPI
      /       &CONT
      ;
      ;
      ;       CLOCK THE MORE SIGNIFICANT BYTE OF ((MAR)) INTO THE AM9520.
      ;
              NOOP
000B  /       &NODLE  &NOIEN  &NOOEY  &NOSRE
      /       &BT2U   &NOJMPI
      /       &CONT
      ;
      ;
      ;       INCREMENT (R4).
      ;       END CHECK BITS PRECALCULATION LOOP.
      ;
000C          SOR     W,INC,SORR,R4
      /       &NODLE  &IEN    &NOOEY  &NOSRE
      /       &CP20   &NOJMPI
      /       &RPCT   PCPREL
```

```
        ;
        ;       PLACE THE AM9520 IN WRITE CHECK BITS MODE.
        ;       INITIALIZE COUNTER FOR STORE CHECK BITS IN BUFFER LOOP.
        ;
000D            SONR    W,MOVE,SOI,NRY
        /       &NODLE  &NOIEN  &OEY    &NOSRE
        /       &NOJMPI
        /       &CONT
        ;
000E            IMME    H#0011
        /       &NODLE  &NOIEN  &OEY    &NOSRE
        /       &BT20   &NOJMPI
        /       &LDCT   2
        ;
        ;
        ;       BEGIN STORE CHECK BITS IN BUFFER LOOP.
        ;       (R4) TO THE MAR.
        ;       CLOCK OUT NEXT MODIFIED FIRE CODE BYTE TO THE
        ;       LESS-SIGNIFICANT MEMORY BUS INTERFACE REGISTER.
        ;
000F SCBIBL:    SOR     W,MOVE,SORY,R4
        /       &NODLE  &NOIEN  &OEY    &NOSRE
        /       &CP20   &MADR   &NOJMPI
        /       &CONT
        ;
                NOOP
0010    /       &NODLE  &NOIEN  &NOOEY  &NOSRE
        /       &CE2L   &NOJMPI
        /       &CONT
        ;
        ;
        ;       NOOP FOR TIMING PURPOSES
        ;
                NOOP
0011    /       &NODLE  &NOIEN  &NOOEY  &NOSRE
        /       &NOJMPI
        /       &CONT
        ;
        ;
        ;       CLOCK OUT NEXT MODIFIED FIRE CODE BYTE TO THE
        ;       MORE-SIGNIFICANT MEMORY BUS INTERFACE REGISTER.
        ;
                NOOP
0012    /       &NODLE  &NOIEN  &NOOEY  &NOSRE
        /       &CP20   &NOJMPI
        /       &CONT
        ;
        ;
        ;       NOOP FOR TIMING PURPOSES
        ;
                NOOP
0013    /       &NODLE  &NOIEN  &NOOEY  &NOSRE
        /       &NOJMPI
        /       &CONT
        ;
        ;       (BUS INTERFACE REGISTER PAIR) TO (MAR).
        ;       INCREMENT (R4).
        ;       END STORE CHECK BITS IN BUFFER LOOP.
        ;
0014            SOR     W,INC,SORR,R4
        /       &NODLE  &IEN    &NOOEY  &NOSRE
        /       &BF2L   &BF2U   &MWRT   &NOJMPI
        /       &RPCT   SCBIBL
        ;
        ;
        ;       ZERO THE UPPER BYTE OF (R5) AND THEN CLOCK THE 7TH AND LAST
        ;       BYTE OF THE MODIFIED FIRE CODE INTO ITS LOWER BYTE.
        ;
0015            SOR     W,MOVE,SOZR,R5
        /       &NODLE  &IEN    &NOOEY  &NOSRE
        /       &CP20   &NOJMPI
        /       &CONT
        ;
                NOOP
0016    /       &NODLE  &NOIEN  &NOOEY  &NOSRE
        /       &CE2L   &NOJMPI
        /       &CONT
        ;
0017            SOR     B,MOVE,SODR,R5
        /       &DLE    &IEN    &NOOEY  &NOSRE
        /       &BF2L   &BT16   &NOJMPI
        /       &CONT
        ;
        ;
        ;       (R4) TO MAR.
        ;
0018            SOR     W,MOVE,SORY,R4
        /       &NODLE  &NOIEN  &OEY    &NOSRE
        /       &MADR   &NOJMPI
        /       &CONT
        ;
        ;
        ;       (R5) TO (MAR).
        ;
0019            SOR     W,MOVE,SORY,R5
        /       &NODLE  &NOIEN  &OEY    &NOSRE
        /       &BF16   &MWRT   &NOJMPI
        /       &CONT
        ;
```

```
      ;
      ;       CONVERT THE FUNCTION CODE IN R0 TO A MICROCODE BRANCH ADDRESS.
      ;
001A CFCODE:  SONR    W,MOVE,SOI,NRA
      /       &NODLE  &IEN    &NOOEY   &NOSRE
      /       &NOJMPI
      /       &CONT
      ;
001B          IMME    BRTABL
      /       &NODLE  &IEN    &NOOEY   &NOSRE
      /       &NOJMPI
      /       &CONT
      ;
001C          TOR1    W,ADD,TORAA,R0
      /       &NODLE  &IEN    &NOOEY   &NOSRE
      /       &NOJMPI
      /       &CONT
      ;
      ;
      ;       CRCF POLYNOMIAL MASK TO ACC.
      ;
001D          SONR    W,MOVE,SOI,NRA
      /       &NODLE  &IEN    &NOOEY   &NOSRE
      /       &NOJMPI
      /       &CONT
      ;
001E          IMME    CRCMSK
      /       &NODLE  &IEN    &NOOEY   &NOSRE
      /       &NOJMPI
      /       &CONT
      ;
      ;
      ;       CLEAR REGISTER USED TO ACCUMULATE CRCF.
      ;
001F          SOR     W,MOVE,SOZR,R4
      /       &NODLE  &IEN    &NOOEY   &NOSRE
      /       &NOJMPI
      /       &CONT
      ;
      ;
      ;       COPY HEAD BYTE AND TRACK BYTE 1 TO R5.
      ;       SET CRCF LOOP COUNTER.
      ;
0020          SOR     W,MOVE,SORY,R1
      /       &DLE    &IEN    &NOOEY   &NOSRE
      /       &NOJMPI
      /       &LDCT   CRCNIT
      ;
0021          SOR     W,MOVE,SODR,R5
      /       &NODLE  &IEN    &NOOEY   &NOSRE
      /       &NOJMPI
      /       &CONT
      ;
      ;       SHIFT R5 AND SET QLINK.
      ;
0022 CRCFL1:  SHFTR   W,SHUPZ,SHRR,R5
      /       &NODLE  &IEN    &NOOEY   &NOSRE
      /       &NOJMPI
      /       &CONT
      ;
      ;
      ;       ACCUMULATE CRCF.
      ;
0023          CRCF    R4
      /       &NODLE  &IEN    &NOOEY   &NOSRE
      /       &NOJMPI
      /       &RPCT   CRCFL1
      ;
      ;
      ;       COPY TRACK BYTE 2 AND SECTOR BYTE TO R5.
      ;       SET CRCF LOOP COUNTER.
      ;
0024          SOR     W,MOVE,SORY,R2
      /       &DLE    &IEN    &NOOEY   &NOSRE
      /       &NOJMPI
      /       &LDCT   CRCNIT
      ;
0025          SOR     W,MOVE,SODR,R5
      /       &NODLE  &IEN    &NOOEY   &NOSRE
      /       &NOJMPI
      /       &CONT
      ;
      ;
      ;       SHIFT R5 AND SET QLINK.
      ;
0026 CRCFL2:  SHFTR   W,SHUPZ,SHRR,R5
      /       &NODLE  &IEN    &NOOEY   &NOSRE
      /       &NOJMPI
      /       &CONT
      ;
      ;
      ;       ACCUMULATE CRCF.
      ;
0027          CRCF    R4
      /       &NODLE  &IEN    &NOOEY   &NOSRE
      /       &NOJMPI
      /       &RPCT   CRCFL2
```

```
        ;
        ;       INITIALIZE SECTOR PASS LOOP COUNTER.
        ;       ENTER INPUT MODE.
        ;       TURN ON READ GATE.
        ;
        ;
                NOOP
   0028 /       &NODLE  &NOIEN  &NOOEY  &NOSRE
        /       &INPT   &NOJMPI &RDGA
        /       &LDCT   NSPASS
        ;
        ;
        ;       BEGIN SECTOR PASS LOOP.
        ;       TURN ON ADDRESS MARK CONTROL.
        ;       RESET BYTE SYNC ACQUISITION CIRCUITRY AND FIFO ARRAY.
        ;
        SECTL1: NOOP
   0029 /       &NODLE  &NOIEN  &NOOEY  &NOSRE
        /       &ADMC   &INPT   &NOJMPI &RDGA   &RFIF
        /       &CONT
        ;
        ;
        ;       PASS WHEN ADDRESS MARK DETECTED.
        ;
                NOOP
   002A /       &NODLE  &NOIEN  &NOOEY  &NOSRE
        /       &ADMC   &INPT   &NOJMPI &RDGA
        /       &IFNOT  SAMD    &CJP    $
        ;
        ;
        ;       TURN OFF ADDRESS MARK CONTROL.
        ;       PASS WHEN INPUT AVAILABLE FROM FIFO ARRAY.
        ;
                NOOP
   002B /       &NODLE  &NOIEN  &NOOEY  &NOSRE
        /       &INPT   &NOJMPI &RDGA
        /       &IFNOT  RDYI    &CJP    $
        ;
        ;       INPUT RECORDED HEAD NUMBER AND MSB OF RECORDED TRACK NUMBER
        ;       TO R5 AND D-LATCH.
        ;       PASS WHEN INPUT AVAILABLE FROM FIFO ARRAY.
        ;
   002C         SOR     W,MOVE,SODR,R5
        /       &DLE    &IEN    &NOOEY  &NOSRE
        /       &BF03   &BT16   &INPT   &NOJMPI &RDGA
        /       &IFNOT  RDYI    &CJP    $
        ;
        ;
        ;       COMPARE THE CONTENTS OF R1 AND R5.
        ;       IF THEY DISAGREE, EXAMINE THE NEXT SECTOR.
        ;
   002D         TOR1    W,EXOR,TODRR,R1
        /       &NODLE  &NOIEN  &NOOEY  &SRE    &CT     Z
        /       &INPT   &NOJMPI &RDGA
        /       &IF     CT16    &CJP    SECTL2
        ;
        ;
        ;       INPUT LSB OF RECORDED TRACK NUMBER AND RECORDED SECTOR NUMBER
        ;       TO R5 AND D-LATCH.
        ;       PASS WHEN INPUT AVAILABLE FROM FIFO ARRAY.
        ;
                NOOP
   002E /       &NODLE  &NOIEN  &NOOEY  &NOSRE
        /       &PF03   &INPT   &NOJMPI &RDGA
        /       &CONT
        ;
   002F         SOR     W,MOVE,SODR,R5
        /       &DLE    &IEN    &NOOEY  &NOSRE
        /       &BF03   &BT16   &INPT   &NOJMPI &RDGA
        /       &IFNOT  RDYI    &CJP    $
```

```
     ;
     ;        COMPARE THE CONTENTS OF R2 AND R5;
     ;        IF THEY DISAGREE, EXAMINE THE NEXT SECTOR.
     ;
0030          TOR1    W,EXOR,TODRR,R2
     /        &NODLE  &NOIEN  &NOOEY  &SRE     &CT      Z
     /        &PF03   &INPT   &NOJMPI &RDGA
     /        &IF     CT16    &CJP    SECTL2
     ;
     ;
     ;        INPUT RECORDED CRCF BYTES 1 AND 0 TO R5 AND D-LATCH.
     ;
0031          SOR     W,MOVE,SODR,R5
     /        &DLE    &IEN    &NOOEY  &NOSRE
     /        &BF03   &BT16   &INPT   &NOJMPI &RDGA
     /        &CONT
     ;
     ;
     ;        COMPARE THE TWO CRCFS.
     ;        IF THEY AGREE, PROCEED TO READ OR WRITE AS SPECIFIED BY R0.
     ;        OTHERWISE, ASSUME BAD HEADER, TRUE ID UNKNOWN,
     ;        AND CONTINUE LOOP.
     ;
0032          TOR1    W,EXOR,TODRR,R4
     /        &NODLE  &NOIEN  &NOOEY  &SRE     &CT      Z
     /        &INPT   &NOJMPI &RDGA
     /        &IF     CT16    &CJP    MATCH1
     ;
     ;
     ;        TURN OFF READ GATE.
     ;        LEAVE INPUT MODE.
     ;        END SECTOR PASS LOOP.
     ;        NOTICE WE HAVE THREE MICROINSTRUCTION CLOCKS LEFT BEFORE
     ;        IT IS TIME TO BEGIN WRITING OR RE-SYNC AND BEGIN READING.
     ;
0033 SECTL2:  SOR     W,MOVE,SORY,R0
     /        &NODLE  &NOIEN  &NOOEY  &NOSRE
     /        &NOJMPI
     /        &RPCT   SECTL1
     ;
     ;
     ;        IF SECTOR SEARCH COUNT EXHAUSTED, LOAD +1 INTO R0 AND RETURN
     ;
0034          BOR2    W,LD2NR,0,R0
     /        &NODLE  &IEN    &NOOEY  &NOSRE
     /        &NOJMPI
     /        &RTN
     ;
     ;        SUCCESSFUL MATCH.
     ;
0035 MATCH1:  SOR     W,MOVE,SORY,R0
     /        &NODLE  &NOIEN  &OEY    &NOSRE
     /        &JMPI
     /        &CONT
     ;
0036 BRTABL:  SOR     W,NEG,SORA,R3
     /        &NODLE  &IEN    &NOOEY  &NOSRE
     /        &NOJMPI &RFIF
     /        &JP     RDSEC1
     ;
0037          SOR     W,NEG,SORA,R3
     /        &NODLE  &IEN    &NOOEY  &NOSRE
     /        &NOJMPI &RFIF
     /        &JP     WRSEC1
```

```
        ;
        ;       READ SECTOR
        ;
        ;       (R3) TO R4 AND MAR.
        ;       ENTER INPUT MODE AGAIN.
        ;       TURN READ GATE BACK ON.
        ;       INITIALIZE COUNTER FOR READ DATA SEGMENT LOOP.
        ;       DURING THAT LOOP, AM9520 READ HIGH SPEED IS PERFORMED ON THE
        ;       FIRST HALF OF THE SEGMENT BUFFER.  THE SECOND HALF OF THE BUFFER
        ;       IS PROCESSED BY THE AM9520 IN THE READ HIGH SPEED COMPLETION LOOP.
        ;
0038 RDSEC1:  SOR     W,NEG,SOAR,R4
        /       &NODLE  &IEN    &OEY    &NOSRE
        /       &INPT   &NOJMPI &MADR   &RDGA   &RFIF
        /       &LDCT   RDITCT
        ;
        ;
        ;       (R3) - 1 TO R5.
        ;       PASS WHEN INPUT AVAILABLE FROM FIFO ARRAY.
        ;
0039            SOR     W,COMP,SOAR,R5
        /       &NODLE  &IEN    &NOOEY  &NOSRE
        /       &INPT   &NOJMPI &RDGA
        /       &IFNOT  RDYI    &CJP    $
        ;
        ;       RESET THE AM9520 AND THEN PLACE IT IN READ HIGH SPEED MODE.
        ;
003A            SONR    W,MOVE,SOI,NRY
        /       &NODLE  &NOIEN  &OEY    &NOSRE
        /       &INPT   &NOJMPI &RDGA
        /       &CONT
        ;
003B            IMME    H#0003
        /       &NODLE  &NOIEN  &OEY    &NOSRE
        /       &INPT   &NOJMPI &RDGA
        /       &CONT
        ;
003C            SONR    W,MOVE,SOI,NRY
        /       &NODLE  &NOIEN  &OEY    &NOSRE
        /       &INPT   &NOJMPI &RDGA
        /       &CONT
        ;
003D            IMME    H#0013
        /       &NODLE  &NOIEN  &OEY    &NOSRE
        /       &BT20   &INPT   &NOJMPI &RDGA
        /       &CONT
        ;
                NOOP
003E    /       &NODLE  &NOIEN  &NOOEY  &NOSRE
        /       &INPT   &NOJMPI &PF03   &RDGA
        /       &CONT
```

```
      ;
      ;        BEGIN READ DATA SEGMENT LOOP.
      ;        TRANSFER NEXT WORD FROM FIFO ARRAY TO (MAR).
      ;        PASS WHEN FIFO INPUT AGAIN BECOMES AVAILABLE.
      ;
      RDSEC2:  NOOP
003F  /        &NODLE   &NOIEN   &NOOEY   &NOSRE
      /        &BF03    &INPT    &NOJMPI  &MWRT    &RDGA
      /        &IFNOT   RDYI     &CJP     $
      ;
      ;
      ;        INCREMENT (R5) AND TRANSFER THIS TO THE MAR.
      ;
0040           SOR      W,INC,SORR,R5
      /        &NODLE   &IEN     &OEY     &NOSRE
      /        &INPT    &NOJMPI  &MADR    &RDGA
      /        &CONT
      ;
      ;
      ;        CLOCK LESS SIGNIFICANT BYTE OF ((MAR)) INTO THE AM9520.
      ;        INCREMENT (R4) AND TRANSFER THIS TO THE MAR.
      ;
0041           SOR      W,INC,SORR,R4
      /        &NODLE   &IEN     &OEY     &NOSRE
      /        &BT2L    &CP20    &INPT    &NOJMPI  &MADR    &MREA    &PF03    &RDGA
      /        &CONT
      ;
      ;
      ;        TRANSFER NEXT WORD FROM FIFO ARRAY TO (MAR).
      ;        PASS WHEN FIFO INPUT AGAIN BECOMES AVAILABLE.
      ;
               NOOP
0042  /        &NODLE   &NOIEN   &NOOEY   &NOSRE
      /        &BF03    &INPT    &NOJMPI  &MWRT    &RDGA
      /        &IFNOT   RDYI     &CJP     $
      ;
      ;        TRANSFER (R5) TO THE MAR AGAIN.
      ;
0043           SOR      W,MOVE,SORR,R5
      /        &NODLE   &IEN     &OEY     &NOSRE
      /        &INPT    &NOJMPI  &MADR    &RDGA
      /        &CONT
      ;
      ;
      ;        THIS TIME, CLOCK THE MORE SIGNIFICANT BYTE OF ((MAR))
      ;        INTO THE AM9520.
      ;        END READ DATA SEGMENT LOOP.
      ;
0044           SOR      W,INC,SORR,R4
      /        &NODLE   &IEN     &OEY     &NOSRE
      /        &BT2U    &CP20    &INPT    &NOJMPI  &MADR    &MREA    &PF03    &RDGA
      /        &RPCT    RDSEC2
      ;
      ;
      ;        INITIALIZE COUNTER FOR READ HIGH SPEED COMPLETION LOOP.
      ;
               NOOP
0045  /        &NODLE   &NOIEN   &NOOEY   &NOSRE
      /        &NOJMPI
      /        &LDCT    RDITCT
      ;
      ;       BEGIN READ HIGH SPEED COMPLETION LOOP.
      ;       INCREMENT (R5) AND TRANSFER THIS TO THE MAR.
      ;
0046 RDSEC3:  SOR     W,INC,SORR,R5
      /       &NODLE  &IEN     &OEY     &NOSRE
      /       &NOJMPI &MADR
      /       &CONT
      ;
      ;
      ;       NOOP FOR TIMING PURPOSES
      ;
              NOOP
0047  /       &NODLE  &NOIEN   &NOOEY   &NOSRE
      /       &NOJMPI
      /       &CONT
      ;
      ;
      ;       CLOCK LESS SIGNIFICANT BYTE OF ((MAR)) INTO THE AM9520.
      ;
              NOOP
0048  /       &NODLE  &NOIEN   &NOOEY   &NOSRE
      /       &BT2L   &CP20    &NOJMPI &MREA
      /       &CONT
      ;
      ;
      ;       NOOP FOR TIMING PURPOSES
      ;
              NOOP
0049  /       &NODLE  &NOIEN   &NOOEY   &NOSRE
      /       &NOJMPI
      /       &CONT
      ;
      ;
      ;       NOOP FOR TIMING PURPOSES
      ;
              NOOP
004A  /       &NODLE  &NOIEN   &NOOEY   &NOSRE
      /       &NOJMPI
      /       &CONT
      ;
      ;       CLOCK MORE SIGNIFICANT BYTE OF ((MAR)) INTO THE AM9520.
      ;       END READ HIGH SPEED COMPLETION LOOP.
      ;
              NOOP
004B  /       &NODLE  &NOIEN   &NOOEY   &NOSRE
      /       &BT2U   &CP20    &NOJMPI &MREA
      /       &RPCT   RDSEC3
      ;
      ;
      ;       WAS AN ERROR DETECTED BY THE AM9520?
      ;
              NOOP
004C  /       &NODLE  &NOIEN   &NOOEY   &NOSRE
      /       &NOJMPI
      /       &IF     ER20     &CJP     RDSEC4
      ;
      ;
      ;       NO; LOAD 0 INTO R0 AND RETURN.
      ;
004D          SOR     W,MOVE,SOZR,R0
      /       &NODLE  &IEN     &NOOEY   &NOSRE
      /       &NOJMPI
      /       &RTN
      ;
      ;
      ;       YES; IF THE ERROR IS A CORRECTABLE ONE, LOCATE AND CORRECT IT.
      ;       THE ERROR IS LOCATED USING THE CORRECT HIGH SPEED EQUATION:
      ;
      ;               L=NK - (M1A1 + M2A2 + M3A3 + M4A4)
      ;
      ;       WHERE  K,A1,A2,A3,A4 ARE CONSTANTS
      ;       AND  M1,M2,M3,M4 MUST BE CALCULATED.
      ;
      ;       THE ERROR IS CORRECTED BY PERFORMING AN EXOR
      ;       FUNCTION ON THE BURST ERROR IN MEMORY WITH AN
      ;       ERROR PATTERN(EP) PROVIDED BY THE BEP(9520).
      ;
      ;       INITIALIZE CORRECT HIGH SPEED,SET P0=1,& REP=1
```

```
004E RDSEC4: SONR     W,MOVE,SOI,NRY
      /      &NODLE  &NOIEN  &OEY    &NOSRE
      /      &NOJMPI
      /      &CONT
      ;
004F         IMME     H#001F
      /      &NODLE  &NOIEN  &OEY    &NOSRE
      /      &NOJMPI &BT20   &BF16
      /      &CONT
      ;
      ;
      ;      CLEAR R8(M1).
      ;
0050         SOR      W,MOVE,SOZR,R8
      /      &NODLE  &IEN    &NOOEY  &NOSRE
      /      &NOJMPI
      /      &CONT
      ;
      ;
      ;      PERIOD FACTOR 1(PF1) TO ACC.
      ;
0051         SONR     W,MOVE,SOI,NRA
      /      &NODLE  &IEN    &NOOEY  &NOSRE
      /      &NOJMPI
      /      &CONT
      ;
0052         IMME     PF1
      /      &NODLE  &IEN    &NOOEY  &NOSRE
      /      &NOJMPI
      /      &CONT

      ;
      ;      TEST FOR ERROR PATTERN PRESENT.
      ;
      M1:    NOOP
0053  /      &NODLE  &NOIEN  &NOOEY  &NOSRE
      /      &NOJMPI
      /      &IF      EP20    &CJP    M234I
      ;
      ;
      ;      EP NOT PRESENT,
      ;      DECREMENT ACC.
      ;      TEST FOR ALIGNMENT EXCEPTION(AE).
      ;
0054         BONR     W,0,S2NA
      /      &NODLE  &IEN    &NOOEY  &SRE
      /      &NOJMPI &CP20
      /      &IF      AE20    &CJP    AE
      ;
      ;
      ;      AE NOT PRESENT,
      ;      ADD 8 TO R8.
      ;
0055         BOR2     W,3,A2NR,R8
      /      &NODLE  &IEN    &NOOEY  &NOSRE
      /      &NOJMPI
      /      &JP      LINK
      ;
      ;
      ;      AE PRESENT;
      ;      INC R8.
      ;
0056  AE:    BOR2     W,0,A2NR,R8
      /      &NODLE  &IEN    &NOOEY  &NOSRE
      /      &NOJMPI
      /      &CONT
      ;
      ;        TEST FOR PERIOD FACTOR EXCEEDED (UNCORRECTABLE ERROR).
      ;
      LINK:    NOOP
0057  /        &NODLE   &NOIEN   &NOOEY   &NOSRE   &CT      N
      /        &NOJMPI
      /        &IFNOT   CT16     &CJP     M1
      ;
      ;
      ;        PERIOD FACTOR EXCEEDED (UNCORRECTABLE ERROR);
      ;        SET R0 = 2.
      ;        RETURN.
      ;
0058  ERR:     BOR2     W,1,LD2NR,R0
      /        &NODLE   &IEN     &OEY     &NOSRE
      /        &NOJMPI  &BT20    &BF16
      /        &RTN
      ;
      ;
      ;        EP PRESENT, CALCULATE M2,M3,M4.
      ;
      ;        PERIOD FACTOR 2(PF2) TO R9(M2).
      ;        PERIOD FACTOR 3(PF3) TO R10(M3).
      ;        PERIOD FACTOR 4(PF4) TO R11(M4).
      ;
0059  M234I:   SOR      W,MOVE,SOI,R9
      /        &NODLE   &IEN     &NOOEY   &NOSRE
      /        &NOJMPI
      /        &CONT
      ;
005A           IMME     PF2
      /        &NODLE   &IEN     &NOOEY   &NOSRE
      /        &NOJMPI
      /        &CONT
      ;
005B           SOR      W,MOVE,SOI,R10
      /        &NODLE   &IEN     &NOOEY   &NOSRE
      /        &NOJMPI
      /        &CONT
      ;
005C           IMME     PF3
      /        &NODLE   &IEN     &NOOEY   &NOSRE
      /        &NOJMPI
      /        &CONT
      ;
005D           SOR      W,MOVE,SOI,R11
      /        &NODLE   &IEN     &NOOEY   &NOSRE
      /        &NOJMPI
      /        &CONT
      ;
005E           IMME     PF4
      /        &NODLE   &IEN     &NOOEY   &NOSRE
      /        &NOJMPI
      /        &CONT
      ;
      ;
      ;        JUMP TABLE ADDRESS(TAB1) TO R12.
      ;
005F           SOR      W,MOVE,SOI,R12
      /        &NODLE   &IEN     &NOOEY   &NOSRE
      /        &NOJMPI
      /        &CONT
      ;
0060           IMME     TAB1
      /        &NODLE   &IEN     &NOOEY   &NOSRE
      /        &NOJMPI
      /        &CONT
      ;
      ;
      ;        JUMP INDIRECT TO TAB1 VIA PM2-4.
      ;
0061           ROTM     W,15,MDRI,R12
      /        &DLE     &IEN     &NOOEY   &NOSRE
      /        &NOJMPI  &PFPM    &BF2U    &BT16
      /        &CONT
      ;
0062           IMME     H#0007
      /        &NODLE   &IEN     &OEY     &NOSRE
      /        &JMPI    &PFPM    &BF2U    &BT16
      /        &JP      $
```

```
     ;
     ;        R9 = PF2 - R9.
     ;
0063 MFIX:    TOR1     W,TORIR,SUBR,R9
     /        &NODLE   &IEN     &NOOEY   &NOSRE
     /        &NOJMPI
     /        &CONT
     ;
0064          IMME     PF2
     /        &NODLE   &IEN     &NOOEY   &NOSRE
     /        &NOJMPI
     /        &CONT
     ;
     ;
     ;        R10 = PF3 - R10.
     ;
0065          TOR1     W,TORIR,SUBR,R10
     /        &NODLE   &IEN     &NOOEY   &NOSRE
     /        &NOJMPI
     /        &CONT
     ;
0066          IMME     PF3
     /        &NODLE   &IEN     &NOOEY   &NOSRE
     /        &NOJMPI
     /        &CONT
     ;
     ;
     ;        R11 = PF4 - R11.
     ;
0067          TOR1     W,TORIR,SUBR,R11
     /        &NODLE   &IEN     &NOOEY   &NOSRE
     /        &NOJMPI
     /        &CONT
     ;
0068          IMME     PF4
     /        &NODLE   &IEN     &NOOEY   &NOSRE
     /        &NOJMPI
     /        &CONT

     ;
     ;        0 TO R7 (LOCATION ACC MSW,LAC).
     ;
0069 M1A1:    SOR      W,MOVE,SOZR,R7
     /        &NODLE   &IEN     &NOOEY   &NOSRE
     /        &NOJMPI
     /        &CONT
     ;
     ;
     ;        A1 TO R12(LSW),R13(MSW).
     ;
006A          SOR      W,MOVE,SOI,R12
     /        &NODLE   &IEN     &NOOEY   &NOSRE
     /        &NOJMPI
     /        &CONT
     ;
006B          IMME     A1LSW
     /        &NODLE   &IEN     &NOOEY   &NOSRE
     /        &NOJMPI
     /        &CONT
     ;
006C          SOR      W,MOVE,SOI,R13
     /        &NODLE   &IEN     &NOOEY   &NOSRE
     /        &NOJMPI
     /        &CONT
     ;
006D          IMME     A1MSW
     /        &NODLE   &IEN     &NOOEY   &NOSRE
     /        &NOJMPI
     /        &LDCT    4

     ;
     ;        R8 TO D.
     ;        LAC = M1A1, MULTIPLY M1A1.
     ;
006E          SOR      W,MOVE,SORY,R8
     /        &DLE     &NOIEN   &OEY     &NOSRE
     /        &NOJMPI
     /        &JS      MUL
```

```
      ;
      ;       A2 TO R12,R13.
      ;
006F M2A2:    IMME     A2LSW
      /       &NODLE   &IEN     &NOOEY   &NOSRE
      /       &NOJMPI
      /       &CONT
      ;
0070          BOR2     W,3,LD2NR,R13
      /       &NODLE   &IEN     &NOOEY   &NOSRE
      /       &NOJMPI
      /       &LDCT    3
      ;
      ;
      ;       R9 TO D.
      ;       LAC = LAC + M2A2.
      ;
0071          SOR      W,MOVE,SORY,R9
      /       &DLE     &NOIEN   &OEY     &NOSRE
      /       &NOJMPI
      /       &JS      MUL
      ;
      ;
      ;       A3'(A3 - 4K) TO R12,R13.
      ;
0072 M3A3:    IMME     A3LSW
      /       &NODLE   &IEN     &NOOEY   &NOSRE
      /       &NOJMPI
      /       &CONT
      ;
0073          BOR2     W,1,LD2NR,R13
      /       &NODLE   &IEN     &NOOEY   &NOSRE
      /       &NOJMPI
      /       &LDCT    6
      ;
      ;
      ;       R10 TO D.
      ;       LAC = LAC + M3A3.
      ;
0074          SOR      W,MOVE,SORY,R10
      /       &DLE     &NOIEN   &OEY     &NOSRE
      /       &NOJMPI
      /       &JS      MUL
      ;
      ;       A4'(A4 - 4K) TO R12,R13.
      ;
0075 M4A4:    IMME     A4LSW
      /       &NODLE   &IEN     &NOOEY   &NOSRE
      /       &NOJMPI
      /       &CONT
      ;
0076          BOR2     W,3,LD2NR,R13
      /       &NODLE   &IEN     &NOOEY   &NOSRE
      /       &NOJMPI
      /       &LDCT    4
      ;
      ;
      ;       R11 TO D.
      ;       LAC = LAC + M4A4.
      ;
0077          SOR      W,MOVE,SORY,R11
      /       &DLE     &NOIEN   &OEY     &NOSRE
      /       &NOJMPI
      /       &JS      MUL
      ;
      ;
      ;       PRESHIFTED DIVISOR(K) TO R12,R13,D.
      ;       LAC = REM(M1A1 + M2A2 + M3A3 + M4A4) / K.
      ;       LAC = -L + K.
      ;
0078          IMME     KL128
      /       &NODLE   &IEN     &NOOEY   &NOSRE
      /       &NOJMPI
      /       &CONT
      ;
0079          SOR      W,MOVE,SOI,R13
      /       &DLE     &IEN     &OEY     &NOSRE
      /       &NOJMPI
      /       &LDCT    6
      ;
007A          IMME     KM128
      /       &DLE     &IEN     &OEY     &NOSRE
      /       &NOJMPI
      /       &JS      DIV
```

```
     ;
     ;         L = LAC = K - LAC.
     ;
007B SUBK:     IMME     KLSW
     /         &NODLE   &IEN      &NOOEY   &NOSRE
     /         &NOJMPI
     /         &CONT
     ;
007C           TOR1     W,TORIR,SUBRC,R7
     /         &NODLE   &IEN      &NOOEY   &NOSRE
     /         &NOJMPI
     /         &CONT
     ;
007D           IMME     KMSW
     /         &NODLE   &IEN      &NOOEY   &NOSRE
     /         &NOJMPI
     /         &CONT
     ;
     ;
     ;         0 TO R8.
     ;
007E XORMEM:   SOR      W,MOVE,SOZR,R8
     /         &NODLE   &IEN      &NOOEY   &NOSRE
     /         &NOJMPI
     /         &CONT
     ;
     ;
     ;         0 TO R9.
     ;
007F           SOR      W,MOVE,SOZR,R9
     /         &NODLE   &IEN      &NOOEY   &NOSRE
     /         &NOJMPI
     /         &CONT
     ;
     ;
     ;         0 TO ACC.
     ;
0080           SOR      W,MOVE,SORA,R9
     /         &NODLE   &IEN      &NOOEY   &NOSRE
     /         &NOJMPI
     /         &CONT

     ;
     ;         ROTATE R6 DOWN BY FOUR TO OBTAIN WORD ADDRESS
     ;         AND STORE IN ACC.
     ;
0081           ROTM     W,12,MRAI,R6
     /         &NODLE   &IEN      &NOOEY   &NOSRE
     /         &NOJMPI
     /         &CONT
     ;
0082           IMME     H#0FFF
     /         &NODLE   &IEN      &NOOEY   &NOSRE
     /         &NOJMPI
     /         &CONT
     ;
     ;
     ;         MASK UPPER 12 BITS OF R6 TO OBTAIN FIRST BIT OF
     ;         BURST ERROR AND STORE IN R6.
     ;
0083           TOR1     W,TORIR,AND,R6
     /         &NODLE   &IEN      &NOOEY   &NOSRE
     /         &NOJMPI
     /         &CONT
     ;
0084           IMME     H#000F
     /         &NODLE   &IEN      &NOOEY   &NOSRE
     /         &NOJMPI
     /         &CONT
     ;
     ;
     ;         JUMP INDIRECT TO TAB2 VIA R6.
     ;
0085           TOR1     W,TORIR,ADD,R6
     /         &NODLE   &IEN      &NOOEY   &NOSRE
     /         &NOJMPI
     /         &CONT
     ;
0086           IMME     TAB2
     /         &NODLE   &IEN      &OEY     &NOSRE
     /         &JMPI
     /         &CONT
     ;
     ;          MADR = ACC = R4 - ACC.
     ;
0087 XOR:       TOR1     W,TORAA,SUBS,R4
     /          &NODLE   &IEN     &NOOEY   &NOSRE
     /          &NOJMPI
     /          &CONT
     ;
0088            BONR     W,0,S2NA
     /          &NODLE   &IEN     &OEY     &NOSRE
     /          &MADR    &NOJMPI
     /          &CONT
     ;
     ;
     ;          R8 = R8 XOR MEM.
     ;
0089            TOR1     W,TODRR,EXOR,R8
     /          &DLE     &IEN     &NOOEY   &NOSRE
     /          &NOJMPI &BT16
     /          &CONT
     ;
     ;
     ;          R8 TO MEMORY.
     ;
008A            SOR      W,MOVE,SORY,R8
     /          &NODLE   &NOIEN   &OEY     &NOSRE
     /          &NOJMPI &MWRT    &BF16
     /          &CONT
     ;
     ;
     ;          MADR = ACC + 1.
     ;
008B            SONR     W,INC,SOA,NRY
     /          &NODLE   &NOIEN   &OEY     &NOSRE
     /          &NOJMPI &MADR
     /          &CONT
     ;
     ;
     ;          R9 = R9 XOR MEM.
     ;
008C            TOR1     W,TODRR,EXOR,R9
     /          &DLE     &IEN     &NOOEY   &NOSRE
     /          &NOJMPI &BT16
     /          &CONT
     ;
     ;          R9 TO MEMORY.
     ;
008D            SOR      W,MOVE,SORY,R9
     /          &NODLE   &NOIEN   &OEY     &NOSRE
     /          &NOJMPI &MWRT    &BF16
     /          &CONT
     ;
     ;
     ;          0 TO R0 (ERROR CORRECTED FLAG).
     ;          RETURN.
     ;
008E            SOR      W,MOVE,SOZR,R0
     /          &NODLE   &IEN     &OEY     &NOSRE
     /          &NOJMPI &BT20    &BF16
     /          &RTN
```

```
        ;       TABLE 1 IS USED TO CALCULATE
        ;       M2,M3,M4 AND DETECT UNCORRECTABLE ERRORS.
        ;       EACH MICROINSTRUCTION PATH DECREMENTS THE
        ;       APPROPRIATE REGISTER(S) AND CHECKS FOR VALUES
        ;       EXCEEDING THE 56-BIT POLYNOMIAL PERIOD
        ;       FACTOR LIMITS.
        ;
                ALIGN   8
0090 ;
0090 TAB1:      BOR2    W,0,S2NR,R9
     /          &NODLE  &IEN     &NOOEY  &SRE
     /          &NOJMPI &CP20    &PFPM
     /          &JP     PM234
     ;
0091            BOR2    W,0,S2NR,R11
     /          &NODLE  &IEN     &NOOEY  &SRE
     /          &NOJMPI &CP20    &PFPM
     /          &JP     TM34
     ;
0092            BOR2    W,0,S2NR,R9
     /          &NODLE  &IEN     &NOOEY  &SRE
     /          &NOJMPI &CP20    &PFPM
     /          &JP     TM24
     ;
0093            BOR2    W,0,S2NR,R11
     /          &NODLE  &IEN     &NOOEY  &SRE
     /          &NOJMPI &CP20    &PFPM
     /          &JP     TM1
     ;
0094            BOR2    W,0,S2NR,R10
     /          &NODLE  &IEN     &NOOEY  &SRE
     /          &NOJMPI &CP20    &PFPM
     /          &JP     TM23
     ;
0095            BOR2    W,0,S2NR,R10
     /          &NODLE  &IEN     &NOOEY  &SRE
     /          &NOJMPI &CP20    &PFPM
     /          &JP     TM1
     ;
0096            BOR2    W,0,S2NR,R9
     /          &NODLE  &IEN     &NOOEY  &SRE
     /          &NOJMPI &CP20    &PFPM
     /          &JP     TM1
     ;
0097            SOR     W,MOVE,SOZR,R6
     /          &NODLE  &IEN     &NOOEY  &SRE
     /          &NOJMPI &PFPM
     /          &JP     MFIX
     TM34:      NOOP
0098 /          &NODLE  &NOIEN   &NOOEY  &NOSRE
     /          &NOJMPI &CP20    &PFPM
     /          &JP     PM34
     ;
     TM24:      NOOP
0099 /          &NODLE  &NOIEN   &NOOEY  &NOSRE
     /          &NOJMPI &CP20    &PFPM
     /          &JP     PM24
     ;
     TM23:      NOOP
009A /          &NODLE  &NOIEN   &NOOEY  &NOSRE
     /          &NOJMPI &CP20    &PFPM
     /          &JP     PM23
     ;
     TM1:       NOOP
009B /          &NODLE  &NOIEN   &NOOEY  &NOSRE
     /          &NOJMPI &CP20    &PFPM
     /          &CONT
     ;
                NOOP
009C /          &NODLE  &NOIEN   &NOOEY  &NOSRE
     /          &NOJMPI &CP20    &PFPM
     /          &JP     PM1
     ;
009D PM234:     BOR2    W,0,S2NR,R11
     /          &NODLE  &IEN     &NOOEY  &SRE     &CT     N
     /          &NOJMPI &CP20    &PFPM
     /          &IF     CT16     &CJP    ERR
     ;
009E PM34:      BOR2    W,0,S2NR,R10
     /          &NODLE  &IEN     &NOOEY  &SRE     &CT     N
     /          &NOJMPI &CP20    &PFPM
     /          &IF     CT16     &CJP    ERR
     ;
009F PM1:       ROTM    W,15,MDRI,R12
     /          &DLE    &IEN     &NOOEY  &NOSRE   &CT     N
     /          &NOJMPI &PFPM    &BT16   &BF2U
     /          &IF     CT16     &CJP    LINK
     ;
00A0            IMME    H#0007
     /          &NODLE  &IEN     &OEY    &NOSRE
     /          &JMPI   &PFPM    &BT16   &BF2U
     /          &JP     $
```

```
00A1 PM24:   BOR2     W,0,S2NR,R11
     /       &NODLE   &IEN     &NOOEY   &SRE     &CT      N
     /       &NOJMPI  &CP20    &PFPM
     /       &IFNOT   CT16     &CJP     PM1
     ;
             NOOP
00A2 /       &NODLE   &NOIEN   &NOOEY   &NOSRE
     /       &NOJMPI
     /       &JP      ERR
     ;
00A3 PM23:   BOR2     W,0,S2NR,R9
     /       &NODLE   &IEN     &NOOEY   &SRE     &CT      N
     /       &NOJMPI  &CP20    &PFPM
     /       &IFNOT   CT16     &CJP     PM1
     ;
             NOOP
00A4 /       &NODLE   &NOIEN   &NOOEY   &NOSRE
     /       &NOJMPI
     /       &JP      ERR
     ;
     ;
     ;       TABLE 2 IS USED FOR ROTATING MEMORY WORD(S)
     ;       VIA ROTATE AND MERGE INSTRUCTION(S) SO THAT
     ;       BURST ERRORS CAN BE ALIGNED WITH THE ERROR
     ;       PATTERN PROVIDED BY THE BEP(9520).
     ;
00A5 TAB2:   ROTM     W,9,MDRI,R8
     /       &DLE     &IEN     &NOOEY   &NOSRE
     /       &NOJMPI  &BT16    &BF2U    &BF2L
     /       &JP      REP1
     ;
00A6         ROTM     W,10,MDRI,R8
     /       &DLE     &IEN     &NOOEY   &NOSRE
     /       &NOJMPI  &BT16    &BF2U    &BF2L
     /       &JP      REP2
     ;
00A7         ROTM     W,11,MDRI,R8
     /       &DLE     &IEN     &NOOEY   &NOSRE
     /       &NOJMPI  &BT16    &BF2U    &BF2L
     /       &JP      REP3
     ;
00A8         ROTM     W,12,MDRI,R8
     /       &DLE     &IEN     &NOOEY   &NOSRE
     /       &NOJMPI  &BT16    &BF2U    &BF2L
     /       &JP      REP4

00A9         ROTM     W,13,MDRI,R8
     /       &DLE     &IEN     &NOOEY   &NOSRE
     /       &NOJMPI  &BT16    &BF2U    &BF2L
     /       &JP      REP5
     ;
00AA         ROTM     W,14,MDRI,R8
     /       &DLE     &IEN     &NOOEY   &NOSRE
     /       &NOJMPI  &BT16    &BF2U    &BF2L
     /       &JP      REP6
     ;
00AB         ROTM     W,15,MDRI,R8
     /       &DLE     &IEN     &NOOEY   &NOSRE
     /       &NOJMPI  &BT16    &BF2U    &BF2L
     /       &JP      REP7
     ;
00AC         ROTM     W,0,MDRI,R8
     /       &DLE     &IEN     &NOOEY   &NOSRE
     /       &NOJMPI  &BT16    &BF2U    &BF2L
     /       &JP      REP8
     ;
00AD         ROTM     W,1,MDRI,R8
     /       &DLE     &IEN     &NOOEY   &NOSRE
     /       &NOJMPI  &BT16    &BF2U    &BF2L
     /       &JP      REP9
     ;
00AE         ROTM     W,2,MDRI,R8
     /       &DLE     &IEN     &NOOEY   &NOSRE
     /       &NOJMPI  &BT16    &BF2U    &BF2L
     /       &JP      REP10
     ;
00AF         ROTM     W,3,MDRI,R8
     /       &DLE     &IEN     &NOOEY   &NOSRE
     /       &NOJMPI  &BT16    &BF2U    &BF2L
     /       &JP      REP11
     ;
00B0         ROTM     W,4,MDRI,R8
     /       &DLE     &IEN     &NOOEY   &NOSRE
     /       &NOJMPI  &BT16    &BF2U    &BF2L
     /       &JP      REP12
     ;
00B1         ROTM     W,5,MDRI,R8
     /       &DLE     &IEN     &NOOEY   &NOSRE
     /       &NOJMPI  &BT16    &BF2U    &BF2L
     /       &JP      REP13
```

```
00B2          ROTM     W,6,MDRI,R8
       /      &DLE     &IEN     &NOOEY   &NOSRE
       /      &NOJMPI  &BT16    &BF2U    &BF2L
       /      &JP      REP14
       ;
00B3          ROTM     W,7,MDRI,R8
       /      &DLE     &IEN     &NOOEY   &NOSRE
       /      &NOJMPI  &BT16    &BF2U    &BF2L
       /      &JP      REP15
       ;
00B4          ROTM     W,8,MDRI,R8
       /      &DLE     &IEN     &NOOEY   &NOSRE
       /      &NOJMPI  &BT16    &BF2U    &BF2L
       /      &JP      REP16
00B5 REP1:    IMME     H#0001
       /      &DLE     &IEN     &NOOEY   &NOSRE
       /      &NOJMPI  &BT16    &BF2U    &BF2L
       /      &JP      RM1
       ;
00B6 REP2:    IMME     H#0003
       /      &DLE     &IEN     &NOOEY   &NOSRE
       /      &NOJMPI  &BT16    &BF2U    &BF2L
       /      &JP      RM2
       ;
00B7 REP3:    IMME     H#0007
       /      &DLE     &IEN     &NOOEY   &NOSRE
       /      &NOJMPI  &BT16    &BF2U    &BF2L
       /      &JP      RM3
       ;
00B8 REP4:    IMME     H#000F
       /      &DLE     &IEN     &NOOEY   &NOSRE
       /      &NOJMPI  &BT16    &BF2U    &BF2L
       /      &JP      RM4
       ;
00B9 REP5:    IMME     H#001F
       /      &DLE     &IEN     &NOOEY   &NOSRE
       /      &NOJMPI  &BT16    &BF2U    &BF2L
       /      &JP      RM5
       ;
00BA REP6:    IMME     H#003F
       /      &DLE     &IEN     &NOOEY   &NOSRE
       /      &NOJMPI  &BT16    &BF2U    &BF2L
       /      &JP      RM6

00BB REP7:    IMME     H#007F
       /      &DLE     &IEN     &NOOEY   &NOSRE
       /      &NOJMPI  &BT16    &BF2U    &BF2L
       /      &JP      RM7
       ;
00BC REP8:    IMME     H#00FF
       /      &DLE     &IEN     &NOOEY   &NOSRE
       /      &NOJMPI  &BT16    &BF2U    &BF2L
       /      &JP      RM8
       ;
00BD REP9:    IMME     H#01FF
       /      &DLE     &IEN     &NOOEY   &NOSRE
       /      &NOJMPI  &BT16    &BF2U    &BF2L
       /      &JP      RM9
       ;
00BE REP10:   IMME     H#03FF
       /      &DLE     &IEN     &NOOEY   &NOSRE
       /      &NOJMPI  &BT16    &BF2U    &BF2L
       /      &JP      RM10
       ;
00BF REP11:   IMME     H#07FF
       /      &DLE     &IEN     &NOOEY   &NOSRE
       /      &NOJMPI  &BT16    &BF2U    &BF2L
       /      &JP      RM11
       ;
00C0 REP12:   IMME     H#0FFF
       /      &DLE     &IEN     &NOOEY   &NOSRE
       /      &NOJMPI  &BT16    &BF2U    &BF2L
       /      &JP      XOR
       ;
00C1 REP13:   IMME     H#1FFE
       /      &DLE     &IEN     &NOOEY   &NOSRE
       /      &NOJMPI  &BT16    &BF2U    &BF2L
       /      &JP      XOR
       ;
00C2 REP14:   IMME     H#3FFC
       /      &DLE     &IEN     &NOOEY   &NOSRE
       /      &NOJMPI  &BT16    &BF2U    &BF2L
       /      &JP      XOR
       ;
00C3 REP15:   IMME     H#7FF8
       /      &DLE     &IEN     &NOOEY   &NOSRE
       /      &NOJMPI  &BT16    &BF2U    &BF2L
       /      &JP      XOR
```

```
00C4 REP16:   IMME      H#FFF0
     /        &DLE      &IEN      &NOOEY    &NOSRE
     /        &NOJMPI   &BT16     &BF2U     &BF2L
     /        &JP       XOR
     ;
00C5 RM1:     ROTM      W,9,MDRI,R9
     /        &DLE      &IEN      &NOOEY    &NOSRE
     /        &NOJMPI   &BT16     &BF2U     &BF2L
     /        &JP       REP17
     ;
00C6 RM2:     ROTM      W,10,MDRI,R9
     /        &DLE      &IEN      &NOOEY    &NOSRE
     /        &NOJMPI   &BT16     &BF2U     &BF2L
     /        &JP       REP18
     ;
00C7 RM3:     ROTM      W,11,MDRI,R9
     /        &DLE      &IEN      &NOOEY    &NOSRE
     /        &NOJMPI   &BT16     &BF2U     &BF2L
     /        &JP       REP19
     ;
00C8 RM4:     ROTM      W,12,MDRI,R9
     /        &DLE      &IEN      &NOOEY    &NOSRE
     /        &NOJMPI   &BT16     &BF2U     &BF2L
     /        &JP       REP20
     ;
00C9 RM5:     ROTM      W,13,MDRI,R9
     /        &DLE      &IEN      &NOOEY    &NOSRE
     /        &NOJMPI   &BT16     &BF2U     &BF2L
     /        &JP       REP21
     ;
00CA RM6:     ROTM      W,14,MDRI,R9
     /        &DLE      &IEN      &NOOEY    &NOSRE
     /        &NOJMPI   &BT16     &BF2U     &BF2L
     /        &JP       REP22
     ;
00CB RM7:     ROTM      W,15,MDRI,R9
     /        &DLE      &IEN      &NOOEY    &NOSRE
     /        &NOJMPI   &BT16     &BF2U     &BF2L
     /        &JP       REP23
     ;
00CC RM8:     ROTM      W,0,MDRI,R9
     /        &DLE      &IEN      &NOOEY    &NOSRE
     /        &NOJMPI   &BT16     &BF2U     &BF2L
     /        &JP       REP24

00CD RM9:     ROTM      W,1,MDRI,R9
     /        &DLE      &IEN      &NOOEY    &NOSRE
     /        &NOJMPI   &BT16     &BF2U     &BF2L
     /        &JP       REP25
     ;
00CE RM10:    ROTM      W,2,MDRI,R9
     /        &DLE      &IEN      &NOOEY    &NOSRE
     /        &NOJMPI   &BT16     &BF2U     &BF2L
     /        &JP       REP26
     ;
00CF RM11:    ROTM      W,3,MDRI,R9
     /        &DLE      &IEN      &NOOEY    &NOSRE
     /        &NOJMPI   &BT16     &BF2U     &BF2L
     /        &JP       REP27
     ;
00D0 REP17:   IMME      H#FFE0
     /        &DLE      &IEN      &NOOEY    &NOSRE
     /        &NOJMPI   &BT16     &BF2U     &BF2L
     /        &JP       XOR
     ;
00D1 REP18:   IMME      H#FFC0
     /        &DLE      &IEN      &NOOEY    &NOSRE
     /        &NOJMPI   &BT16     &BF2U     &BF2L
     /        &JP       XOR
     ;
00D2 REP19:   IMME      H#FF80
     /        &DLE      &IEN      &NOOEY    &NOSRE
     /        &NOJMPI   &BT16     &BF2U     &BF2L
     /        &JP       XOR
     ;
00D3 REP20:   IMME      H#FF00
     /        &DLE      &IEN      &NOOEY    &NOSRE
     /        &NOJMPI   &BT16     &BF2U     &BF2L
     /        &JP       XOR
     ;
00D4 REP21:   IMME      H#FE00
     /        &DLE      &IEN      &NOOEY    &NOSRE
     /        &NOJMPI   &BT16     &BF2U     &BF2L
     /        &JP       XOR
     ;
00D5 REP22:   IMME      H#FC00
     /        &DLE      &IEN      &NOOEY    &NOSRE
     /        &NOJMPI   &BT16     &BF2U     &BF2L
     /        &JP       XOR
```

```
00D6 REP23:   IMME     H#F800
      /       &DLE     &IEN     &NOOEY   &NOSRE
      /       &NOJMPI  &BT16    &BF2U    &BF2L
      /       &JP      XOR
      ;
00D7 REP24:   IMME     H#F000
      /       &DLE     &IEN     &NOOEY   &NOSRE
      /       &NOJMPI  &BT16    &BF2U    &BF2L
      /       &JP      XOR
      ;
00D8 REP25:   IMME     H#E000
      /       &DLE     &IEN     &NOOEY   &NOSRE
      /       &NOJMPI  &BT16    &BF2U    &BF2L
      /       &JP      XOR
      ;
00D9 REP26:   IMME     H#C000
      /       &DLE     &IEN     &NOOEY   &NOSRE
      /       &NOJMPI  &BT16    &BF2U    &BF2L
      /       &JP      XOR
      ;
00DA REP27:   IMME     H#8000
      /       &DLE     &IEN     &NOOEY   &NOSRE
      /       &NOJMPI  &BT16    &BF2U    &BF2L
      /       &JP      XOR
      ;
      ;       SUBROUTINE MULTIPLY
      ;
      ;       THIS SUBROUTINE MULTIPLIES A DOUBLE PRECISION
      ;       WORD BY A SINGLE PRECISION WORD AND ASSUMES
      ;       A DOUBLE PRECISION ANSWER.  THE MULTIPLIER
      ;       IS IN D, THE MULTIPLICAND IS IN R12,R13, AND
      ;       THE ANSWER APPEARS IN R6,R7(LAC). NOTE, UPON
      ;       RETURNING TO THE MAIN PROGRAM THE SUBROUTINE
      ;       INITIATES AN IMMEDIATE MOVE TO R12.
      ;
00DB MUL:     SHFTR    W,SHDR,SHDNZ,R8
      /       &NODLE   &IEN     &NOOEY   &SRE
      /       &NOJMPI
      /       &CONT
      ;
00DC          SOR      W,MOVE,SORY,R13
      /       &DLE     &NOIEN   &OEY     &NOSRE
      /       &NOJMPI
      /       &CONT
      ;
00DD CYC:     SOR      W,MOVE,SORA,R12
      /       &NODLE   &IEN     &NOOEY   &NOSRE   &CT      L
      /       &NOJMPI
      /       &IFNOT   CT16     &CJP     NOTQ
      ;
00DE          TOR1     W,TORAR,ADD,R6
      /       &NODLE   &IEN     &NOOEY   &SRE
      /       &NOJMPI
      /       &CONT
      ;
00DF          TOR1     W,TODRR,ADDC,R7
      /       &NODLE   &IEN     &NOOEY   &NOSRE
      /       &NOJMPI
      /       &CONT
```

```
00E0 NOTQ:   SHFTR   W,SHRR,SHUPZ,R12
     /       &NODLE  &IEN    &NOOEY  &SRE
     /       &NOJMPI
     /       &CONT
     ;
00E1         SHFTR   W,SHRR,SHUPL,R13
     /       &DLE    &IEN    &OEY    &NOSRE
     /       &NOJMPI
     /       &CONT
     ;
00E2         SHFTR   W,SHRR,SHDNZ,R8
     /       &NODLE  &IEN    &NOOEY  &SRE
     /       &NOJMPI
     /       &RPCT   CYC
     ;
00E3         SOR     W,MOVE,SOI,R12
     /       &NODLE  &IEN    &NOOEY  &NOSRE
     /       &NOJMPI
     /       &RTN

     ;
     ;       SUBROUTINE DIVIDE
     ;       THIS SUBROUTINE DIVIDES A DOUBLE PRECISION
     ;       NUMBER BY A DOUBLE PRECISION NUMBER LEAVING
     ;       ONLY A REMAINDER.  THE DIVISOR IS IN R12,R13,D
     ;       AND THE DIVIDEND/REMAINDER APPEARS IN R6,R7.
     ;       NOTE, UPON RETURN AN IMMEDIATE SUBTRACT IS
     ;       INITIATED.
     ;
00E4 DIV:    SOR     W,MOVE,SORA,R12
     /       &NODLE  &IEN    &NOOEY  &NOSRE
     /       &NOJMPI
     /       &CONT
     ;
00E5 CYC1:   TOR1    W,TORAR,SUBS,R6
     /       &NODLE  &IEN    &NOOEY  &SRE
     /       &NOJMPI
     /       &CONT
     ;
00E6         TOR1    W,TODRR,SUBSC,R7
     /       &NODLE  &IEN    &NOOEY  &SRE
     /       &NOJMPI
     /       &CONT
     ;
             NOOP
00E7 /       &NODLE  &IEN    &NOOEY  &NOSRE  &CT     N
     /       &NOJMPI
     /       &IFNOT  CT16    &CJP    POS
     ;
00E8         TOR1    W,TORAR,ADD,R6
     /       &NODLE  &IEN    &NOOEY  &SRE
     /       &NOJMPI
     /       &CONT
     ;
00E9         TOR1    W,TODRR,ADDC,R7
     /       &NODLE  &IEN    &NOOEY  &NOSRE
     /       &NOJMPI
     /       &CONT
```

```
00EA POS:     SHFTR    W,SHRR,SHDNZ,R13
       /      &DLE     &IEN     &OEY     &SRE
       /      &NOJMPI
       /      &CONT
       ;
00EB          SHFTR    W,SHA,SHDNL,R12
       /      &NODLE   &IEN     &NOOEY   &NOSRE
       /      &NOJMPI
       /      &RPCT    CYC1
       ;
00EC          TOR1     W,TORIR,SUBR,R6
       /      &NODLE   &IEN     &NOOEY   &SRE
       /      &NOJMPI
       /      &RTN

       ;
       ;      WRITE SECTOR
       ;
       ;      (R3) TO R4 AND MAR.
       ;      ENTER OUTPUT MODE. (TURNS ON WRITE CLOCK.)
       ;      INITIALIZE COUNTER FOR OUTPUT DATA PREAMBLE LOOP.
       ;
00ED WRSEC1:  SOR      W,NEG,SOAR,R4
       /      &NODLE   &IEN     &NOOEY   &NOSRE
       /      &MADR    &NOJMPI  &OUPT
       /      &LDCT    5
       ;
       ;
       ;      BEGIN WRITE DATA PREAMBLE LOOP.
       ;      TURN ON WRITE GATE.
       ;      PASS WHEN FIFO ARRAY READY FOR OUTPUT.
       ;
     WRSEC2:  NOOP
00EE   /      &NODLE   &NOIEN   &NOOEY   &NOSRE
       /      &NOJMPI  &OUPT    &WRGA
       /      &IFNOT   RDYO     &CJP     $
       ;
       ;
       ;      OUTPUT H#0000 TO FIFO ARRAY.
       ;
00EF          SONR     W,MOVE,SOZ,NRY
       /      &NODLE   &NOIEN   &OEY     &NOSRE
       /      &NOJMPI  &OUPT    &WRGA
       /      &CONT
       ;
       ;
       ;      END WRITE DATA PREAMBLE LOOP.
       ;
00F0          SONR     W,MOVE,SOZ,NRY
       /      &NODLE   &NOIEN   &OEY     &NOSRE
       /      &BF16    &BT03    &NOJMPI  &OUPT    &PL03    &WRGA
       /      &RPCT    WRSEC2
       ;
       ;
       ;      OUTPUT LAST DATA PREAMBLE BYTE AND THE H#FE DATA SYNC BYTE.
       ;      INITIALIZE COUNTER FOR WRITE DATA SEGMENT LOOP.
       ;      PASS WHEN FIFO ARRAY AGAIN READY FOR OUTPUT.
       ;
00F1 WRSEC3:  SONR     W,MOVE,SOI,NRY
       /      &DLE     &NOIEN   &OEY     &NOSRE
       /      &NOJMPI  &OUPT    &WRGA
       /      &LDCT    131
       ;
00F2          IMME     H#FE00
       /      &DLE     &NOIEN   &OEY     &NOSRE
       /      &NOJMPI  &OUPT    &WRGA
       /      &IFNOT   RDYO     &CJP     WRSEC3
       ;
00F3          SONR     W,MOVE,SOD,NRY
       /      &NODLE   &NOIEN   &OEY     &NOSRE
       /      &BF16    &BT03    &NOJMPI  &OUPT    &PL03    &WRGA
       /      &CONT
```

```
        ;
        ;        BEGIN WRITE DATA SEGMENT LOOP.
        ;        (R4) TO MAR.
        ;
00F4 WRSEC4:  SOR      W,MOVE,SORY,R4
        /        &NODLE   &NOIEN  &OEY     &NOSRE
        /        &MADR    &NOJMPI &OUPT    &WRGA
        /        &CONT
        ;
        ;
        ;        INCREMENT (R4).
        ;
00F5             SOR      W,INC,SORR,R4
        /        &NODLE   &IEN    &NOOEY   &NOSRE
        /        &NOJMPI  &OUPT   &WRGA
        /        &CONT
        ;
        ;
        ;        ((MAR)) TO FIFO ARRAY.
        ;        END WRITE DATA SEGMENT LOOP.
        ;
                 NOOP
00F6    /        &NODLE   &NOIEN  &NOOEY   &NOSRE
        /        &BT03    &MREA   &NOJMPI  &OUPT    &PL03    &WRGA
        /        &RPCT    WRSEC4
        ;
        ;
        ;        CLEAR (R4).
        ;        INITIALIZE COUNTER FOR WRITE DATA POSTAMBLE LOOP.
        ;
00F7             SOR      W,MOVE,SOZR,R4
        /        &NODLE   &IEN    &NOOEY   &NOSRE
        /        &NOJMPI  &OUPT   &WRGA
        /        &LDCT    1
        ;
        ;        BEGIN WRITE DATA POSTAMBLE LOOP.
        ;        OUTPUT H#0000 TO FIFO ARRAY.
        ;
00F8 WRSEC5:  SOR      W,MOVE,SORY,R4
        /        &NODLE   &NOIEN  &OEY     &NOSRE
        /        &NOJMPI  &OUPT   &WRGA
        /        &IFNOT   RDYO    &CJP     $
        ;
00F9             SOR      W,MOVE,SORY,R4
        /        &NODLE   &NOIEN  &OEY     &NOSRE
        /        &BF16    &BT03   &NOJMPI  &OUPT    &PL03    &WRGA
        /        &CONT
        ;
        ;
        ;        (NOOP FOR TIMING PURPOSES)
        ;        END WRITE DATA POSTAMBLE LOOP.
        ;
                 NOOP
00FA    /        &NODLE   &NOIEN  &NOOEY   &NOSRE
        /        &NOJMPI  &OUPT   &WRGA
        /        &RPCT    WRSEC5
        ;
        ;
        ;        TURN OFF WRITE GATE.
        ;
                 NOOP
00FB    /        &NODLE   &NOIEN  &NOOEY   &NOSRE
        /        &NOJMPI  &OUPT
        /        &CONT
        ;
        ;
        ;        THEN LEAVE OUTPUT MODE. (TURNS OFF WRITE CLOCK.)
        ;        LOAD 0 INTO R0.
        ;        RETURN.
        ;
00FC             SOR      W,MOVE,SOZR,R0
        /        &NODLE   &IEN    &NOOEY   &NOSRE
        /        &NOJMPI
        /        &RTN
        ;
                 END
```

```
0000 1100000111100000 0010011000110000 011010010011XXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
0001 1111100011100000 XXXX001011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
0002 0000000000000000 XXXX001011101111 111111XXXXXXXXXX XXXXXXXX0XXXXX10 XXXXXXXXXXXXXXXX
0003 1111100011100000 XXXX001011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
0004 0000000000010000 XXXX001011000001 111111XXXXXXXXXX XXXXXXXX0XXXXX10 XXXXXXXXXXXXXXXX
0005 1101100001000011 XXXX001111101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
0006 1101100011000100 XXXX010011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
0007 1101100001000100 XXXX001011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 0XXXXXXXXXXXXXXX
0008 0111000101000000 XXXX011011101111 111111XXXXXXXXXX XXXXXX0XXXXXXX10 XXXXXXXXXXXXXXXX
0009 0111000101000000 XXXX011011101111 111111XXXXXXXXXX XXXXXXXXXXX0XX10 XXXXXXXXXXXXXXXX
000A 0111000101000000 XXXX011011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
000B 0111000101000000 XXXX011011101111 111111XXXXXXXXXX XXXXXXX0XXXXXX10 XXXXXXXXXXXXXXXX
000C 1101110101100100 XXXX010010010000 000111XXXXXXXXXX XXXXXXXXXXX0XX10 XXXXXXXXXXXXXXXX
000D 1111100011100000 XXXX001011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
000E 0000000000010001 XXXX001011000000 000010XXXXXXXXXX XXXXXXXX0XXXXX10 XXXXXXXXXXXXXXXX
000F 1101100001000100 XXXX001011101111 111111XXXXXXXXXX XXXXXXXXXXX0XX10 0XXXXXXXXXXXXXXX
0010 0111000101000000 XXXX011011101111 111111XXXXXXXXXX XXXXXXXXX0XXXX10 XXXXXXXXXXXXXXXX
0011 0111000101000000 XXXX011011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
0012 0111000101000000 XXXX011011101111 111111XXXXXXXXXX XXXXXXXXXXX0XX10 XXXXXXXXXXXXXXXX
0013 0111000101000000 XXXX011011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
0014 1101110101100100 XXXX010010010000 001111XXXXXXXXXX X00XXXXXXXXXXX10 XX0XXXXXXXXXXXXX
0015 1101100100000101 XXXX010011101111 111111XXXXXXXXXX XXXXXXXXXXX0XX10 XXXXXXXXXXXXXXXX
0016 0111000101000000 XXXX011011101111 111111XXXXXXXXXX XXXXXXXXX0XXXX10 XXXXXXXXXXXXXXXX
0017 0101100011000101 XXXX010111101111 111111XXXXXXXXXX X0XXX0XXXXXXXX10 XXXXXXXXXXXXXXXX
0018 1101100001000100 XXXX001011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 0XXXXXXXXXXXXXXX
0019 1101100001000101 XXXX001011101111 111111XXXXXXXXXX 0XXXXXXXXXXXXX10 XX0XXXXXXXXXXXXX
001A 1111100011100001 XXXX010011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
001B 0000000000110110 XXXX010011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
001C 1000100000000000 XXXX010011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
001D 1111100011100001 XXXX010011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
001E 1100000000000001 XXXX010011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
001F 1101100100000100 XXXX010011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
0020 1101100001000001 XXXX010111000000 010000XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
0021 1101100011000101 XXXX010011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
0022 1100000011000101 XXXX010011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
0023 1100110001100100 XXXX010010010000 100010XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
0024 1101100001000010 XXXX010111000000 010000XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
0025 1101100011000101 XXXX010011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
0026 1100000011000101 XXXX010011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
0027 1100110001100100 XXXX010010010000 100110XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
0028 0111000101000000 XXXX011011000001 000000XXXXXXXXXX XXXXXXXXXXXXX010 XXXXXXXXX0XXXXXX
0029 0111000101000000 XXXX011011101111 111111XXXXXX0XXX XXXXXXXXXXXXX010 XXXXXXXXX00XXXXX
002A 0111000101000000 XXXX011000110000 1010101010010XXX XXXXXXXXXXXXX010 XXXXXXXXX0XXXXXX
002B 0111000101000000 XXXX011000110000 101011011011XXXX XXXXXXXXXXXXX010 XXXXXXXXX0XXXXXX
002C 1101100011000101 XXXX010100110000 101100011011XXX0 XXXXX0XXXXXXX010 XXXXXXXXX0XXXXXX
002D 1001000111100001 0010111000110000 110011010010XXXX XXXXXXXXXXXXX010 XXXXXXXXX0XXXXXX
002E 0111000101000000 XXXX011011101111 111111XXXXXXXXXX XXXXXXXXXXXXX010 XXXXXX0XX0XXXXXX
002F 1101100011000101 XXXX010100110000 101111011011XXX0 XXXXX0XXXXXXX010 XXXXXXXXX0XXXXXX
0030 1001000111100010 0010111000110000 110011010010XXXX XXXXXXXXXXXXX010 XXXXXX0XX0XXXXXX
0031 1101100011000101 XXXX010111101111 111111XXXXXXXXX0 XXXXX0XXXXXXX010 XXXXXXXXX0XXXXXX
0032 1001000111100100 0010111000110000 110101010010XXXX XXXXXXXXXXXXX010 XXXXXXXXX0XXXXXX
0033 1101100001000000 XXXX011010010000 101001XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
0034 1101100000000000 XXXX010010101111 111111011110XXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
0035 1101100001000000 XXXX001011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX01 XXXXXXXXXXXXXXXX
0036 1101111000000011 XXXX010000110000 111000011110XXXX XXXXXXXXXXXXXX10 XXXXXXXXXX0XXXXX
0037 1101111000000011 XXXX010000110011 101101011110XXXX XXXXXXXXXXXXXX10 XXXXXXXXXX0XXXXX
0038 1101111010000100 XXXX000011000001 000001XXXXXXXXXX XXXXXXXXXXXXX010 0XXXXXXXX00XXXXX
0039 1101101010000101 XXXX010000110000 111001011011XXXX XXXXXXXXXXXXX010 XXXXXXXXX0XXXXXX
```

```
003A 1111100011100000 XXXX001011101111 111111XXXXXXXXXX XXXXXXXXXXXXX010 XXXXXXXXX0XXXXXX
003B 0000000000000011 XXXX001011101111 111111XXXXXXXXXX XXXXXXXXXXXXX010 XXXXXXXXX0XXXXXX
003C 1111100011100000 XXXX001011101111 111111XXXXXXXXXX XXXXXXXXXXXXX010 XXXXXXXXX0XXXXXX
003D 0000000000010011 XXXX001011101111 111111XXXXXXXXXX XXXXXXXX0XXXX010 XXXXXXXXX0XXXXXX
003E 0111000101000000 XXXX011011101111 111111XXXXXXXXXX XXXXXXXXXXXXX010 XXXXXX0XX0XXXXXX
003F 0111000101000000 XXXX011000110000 111111011011XXX0 XXXXXXXXXXXXX010 XX0XXXXXX0XXXXXX
0040 1101110101100101 XXXX000011101111 111111XXXXXXXXXX XXXXXXXXXXXXX010 0XXXXXXXX0XXXXXX
0041 1101110101100100 XXXX000011101111 111111XXXXXXXXXX XXXXXX0XXXX0X010 00XXXX0XX0XXXXXX
0042 0111000101000000 XXXX011000110001 000010011011XXX0 XXXXXXXXXXXXX010 XX0XXXXXX0XXXXXX
0043 1101100101100101 XXXX000011101111 111111XXXXXXXXXX XXXXXXXXXXXXX010 0XXXXXXXX0XXXXXX
0044 1101110101100100 XXXX000010010000 111111XXXXXXXXXX XXXXXXX0XXX0X010 00XXXX0XX0XXXXXX
0045 0111000101000000 XXXX011011000001 000001XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
0046 1101110101100101 XXXX000011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 0XXXXXXXXXXXXXXX
0047 0111000101000000 XXXX011011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
0048 0111000101000000 XXXX011011101111 111111XXXXXXXXXX XXXXXX0XXXX0XX10 X0XXXXXXXXXXXXXX
0049 0111000101000000 XXXX011011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
004A 0111000101000000 XXXX011011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
004B 0111000101000000 XXXX011010010001 000110XXXXXXXXXX XXXXXXX0XXX0XX10 X0XXXXXXXXXXXXXX
004C 0111000101000000 XXXX011000110001 001110010110XXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
004D 1101100100000000 XXXX010010101111 111111011110XXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
004E 1111100011100000 XXXX001011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
004F 0000000000011111 XXXX001011101111 111111XXXXXXXXXX 0XXXXXXX0XXXXX10 XXXXXXXXXXXXXXXX
0050 1101100100001000 XXXX010011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
0051 1111100011100001 XXXX010011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
0052 0000000000010110 XXXX010011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
0053 0111000101000000 XXXX011000110001 011001010100XXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
0054 1110000110000101 XXXX110000110001 010110010000XXXX XXXXXXXXXXX0XX10 XXXXXXXXXXXXXXXX
0055 1100011111001000 XXXX010000110001 010111011110XXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
0056 1100000111001000 XXXX010011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
0057 0111000101000000 0111011000110001 010011010011XXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
0058 1100001110000000 XXXX000010101111 111111011110XXXX 0XXXXXXX0XXXXX10 XXXXXXXXXXXXXXXX
0059 1101100011101001 XXXX010011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
005A 0000000000001101 XXXX010011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
005B 1101100011101010 XXXX010011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
005C 0000000001011001 XXXX010011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
005D 1101100011101011 XXXX010011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
005E 0000000000010111 XXXX010011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
005F 1101100011101100 XXXX010011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
0060 0000000010010000 XXXX010011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
0061 1011111100101100 XXXX010111101111 111111XXXXXXXXXX XX0XX0XXXXXXXX10 XXXXX0XXXXXXXXXX
0062 0000000000000111 XXXX000000110001 100010011110XXXX XX0XX0XXXXXXXX01 XXXXX0XXXXXXXXXX
0063 1001110000001001 XXXX010011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
0064 0000000000001101 XXXX010011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
0065 1001110000001010 XXXX010011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
0066 0000000001011001 XXXX010011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
0067 1001110000001011 XXXX010011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
0068 0000000000010111 XXXX010011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
0069 1101100100000111 XXXX010011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
006A 1101100011101100 XXXX010011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
006B 1110011100100011 XXXX010011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
006C 1101100011101101 XXXX010011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
006D 0000000000000110 XXXX010011000000 000100XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
006E 1101100001001000 XXXX001100010011 011011011110XXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
006F 1011111110101000 XXXX010011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
0070 1100011110001101 XXXX010011000000 000011XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
0071 1101100001001001 XXXX001100010011 011011011110XXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
0072 1101010100110000 XXXX010011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
0073 1100001110001101 XXXX010011000000 000110XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
```

```
0074 1101100001001010 XXXX001100010011 011011011110XXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
0075 1010100100101000 XXXX010011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
0076 1100011110001101 XXXX010011000000 000100XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
0077 1101100001001011 XXXX001100010011 011011011110XXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
0078 0111000100000000 XXXX010011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
0079 1101100011101101 XXXX000111000000 000110XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
007A 0000010001110111 XXXX000100010011 100100011110XXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
007B 1110111011100010 XXXX010011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
007C 1001110000100111 XXXX010011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
007D 0000000000001000 XXXX010011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
007E 1101100100001000 XXXX010011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
007F 1101100100001001 XXXX010011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
0080 1101100000001001 XXXX010011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
0081 1011100111000110 XXXX010011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
0082 0000111111111111 XXXX010011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
0083 1001110011000110 XXXX010011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
0084 0000000000001111 XXXX010011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
0085 1001110010000110 XXXX010011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
0086 0000000010100101 XXXX000011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX01 XXXXXXXXXXXXXXXX
0087 1000000001000100 XXXX010011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
0088 1110000110000101 XXXX000011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 0XXXXXXXXXXXXXXX
0089 1001111100001000 XXXX010111101111 111111XXXXXXXXXX XXXXX0XXXXXXXX10 XXXXXXXXXXXXXXXX
008A 1101100001001000 XXXX001011101111 111111XXXXXXXXXX 0XXXXXXXXXXXXX10 XX0XXXXXXXXXXXXX
008B 1111110010000000 XXXX001011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 0XXXXXXXXXXXXXXX
008C 1001111100001001 XXXX010111101111 111111XXXXXXXXXX XXXXX0XXXXXXXX10 XXXXXXXXXXXXXXXX
008D 1101100001001001 XXXX001011101111 111111XXXXXXXXXX 0XXXXXXXXXXXXX10 XX0XXXXXXXXXXXXX
008E 1101100100000000 XXXX000010101111 111111011110XXXX 0XXXXXXX0XXXXX10 XXXXXXXXXXXXXXXX
0090 1100000111101001 XXXX110000110010 011101011110XXXX XXXXXXXXXXX0XX10 XXXXX0XXXXXXXXXX
0091 1100000111101011 XXXX110000110010 011000011110XXXX XXXXXXXXXXX0XX10 XXXXX0XXXXXXXXXX
0092 1100000111101001 XXXX110000110010 011001011110XXXX XXXXXXXXXXX0XX10 XXXXX0XXXXXXXXXX
0093 1100000111101011 XXXX110000110010 011011011110XXXX XXXXXXXXXXX0XX10 XXXXX0XXXXXXXXXX
0094 1100000111101010 XXXX110000110010 011010011110XXXX XXXXXXXXXXX0XX10 XXXXX0XXXXXXXXXX
0095 1100000111101010 XXXX110000110010 011011011110XXXX XXXXXXXXXXX0XX10 XXXXX0XXXXXXXXXX
0096 1100000111101001 XXXX110000110010 011011011110XXXX XXXXXXXXXXX0XX10 XXXXX0XXXXXXXXXX
0097 1101100100000110 XXXX110000110001 100011011110XXXX XXXXXXXXXXXXXX10 XXXXX0XXXXXXXXXX
0098 0111000101000000 XXXX011000110010 011110011110XXXX XXXXXXXXXXX0XX10 XXXXX0XXXXXXXXXX
0099 0111000101000000 XXXX011000110010 100001011110XXXX XXXXXXXXXXX0XX10 XXXXX0XXXXXXXXXX
009A 0111000101000000 XXXX011000110010 100011011110XXXX XXXXXXXXXXX0XX10 XXXXX0XXXXXXXXXX
009B 0111000101000000 XXXX011011101111 111111XXXXXXXXXX XXXXXXXXXXX0XX10 XXXXX0XXXXXXXXXX
009C 0111000101000000 XXXX011000110010 011111011110XXXX XXXXXXXXXXX0XX10 XXXXX0XXXXXXXXXX
009D 1100000111101011 0111110000110001 011000010010XXXX XXXXXXXXXXX0XX10 XXXXX0XXXXXXXXXX
009E 1100000111101010 0111110000110001 011000010010XXXX XXXXXXXXXXX0XX10 XXXXX0XXXXXXXXXX
009F 1011111100101100 0111010100110001 010111010010XXXX XX0XX0XXXXXXXX10 XXXXX0XXXXXXXXXX
00A0 0000000000000111 XXXX000000110010 100000011110XXXX XX0XX0XXXXXXXX01 XXXXX0XXXXXXXXXX
00A1 1100000111101011 0111110000110010 011111010011XXXX XXXXXXXXXXX0XX10 XXXXX0XXXXXXXXXX
00A2 0111000101000000 XXXX011000110001 011000011110XXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
00A3 1100000111101001 0111110000110010 011111010011XXXX XXXXXXXXXXX0XX10 XXXXX0XXXXXXXXXX
00A4 0111000101000000 XXXX011000110001 011000011110XXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
00A5 1011001100101000 XXXX010100110010 110101011110XXXX X00XX0XXXXXXXX10 XXXXXXXXXXXXXXXX
00A6 1011010100101000 XXXX010100110010 110110011110XXXX X00XX0XXXXXXXX10 XXXXXXXXXXXXXXXX
00A7 1011011100101000 XXXX010100110010 110111011110XXXX X00XX0XXXXXXXX10 XXXXXXXXXXXXXXXX
00A8 1011100100101000 XXXX010100110010 111000011110XXXX X00XX0XXXXXXXX10 XXXXXXXXXXXXXXXX
00A9 1011101100101000 XXXX010100110010 111001011110XXXX X00XX0XXXXXXXX10 XXXXXXXXXXXXXXXX
00AA 1011110100101000 XXXX010100110010 111010011110XXXX X00XX0XXXXXXXX10 XXXXXXXXXXXXXXXX
00AB 1011111100101000 XXXX010100110010 111011011110XXXX X00XX0XXXXXXXX10 XXXXXXXXXXXXXXXX
00AC 1010000100101000 XXXX010100110010 111100011110XXXX X00XX0XXXXXXXX10 XXXXXXXXXXXXXXXX
00AD 1010001100101000 XXXX010100110010 111101011110XXXX X00XX0XXXXXXXX10 XXXXXXXXXXXXXXXX
00AE 1010010100101000 XXXX010100110010 111110011110XXXX X00XX0XXXXXXXX10 XXXXXXXXXXXXXXXX
```

```
00AF 1010011100101000 XXXX010100110010 111111011110XXXX X00XX0XXXXXXXX10 XXXXXXXXXXXXXXXX
00B0 1010100100101000 XXXX010100110011 000000011110XXXX X00XX0XXXXXXXX10 XXXXXXXXXXXXXXXX
00B1 1010101100101000 XXXX010100110011 000001011110XXXX X00XX0XXXXXXXX10 XXXXXXXXXXXXXXXX
00B2 1010110100101000 XXXX010100110011 000010011110XXXX X00XX0XXXXXXXX10 XXXXXXXXXXXXXXXX
00B3 1010111100101000 XXXX010100110011 000011011110XXXX X00XX0XXXXXXXX10 XXXXXXXXXXXXXXXX
00B4 1011000100101000 XXXX010100110011 000100011110XXXX X00XX0XXXXXXXX10 XXXXXXXXXXXXXXXX
00B5 0000000000000001 XXXX010100110011 000101011110XXXX X00XX0XXXXXXXX10 XXXXXXXXXXXXXXXX
00B6 0000000000000011 XXXX010100110011 000110011110XXXX X00XX0XXXXXXXX10 XXXXXXXXXXXXXXXX
00B7 0000000000000111 XXXX010100110011 000111011110XXXX X00XX0XXXXXXXX10 XXXXXXXXXXXXXXXX
00B8 0000000000001111 XXXX010100110011 001000011110XXXX X00XX0XXXXXXXX10 XXXXXXXXXXXXXXXX
00B9 0000000000011111 XXXX010100110011 001001011110XXXX X00XX0XXXXXXXX10 XXXXXXXXXXXXXXXX
00BA 0000000000111111 XXXX010100110011 001010011110XXXX X00XX0XXXXXXXX10 XXXXXXXXXXXXXXXX
00BB 0000000001111111 XXXX010100110011 001011011110XXXX X00XX0XXXXXXXX10 XXXXXXXXXXXXXXXX
00BC 0000000011111111 XXXX010100110011 001100011110XXXX X00XX0XXXXXXXX10 XXXXXXXXXXXXXXXX
00BD 0000000111111111 XXXX010100110011 001101011110XXXX X00XX0XXXXXXXX10 XXXXXXXXXXXXXXXX
00BE 0000001111111111 XXXX010100110011 001110011110XXXX X00XX0XXXXXXXX10 XXXXXXXXXXXXXXXX
00BF 0000011111111111 XXXX010100110011 001111011110XXXX X00XX0XXXXXXXX10 XXXXXXXXXXXXXXXX
00C0 0000111111111111 XXXX010100110010 000111011110XXXX X00XX0XXXXXXXX10 XXXXXXXXXXXXXXXX
00C1 0001111111111110 XXXX010100110010 000111011110XXXX X00XX0XXXXXXXX10 XXXXXXXXXXXXXXXX
00C2 0011111111111100 XXXX010100110010 000111011110XXXX X00XX0XXXXXXXX10 XXXXXXXXXXXXXXXX
00C3 0111111111111000 XXXX010100110010 000111011110XXXX X00XX0XXXXXXXX10 XXXXXXXXXXXXXXXX
00C4 1111111111110000 XXXX010100110010 000111011110XXXX X00XX0XXXXXXXX10 XXXXXXXXXXXXXXXX
00C5 1011001100101001 XXXX010100110011 010000011110XXXX X00XX0XXXXXXXX10 XXXXXXXXXXXXXXXX
00C6 1011010100101001 XXXX010100110011 010001011110XXXX X00XX0XXXXXXXX10 XXXXXXXXXXXXXXXX
00C7 1011011100101001 XXXX010100110011 010010011110XXXX X00XX0XXXXXXXX10 XXXXXXXXXXXXXXXX
00C8 1011100100101001 XXXX010100110011 010011011110XXXX X00XX0XXXXXXXX10 XXXXXXXXXXXXXXXX
00C9 1011101100101001 XXXX010100110011 010100011110XXXX X00XX0XXXXXXXX10 XXXXXXXXXXXXXXXX
00CA 1011110100101001 XXXX010100110011 010101011110XXXX X00XX0XXXXXXXX10 XXXXXXXXXXXXXXXX
00CB 1011111100101001 XXXX010100110011 010110011110XXXX X00XX0XXXXXXXX10 XXXXXXXXXXXXXXXX
00CC 1010000100101001 XXXX010100110011 010111011110XXXX X00XX0XXXXXXXX10 XXXXXXXXXXXXXXXX
00CD 1010001100101001 XXXX010100110011 011000011110XXXX X00XX0XXXXXXXX10 XXXXXXXXXXXXXXXX
00CE 1010010100101001 XXXX010100110011 011001011110XXXX X00XX0XXXXXXXX10 XXXXXXXXXXXXXXXX
00CF 1010011100101001 XXXX010100110011 011010011110XXXX X00XX0XXXXXXXX10 XXXXXXXXXXXXXXXX
00D0 1111111111100000 XXXX010100110010 000111011110XXXX X00XX0XXXXXXXX10 XXXXXXXXXXXXXXXX
00D1 1111111111000000 XXXX010100110010 000111011110XXXX X00XX0XXXXXXXX10 XXXXXXXXXXXXXXXX
00D2 1111111110000000 XXXX010100110010 000111011110XXXX X00XX0XXXXXXXX10 XXXXXXXXXXXXXXXX
00D3 1111111100000000 XXXX010100110010 000111011110XXXX X00XX0XXXXXXXX10 XXXXXXXXXXXXXXXX
00D4 1111111000000000 XXXX010100110010 000111011110XXXX X00XX0XXXXXXXX10 XXXXXXXXXXXXXXXX
00D5 1111110000000000 XXXX010100110010 000111011110XXXX X00XX0XXXXXXXX10 XXXXXXXXXXXXXXXX
00D6 1111100000000000 XXXX010100110010 000111011110XXXX X00XX0XXXXXXXX10 XXXXXXXXXXXXXXXX
00D7 1111000000000000 XXXX010100110010 000111011110XXXX X00XX0XXXXXXXX10 XXXXXXXXXXXXXXXX
00D8 1110000000000000 XXXX010100110010 000111011110XXXX X00XX0XXXXXXXX10 XXXXXXXXXXXXXXXX
00D9 1100000000000000 XXXX010100110010 000111011110XXXX X00XX0XXXXXXXX10 XXXXXXXXXXXXXXXX
00DA 1000000000000000 XXXX010100110010 000111011110XXXX X00XX0XXXXXXXX10 XXXXXXXXXXXXXXXX
00DB 1100111010001000 XXXX110011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
00DC 1101100001001101 XXXX001111101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
00DD 1101100000001100 1000010000110011 1000000100 11XXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
00DE 1001100010000110 XXXX110011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
00DF 1001111010100111 XXXX010011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
00E0 1100110000001100 XXXX110011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
00E1 1100110001001101 XXXX000111101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
00E2 1100110010001000 XXXX110010010011 011101XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
00E3 1101100011101100 XXXX010010101111 111111011110XXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
00E4 1101100000001100 XXXX010011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
00E5 1001100001000110 XXXX110011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
00E6 1001111001100111 XXXX110011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
00E7 0111000101000000 0111010000110011 101010010011XXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
00E8 1001100010000110 XXXX110011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
```

```
00E9 1001111010100111 XXXX010011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
00EA 1100110010001101 XXXX100111101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
00EB 1100110011001100 XXXX010010010011 100101XXXXXXXXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
00EC 1001110000000110 XXXX110010101111 111111011110XXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
00ED 1101111010000100 XXXX010011000000 000101XXXXXXXXXX XXXXXXXXXXXXXX10 0XX0XXXXXXXXXXXX
00EE 0111000101000000 XXXX011000110011 101110011101XXXX XXXXXXXXXXXXXX10 XXX0XXXXXXXX0XXX
00EF 1111100100000000 XXXX001011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXX0XXXXXXXX0XXX
00F0 1111100100000000 XXXX001010010011 101110XXXXXXXXXX 0XXX0XXXXXXXXX10 XXX0XXX0XXXX0XXX
00F1 1111100011100000 XXXX001111000010 000011XXXXXXXXXX XXXXXXXXXXXXXX10 XXX0XXXXXXXX0XXX
00F2 1111111000000000 XXXX001100110011 110001011101XXXX XXXXXXXXXXXXXX10 XXX0XXXXXXXX0XXX
00F3 1111100011000000 XXXX001011101111 111111XXXXXXXXXX 0XXX0XXXXXXXXX10 XXX0XXX0XXXX0XXX
00F4 1101100001000100 XXXX001011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 0XX0XXXXXXXX0XXX
00F5 1101110101100100 XXXX010011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXX0XXXXXXXX0XXX
00F6 0111000101000000 XXXX011010010011 110100XXXXXXXXXX XXXX0XXXXXXXXX10 X0X0XXX0XXXX0XXX
00F7 1101100100000100 XXXX010011000000 000001XXXXXXXXXX XXXXXXXXXXXXXX10 XXX0XXXXXXXX0XXX
00F8 1101100001000100 XXXX001000110011 111000011101XXXX XXXXXXXXXXXXXX10 XXX0XXXXXXXX0XXX
00F9 1101100001000100 XXXX001011101111 111111XXXXXXXXXX 0XXX0XXXXXXXXX10 XXX0XXX0XXXX0XXX
00FA 0111000101000000 XXXX011010010011 111000XXXXXXXXXX XXXXXXXXXXXXXX10 XXX0XXXXXXXX0XXX
00FB 0111000101000000 XXXX011011101111 111111XXXXXXXXXX XXXXXXXXXXXXXX10 XXX0XXXXXXXXXXXX
00FC 1101100100000000 XXXX010010101111 111111011110XXXX XXXXXXXXXXXXXX10 XXXXXXXXXXXXXXXX
```

SYMBOLS

A1LSW	E723
A1MSW	0006
A2LSW	BFA8
A2NA	0004
A2NDY	0014
A2NR	000E
A3LSW	D530
A4LSW	A928
ADD	0004
ADDC	0005
AE	0056
AE20	0008
AND	0006
ASCEBC	0000
ATTN	0010
B	0000
BACK	0011
BCDEBC	0001
BRTABL	0036
BUSY	0012
C	0005
CDAI	0002
CDRA	0004
CDRI	0003
CFCODE	001A
COMP	000D
CRAI	0005
CRCFL1	0022
CRCFL2	0026
CRCMSK	C001
CRCNIT	0010
CT16	0009
CYC	00DD
CYC1	00E5
DIV	00E4
EBCASC	0002
EBCBCD	0003
EP20	000A
ER20	000B
ERR	0058
EXNOR	000B
EXOR	0008
F1	0009
F2	000A
F3	000B
FAIL	000C
INC	000E
INDX	0013
KL128	7100
KLSW	EEE2
KM128	0477
KMSW	0008
L	0008
LD2NA	0006
LD2NR	000C
LD2NY	0016
LDC2NA	0007
LDC2NR	000D
LDC2NY	0017
LINK	0057
LOW	0004
M1	0053
M1A1	0069
M234I	0059
M2A2	006F
M3A3	0072
M4A4	0075
MARI	000C
MATCH1	0035
MDAI	0007
MDAR	0008
MDRA	000A
MDRI	0009
MFIX	0063
MOVE	000C
MRAI	000E
MUL	00DB
N	0007
N0	0000
N1	0001
N2	0002
N3	0003
N4	0004
N5	0005
N6	0006
N7	0007
N8	0008
N9	0009
NA	000A
NAND	0007
NB	000B
NC	000C
ND	000D
NE	000E
NEG	000F
NF	000F
NO	0001
NOR	0009
NOTQ	00E0
NOZ	0000
NRA	0001
NRAS	0005
NRS	0004
NRY	0000

NSPASS	0040	R9	0009
OR	000A	RDITCT	0041
OVR	0003	RDSEC1	0038
PCPREL	0007	RDSEC2	003F
PF1	0016	RDSEC3	0046
PF2	000D	RDSEC4	004E
PF3	0059	RDYI	000D
PF4	0017	RDYO	000E
PM1	009F	REP1	00B5
PM2	0015	REP10	00BE
PM23	00A3	REP11	00BF
PM234	009D	REP12	00C0
PM24	00A1	REP13	00C1
PM3	0016	REP14	00C2
PM34	009E	REP15	00C3
PM4	0017	REP16	00C4
POS	00EA	REP17	00D0
PR1A	0008	REP18	00D1
PR1D	0009	REP19	00D2
PR1R	000B	REP2	00B6
PR1Y	000A	REP20	00D3
PR2A	0000	REP21	00D4
PR2Y	0002	REP22	00D5
PR3A	0004	REP23	00D6
PR3D	0006	REP24	00D7
PR3R	0003	REP25	00D8
PRA	0008	REP26	00D9
PRI	000B	REP27	00DA
PRT1A	0007	REP3	00B7
PRTA	0004	REP4	00B8
PRTD	0006	REP5	00B9
PRZ	000A	REP6	00BA
R0	0000	REP7	00BB
R1	0001	REP8	00BC
R10	000A	REP9	00BD
R11	000B	RF1	0006
R12	000C	RF2	0009
R13	000D	RF3	000A
R14	000E	RL	0005
R15	000F	RM1	00C5
R16	0010	RM10	00CE
R17	0011	RM11	00CF
R18	0012	RM2	00C6
R19	0013	RM3	00C7
R2	0002	RM4	00C8
R20	0014	RM5	00C9
R21	0015	RM6	00CA
R22	0016	RM7	00CB
R23	0017	RM8	00CC
R24	0018	RM9	00CD
R25	0019	RONCZ	0003
R26	001A	RSTNA	0001
R27	001B	RSTND	0011
R28	001C	RSTNR	000E
R29	001D	RTAA	001D
R3	0003	RTAR	0000
R30	001E	RTAY	001C
R31	001F	RTDA	0019
R4	0004	RTDR	0001
R5	0005	RTDY	0018
R6	0006	RTRA	000C
R7	0007	RTRR	000F
R8	0008	RTRY	000E

```
S2NA      0005
S2NDY     0015
S2NR      000F
SAMD      0014
SCBIBL    000F
SECTIO    0000
SECTL1    0029
SECTL2    0033
SETNA     0002
SETND     0012
SETNR     000D
SF1       0006
SF2       0009
SF3       000A
SHA       0006
SHD       0007
SHDN1     0005
SHDNC     0007
SHDNL     0006
SHDNOV    0008
SHDNZ     0004
SHDR      0007
SHRR      0006
SHUP1     0001
SHUPL     0002
SHUPZ     0000
SL        0005
SOA       0004
SOAR      0004
SOD       0006
SODR      0006
SOI       0007
SOIR      0007
SONZC     0003
SORA      0000
SORR      000B
SORS      0003
SORY      0002
SOSE      000A
SOSER     000A
SOZ       0008
SOZE      0009
SOZER     0009
SOZR      0008
SUBK      007B
SUBR      0000
SUBRC     0001
SUBS      0002
SUBSC     0003
```

```
SUCC      000F
TAB1      0090
TAB2      00A5
TC        000A
TF1       0012
TF2       0014
TF3       0016
TL        0010
TLOW      0008
TM1       009B
TM23      009A
TM24      0099
TM34      0098
TN        000E
TNO       0002
TNOZ      0000
TOAI      0002
TOAIR     0002
TODA      0001
TODAR     0001
TODI      0005
TODIR     0005
TODRA     0003
TODRR     000F
TODRY     000B
TORAA     0000
TORAR     000C
TORAY     0008
TORIA     0002
TORIR     000E
TORIY     000A
TOVR      0006
TSTNA     0000
TSTND     0010
TSTNR     000F
TZ        0004
TZC       000C
W         0001
WRSEC1    00ED
WRSEC2    00EE
WRSEC3    00F1
WRSEC4    00F4
WRSEC5    00F8
XOR       0087
XORMEM    007E
Z         0002
ZC        0006

TOTAL PHASE 2 ERRORS =    0
```

The International Standard of Quality guarantees these electrical AQLs on all parameters over the operating temperature range: 0.1% on MOS RAMs & ROMs; 0.2% on Bipolar Logic & Interface; 0.3% on Linear, LSI Logic & other memories.

INT-STD-123

ADVANCED MICRO DEVICES 901 Thompson Pl., P.O. Box 453, Sunnyvale, CA 94086, USA
TEL: (408) 732-2400 • TWX: 910-339-9280 • TELEX: 34-6306 • TOLL FREE: (800) 538-8450

ADVANCED MICRO DEVICES INTERNATIONAL OFFICES

BELGIUM
Advanced Micro Devices
Overseas Corporation
Avenue de Tervueren, 412, bte 9
B-1150 Bruxelles
Tel: (02) 771 99 93
TELEX: 61028

FRANCE
Advanced Micro Devices, S.A.
Silic 314, Immeuble Helsinki
74, rue d'Arcueil
F-94588 Rungis Cedex
Tél: (01) 687.36.66
TELEX: 202053

GERMANY
Advanced Micro Devices GmbH
Rosenheimer Str. 139
D-8000 Muenchen 80
Tel: (089) 40 19 76
TELEX: 523883

Advanced Micro Devices GmbH
Harthaeuser Hauptstrasse 4
D-7024 Filderstadt 3
Tel: (07158) 30 60
TELEX: 721211

ITALY
Advanced Micro Devices S.R.L.
Centro Direzionale
Palazzo Vasari, 3º Piano
I-20090 MI2 – Segrate (MI)
Tel: (02) 215 4913-4-5
TELEX: 315286

JAPAN
Advanced Micro Devices, K.K.
Dai 3 Hoya Building
8-17, Kamitakaido 1-chome
Suginami-ku, Tokyo 168
Tel: (03) 329-2751
TELEX: 2324064

SWEDEN
Advanced Micro Devices AB
Box 7013
S-172 07 Sundbyberg
Tel: (08) 98 12 35
TELEX: 11602

UNITED KINGDOM
Advanced Micro Devices (U.K.) Ltd.
A.M.D. House,
Goldsworth Road,
Woking,
Surrey GU21 1JT
Tel: Woking (04862) 22121
TELEX: 859103