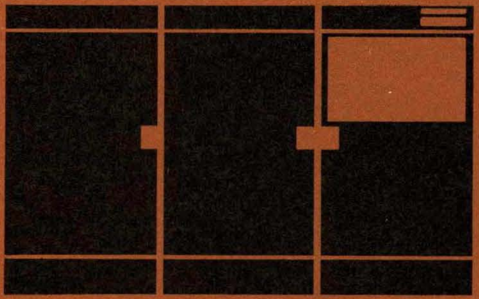
DATANET-30 Programming Reference Manual



GENERAL ELECTRIC

CPB-1019

DATANET-30 PROGRAMMING

REFERENCE MANUAL

JANUARY 1964

GENERAL ELECTRIC
COMPUTER DEPARTMENT

FOREWORD

This manual will cover the aspects of programming the General Electric DATANET-30 Communications Processor. The assumptions are that the individual doing the programming is already familiar with programming techniques, and has a comprehensive understanding of the communications system in which the DATANET-30 is operating.

References to be used in addition to this manual are the DATANET-30 system manual and the glossary of terms of the X3.3.2 committee of the American Standards Association. Familiarity with the document is important before proceeding into the actually programming of the DATANET-30.

CONTENTS

APPENDIX

ILLUSTRATIONS

Figure | Page

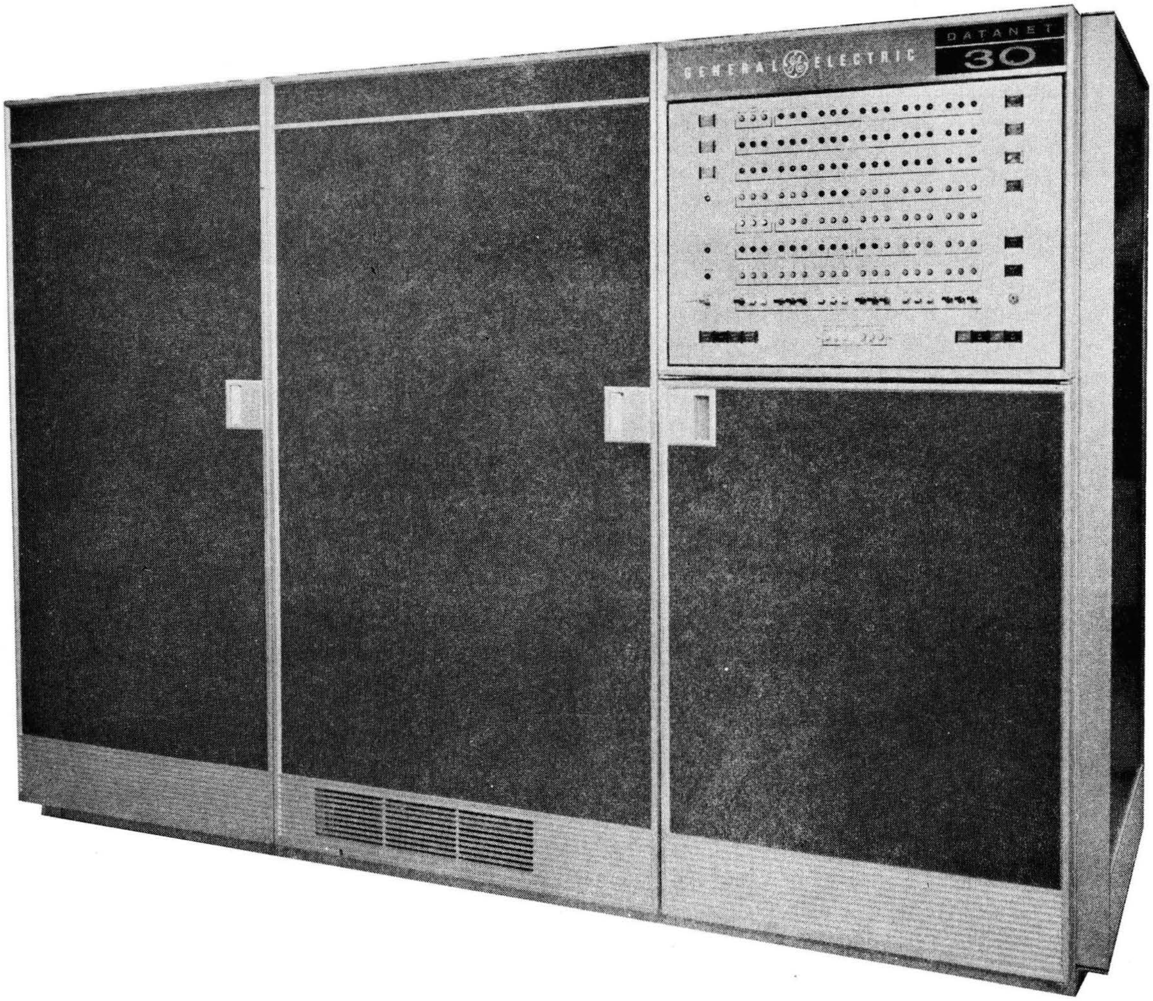


Figure 1. DATANET - 30

I. GENERAL DESCRIPTION

The DATANET-30 is a single address, stored program, special purpose, digital computer which operates primarily in a straight binary mode but processes both alphanumeric and binary information. It performs computation (arithmetic) operations and acts as central control for the DATANET-30 system. Programs to be executed and data to be operated upon are stored in a magnetic core memory where each core represents a binary digit (bit) of an instruction or data word. A word is the basic unit of addressable information in the memory.

The overall function is to simultaneously receive, store, process and transmit data in a communications oriented system.

The system can accommodate any standard transmission speed ranging from 45 to 3,000 bits per second. The basic DATANET-30 controls the transmission of digital data information over normal common carrier facilities to either another DATANET-30, a DATANET-15, a DATANET-600, or any of the standard teletype terminal units in use, such as the Automatic Send Receive (ASR), Keyboard Send Receive (KSR), or Receive Only (RO) units.

The instruction repertoire contains 78 basic instructions and the hardware is capable of executing over 144,000 instructions per second.

Figure 1 shows the major functional sections of the DATANET-30 Communication System, consisting of:

1. The buffer selector and associated buffer units
2. The controller selector and associated high-speed controllers
3. The DATANET-30 Data Communications Processor.

THE MEMORY UNIT

The DATANET-30 uses a magnetic core memory to store program instructions, alphanumeric information, and binary data. Standard memory units are available in 4096, 8192 and 16,384 word sizes. Each word consists of 18 bits. An 18-bit word can contain three 6-bit characters, two 8-bit characters, or one machine instruction.

The memory cycle time is 6.94 microseconds for a read-restore cycle, a clear-write cycle, or a read-compute-write cycle.

During a read-restore cycle, 18 bits of information are read from the memory and transferred to the data communications processor.

During a clear-write cycle, 18 bits of information are transferred from the data communications processor and written into memory.

During a read-compute-write cycle, 18 bits of information are read from memory, changed by the data communications processor, and then the new information is written back into memory.

THE BUFFER SELECTOR

All units connected directly to the buffer selector are referred to as "buffers." Information flows via the buffers and the buffer selector to and from the data communications processor.

The buffer selector contains 128 channels numbered 0 to 127. Each buffer occupies one channel address of the buffer selector, whether the channel is simplex, half-duplex, or full-duplex. The buffer selector channel address for each buffer is established by the wiring of an address plug. The address can be changed or new addresses (buffers) added by changing the existing plug wiring or inserting a new address plug. The channel addresses in any given buffer module need not be sequential. However the addresses for bit buffers must be sequential. Channel 0 is always reserved for the paper tape reader.

THE BIT BUFFER UNIT MODULE

General

The bit buffer units contain a control section and up to ten bit buffer channels.

The bit buffer unit control section contains hardware that is common to all the bit buffer channels in the module. A bit buffer module may terminate from 1 to 10 full-duplex or half-duplex transmission lines which are all operating at the same bit rate.

Bit Buffer Channel

The function of a bit buffer channel is to transmit data to and receive data from a remote terminal on a bit basis.

Each bit buffer channel in a module is assigned a buffer selector address by the address plug for that module. The address applies to both the receive and the transmit section. The addresses for the bit buffers in a module can be whatever is desired for the system and they need not be sequential. Thus, a bit buffer may be added to a module and given an address without disturbing the existing address arrangement. However, the addresses of all bit buffers must be sequential.

The bit buffer provides the interface between the DATANET-30 and one full-duplex, half-duplex, or simplex transmission line on a bit basis. Usually system considerations will limit the bit buffer lines to an operating speed of less than 300 bits per second. Standard teletype rates of 45, 50, 56.26, 75, 110, and 150 bits per second are selected with the timing connector plug. The selected bit rate will apply to all the bit buffer channels physically located in that module. If more than one bit rate is in use in an existing system, the different bit rates must be terminated in separate bit buffer modules. Since the bit buffer channel communicates with the remote terminals on a bit basis, the code level can be different in the separate bit buffers. The code level of individual bit buffers is recognized by the program.

THE CHARACTER/ WORD BUFFER UNIT (CWU)

The character/word buffer unit module can contain either two character buffer channels (CBC), two word buffer channels (WBC), or one of each. Each character/word buffer has a control section.

The Character Buffer Channel (CBC)

The function of a character buffer is to transmit data to and receive data from a remote terminal on a character basis. Transmission to and from a remote terminal is on a bit serial, asynchronous basis.

The character buffer control unit contains hardware to control the bit rate and character length. The character buffers in a module may be operating at different bit rates and different character lengths. The standard bit rates are 300, 600, 1200, 1800, 2000, 2400, or 3000 bits per second. The code level may be any one of 5-, 6-, 7-, or 8-level codes with start-stop bit synchronization. Both the bit rate and code level (character length) may be selected or changed by means of a connector for each buffer. The timing connector plug is available in any one of the standard bit rates. The code level plug is available for 5-, 6-, 7-, or 8-level codes. Thus, by changing plug connectors, both bit rate and code level may be changed to suit changing remote terminal operations.

One character buffer channel provides the interface between the DATANET-30 and a half-duplex transmission line.

Usually, a character buffer channel operates with a character oriented device at speeds higher than 300 bits per second. At this higher rate it is necessary to have some kind of digital subset (DSS) on each end of the transmission line.

The Word Buffer Channel (WBC)

The function of a word buffer channel is to transmit data to and receive data from another DATANET-30 or a DATANET-600.

The word buffer can operate at the same standard bit rates as the character buffer. The bit rate is established by a timing connector plug. The word length is not variable. It is established at 18 bits for a DATANET-30 word, plus one parity bit and one control bit, giving a total of 20 bits per word. This word length is established by a 20-bit code level connector. The DATANET-600 word is similarly established at 14 bits.

THE RECEIVE PARALLEL UNIT BUFFER MODULE

The receive parallel unit buffer module can contain 1 or 2 receive parallel units (RPU). Each unit receives information from one communications channel. Each RPU has a control section. The buffer selector address for each channel (RPU) is specified by the address plug for the module and each RPU is addressed independently of the other. The code level for each RPU may be different.

The Receive Parallel Unit

The receive parallel unit provides buffering, on a receive-only basis, for one character of information in any parallel code up to a 14-channel code level. The RPU buffers the input from a local DATANET-3101 with an 8-level code. A timing connector plug is not used. The upper limit of speed of transmission (receiving information) is determined by the scan rate of the program and should be consistent with the system rate. Operation is asynchronous, timed by the transmitting device.

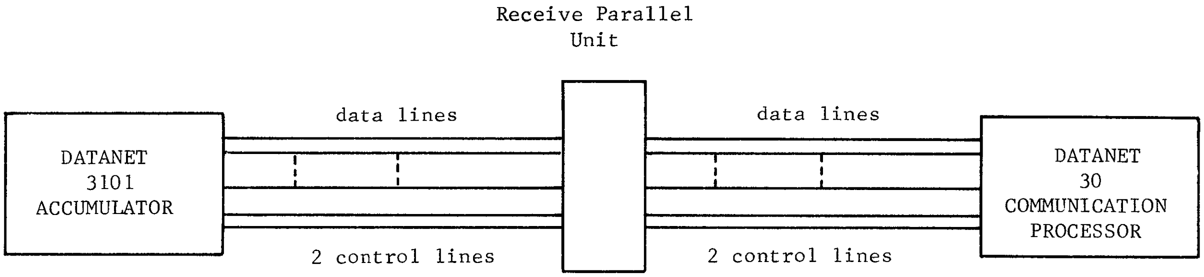


Figure 2. The Receive Parallel Unit with the DATANET-3101

Receive Sequence

The RPU and the DATANET-3101 accumulator are directly connected (on line at all times). When no signals are present on the line, the receive flag is not set and the program ignores the buffer. When a character is received, the receive flag is set and the program must take the character before the next one is transmitted.

The DATANET-3101 system uses answer-back lines to acknowledge the transmission or provide a signal indicating that an error has occurred.

THE CIU-930 COMPUTER INTERFACE UNIT

For those systems requiring a combination data communication-information processing system, a CIU-930 Computer Interface Unit is provided. This unit permits attaching a DATANET-30 data communication processor to a General Electric Compatibles/200 Information Processing System. With this combination, the DATANET-30 is responsible for the communications half of the system, while the Compatibles/200 system is responsible for the data processing.

Twenty-one-bit words are transferred in parallel to and from the information processing system via the Computer Interface Unit. The memory address is also transferred in parallel from the address register in the CIU-930 to the processing system prior to the data transfer.

The CIU allows addressing any location in the central processor memory. The CIU-930 connects into any channel of the DATANET-30 buffer selector in the same manner as any other buffer. The buffer selector address of the CIU-930 is specified by the wiring of the buffer selector address plug for the module. There is no DATANET-30 hardware restriction on the number of CIU's which may be used, other than the physical space occupied. On processing system side, the CIU-930 can connect into any GE-215/225/235 priority control channel.

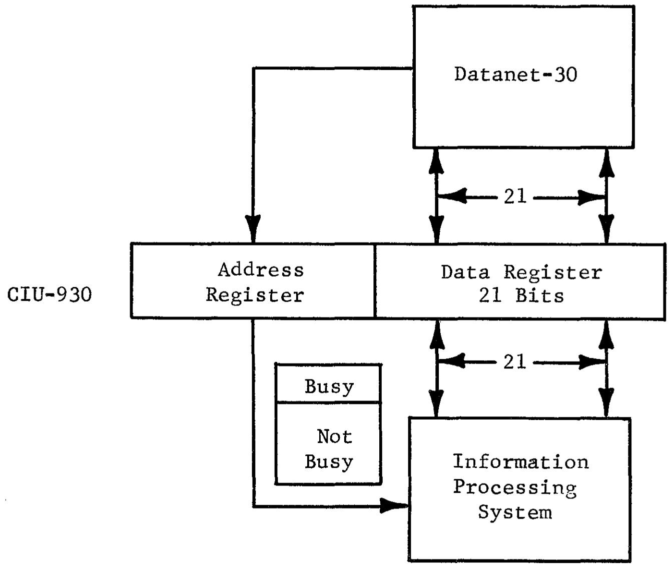


Figure I-3. Computer Interface Block Diagram

The CIU can be tested for a busy/not-busy condition by the DATANET-30. This busy/not-busy test tells the DATANET-30 whether or not it can put data into the data and address registers of the CIU-930, and whether or not it can take data from the data register.

The DATANET-30 communicates with the Compatibles/200 central processor only on a memory interrupt basis. The DATANET-30, under program control, puts data and address information into the CIU to interrupt the central processor. The central processor cannot control the DATANET-30, as is possible with other peripheral equipment. Since both the DATANET-30 and the central processor have stored programs and since the DATANET-30 operates in real time, the DATANET-30 must have control and priority between the two programs.

When the Information Processing system has data for the DATANET-30, it will set a flag in a memory location of the central processor, which is periodically interrogated by the DATANET-30. When the DATANET-30 is ready to accept the traffic, a control instruction is sent to the central processor, the processing system program is interrupted, and the traffic is transmitted to the DATANET-30. The DATANET-30 then processes the traffic and sends it on to the designated remote station. Thus the information processing system and the DATANET-30 exchange control words, instructions, and traffic under control of the DATANET-30.

THE CIU-931 COMPUTER INTERFACE UNIT

The CIU-931 Computer Interface Unit of the DATANET-30 is an 18-bit buffer within the DATANET-30 that provides the connecting link between the DATANET-30 and a General Electric Compatibles/400 system. The CIU connects into the buffer selector of the DATANET-30 and one standard input/output channel of a Compatibles/400 system. The channel may be either a word channel or a character channel for input and output. Direction of data flow is under program control.

The transfer rate is up to 43,200 characters per second or 14,400 DATANET-30 words per second. The actual transfer rate will be determined by the DATANET-30 program.

The CIU permits both the DATANET-30 and the Compatibles/400 system to execute programs concurrently with the transfer of data in either direction. The CIU is able to respond to the processing system without the need of service from the DATANET-30 program. When the CIU responds to the processing system, a signal is generated to indicate to the DATANET-30 program that service is required. The information from the Compatibles/400 command will be stored in the CIU until the DATANET-30 program is able to service the request. Conversely, the CIU will request service from the processing system and store the request until the latter can respond.

All data transferred thru the CIU-931 is parity checked for accuracy. In the event of a parity error, an appropriate signal is generated in the CIU.

THE CONTROLLER SELECTOR

The controller selector permits attaching computer-type peripherals to the DATANET-30.

Eight high-speed channels may be connected to the controller selector enabling the transfer of data to and from the DATANET-30 on a memory interrupt basis. The eight high-speed channels, numbered 0 through 7, operate on a priority basis, with channel 0 having the highest priority and channel 7 the lowest.

The controller selector channel priority assignment is:

Channels 0-5	-	Any combination of:
		Single-access disc storage units Dual-access disc storage unit Magnetic tape controller

Each disc storage unit controller may have 4 disc storage units.

Each magnetic tape controller may have 8 tape handlers.

DATA COMMUNICATIONS PROCESSOR

Data Flow

The DATANET-30 is organized on an 18-bit parallel, bus logic arrangement. Figure 4 is a basic diagram of the principal internal working units of the communications processor. The data is transferred from memory to the arithmetic unit or from a working register through the lower data bus and the Y-register to the arithmetic unit. The Y-register holds the data while it is being processed by the arithmetic unit. After the data has been processed by the arithmetic unit, it is sent to the Z drivers, which are a common distribution center for all data coming from the arithmetic unit and going to a working register, memory, control unit, or an input/output channel. The plus, zero, and even flip-flops also connected to the Z drivers will reflect the branch conditions of any data sent through the Z drivers. For example, if a word coming from memory and going to a working register is plus, non-zero and odd, the branch conditions would be plus, non-zero, and odd. If the data word was all zeros the branch conditions would be plus, zero, and even. From the Z drivers the data flows along the upper data bus to a working register, an input/output channel, or to the memory, according to the instruction currently being executed.

In Figure 5, the buffer selector and controller selector have been added to Figure 4. Data coming from a working register, going to a transmit data line, flows under program control from a specified register to the lower data bus into the Y-register. From the Y-register the

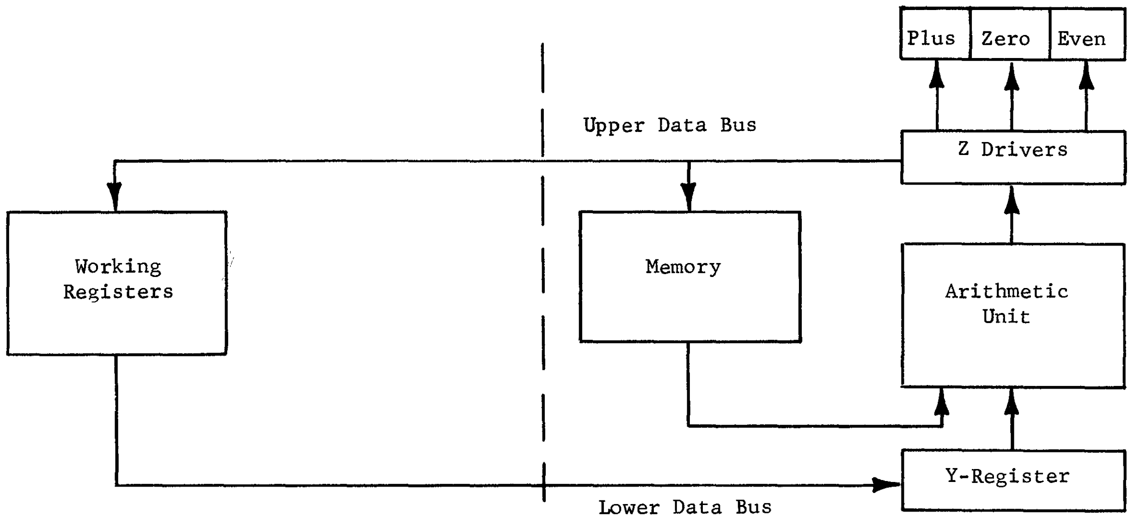


Figure 4. Basic Block Diagram

data flows through the arithmetic unit and the Z drivers onto the upper data bus, where it is then distributed to the buffer selector. The buffer selector then passes the data along to the proper output channel.

Data being received from a specified remote terminal is temporarily stored in a bit buffer, word buffer, or character buffer. The buffer selector then passes the data from the receive buffer channel through the receive data lines to the lower data bus, where it is then sent to the Y-register. From the Y-register the data is sent through the arithmetic unit to the Z drivers, where it is then distributed to the proper working register under program control.

The flow of data to and from the controller selector follows the same paths as for the buffer selector, with the exception that data going to a high-speed peripheral comes from memory and data coming from a high-speed peripheral is put into memory without first going through a working register.

Data flows to and from the controller selector under automatic control of the DATANET-30 circuitry.

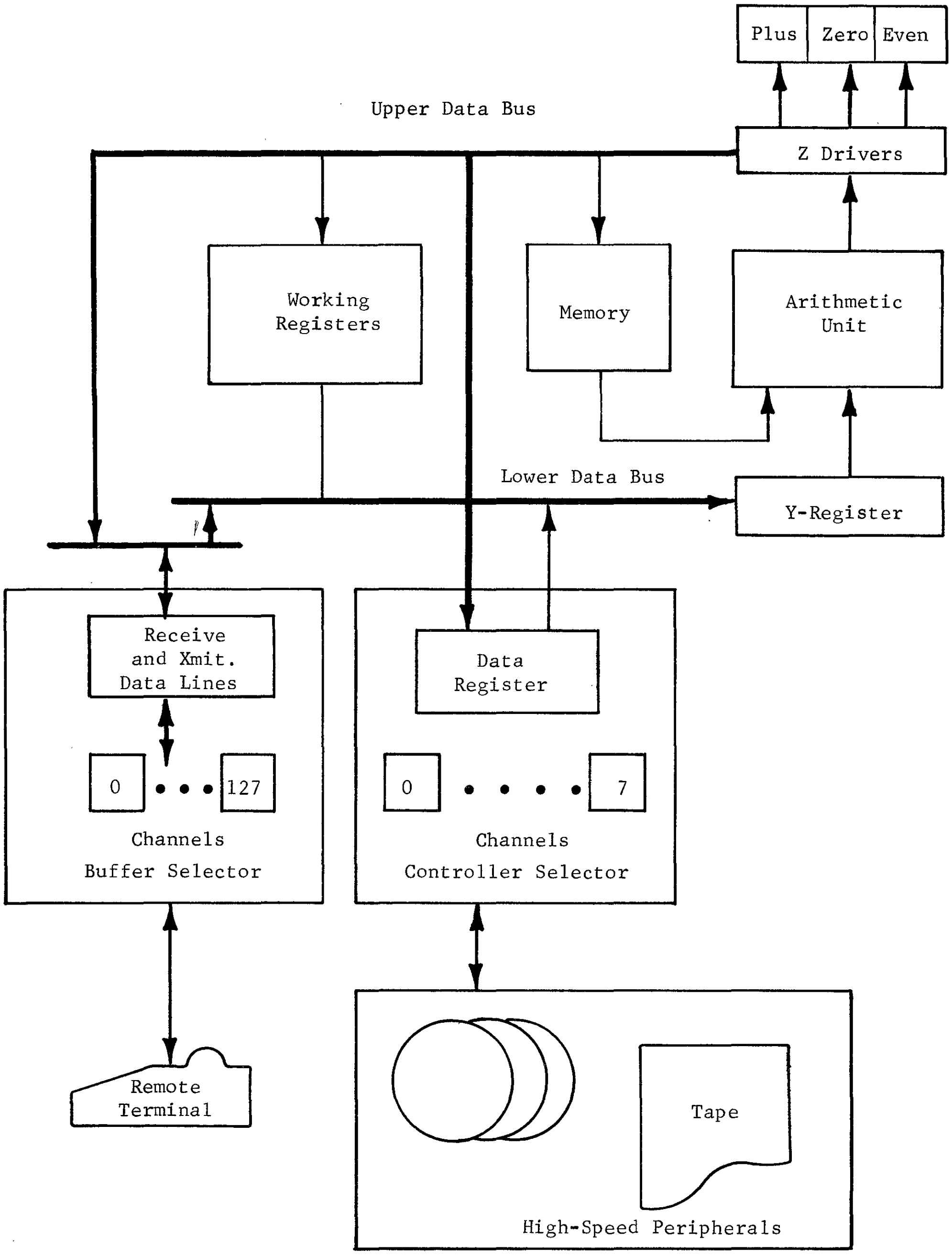


Figure 5. Basic Block Diagram

Detailed Block Diagram

The detailed block diagram (Figure 6) shows many more data paths of the communications processor, including those for the memory unit, the buffer selector, and the controller selector; but the overall pattern of data flow still applies. In general, data flows from one or more registers to the lower data bus, through the Y-register to the arithmetic unit, to the Z drivers, and then to one or more of the registers connected to the upper data bus. Data may also go from the memory to the arithmetic unit at the same time that data is coming from the Y-register.

The register transfer instructions, a major class of instructions, permit any combination of up to six (specific) registers to be combinedin the Y-register, to be manipulated in some selected manner, and then have the result putin any combination of up to four (specific) registers. Further details of the register transfer instructions are given in the discussion of the instruction repertorie.

Description of Registers

This section contains information about each of the blocks on the detailed block diagram. Certain conventions are followed:

First Item: The size of the register.

Second Item: The abbreviation for the name of the register (no abb. means no abbreviation is used).

Third Item: A or N, to indicate that the register is accessible or is not directly accessible to the program.

A-Register (18 bits, A, A)

B-Register (18 bits, B, A)

The A and B registers are the principal working registers of the DATANET-30. They are identical and have identical functions and instructions except for the parity network, which is connected to just the B-register.

C-Register (7 bits, C, A)

The C-register is used to specify a particular input/output channel of the buffer selector. In addition, C can be used as a normal index register when indirect addressing is used.

L-Register (14 bits, L, N)

The L-register contains the address of the next memory location to be accessed. In the step/stop mode, the register will contain the operand address of the instruction last executed.

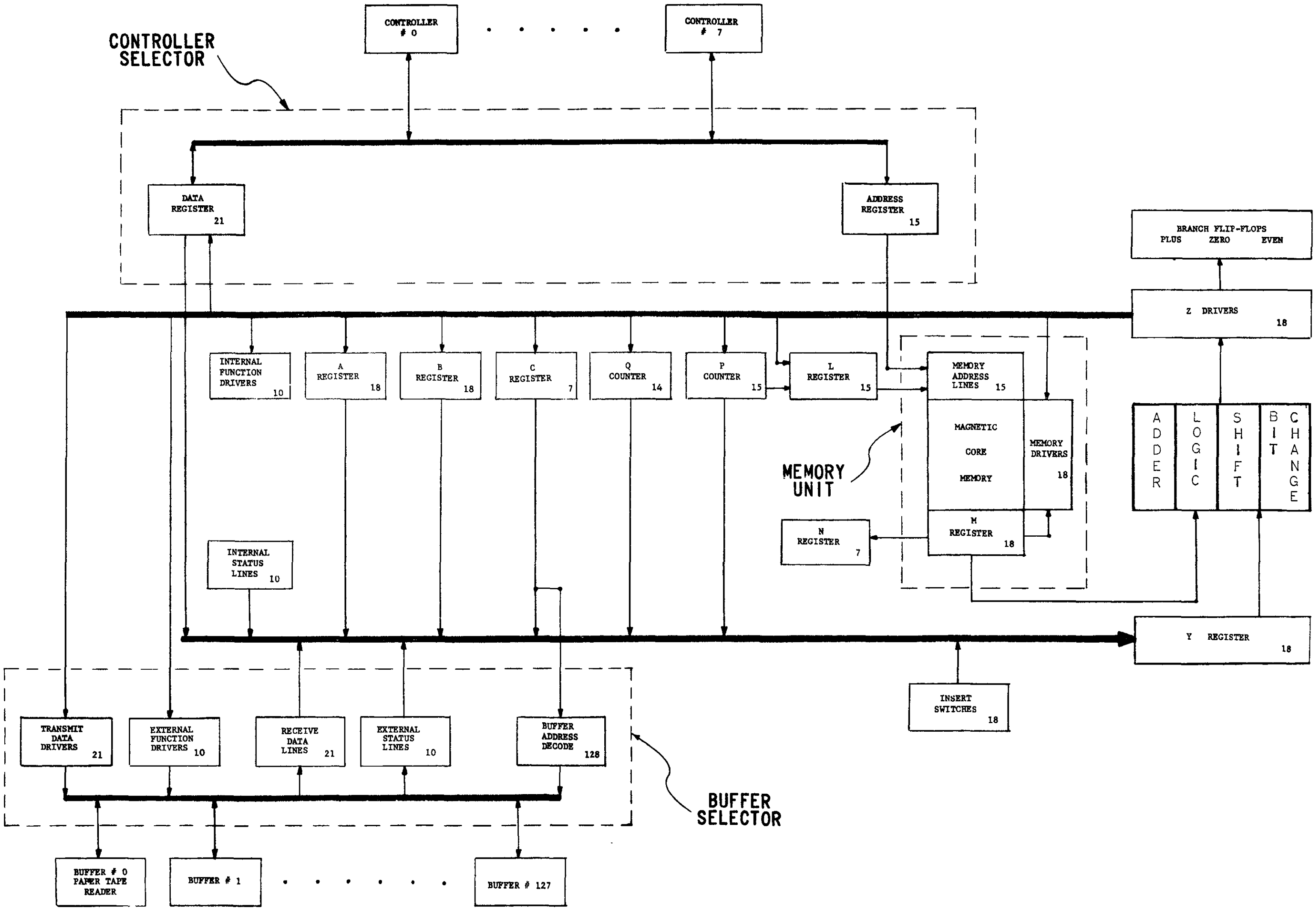


Figure 6. Detailed Block Diagram

N-Register (7 bits, N, N)

The N-register is used to facilitate the instruction decoding process. The register contains the high order 7 bits of the instruction to be executed. In the step/stop mode, the register will contain the operation code of the last instruction executed.

P-Counter (14 bits, P, A)

The P-counter contains the address of the next instruction to be executed. Some bits of the P-counter are used for generating addresses. The P-counter will count up through program banks.

Q-Counter (14 bits, Q, A)

The Q-counter serves as the elapsed time clock.

Y-Register (18 bits, Y, N)

The Y-register is used to form and hold the intermediate operand for an instruction.

Z Drivers (18 bits, Z, N)

The Z drivers are a common data distribution center for all data coming from the arithmetic unit and going to a working register, memory, control unit, or an input/output (I/O) channel. Data passes through the Z drivers without delay enroute to the destination determined by the instruction being executed at the time that the data exists in the drivers.

Arithmetic Unit (18 bits, no abb., N)

The arithmetic unit performs the following functions on the contents of Y and/or M and puts the result into the Z drivers:

1. Binary addition
2. Logical AND
3. Logical OR
4. Logical EXCLUSIVE OR
5. Shift left, right, circulate
6. Bit change
7. Address modification.

Branch Flip-Flops (BFF's, A)

The plus, zero, and even flip-flops are connected to the Z drivers. These three flip-flops are set at the completion of every non-branch instruction and will reflect the branch conditions of any data passing through the Z drivers. The plus FF (PFF) stores the status of the high order bit of the result Z(18). The zero FF (ZFF) stores the status of the entire result Z(1-18). The even FF (EFF) stores the status of the low order bit Z(1) of the result. The results of an operation is available for test on the next instruction. When the branch is based on contents of the C-register, only Z(1-7) are reflected in ZFF and EFF. When the branch is based on the internal status lines, only Z(1-10) are reflected in ZFF and EFF.

Plus Flip-Flop (1 bit, PFF, A)

The PFF records (for testing) the condition of Z(18) at the end of an instruction. If Z(18) was zero, the PFF would be plus; but if Z(18) was one, the PFF would be minus. The notation Z(18) refers to bit position 18 of Z -- that is, the high order position of Z.

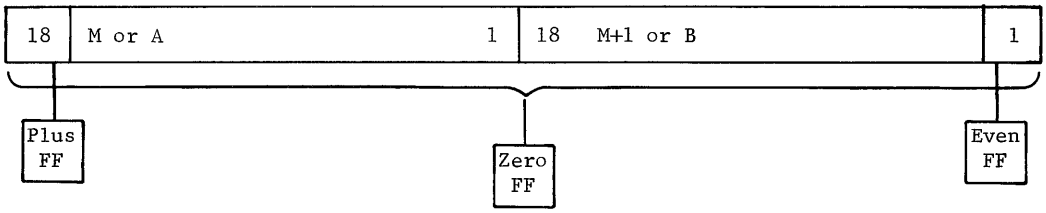
Zero Flip-Flop (1 bit, ZFF, A)

The ZFF records (for testing) the condition of Z at the end of an instruction. If all of the Z drivers were zero, the ZFF would be zero; but if any one of the Z drivers were non-zero, the ZFF would be non-zero.

Even Flip-Flop (1 bit, EFF, A)

The EFF records (for testing) the condition of Z(1) at the end of an instruction. If Z(1) was zero, the EFF would be even; but if Z(1) was one, the EFF would be odd.

On double length instructions (AMD, LDD, STD) the branch flip-flops indicate the following:



Thus the last word through the Z drivers can be tested for being:

1. Plus or minus (sign bit)
2. Odd or even (numerical sense)
3. All zeros or not all zeros.

Insert Switches (18 switches, S, A)

The switches are located on the control console and are described in the discussion of the control console. They can be gated in under program control.

Internal Function Drivers (10 drivers, IFD, A)

These drivers can activate special control functions. These functions are listed under "Special Instructions" as the Drive Internal Function (DIF) instructions.

Internal Status Lines (10 lines, ISL, A)

These lines are used to test the status of various special conditions. These conditions are listed under "Special Instructions" as the AND Internal Status (NIS) instructions.

THE MEMORY UNIT

M-Register (18 bits, no abb., N)

The M-register is the memory output register. References to M in many places in this manual refer to the contents of a memory location, which is actually made available in the M-register. In the step/stop mode, the register will contain the contents of the last memory location accessed as specified by L.

Memory Drivers (18 drivers, no abb., N)

The memory drivers are used to write a new word into the memory and to regenerate a word when it is read out of the memory.

Memory Address Lines (14 lines, no abb., N)

These contain the address of the memory location being accessed.

THE BUFFER SELECTOR

Receive Data Lines (21 lines, R, A)

These lines are used to receive data from all buffer units on the buffer selector.

Transmit Data Drivers (21 drivers, T, A)

These drivers are used to send data to all buffer units on the buffer selector.

External Function Drivers (10 drivers, EFD, A)

These drivers are used to send control signals to a buffer unit. The function of each driver depends on the particular type of buffer unit. The functions are listed under "Buffer Selector Instructions" as the DEF instructions.

External Status Lines (10 lines, ESL, A)

These lines are used to test various conditions in a buffer unit. The condition tested by each line depends on the particular buffer unit. The conditions are listed under "Buffer Selector Instructions" as the NES instructions.

Buffer Address Decode (128, N)

This unit decodes the C-register into a 1 out of 128 signal to select the desired buffer address.

THE CONTROLLER SELECTOR

Data Register (21 bits, no abb., N)

The controller selector data register contains the data being transferred between the controller selector and the DATANET-30.

Address Register (14 bits, no abb., N)

The controller selector address register contains the address of the next memory location to be accessed by the controller selector.

PARITY NETWORKS (21 bits, no abb., A)

Although not shown on the block diagram, the parity networks are attached to the B-register and consist of a word parity network and a character parity network.

There are two outputs from the parity network, one for character parity and one for word parity. Either output may be tested to check incoming data. The appropriate output is automatically sent to a buffer unit when information is transmitted.

The input to the word parity network consists of the 18 bits of the B-register and the control bit 1 and control bit 2 flip-flops. The output of the word parity network is bit 21 and is used with the word buffer channel and CIU. The inputs to the character parity network are bits 1-6 of the B-register and the control bit 1 and 3 flip-flops. The character parity is used almost exclusively for generating correct parity on 8-level teletype characters. Each time a word is brought into the B-register, the word parity network will generate correct parity on it. At the same time, proper character parity will be generated on bits 1-6 of the B-register.

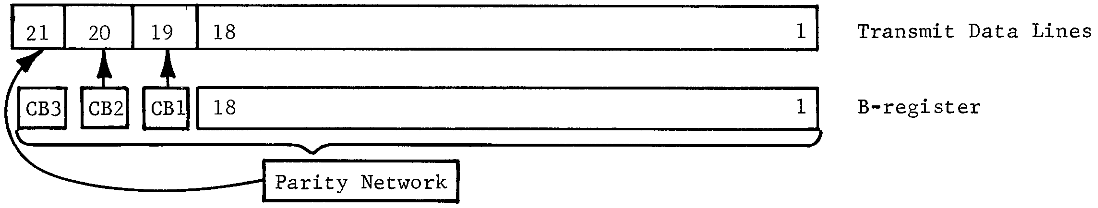
CONTROL BITS 1, 2 and 3

The control bits are special-purpose flip-flops and are used as needed. Since there are 21 receive data lines and the registers are 18-bit registers, the receive data lines 19, 20, and 21 go to control bits 1, 2, and 3, respectively. Control bit 3 is also referred to as the "parity bit." The following chart shows the instructions and conditions affecting the control bits.

	CB1	CB2	CB3 (Parity)
Buffer Selector Receive Data Lines	19	20	21
Instructions BCO	Y09	Resets only	Y06
NIS	NIS 8	NIS 9	NIS 0
DIF	DIF 8	DIF 9	DIF 0
LDF	Z08	Z09	Z10
STF	Z08	Z09	Z10

The paper tape reader also uses the control bits in a special way when reading paper tape under program control.

The transmit data lines use the control bits as follows.



When transferring data to a word buffer or a CIU, where a parity bit is needed, put a word in the B-register, set bits 19 and 20 as required (DIF instructions) and when a Register Transfer instruction is executed, the proper parity will go to line 21.

Instruction Cycles

The following examples illustrate typical situations and the flow of information by large lines with arrowheads indicating the direction of flow. The steps are numbered to tie in with the corresponding explanation. These examples are for one 6.94 microsecond word time each.

The function the instruction cycle (Figure 7) performs is the initial decoding of the instruction and the generation of the desired memory address and its transfer to the L-register. This prepares the DATANET-30 for the execution cycles to follow:

1. At the very start of the instruction cycle (actually slightly before) the address of the next instruction is transferred from P to L. After this takes place, P is incremented by plus 1.
2. The L-register is transferred to the memory address lines.
3. When the instruction is read out, it is transferred from M to N where, in this example, a non general instruction is decoded.
4. After the instruction is decoded the address modification mode is decoded and the correct section of the arithmetic unit enabled (see "Addressing Memory").
5. The desired memory address is transferred from the arithmetic unit to Z.
6. The address is then sent to L to prepare for addressing memory on the next cycle.
7. Simultaneously with steps 3, 4, and 5, the contents of M are being regenerated by the memory drivers.

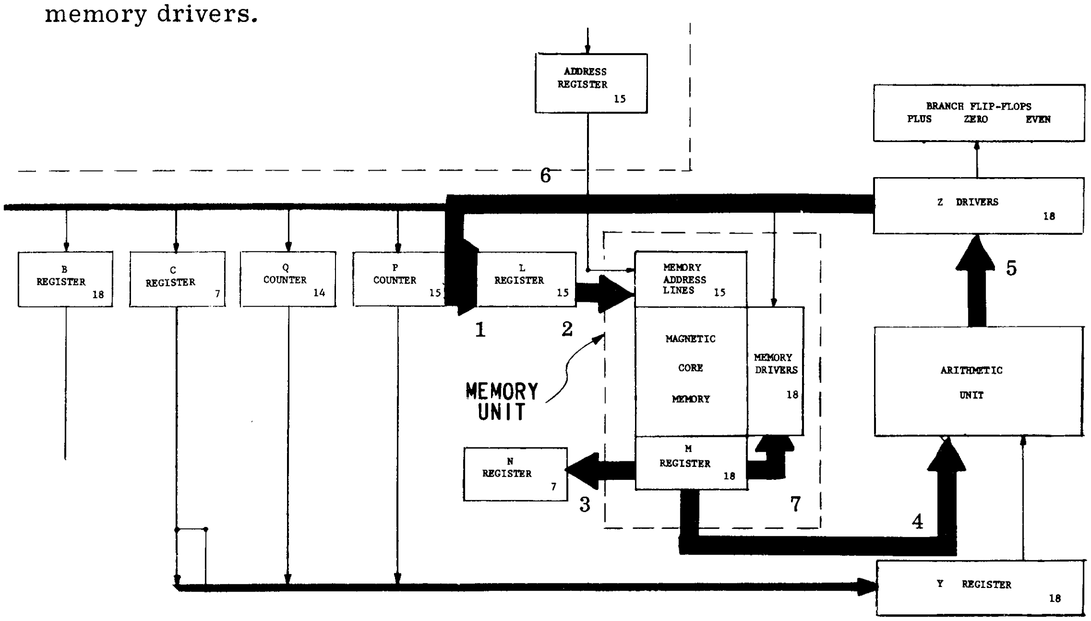


Figure 7. Detailed Block Diagram DATANET-30 Instruction Cycle

LOAD A-REGISTER (LDA) EXECUTION CYCLE. This instruction performs the function of transferring information from M to A (Figure 8):

1. The operand address in L is transferred to the memory address lines for accessing the memory.
2. The contents of M are transferred to the arithmetic unit.
3. The contents of M are transferred through the arithmetic unit to Z.
4. The contents of M are transferred from Z to A, thus loading A with the contents of M.
5. Simultaneously with steps 2, 3, and 4, the contents of M are being regenerated by the memory drivers.
6. The branch flip-flops store the plus, zero, and even conditions of the contents of memory.

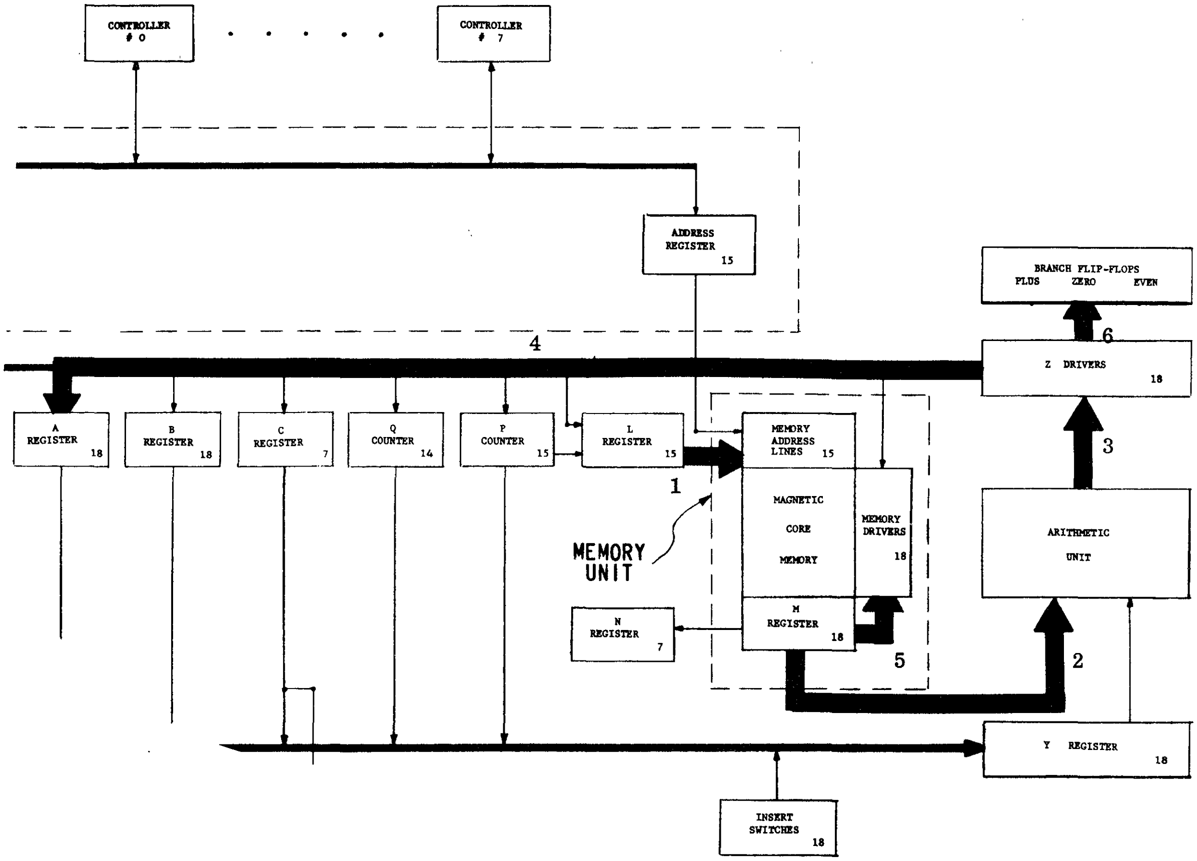


Figure 8. Detailed Block Diagram DATANET-30
Load A (LDA)

STORE B-REGISTER (STB) EXECUTION CYCLE. Information is again transferred from B to the memory (Figure 9):

1. The operand address in L is transferred to the memory address lines for accessing the memory.

2. The contents of B is transferred to Y while the memory is being read out and cleared.

3. B is transferred from Y to the arithmetic unit.

4. B is then transferred to Z.

5. The contents of B is then transferred from Z to the memory drivers for the generation in memory of the new information.

6. The branch flip-flops store the plus, zero, and even conditions of the contents of B.

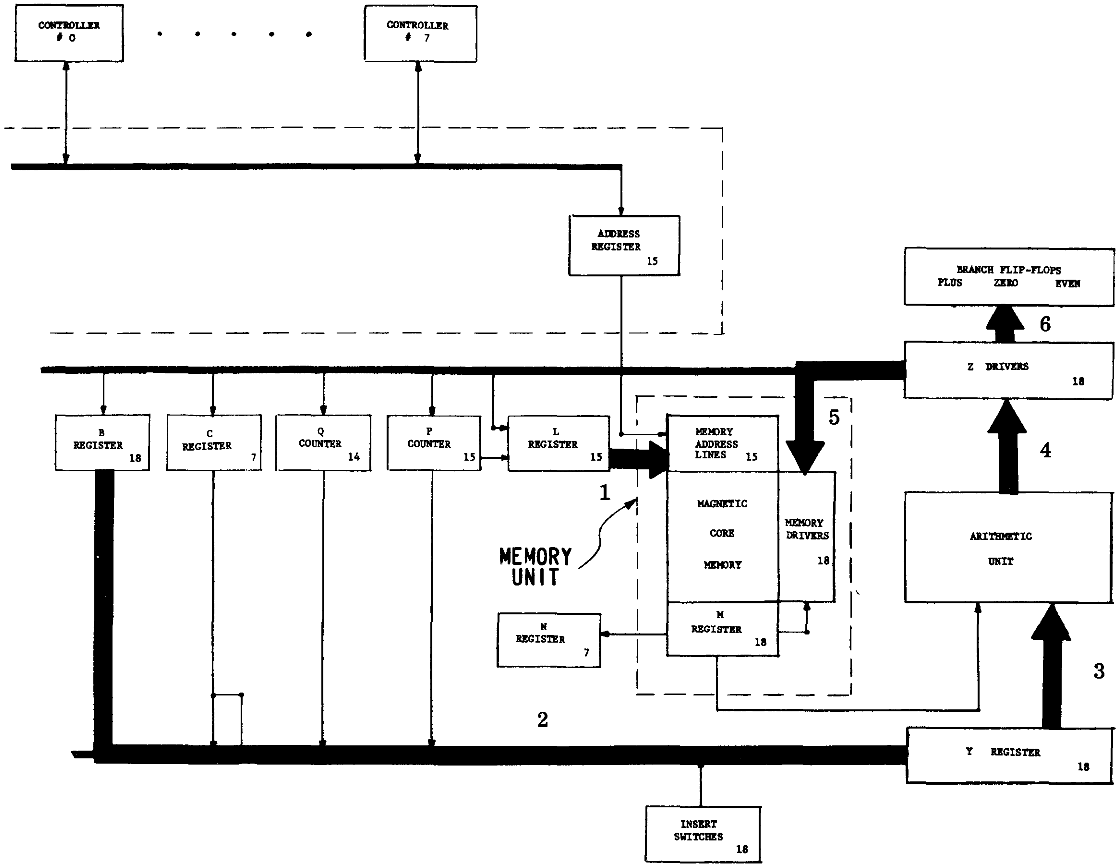


Figure 9. Detailed Block Diagram DATANET-30
Store B (STB)

ADD MEMORY TO A-REGISTER (AMA) EXECUTION CYCLE. This instruction replaces A with the sum of A and M, and regenerates M (Figure 10):

1. The operand address in L is transferred to the memory address lines for accessing memory.

2. The contents of A is transferred to Y while the memory is being read out.

3. The contents of M is read from memory and transferred to the arithmetic unit.

4. The contents of A is transferred through Y to the arithmetic unit.

5. The binary arithmetic sum of M andA is generated by the arithmetic unit and transferred to Z.

6. The sum in Z is transferred to A.

7. Simultaneously with steps 3, 4, 5, and 6, the contents of M are being regenerated by the memory drivers.

8. The branch flip-flops store the plus, zero, and even conditions of the binary arithmetic sum of A and M.

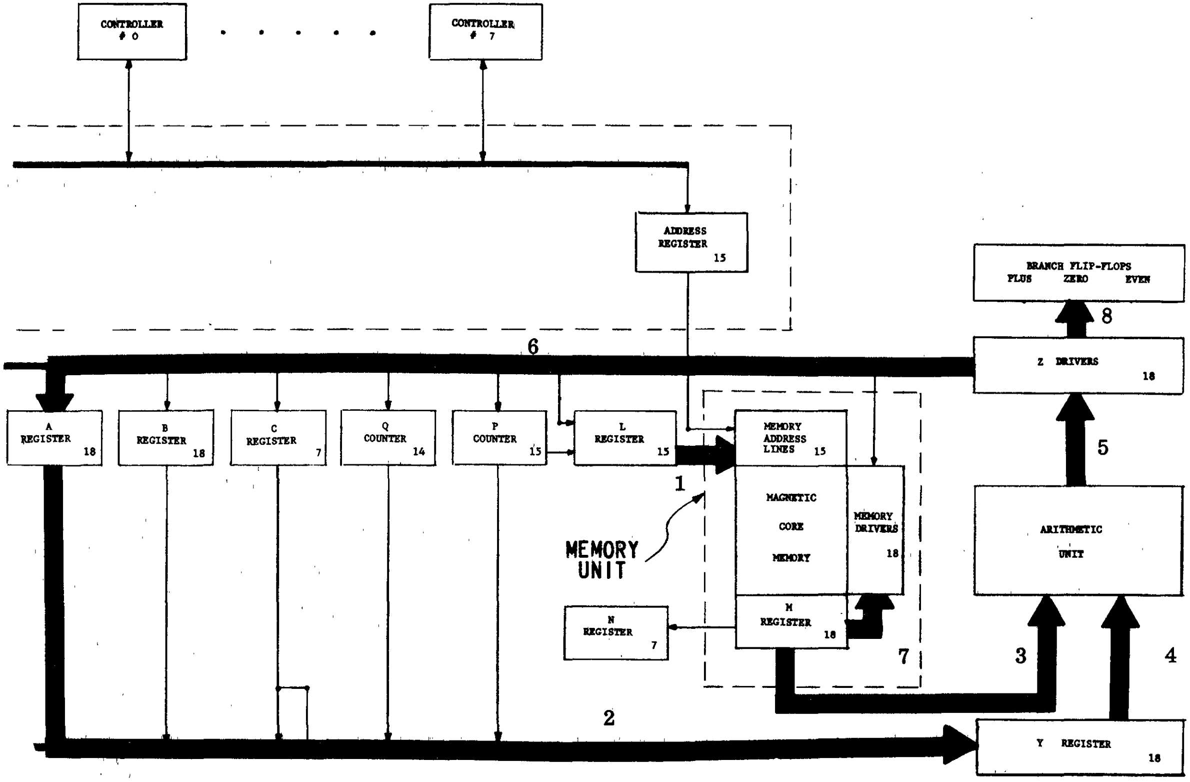


Figure 10. Detailed Block Diagram DATANET-30
Add Memory to A (AMA)

SHIFT RIGHT ONE (SR1) BR,B CYCLE. This instruction performs the Shift Right One (SR1) function in one word time (Figure 11):

1. At the very start of the instruction cycle (actually slightly before) the address of the next instruction is transferred from P to L. After this takes place, P is incremented by plus 1.
2. The L-register is transferred to the memory address lines.
3. When the instruction is read out, it is transferred from M to N where, in this example, a general instruction (SR1 BR,B) is decoded.
4. After the instruction is decoded, the contents of B are transferred to Y.
5. Simultaneously with step 3, the contents of R are transferred to Y.
6. The logical OR of B and R is done in Y and transferred to the arithmetic unit.
7. The arithmetic unit performs a SR1 function on Y and transfers the result to Z.
8. The result in Z is transferred to B.
9. Simultaneously with steps 3, 4, 5, 6, and 7, the contents of M are being regenerated by the memory drivers.
10. The branch flip-flops store the plus, zero, and even conditions of the new contents of B.

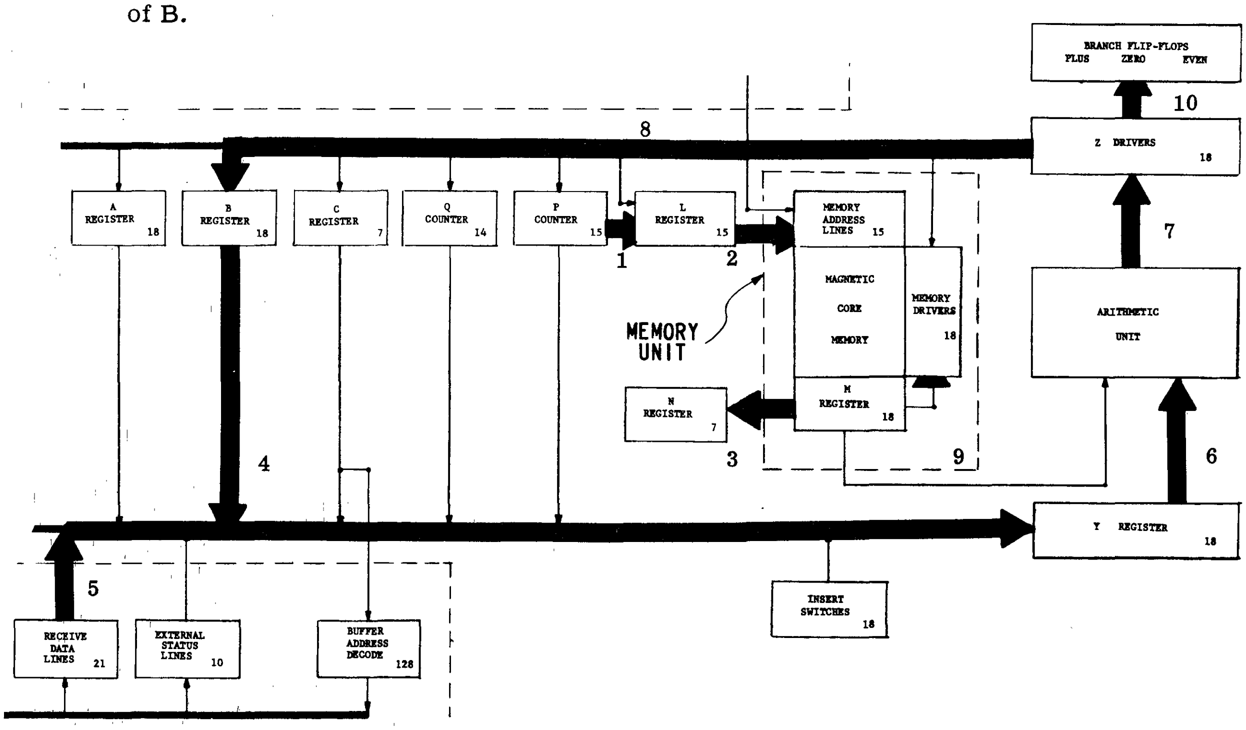


Figure 11. Detailed Block Diagram DATANET-30 Shift Right 1 Receive Lines to B-register (SR1 BR, B)

Paper Tape Reader

The paper tape reader will read 5-, 6-, 7-, or 8-level tape under program control, or 8-level tape under hardware control. When reading is done under hardware control, this is referred to as "hardware load." Normally, 8-level tape is used in both cases.

The reader is permanently tied to buffer selector address 0. It operates like any other remote terminal connected to the buffer selector when under program control, in the sense that it uses the external function drivers for control and the external status lines for testing. As information is read, it is transferred into input buffer 0 and the receive flag is set to indicate that data is present. This flag may be tested by an NES command.

The primary function of the paper tape reader is to contain either a bootstrap program to be used at the start of a day, or a special restart and error recovery program to be used in the event that an error condition develops in the execution of the normal program.

The secondary function of Hardware Load and the paper tape reader is to initially load the programs into memory. Once the programs are loaded,· they may be stored in the disc storage unit or on magnetic tape and recalled as necessary.

The third possible function is to enter data via the paper tape reader under program control. This is not a normal usage, however, and is more of an exception than a rule to the intended use of the reader.

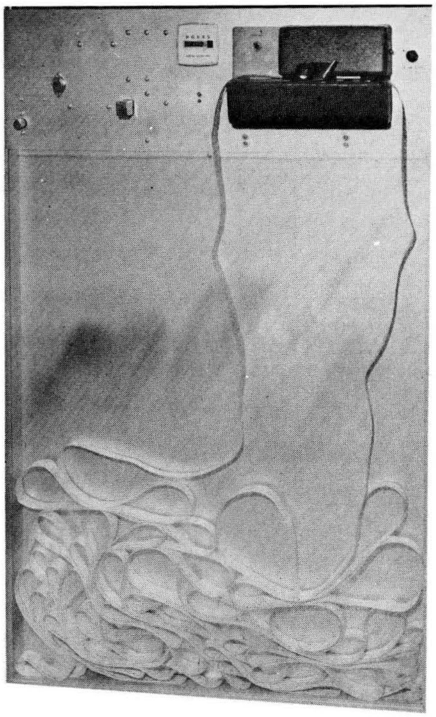


Figure 12. Paper Tape Reader

Hardware Load

Hardware Load is a process whereby data is transferred from the paper tape reader to memory under hardware control. This is used for initial loading of programs, for the loading of maintenance diagnostics when necessary, and for the automatic restart of an operating program upon discovery of a fault condition.

Hardware load may be initiated in five ways.

1. Manually from the control console.
2. By execution of a DIF 4 instruction.
3. When Q counts down to -32.
4. When the second LDQ instruction is executed after a program interrupt occurs while in the operate mode.
5. When in the operate mode and a halt occurs.

Hardware Load has a special format. The generation of paper tape in the hardware load format is described in the section on programming the paper tape reader.

The Elapsed Time Clock (Q-counter)

The DATANET-30 is a real time data communications processor. Real time programs have a periodic nature of operation. The elapsed time clock (the Q-counter) provides an efficient technique for achieving this.

The Q-counter is loaded by the program, and is counted down one each word time. This serves as a word/time counter. Q can be loaded with any number between -32 and +16,351. If loaded with 16,351, this is equal to approximately 112 milliseconds.

When Q counts down to zero, a program interrupt is initiated, thus permitting the periodic execution of programs at any period up to 112 milliseconds. The Q-counter may be used as a relatively accurate real time clock by counting the number of program interrupts when they occur. For example, if a delay of 900 milliseconds is desired and the communication lines are scanned every 12.5 milliseconds, then a count of 72 interrupts equals 900 milliseconds.

The Q-counter and Hardware Load

The Q-counter also serves as a reliability check on the system. When Q counts down to -32, the DATANET-30 assumes a circuit failure and automatically initiates loading a restart program by initiating hardware load. Successful operation of the programs depends on preventing Q from counting to -32 and reading in a restart program. This is achieved in the Program Interrupt Routine by loading the Q-counter before it counts down to -32. Also, in the operate mode, protection against a "dead loop" which includes an instruction to load the Q-counter, has been achieved by counting the number of times the counter has been loaded since the last program interrupt. Hardware load will be initiated upon execution of the second Load Q instruction. This assures that the Program Interrupt Routine is executed periodically. The Program Interrupt Routine may be written to check the program and initiate a hardware load if a fault is found. This hardware-software feature provides a very adequate check on the proper operation of the program. In the event that certain programs do not require a periodic interrupt, this feature may be inhibited by the Q-counter switch on the operating panel.

Upon the completion of loading the restart program, control is returned to the program and the necessary details involved in the restart process are completed.

INSTRUCTION FORMATS

There are two main groups of instructions:

1. Non-general instructions - Those for which the low-order bits specify a memory address -- for example, memory reference instructions which may be subject to address modification.

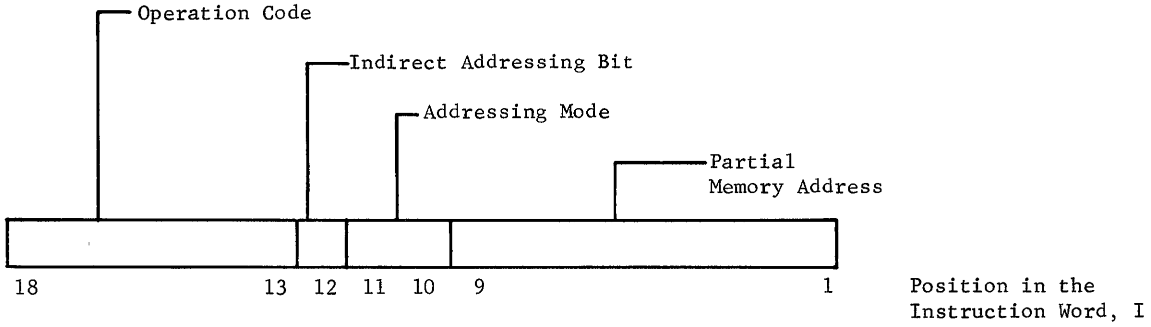
2. General instructions - Those for which the low-order bits contain information to be used by the instruction.

The notation I ( ) refers to the contents of an instruction word. General instructions may be recognized by the fact that the three high-order bits, I (16-18), are all zeros. (When expressed in octal notation, the general instructions start with a 0 in the high order position).

There is one format for non-general instructions and three for general instructions (register transfer, status line and function driver, and C-register instructions).

Non-general Instructions

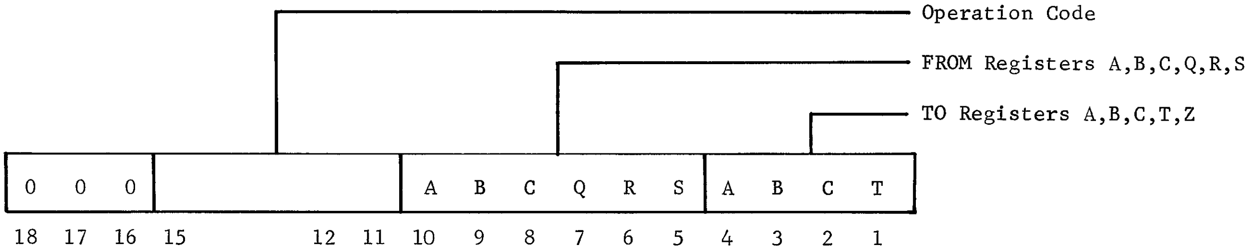
The non-general, or memory reference, instructions have four fields:



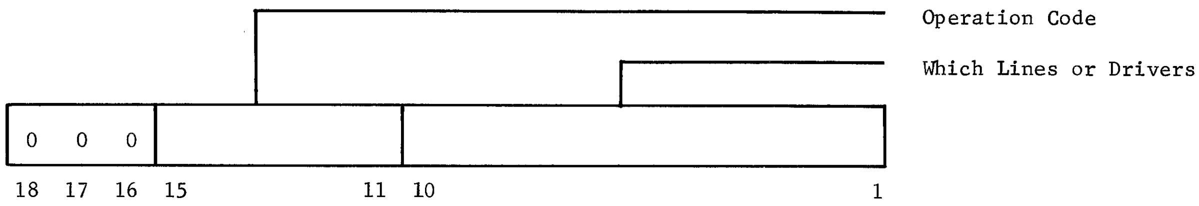
General Instructions

The fields for the three types of general instructions are as follows:

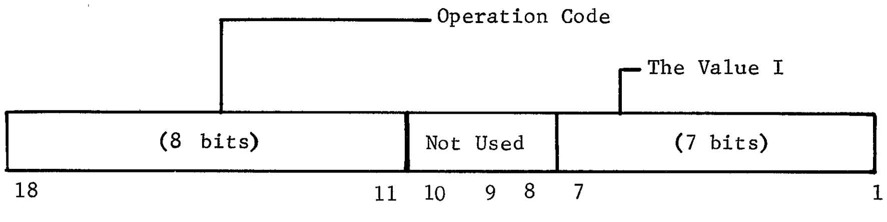
1. The register transfer instructions have three fields:



2. The status line and function driver instructions have two fields:



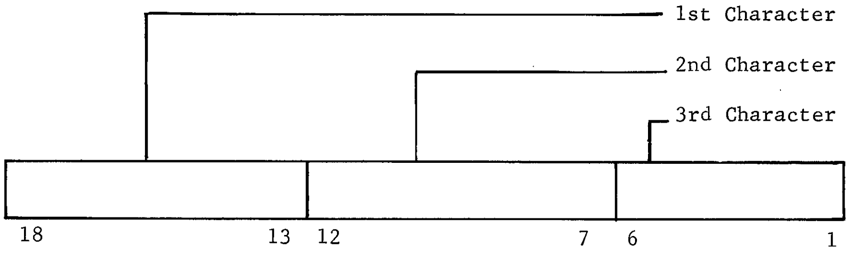
3. The C-register instructions have two pertinent fields:



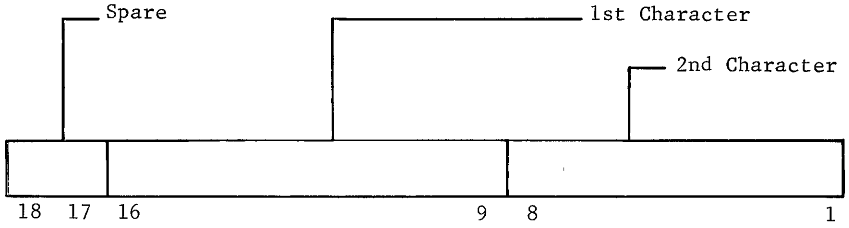
REPRESENTATION OF INFORMATION IN MEMORY

Alphanumeric Data

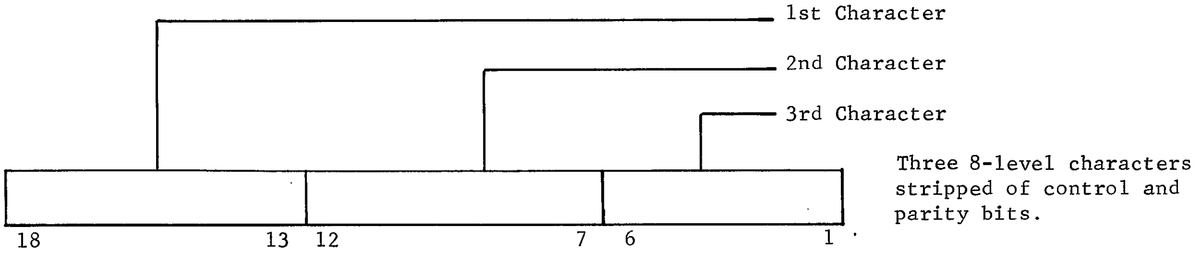
Each DATANET-30 word can contain three six-bit alphanumeric characters. The 64 possible bit combinations can be assigned to 64 symbols in any manner desired, because the DATANET-30 does not use alphanumeric data as a unique code. Therefore, other system conditions will determine the actual bit-pattern-to-symbol assignment. An alphanumeric data word would look like this in memory:

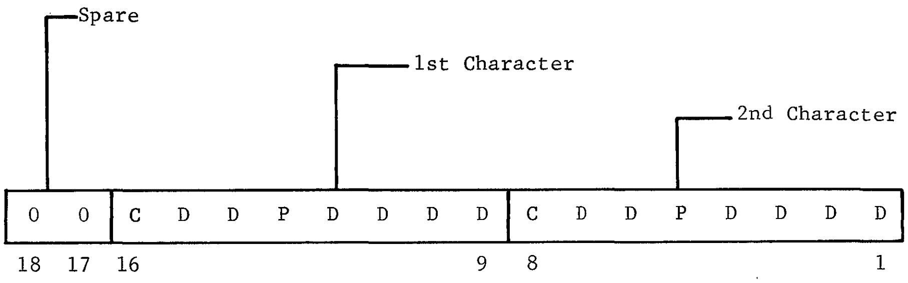


Each DATANET-30 word can contain two eight-bit alphanumeric characters. The particular code set used is dependent primarily on the remote terminals. This word might appear as follows:



Eight-level teletype characters can be stored conveniently in memory as six-bit characters. The DATANET-30 has two special instructions to facilitate stripping off and checking the parity and control bits when a character is received, and generation and insertion of parity and control bits when a character is to be transmitted. If desired for some applications, two eight-level characters could be stored in a word as eight-bit characters including the parity and control bits.


1st Character
2nd Character
3rd Character
Three 8-level characters stripped of control and parity bits.
18
13
12
7
6
1

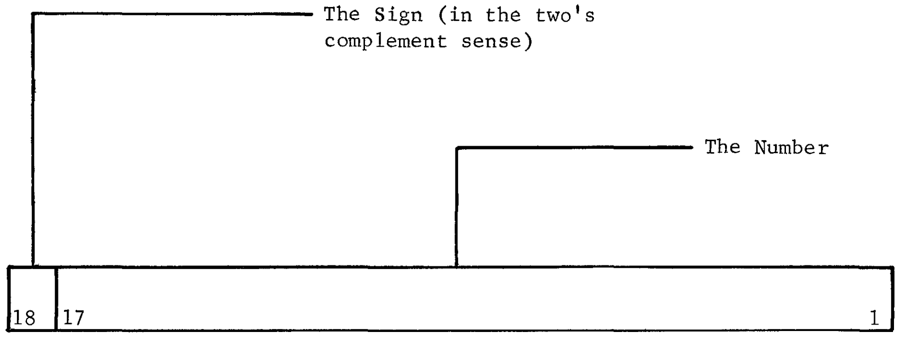


Two 8-level characters still containing parity and control bits, where:

C = Control Bit
D = Data Bit
P = Parity Bit

Numeric Data

Positive numbers are represented by integers. Negative numbers are represented in the 2's complement form. The DATANET-30 utilizes 2's complement arithmetic. Therefore, the high-order bit is properly thought of as the sign bit, when it is understood that the sign is a 2's complement sign, not an algebraic sign. The bits are shown in groups merely to simplify the presentation. There is no hardware sign bit in either the A or B registers. The sign is always programmed.



The number is considered a 17-bit number with bit 18 as the sign bit. In case of overflow of a positive number into bit 18 position, the sign changes and goes negative. Conversely, with a negative number, bit 18 will change in the event of overflow. This condition is tested with a Branch On Plus or Branch On Minus instruction.

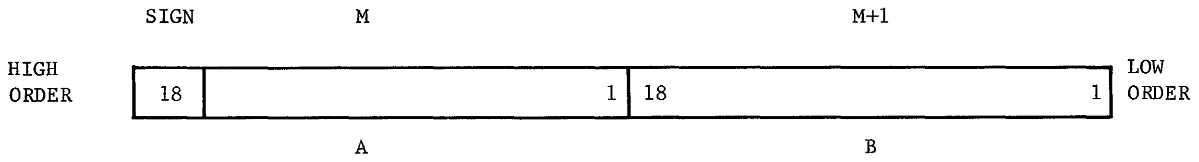
Examples of binary representation of numeric data are shown below:

SIGN	2^{16} 2^{15}	2^{14} 2^{13} 2^{12}	2^{11} 2^{10} 2^{9}	2^{8} 2^{7} 2^{6}	2^{5} 2^{4} 2^{3}	2^{2} 2^{1} 2^{0}	
0	00	000	000	000	000	000	0 (negative zero is not permissible)
18	17					1	
0	00	000	000	000	000	101	+5
18	17					1	
1	11	111	111	111	111	011	-5
18	17					1	
1	11	111	111	111	111	111	-1
18	17					1	
1	00	000	000	000	000	001	-131,071 (the largest negative number)
18	17					1	
0	11	111	111	111	111	111	+131,071 (the largest positive number)
18	17					1	

Double Length Binary Data

There are instructions which perform operation on double length words (36 bits). The numerical range is increased from (-131,071 to +131,071) to (-34, 353, 367 368 to +34, 359, 738, 367).

These double length words are stored in memory and the registers as below, where M(18), A(18) is a "two" complement sign. M must be even for all double length instructions.



The branch flip-flops are treated in a special manner by the three double length instructions (LDD, STD, AMD). The plus flip-flop is set on A(18). The zero flip-flop is set on the entire 36 bits of the double length result. The even flip-flop is set on B(1). The sign is programmed.

II. INSTRUCTION REPERTOIRE

There are over 78 basic instructions with many variations of some of them. These are classified into three groups:

1. Internal instructions
2. Buffer selector instructions
3. Controller selector instructions.

INTERNAL INSTRUCTIONS

The internal instructions are further classified into eight subgroups:

1. Load
2. Store
3. Arithmetic
4. Logical
5. Register Transfer
6. Branch
7. Macro
8. Special

In the following discussion, an M in the "Operand" column means that the instruction refers to a memory location. All such instructions use one of the addressing modes; therefore, no specific mention is made of these modes here.

I or FROM, TO in the operand column means that the information to be used in executing the instruction is made up of the bits in the low-order part of the instruction itself.

For brevity, the notation I (1-7) will be used for the 7 low-order bits of the instruction word. B (18) stands for the high-order bit of B. M stands for all 18 bits of the memory location; B stands for all 18 bits of the B-register; C stands for all 7 bits of the C-register, etc.

At times the discussion will refer to M as a memory location. It should be understood that what is really meant is the effective address -- that is, the memory location specified by M and the addressing mode. M is used for brevity.

The following word times assume that direct addressing is used. Add one additional word time when using indirect addressing. All instructions that address memory are also indirectly addressable.

Load Instructions

Mnemonic	Operand	Word Times
LDA	M	2
LOAD A.		The contents of M replace the contents of A. The contents of M are unchanged.
LDB	M	2
LOAD B.		The contents of M replace the contents of B. The contents of M are unchanged.
LDC	M	2
LOAD C.		The contents of M (1-7) replace the contents of C. The high order bits of M are ignored and M is unchanged.
LDD	M	3
LOAD DOUBLE.		The contents of M (1-18) replace the contents of A. The contents of M+1 replace the contents of B. M must be even. M and M+1 are unchanged.
LDQ	M	2
LOAD Q.		The contents of M replace the contents of Q. The contents of M are unchanged.
LDZ	M	2
LOAD Z.		The contents of M is placed only in Z and the branch flip-flops. M remains unchanged. Z sets up the branch flip-flops.
CMA	M	2
COMPLEMENT MEMORY TO A.		The 1's complement of the contents of M replaces the contents of A. The contents of M are unchanged.

Mnemonic	Operand	Word Times
CMB	M	2
COMPLEMENT MEMORY TO B.	The 1's complement of the contents of M replaces the contents of B. The contents of M remain unchanged.	
PIC	I	1
PLACE I IN C.	I (1-7) is placed in C. I is bits 1-7 of the instruction.	

Store Instructions

Mnemonic	Operand	Word Times
STA	M	2
STORE A.		The contents of A replace the contents of M. The contents of A remain unchanged.
STB	M	2
STORE B.		The contents of B replace the contents of M. The contents of B remain unchanged.
STC	M	2
STORE C.		The contents of C are stored in M (1-7). The contents of M (8-18) are reset to zero and C remains unchanged.
STD	M	3
STORE DOUBLE.		The contents of A are stored in M and the contents of B are stored in M+1. M must be even. The contents of A and B are unchanged.
STZ	M	2
STORE ZERO.		A zero is stored in M.
CAM	M	2
COMPLEMENT A TO MEMORY.		The 1's complement of the contents of A is stored in M, The contents of A remain unchanged.
CBM	M	2
COMPLEMENT B TO MEMORY.		The 1's complement of the contents of B is stored in M. The contents of B remain unchanged.
CMM	M	2
COMPLEMENT MEMORY TO MEMORY.		The 1's complement of the contents of M is stored in M, the same memory location.

Arithmetic Instructions

Mnemonic	Operand	Word Times
AMA	M	2
ADD MEMORY TO A.		The contents of M are added to the contents of A and the result is placed in A.
AMB	M	2
ADD MEMORY TO B.		The contents of M are added to the contents of B and the result is placed in B.
AIC	I	1
ADD I TO C.		I (1-7) are added to the contents of C and the result is placed in C.
AMD	M	3
ADD MEMORY DOUBLE.		The contents of M+1 are added to the contents of B and the result is placed in B, and the contents of M and a carry from the first are added to the contents of A and the result is placed in A. M must be even. M and M+1 are unchanged.
AAM	M	2
ADD A TO MEMORY.		The contents of A are added to the contents of M and the result is stored in M. A remains unchanged.
ABM	M	2
ADD B TO MEMORY.		The contents of B are added to the contents of M and the result is stored in M. B remains unchanged.
ADO	M	2
ADD ONE.		One is added to the contents of M and the result is stored in M.

Mnemonic	Operand	Word Times
SBO	M	2
SUBTRACT ONE.		One is subtracted from the contents of M and the result is stored in M.
AAZ	M	2
ADD A TO Z.		The contents of A are added to the contents of M. The result in the Z drivers is placed only in the branch flip-flops. A and M are unchanged.
ABZ	M	2
ADD B TO Z.		The contents of B are added to the contents of M. The result in the Z drivers is placed only in the branch flip-flops. B and M remain unchanged.

Logical Instructions

The truth table for the logical AND function is:

Y (A,B,C)	M (M,I)	Z (A,B,M)
0	0	0
0	1	0
1	0	0
1	1	1

Mnemonic	Operand	Word Times
NMA	M	2

AND MEMORY TO A.

A logical AND is performed with the contents of M and the contents of A. The result is placed in A.

NMB	M	2

AND MEMORY TO B.

A logical AND is performed with the contents of M and the contents of B. The result is placed in B.

NAM	M	2

AND A TO MEMORY.

A logical AND is performed with the contents of A and the contents of M. The result is stored in M.

NBM	M	2

AND B TO MEMORY.

A logical AND is performed with the contents of B and the contents of M. The result is stored in M.

NAZ	M	2

AND A TO Z.

A logical AND is performed on the contents of A and the contents of M. The result in the Z drivers is placed only in the branch flip-flops. A and M remain unchanged.

Mnemonic	Operand	Word Times
NBZ	M	2

AND B TO Z.

A logical AND is performed on the contents of B and the contents of M. The result in the Z drivers is placed only in the branch flip-flops. B and M remain unchanged.

Mnemonic	Operand	Word Times
NCZ	I	1

AND C TO Z.

A logical AND is performed on I (1-7) and the contents of C. The result in the Z drivers is placed only in the branch flip-flops. C remains unchanged.

The truth table for the logical OR function is:

Y (A,B)	M	Z (A,B,M)
0	0	0
0	1	1
1	0	1
1	1	1

Mnemonic	Operand	Word Times
RMA	M	2

OR MEMORY TO A.

A logical OR is performed with the contents of M and the contents of A. The result is placed in A.

Mnemonic	Operand	Word Times
RMB	M	2

OR MEMORY TO B.

A logical OR is performed with the contents of M and the contents of A. The result is placed in A.

Mnemonic	Operand	Word Times
RAM	M	2

OR A TO MEMORY.

A logical OR is performed with the contents of A and the contents of M. The result is stored in M.

Mnemonic	Operand	Word Times
RBM	M	2
OR B TO MEMORY.		A logical OR is performed with the contents of B and the contents of M. The result is stored in M.

The truth table for the logical EXCLUSIVE OR function is:

Y (A,B,C)	M (M,I)	Z (A,B,M)
0	0	0
0	1	1
1	0	1
1	1	0

XMA	M	2
EXCLUSIVE OR MEMORY TO A.		A logical EXCLUSIVE OR is performed with the contents of M and the contents of A. The result is placed in A.
XMB	M	2
EXCLUSIVE OR MEMORY TO B.		A logical EXCLUSIVE OR is performed with the contents of M and the contents of B. The result is placed in B.
XAM	M	2
EXCLUSIVE OR A TO MEMORY.		A logical EXCLUSIVE OR is performed with the contents of A and the contents of M. The result is stored in M.
XBM	M	2
EXCLUSIVE OR B TO MEMORY.		A logical EXCLUSIVE OR is performed with the contents of B and the contents of M. The result is stored in M.

Mnemonic	Operand	Word Times
XAZ	M	2
EXCLUSIVE OR A TO Z.		A logical EXCLUSIVE OR is performed on the contents of A and M. The result in the Z drivers is placed only in the branch flip-flops. A and M remain unchanged.
XBZ	M	2
EXCLUSIVE OR B TO Z.		A logical EXCLUSIVE OR is performed on the contents of B and the contents of M. The result in the Z drivers is placed only in the branch flip-flops. B and M remain unchanged.
XCZ	I	1
EXCLUSIVE OR C TO Z.		A logical EXCLUSIVE OR is performed on I (1-7) and the contents of C. The result in Z is placed only in the branch flip-flops. C remains unchanged.

Register Transfer Instructions

All of the register transfer instructions use the low order bits of the instruction to specify which locations are to be included in the FROM group and which in the TO group. The possibilities are:

				Bit Position in I
FROM:	A	The A-register	-	10
	B	The B-register	-	9
	C	The C-counter	-	8
	Q	The Q-counter	-	7
	R	The receive data lines (From X, the address of a particular buffer)	-	6
	S	The insert switches	-	5
	Ø	Zero is transferred to the specified TO location		
TO:	A	The A-register	-	4
	B	The B-register	-	3
	C	The C-counter	-	2
	T	The transmit data lines (To X, the address of a particular buffer)	-	1
	Z	The Z-drivers; FROM remains unchanged.		

If R, S, or T is specified, the control bit 1, control bit 2, and parity flip-flops (internal functions) are used for the "extra" positions, since R, S and T are all more than 18 bits.

Any register specified in the FROM group will remain unchanged after the register transfer operation if it does not appear in the TO group. If R is specified in the FROM group, after the data is transferred, the receive flag and receive data buffer are reset by an automatically generated signal activating external function driver 1 (DEF1).

With the exception of T in the TO group, the TO register will contain the result after a register transfer instruction. If T is specified in the TO group, before the data is transferred, the transmit flag and transmit buffer are reset by an automatically generated signal activating external function driver 2 (DEF2).

When a register transfer instruction is executed, the contents of those registers which are specified to be used as the FROM group for this instruction are logically OR-ed together into the Y-register. Then the data goes from Y to Z with the operation specified by the instruction being performed on the data as it goes from Y to Z. Finally the result goes from the Z drivers to all of those registers which are specified in the TO group. The plus, zero, and even flip-flops will take on their new states in the normal manner. If no registers are specified in the FROM group, the output from the Y-register will be zero. If no registers are specified in the TO group, the only outputs are the new states of the plus, zero, and even flip-flops. Register transfer instructions with more than one register in the FROM and TO groups can be specified. For example: TRA O,ABC; TRA ABC,Z; SL6 BC,AB.

Mnemonic	Operand	Word Times
TRA	FROM, TO	1
TRANSFER.		In going from Y to Z, no change is made in the data.
TRC	FROM, TO	1
TRANSFER COMPLEMENT.		In going from Y to Z, the data is changed into its 1's complement.
SL1	FROM, TO	1
SHIFT LEFT ONE.		In going from Y to Z, the data is shifted left one position. The high-order bit is lost and a zero goes into the low-order position.
SR1	FROM, TO	1
SHIFT RIGHT ONE.		In going from Y to Z, the data is shifted right one position. The low-order bit is lost and a zero goes into the high-order position.
SL6	FROM, TO	1
SHIFT LEFT SIX.		In going from Y to Z, the data is shifted left six positions. The six high-order bits are lost and zeros go into the six low-order positions.
SR6	FROM, TO	1
SHIFT RIGHT SIX.		In going from Y to Z, the data is shifted right six positions. The six low-order bits are lost and zeros go into the six high-order positions.

Mnemonic	Operand	Word Times
CL1	FROM, TO	1

CIRCULATE LEFT ONE.

In going from Y to Z, the data is circulated left one position. The high-order bit goes into the low-order position; no bits are lost.

CR1	FROM, TO	1

CIRCULATE RIGHT ONE.

In going from Y to Z, the data is circulated right one position. The low-order bit goes into the high-order position; no bits are lost.

CL6	FROM, TO	1

CIRCULATE LEFT SIX.

In going from Y to Z, the data is circulated left six positions. The six high-order bits go into the six low-order positions; no bits are lost.

CR6	FROM, TO	1

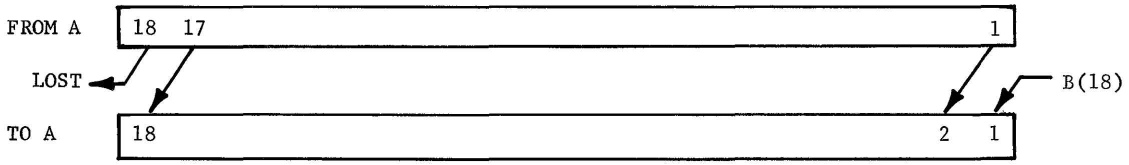
CIRCULATE RIGHT SIX.

In going from Y to Z, the data is circulated right six positions. The six low-order bits go into the six high-order positions; no bits are lost.

SLS	FROM A, TO A	1

SHIFT LEFT SPECIAL.

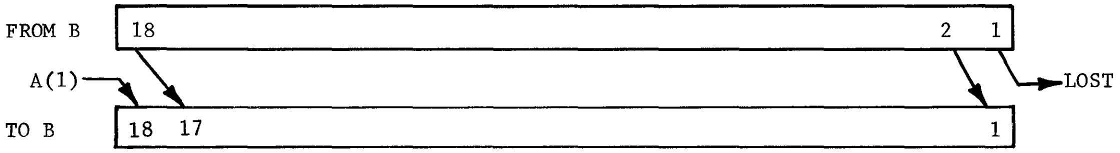
This instruction is a SL1 instruction with one added function - Z (1) = B (18). Bit B (18) is shifted into Bit A (1).



SRS	FROM B, TO B	1

SHIFT RIGHT SPECIAL.

This instruction is a SR1 instruction with one added function - Z (18) = A (1). Bit A (1) is shifted into Bit B (18).



Mnemonic	Operand	Word Times
BCO	FROM, TO	1

BIT CHANGE ZERO.

This is a special instruction for use with eight-level Frieden data. In going from Y to Z, the data is rearranged from the eight-level format used on a transmission line to the six-bit alphanumeric format used in computers. The other two bits, the parity and control bits, are put in the CB1 and CB3 flip-flops.

FROM	X	X	X	X	X	X	X	X	X	C	D_6	D_5	P	D_4	D_3	D_2	D_1	X	Y
TO	0	0	0	0	0	0	0	0	0	0	0	0	D_6	D_5	D_4	D_3	D_2	D_1	Z

P goes to the parity flip-flop (CB3)

C goes to the control bit flip-flop 1 (CB1)

BC1	FROM, TO	1

BIT CHANGE ONE.

This is the reverse operation of BCO. In going from Y to Z, the data is rearranged from the six-bit alphanumeric format into the eight-level format used on a transmission line. The control bit comes from BC1 and the parity bit comes from the output of the character parity network.

FROM	X	X	X	X	X	X	X	X	X	X	X	X	D_6	D_5	D_4	D_3	D_2	D_1	Y
TO	0	0	0	0	0	0	1	1	1	C	D_6	D_5	P	D_4	D_3	D_2	D_1	0	Z

P is the output from the character parity network

C is the control bit 1 flip-flop (CB1)

Branch Instructions

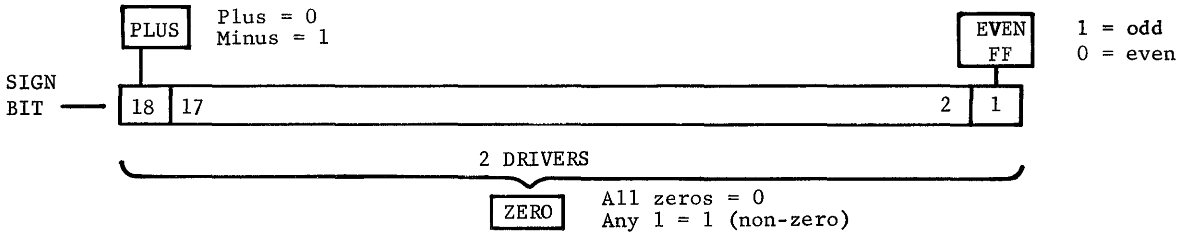
The states of the plus, zero, and even flip-flops are not changed by any branch instruction.

Mnemonic	Operand	Word Times
BRU	M	1
BRANCH UNCONDITIONALLY.		Control is transferred to the instruction in M within the same program bank. When indirect addressing is specified, control is transferred to the address in M.
BRS	M	3
BRANCH TO SUBROUTINE.		The location of the instruction following the BRS is stored in M; then, control is transferred to the location specified by the contents of M+1. M must be even.

The remaining branch instructions are conditional branches. Control is transferred to M if the appropriate conditional test is satisfied. Otherwise, control goes to the next instruction - that is, the instruction following the branch instruction.

BZE	M	1
BRANCH ON ZERO.		If the ZFF is zero, control is transferred to M.
BNZ	M	1
BRANCH ON NON-ZERO.		If the ZFF is non-zero, control is transferred to M.
BPL	M	1
BRANCH ON PLUS.		If the plus flip-flop is plus, control is transferred to M.
BMI	M	1
BRANCH ON MINUS.		If the plus flip-flop is minus, control is transferred to M.

Mnemonic	Operand	Word Times
BEV	M	1
BRANCH ON EVEN.	If the even flip-flop is even, control is transferred to M.	
BOD	M	1
BRANCH ON ODD.	If the even flip-flop is odd, control is transferred to M.	



Macro-Instructions

The following instructions are macro-instructions. That is, they are not actual machine instructions; however, the General Assembly Program will recognize the mnemonics for the macro-instructions and generate the appropriate series of instructions to do the specified operation.

Mnemonic	Operand	Word Times
CL2	FROM, TO	2
CIRCULATE LEFT 2.		The contents of the specified FROM location is shifted left 2 places. The bits leaving position 18 are shifted into position 1 of the TO location.
CL3	FROM, TO	3
CIRCULATE LEFT 3.		
CL4	FROM, TO	3
CIRCULATE LEFT 4.		
CL5	FROM, TO	2
CIRCULATE LEFT 5.		
CL7	FROM, TO	2
CIRCULATE LEFT 7.		
CL8	FROM, TO	3
CIRCULATE LEFT 8.		
CL9	FROM, TO	4
CIRCULATE LEFT 9.		
CR2	FROM, TO	2
CIRCULATE RIGHT 2.		The contents of the specified FROM location are shifted right 2 places. Bits leaving position 1 are shifted into position 18 of the TO location.

Mnemonic	Operand	Word Times
CR3	FROM, TO	3
CIRCULATE RIGHT 3.		
CR4	FROM, TO	3
CIRCULATE RIGHT 4.		
CR5	FROM, TO	2
CIRCULATE RIGHT 5.		
CR7	FROM, TO	2
CIRCULATE RIGHT 7.		
CR8	FROM, TO	3
CIRCULATE RIGHT 8.		
CR9	FROM, TO	4
CIRCULATE RIGHT 9.		
SAM	M	7
SUBTRACT B FROM MEMORY.		The contents of the A-register are subtracted from the specified memory location M. The result is placed in M.
SBM	M	7
SUBTRACT B FROM MEMORY.		The contents of the B-register are subtracted from the specified memory location M. The result is placed in M.
SL2	FROM, TO	2
SHIFT LEFT 2.		The contents of the FROM location are shifted left 2 binary places and put into the TO location.

Mnemonic	Operand	Word Times
SL3	FROM, TO	3
SHIFT LEFT 3.		
SL4	FROM, TO	4
SHIFT LEFT 4.		
SL5	FROM, TO	5
SHIFT LEFT 5.		
SL7	FROM, TO	2
SHIFT LEFT 7.		
SL8	FROM, TO	3
SHIFT LEFT 8.		
SL9	FROM, TO	4
SHIFT LEFT 9.		
SLD	I	2(I)
SHIFT LEFT DOUBLE.		The contents of registers A and B are shifted left double I number of times. Bits shifted out of B (18) enter A (1). Bits shifted out of A (18) are lost. The vacated positions of the B-register are filled with zeros.
SMA	M	4
SUBTRACT MEMORY FROM A.		The contents of the specified memory location M are subtracted from the contents of the A-register. The result is placed in A.

Mnemonic	Operand	Word Times
SMB	FROM, TO	4
SUBTRACT MEMORY FROM B.		The contents of the specified memory location M are subtracted from the contents of the B-register. The result is placed in B.
SR2	FROM, TO	2
SHIFT RIGHT 2.		The contents of the FROM location are shifted right 2 binary places and placed in the TO location.
SR3	FROM, TO	3
SHIFT RIGHT 3.		
SR4	FROM, TO	
SHIFT RIGHT 4.		
SR5	FROM, TO	5
SHIFT RIGHT 5.		
SR7	FROM, TO	2
SHIFT RIGHT 7.		
SR8	FROM, TO	4
SHIFT RIGHT 8.		
SR9	FROM, TO	4
SHIFT RIGHT 9.		
SRD	I	2(I)
SHIFT RIGHT DOUBLE.		The contents of registers A and B are shifted right I places. The vacated positions of the A-register are filled with zeros. Bits shifted out of A (1) go into B (18). Bits shifted out of B (1) are lost.

Special Instructions

INTERNAL FUNCTION DRIVERS

Mnemonic	Operand	Word Times
DIF	I	1

DRIVE INTERNAL FUNCTION.

A signal will be sent to those internal function drivers which correspond to 1-bits in I.

	Function
DIF 1	Reset control bit flip-flops 1 and 2, and parity bit flip-flop.
DIF 2	Reset the buzzer flip-flop.
DIF 3	Set the buzzer flip-flop.
DIF 4	Initiate the hardware load process.
DIF 5-6	Not assigned.
DIF 7	This is the SEL instruction.
DIF 8	Set control bit flip-flop 1.
DIF 9	Set control bit flip-flop 2.
DIF 0	Set the parity bit flip-flop.

INTERNAL STATUS LINES

NIS	I	1

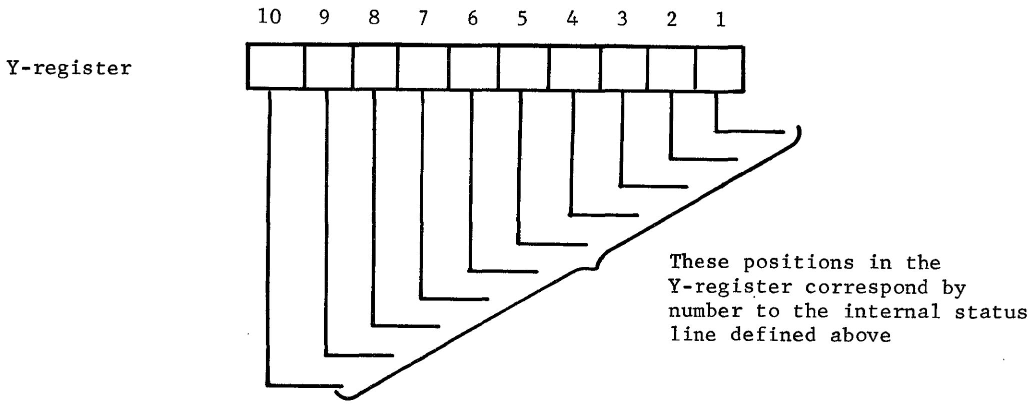
AND INTERNAL STATUS LINES TO Z.

The NIS instructions allow the program to interrogate the status of the I internal status lines. A logical AND is performed with I (1-10) and the internal status lines.

The result of the AND sets the branch flip-flops in accordance with the results of the AND.

If the tested condition is true, the zero flip-flop will have been set $\neq 0$. A 1 is a true condition. If the zero flip-flop is to be 0, then Z (1-10) must all have been 0.

Mnemonic		Operand	Word Times
NIS 1	Will be true if		The character parity output of the parity network is a 1.
NIS 2	Will be true if		The word parity output of the parity network is a 1.
NIS 3	Will be true if		Control bit flip-flop 2 and the word parity output of the parity network are identical. This is intended for use when transmitting data with error-correcting techniques.
NIS 4	Will be true if		The OPERATING MODE/MAINTENANCE MODE switch is in the MAINTENANCE MODE position.
NIS 5-6	Will be true if		Not assigned.
NIS 7	Will be true if		Controller selector is ready.
NIS 8	Will be true if		Control bit flip-flop 1 is a 1.
NIS 9	Will be true if		Control bit flip-flop 2 is a 1.
NIS 0	Will be true if		The parity bit flip-flop is a 1.



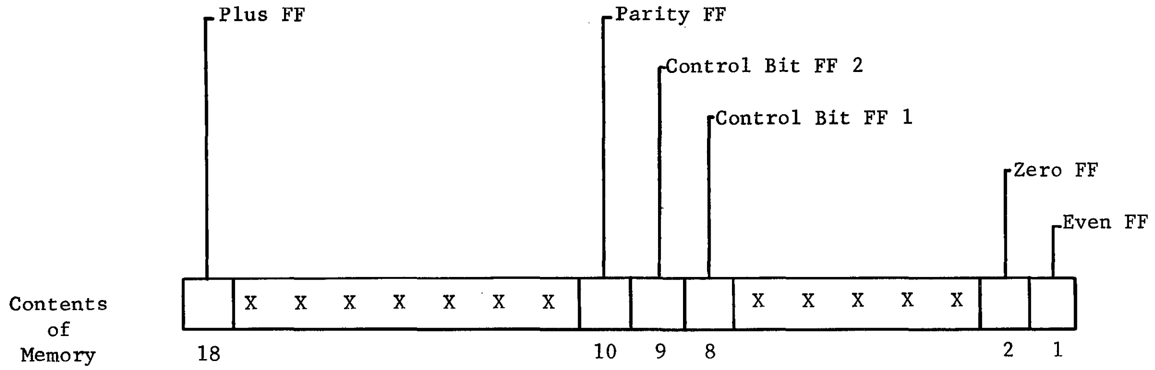
These positions in the Y-register correspond by number to the internal status line defined above

Bit Position of Internal Status Lines

LDF M 2

LOAD SPECIAL FLIP-FLOPS.

Selected bits from the contents of M are used to restore the conditions (saved by a STF instruction) of the plus, zero, even, control bit 1, control bit 2, and parity flip-flops. Bit position 1 goes to the even flip-flop. Bit position 2 goes to the zero flip-flop and bit position 18 goes to the plus flip-flop. Bits 8, 9, and 10 go to control bit flip-flops 1 and 2 and the parity flip-flop, respectively.



Mnemonic **Operand** **Word Times**

STF M 2

STORE SPECIAL FLIP-FLOPS.

The conditions of the plus, zero, even, control bit 1, control bit 2, and parity flip-flops are stored in M in positions 18, 2, 1, 8, 9, and 10, respectively (same as in LDF).

HLT I 1

CONDITIONAL HALT.

The DATANET-30 will halt if this instruction is executed when the INHIBIT HALT switch on the MAINTENANCE panel is in DISABLE position. If the INHIBIT HALT switch is in the OPERATE position, hardware load will be initiated when this instruction is executed.

Buffer Selector Instructions

There are six buffer selector instructions. The register transfer FROM R, and the register transfer TO T have already been covered.

Mnemonic	Operand	Word Times
LDT	M	2

LOAD T.

The contents of M are sent to the transmit data drivers and from there to whichever channel has been preselected by the contents of the C-counter. The contents of M are unchanged (used only with the CIU-930).

EXTERNAL FUNCTION DRIVERS

DEF	I	1

DRIVE EXTERNAL FUNCTION.

A signal will be sent to those external function drivers which correspond to 1's in I. The signal(s) will actually get to only the buffer unit which has been preselected by the C-counter. The meaning of each driver varies with the particular input/output device.

	BBC	CBC	WBC	RPU
DEF 1	A	A	A	
2	B	B	B	
3	C	C	C	
4	D	D	D	
5	E	NU	NU	
6	NU	NU	NU	
7	NU	NU	NU	
8	NU	NU	NU	
9	NU	F	NU	
0	NU	G	NU	

A - Reset receive flag and receive data buffer.
B - Reset transmit flag and data buffer.
C - Set receive mode. (Turn carrier off.)
D - Set transmit mode. (Turn carrier on.)
E - Reset receive clock.
NU - Not used

EXTERNAL STATUS LINES

Mnemonic	Operand	Word Times
NES	I	1

AND EXTERNAL STATUS LINES TO Z.

A logical AND is performed with I (1-10) and the external status lines. The only results are the new states of the plus, zero, and even flip-flops. The meaning of each line varies with the particular input/output device.

	BBC	CBC	WBC	RPU
NES 1	A	E	G	see page
2	B	F	H	
3	NU	J	NU	
4	NU	K	↓	
5	C	G	↓	
6	D	M	↓	
7	NU	N	↓	
8	NU	NU	↓	
9	NU	NU	↓	
0	NU	NU	NU	

A - Receive flag is set (data buffer contains a new bit).
B - Transmit flag is set (data buffer is ready for a new bit).
C - Interlock on.
D - Carrier on.
E - Receive flag is set (data register contains a new character).
F - Transmit flag is set (data register is ready for a new character).
G - Receive flag is set (data register contains a new word).
H - Transmit flag is set (data register is ready for a new word).
NU - Not used.

Mnemonic	Operand	Word Time
SCN	I	1+3N

SCAN.

The bit buffer channels are scanned starting with channel I. N equals the number of channels scanned. The instruction is terminated upon detection of the end scan plug in scan word 2, field 2.

III. ADDRESSING MEMORY

General Description

The address field of the instruction is divided into a partial memory address and an addressing mode.

12 10	9 1
MODE	ADDRESS

The four modes for addressing memory are:

1. Program Bank addressing
2. Common Data Bank addressing
3. Channel Table addressing
4. Indirect

Bit Positions

12	11	10	
0	0	X	Program Bank addressing
0	1	0	Common Data Bank addressing
0	1	1	Channel Table Address
1	X	X	To any of the 3 above

DETAILED DESCRIPTION

The following descriptions of the hardware aspects of memory addressing are given for use when debugging programs. The General Assembly Program automatically assigns proper addressing for each instruction.

Program Bank Addressing

Program bank addressing can only address locations in the common data bank or another location in the same program bank. The addresses within 1024 memory locations of the base location of the program bank in which the instruction is located may be directly addressed by an instruction within the program bank.

The eight 1024-word program banks for an 8192-word memory are listed in the table below:

	Memory Locations				
Program Bank	Start			End	
	Decimal	Octal		Decimal	Octal
1	0000	0000	to	1023	1777
2	1024	2000	to	2047	3777
3	2048	4000	to	3071	5777
4	3072	6000	to	4095	7777
5	4096	10000	to	5119	11777
6	5120	12000	to	6143	13777
7	6144	14000	to	7167	15777
8	7168	16000	to	8191	17777

Each program bank has upper and lower limits for direct addressing. When it is necessary to go from one program bank to another, indirect addressing is used. When approaching the upper limit of a program bank,. some caution is necessary regarding the type of instruction placed in the last location of the program bank. Upon the execution of the last instruction in a program bank, the P-counter contains the address of the first instruction in the next program bank. If a branch instruction is in the last location, the program will branch to the corresponding address in the next program bank.

There are two ways to change from one program bank to another:

1. The P-counter counts up past the program bank boundary.
2. A branch instruction is given in the indirect mode.

```
   Location    Instruction     Symbol     OPR       Operand                  Remarks

                 01750                    ORG       1000       ORIGIN IN 1ST PROGRAM BANK
    01750       000001         FIRST      DEC       1
                 03720                    ORG       2000       ORIGIN IN 2ND PROGRAM BANK
    03720       000002         SECOND     DEC       2
                 05670                    ORG       3000       ORIGIN IN 3RD PROGRAM BANK
    05670       000003         THIRD      DEC       3
                 07640                    ORG       4000       ORIGIN IN 4TH PROGRAM BANK
    07640       000004         FOURTH     DEC       4

    START EXAMPLE PROGRAM

                 01604                    ORG       900        ORIGIN LOCATION
    01604       401750                    LDA       FIRST      PROGRAM BANK ADDRESSING APPEARS.
                                                               PROGRAM BANK ADDRESSING CAN BE NOTED
                                                               BY A BINARY 01 IN BIT POSITIONS 11
                                                               AND 10.  THIS CAN BE SEEN AS AN
                                                               OCTAL 01 IN THE MACHINE INSTRUCTION.

A   01605       400000                    LDA       FOURTH     THIS INSTRUCTION PRODUCES AN ERROR
                                                               TAG (A) BECAUSE THE SYMBOL "FOURTH"
                                                               IS NOT IN THE SAME PROGRAM BANK OR
                                                               THE COMMON DATA BANK********

                 03554                    ORG       1900       ORIGIN LOCATION
    03554       401720                    LDA       SECOND     NOTE PROGRAM BANK ADDRESSING

                 05524                    ORG       2900       ORIGIN LOCATION
    05524       401670                    LDA       THIRD      NOTE PROGRAM BANK ADDRESSING

                 07474                    ORG       3900       ORIGIN LOCATION
    07474       401640                    LDA       FOURTH     NOTE PROGRAM BANK ADDRESSING

                                                               THE PROGRAM BANK ADDRESSING CAN BE
                                                               NOTED BY THE 3RD OCTAL DIGIT IN EACH
                                                               OF THE PRECEDING LDA INSTRUCTIONS.

   101750                                 END       1000
```

Common Data Bank Addressing

The common data bank is the first 512 words of memory and may be addressed directly from any location in memory. In the following example, common data bank addressing is denoted by the 2 in the third digit of the octal instruction. All instructions that refer to an address in the common data bank will always be assigned commondata bank addressing by the General Assembly Program.

Location	Instruction	OPR	Operand	Remarks
	11610	ORG	5000	
11610	402024	LDA	50	LOAD A Register with contents cell 20_{10}
11611	702231	STB	153	STORE B Register in location 153_{10}
11612	342764	ADO	500	ADD one to location 500_{10}

Channel Table Addressing

A channel table is a table with a mnemonic that is symbolic and starts with the character $. The starting locations of the channel table must be a multiple of 16 decimal and located in the first 8192 words of memory. The channel table may be addressed directly from anywhere in memory. The maximum table length is 128 locations. When referred to, the base address (starting location) is automatically indexed by the C-register. The channel table addressing mode will be assigned to any instruction which refers to a channel table ($ - -).

Example 1:

```
                ORG    512
        $SW1    DEC 0            Scan Word Table Channel 0
                 .
                 .               Scan Word Table Channel 1
```

Example 2:

```
                ORG    608
        $POINT  DEC 0            Pointer for Channel 0
                 .
                 .               Pointer for Channel 1
```

Example 3:

```
                ORG    2048
                PIC    1
4000    403040  LDA    $SW1      The A-register is loaded with the contents of location 513
                                 (Location 512 + value of C-register)
```

If the number of channels (table size) exceeds 16, the location of the table must be a multiple of the next higher power of 2.

Example:

Number of Channels	Starting location must be a multiple of
0-16	16
17-32	32
33-64	64
65-128	128

Indirect Addressing

Indirect addressing (2nd level addressing) is where the address part of an instruction is the location in memory where the address of the operand may be found or is to be stored.

Indirect addressing is specified in an instruction when an X is placed in the index column (col. 20) of the coding sheet.

Indirect addressing must be used to access an address in another Program Bank, with the exception of the Common Data Bank or Channel Table. It must also be used to branch across bank boundries.

Indirect address (second level address) example:

Location	Instruction	OPR	Operand	X	Remarks
	*	ORG	2048		
4000	404030	LDA	POINT	X	Load Register A with alpha
	*	.			
	*	.			
4030	POINT	IND	ALPHA		
		.			
		.			
7760	ALPHA	OCT	000174		

Indexing

During indirect addressing, the first operand address can be indexed by any one of A-, B-, or C-registers by specifying which register in the pointer. Bits 16-17 of the indirect address word specify which register to be used for indexing as follows:

Bits (18-17-16)	Function	Pseudo-Operation	
000	No indexing	IND	
001	Index by A	INA	Base address indexed by contents of A
010	Index by B	INB	Base address indexed by contents of B
011	Index by C	INC	Base address indexed by contents of C

The pseudo-operations IND, INA, INB, and INC are used by the General Assembly Program to automatically add these bits as required.

```
 LOC    INSTRUCTION          OPR  OPERAND       X              REMARKS

                             ORG  2048
                      *
                      *   Convert Octal digit to baudot
                      *
04000   601100               LDB  DIGIT                     Pick up octal digit
04001   404400               LDA  BAUDOT        X           Convert
  .                   *
  .                   *   BAUDOT CONVERSION TABLE
  .                   *
04400   204401        BAUDOT INB  *+1                       Octal to Baudot Conv Table
04401   000054               OCT   54                       Baudot Char 0
04402   000056               OCT   56                                 "  1
04403   000046               OCT   46                                    2
04404   000002               OCT   02                                    3
                              .
                              .
                              .
                              .
05100   000002        DIGIT  OCT  000002
                      *
                      *   Branch to switch table
                      *   Depending on contents of C-register
                      *
                             ORG  2048
04000   201100               LDC  DIGIT                     Pick up value in C-reg
04001   104400               BRU  $POINT        X
                              .
                              .
                              .
                             ORG  4096
10000   010200        $POINT IND  ENTER 0                   GO TO ENTER 0 IF C = 0
10001   010300               IND  ENTER 1                             "  1       1
10002   010400               IND  ENTER 2                             "  2       2
10003   010500               IND  ENTER 3                             "  3       3
```

Subroutine Linkage

Indirect addressing and a special Branch Subroutine (BRS) instruction provide a means for getting to and from subroutines and program banks. The BRS command is a 3-word-time instruction which, during the first execution cycle, stores P+1 (the address of the word following the BRS) in memory location M and during the second cycle loads the contents of (M+1) into the P-counter, as follows:

```
ALPHA     BRS    SUBRN                    Transfer to Subroutine
           .
          LDA      0                      Continue
           .
           .
SUBRN     IND      0                      Subroutine linkage
          IND    SUBRN1
SUBRN1    LDB    SUBRN                    Start of subroutine
           .

           .
          BRU    SUBRN      X             Exit from subroutine
```

When the BRS at location ALPHA is executed:

1. The P-counter + 1 is stored in SUBRN.
2. The program branches to location contained in SUBRN+1.
3. The subroutine is executed. This subroutine may be located anywhere in memory.
4. The exit from the subroutine via the BRU SUBRN X causes the contents of SUBRN (location ALPHA+1) to be loaded into P.
5. The LDA instruction following the BRS is executed after execution of the subroutine.

Thus, 1 instruction (BRS), 2 words in memory (SUBRN and SUBRN+1), and 5 word times (BRS and BRU X) are needed for the general subroutine linkage, since the two linkage words are normally in the common data bank and can be accessed from anywhere in memory.

This technique of subroutine linkage has these advantages:

1. Only 1 instruction is needed in the main program to call a subroutine.
2. The subroutine may be located anywhere in memory at no sacrifice in time or memory.
3. The subroutine may be called from anywhere in memory at no sacrifice in time or memory.

4. All subroutine linkage bookkeeping is handled by hardware and not by the main program or the subroutine.

5. All three registers, A, B, and C, may be used for input to the subroutine, since no register is used for linkage.

The following rules must be observed when using the subroutine BRS command.

1. The first word of the subroutine linkage must be in an even location. (The General Assembly Program will error tag an odd location or force it to an even location.)

2. The subroutine linkage must be placed in a common location to both program points, i.e., common data bank, same program bank.

IV. CONTROL CONSOLE

The control console (Figure 13) serves both operator and maintenance functions. The control exercised by the console is not normally used during normal program execution. Control from the console is concerned with initially loading the program into memory, starting the execution thereof, monitoring the progress of the program, and program debugging.

The switches and lights and their more important functions are:

1. The contents of the A, B, C and P registers may be modified directly from the control console.
2. The contents of memory may be displayed in the M-register. The P-counter is used to specify the memory location to be displayed.
3. The P-counter is automatically incremented so that sequential locations in memory may be displayed by depressing the SINGLE CYCLE button.
4. The contents of memory may be modified by the 18 INSERT SWITCHES.
5. The automatic loading of a program may be initiated from the control console (hardware load).

THE MODE SELECT PUSHBUTTON SWITCHES

The SET A, B, C, and P Button

The following steps are used to set the A, B or C registers and the P-counter to a desired configuration.

1. Press the Set A, B, C, or P button.
2. Lift the INSERT SWITCHES under the register position to be inserted.
3. The inserted configuration is immediately set up in the desired register (counter).

The INSERT MEMORY Button

The following steps are used to insert data into memory:

1. Press SET P button.

2. Put desired memory address in the P-counter.

3. Press the INSERT MEMORY button.

4. Lift the INSERT SWITCHES to the desired input. The input is indicated in the Y-register.

5. Press the SINGLE CYCLE button. The input from the Y-register is transferred to the memory location specified by the P-counter. The P-counter will count up 1.

6. Insert the next desired input into the Y-register with the insert switches.

7. Press the SINGLE CYCLE button. The input in the Y-register is transferred to memory location specified by the P-counter.

8. Continue steps 4 and 5 until all input has been inserted into memory.

9. Press the PROGRAM RUN button, then the RUN button to start the program. The program will start at the location specified by the P-counter.

The DISPLAY MEMORY Button

The following steps allow the contents of memory to be displayed:

1. Press the SINGLE CYCLE button to halt.

2. Press DISPLAY MEMORY button.

3. Press SINGLE CYCLE. The contents of memory location as specified by the P-counter are displayed in the M-register. The P counter counts up 1.

4. The contents of the other registers will be as previously defined under description of registers.

THE ERROR LIGHT AND BUZZER

The ERROR light and buzzer are used to indicate that data read out of memory does not agree with the INSERT SWITCHES.

If a DIF 3 instruction is executed, the error light will turn on. This does not indicate an alert halt and the program will continue to run.

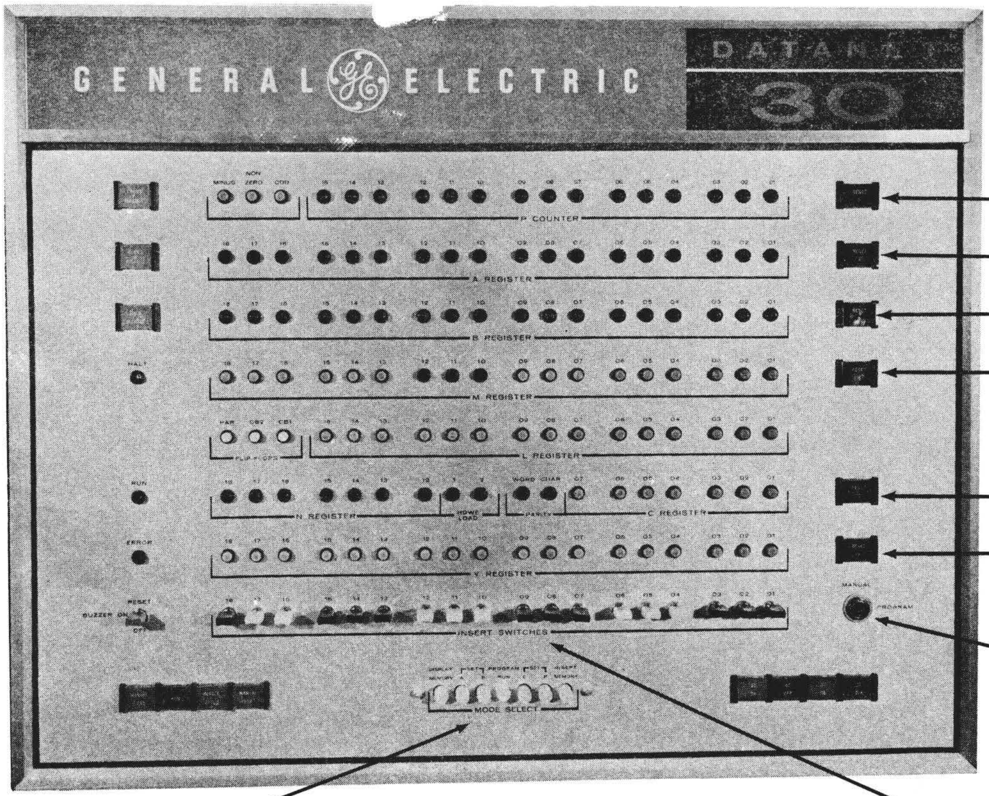
The error light and buzzer only work in either the DISPLAY MEMORY or INSERT MEMORY mode. The error light does not refer to an error in an operating program. The error light and buzzer are both turned on and off with the DIF 3 and DIF 2 instructions.

If the INSERT MEMORY or DISPLAY MEMORY mode is set, the RUN button has been pressed, and the HALT/DISABLE switch is in the HALT position, the error light turning on will indicate an error, halt the DATANET-30, and the location of the error will be indicated in the L-register. This is mainly a maintenance feature.

POWER-ON SEQUENCE

The power-on sequence is shown below:

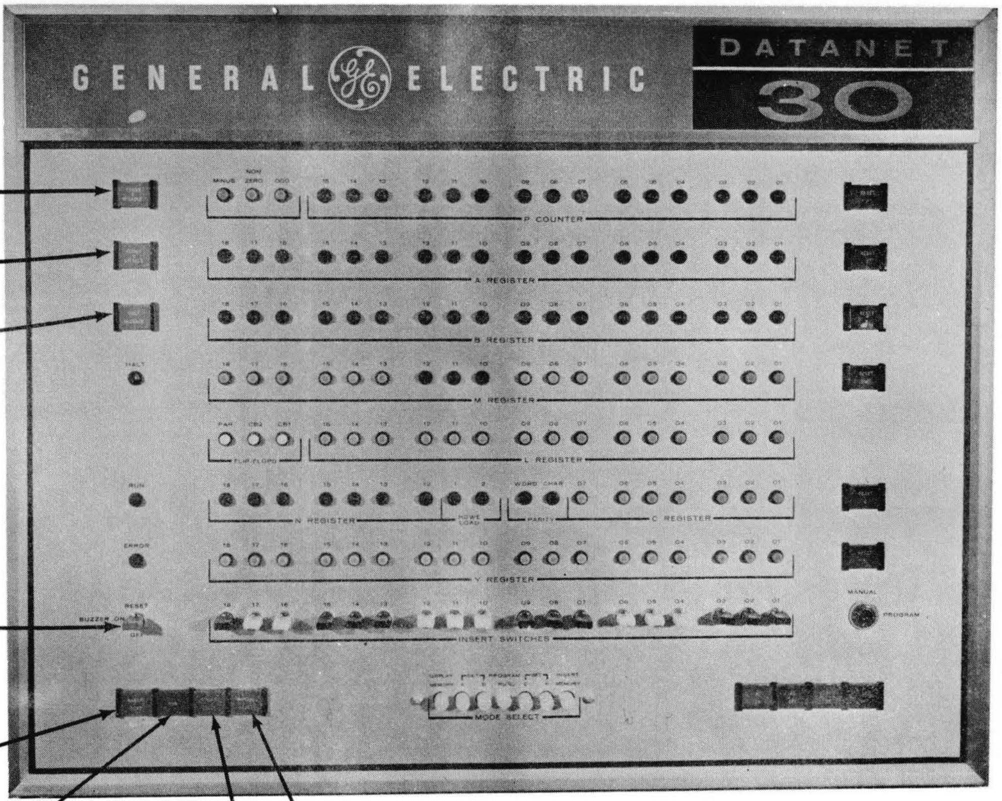
1. Turn on main circuit breaker located behind the front panel of rack 3.
2. Press AC ON button
3. Wait 10 seconds, then press DC ON button.
4. Press MANUAL RESET.



RESET P	Pressed	Resets P.
RESET A	Pressed	Resets A.
RESET B	Pressed	Resets B.
RESET M	Pressed	Resets M.
RESET C	Pressed	Resets C.
RESET Y	Pressed	Resets Y.
MANUAL/ PROGRAM	Program	Run Flip-Flop is locked on. COUNT P, COUNT Q and HALT switches are bypassed such that P COUNTS, Q COUNTS and HALT conditions are ignored. All other switches are inoperative except the INSERT and POWER switches.
	Manual	All switches are operative.
INSERT SWITCHES	Up	The switch position equals 1.
	Center	The switch position equals 0.
	Down	The switch position equals 1.

MODE SELECT	Set A	The Insert Switches can set A.
	Set B	The Insert Switches can set B.
	Set C	The Insert Switches can set C.
	Set P	The Insert Switches can set P.
	Program Run	Instructions are executed in the normal manner. This mode must be selected for the program to be executed.
	Display Memory	The contents of the location specified by P can be displayed. The contents will be compared to the Insert Switches. A discrepancy will result in a halt condition.
	Insert Memory	The data in the Insert Switches can be stored in the memory location specified by P. After storage, the location's contents will be read out and compared to the switches. A discrepancy will result in a halt condition and will turn on the buzzer.

Figure 13-A. Control Console Switches



COUNT P	Off	P counts normally.
	On	P does not count.
COUNT Q	Off	Q counts normally.
	On	Q does not count.
HALT DISABLE	Off	A halt condition will halt the DATANET-30 if the key switch is in manual position.
	On	Halt conditions are ignored.
BUZZER	Up	Resets the Buzzer.
	Center	Will buzz.
	Down	Prevents the buzzer from turning on.
MANUAL LOAD	Depressed	Initiates the Hardware Load process, if the Program Run button has been pressed.
RUN	Depressed	Starts the DATANET-30 program running continuously, if in the Program Run, Display Memory, or Insert Memory mode.
MANUAL RESET	Depressed	Resets all Registers, Counters and Flip-Flops.
SINGLE CYCLE	Depressed	When running in Program Run mode, halts the DATANET-30. When already halted, one instruction or action will be executed each time the switch is depressed. The action depends on the position of the mode selected.

Figure 13-B. Control Console Switches

V. PROGRAMMING CONSIDERATIONS

PROGRAMMING THE BUFFERS

Service Rate

When servicing transmission lines on a bit basis there are certain timing factors which must be taken into account. The following table shows the service rate for six standard teletype transmission speeds:

Bits per Second	Service Rate (milliseconds)
45	22.2
50	20.0
56.25	17.7
75	13.3
110	9.09
150	6.67

In each case, the service rate can be defined as the operation of the receive or transmit flag of the bit buffer.

When scanning the bit buffers, the service rate is taken into account and the Program Interrupt Executive initiates scanning at a rate slightly faster than the service rate. For a 45-bit/second transmission line having a service time of 22.2 milliseconds, the line would be scanned approximately every 21.0 milliseconds to ensure that any speed variations in the remote terminal would not result in data lost at the DATANET-30.

Basic Program Cycle

A real time program response time to certain events must be very small. The communications programs must be divided into the following events:

1. Receive bits
2. Assemble bits into characters

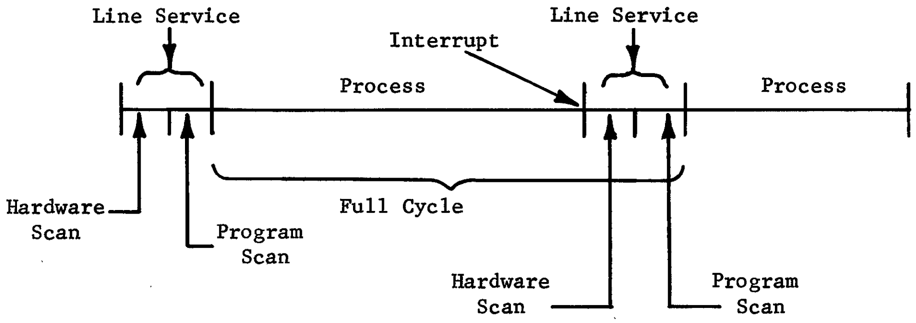
3. Assemble characters into words
4. Assemble words into blocks
5. Assemble blocks into messages
6. Assign message routing
7. Disassemble blocks into words for transmission
8. Disassemble words into characters
9. Put the character in the buffer for transmission.

The program to do this is divided into two basic cycles.

1. Line service cycle (hardware scan and program scan) -- when each buffer is sampled within a bit or character time and the bit or character present is moved to or from the buffer.

2. Processing cycle -- when all the rest of the processing to be done by the program must be accomplished. The bit buffer assembly areas and the other buffers are serviced on a character time bases.

Since a basic premise of the DATANET-30 is to receive (or transmit) each bit or character within rigid time limitations, the line service cycle must be initiated within a certain amount of time.

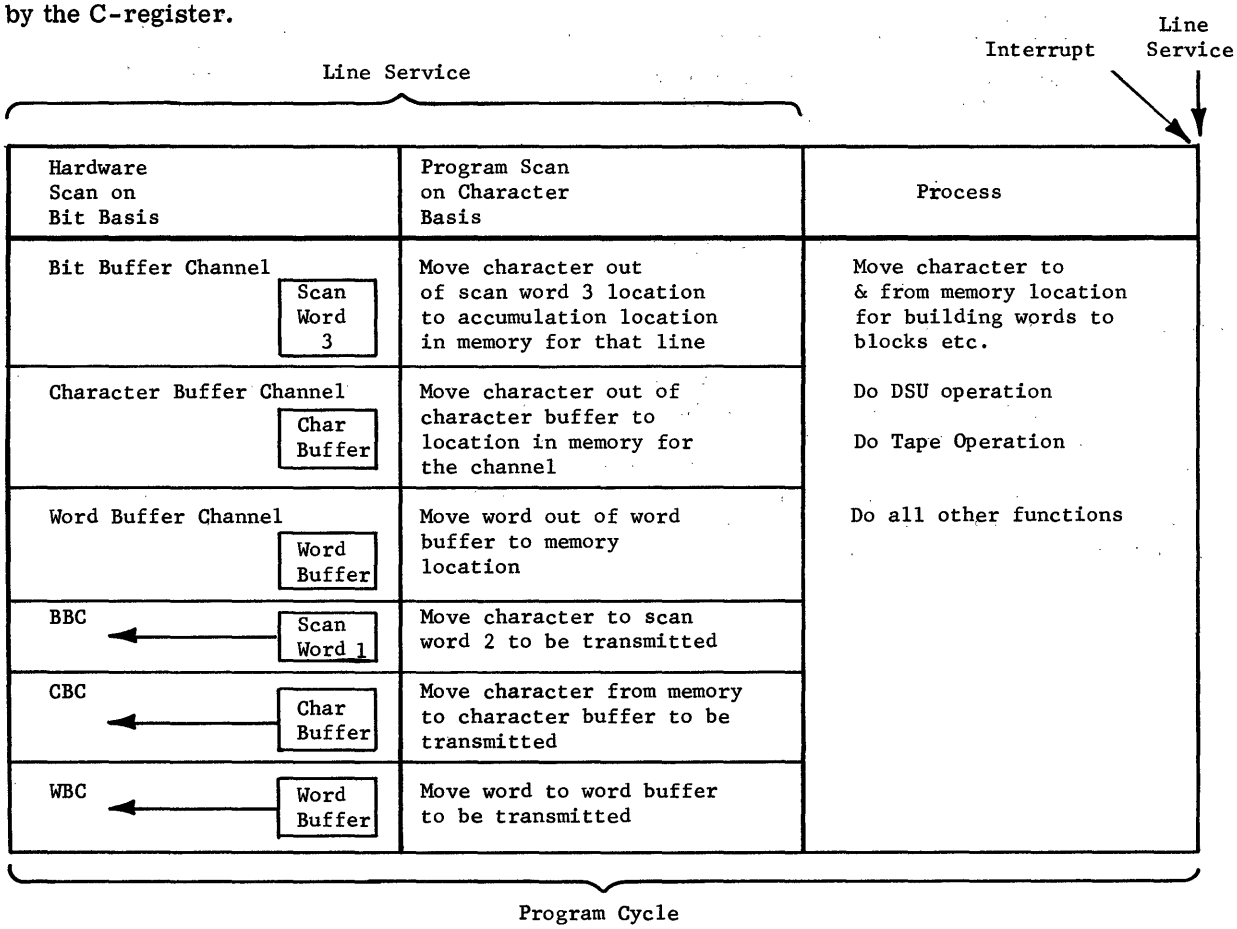


The time will vary with the line service rate required by the remote terminals. One full cycle must therefore be completed at a rate slightly faster than the fastest service rate. In order to do this, processing must be interrupted to allow the hardware scan instruction to service the lines (3 word times per line). The interruption must be timed so that, from the end of one scan cycle to the end of the next scan cycle, the total elapsed time is less than one bit time. Consideration must also be given to memory cycles used during the scan by the controller selector peripherals.

Although the above only discussed the bit time for the bit buffers, the scanning and processing of character and word buffers follow the same rules. The scanning of character and word buffers however is done by programming for each buffer.

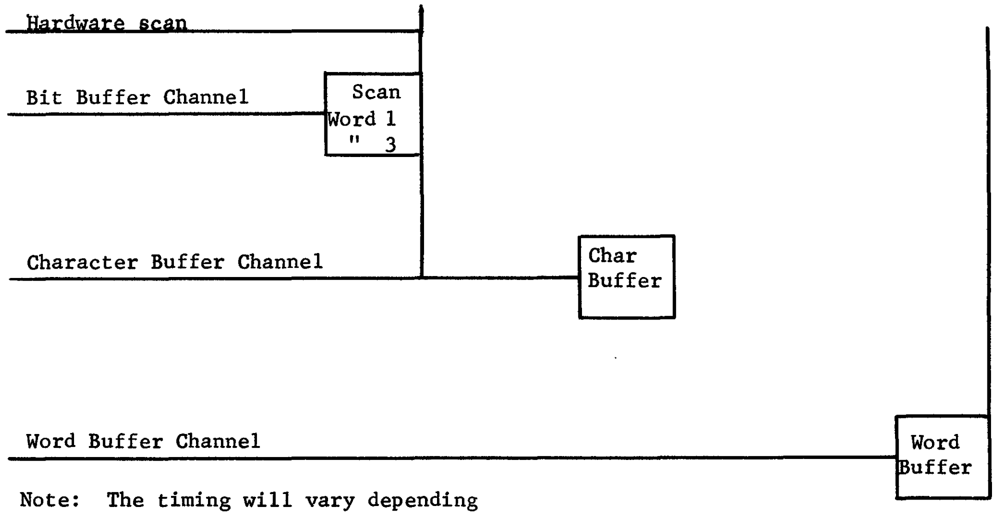
The control of data transfer going to or from a buffer is accomplished by the register transfer instructions, the C-register and the transmit/receive data lines. The receive buffer address in the C-register allows the character or word in the receive buffer to be set up on the receive data lines. The register transfer instruction -- that is, TRA R, B -- then transfers the configuration of the receive data lines to the designated working register.

The transmit sequence using the transmit data lines is basically the opposite of the sequence using the receive data lines. The address of the transmit buffer is first set up in the C-register. Then the transfer of the configuration in one of the FROM registers, again using a register transfer instruction, is transferred to the transmit data lines. The only transmit buffer that will be able to accept the configuration on the transmit data lines will be the one addressed by the C-register.



Hardware Scan on Bit Basis	Program Scan on Character Basis	Process
Bit Buffer Channel Scan Word 3	Move character out of scan word 3 location to accumulation location in memory for that line	Move character to & from memory location for building words to blocks etc.
Character Buffer Channel Char Buffer	Move character out of character buffer to location in memory for the channel	Do DSU operation Do Tape Operation
Word Buffer Channel Word Buffer	Move word out of word buffer to memory location	Do all other functions
BBC ← Scan Word 1	Move character to scan word 2 to be transmitted	
CBC ← Char Buffer	Move character from memory to character buffer to be transmitted	
WBC ← Word Buffer	Move word to word buffer to be transmitted	

Figure 14. General Timing Diagram



Note: The timing will vary depending upon the speed of transmission for the various buffer.

Figure 15. Relative Timing for Scanning Buffers

Functional Sequence

The normal flow of data occurs as shown below. The program periodically halts to allow the SCN instruction to take bits from the bit buffers to form characters in memory. When a character is formed, it is transferred over to another area of memory where the program accumulates characters into words. The words are accumulated into blocks of variable lengths and then transferred to the disc storage unit, where the queue, journal, intercept, and in-transit storage areas are established under program control. The same basic process occurs for the character and word buffers. However, all other buffers must be scanned by the program.

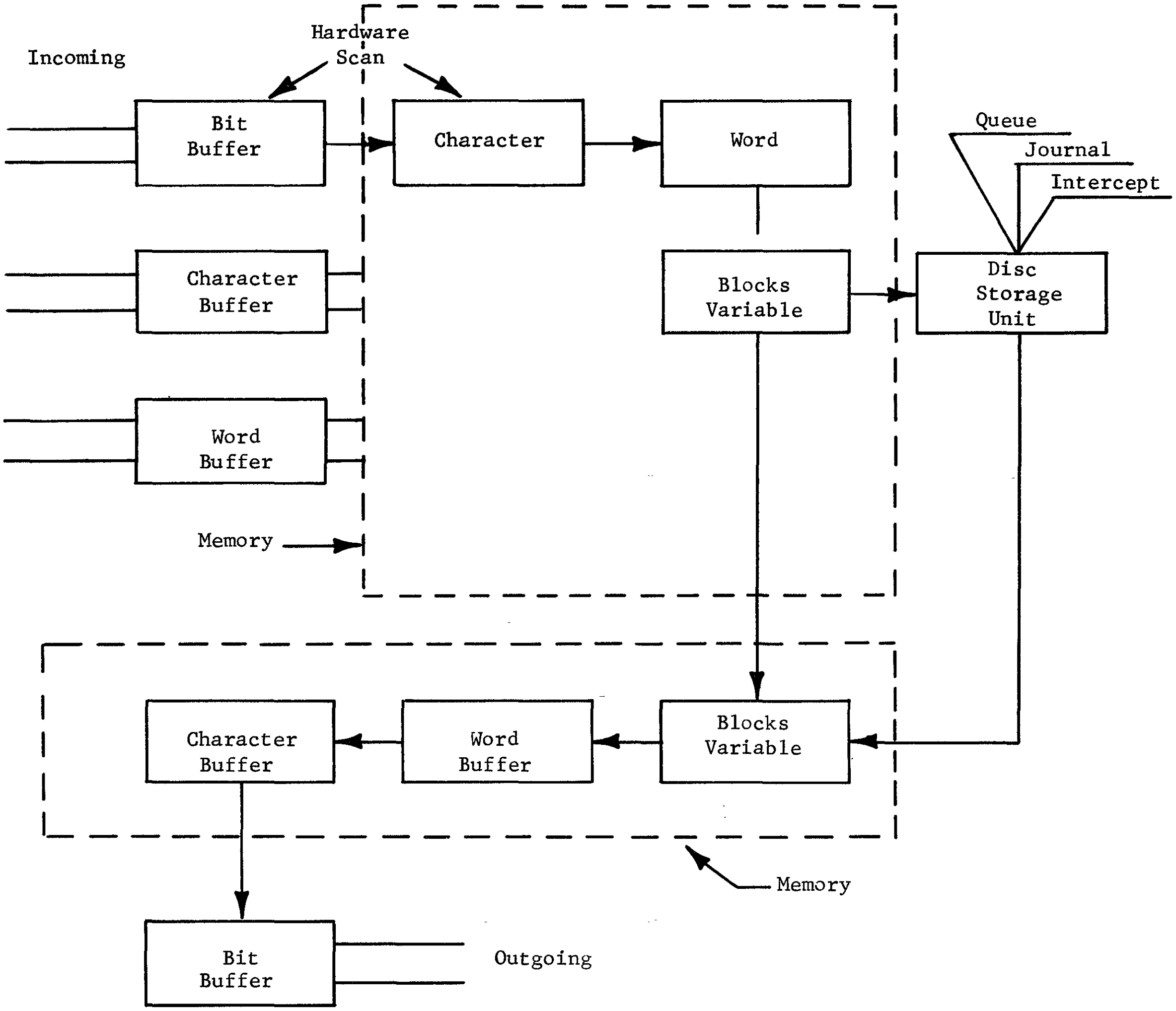


Figure 16. Data Flow Functional Block Diagram

PROGRAMMING CONVENTIONS

In writing programs for the DATANET-30, there are a few conventions which should be considered. The suggestions made here are not hard and fast rules, but must be considered for maximum programming efficiency:

1. Do not use locations 0 and 1 in memory; these locations are used by program interrupt. When the Q-counter counts down to zero, P+1 is stored in location 0 and control is transferred to the location specified by location 1.

2. Do not use cells 3, 4, and 5. These locations are used by the controller selector unit for storage of command words.

3. If possible, all subroutine linkages and constants should be located in the common data bank (cells 8 - 511 in memory).

4. Channel tables must be located in the first 8192 words of memory.

5. Utility routines should be stored at the top of the memory, so that they will not be destroyed when reading in later programs.

6. The following checks should be made:

   a. Before issuing any SEL instruction, check the ready status of the controller with the CSR instruction.

   b. Before issuing any CSR instruction, check for the completion of the previous SEL sequence with an NIS 7 instruction.

   c. Before changing memory locations 3, 4, and 5, check for completion of the previous SEL sequence with an NIS 7 instruction.

7. When closing a file on magnetic tape always write an end of file on the tape.

8. When branching to a subroutine, the symbolic name of the subroutine link will be followed by 1:

```
          BRS      REPRT            Go to report subroutine


REPRT     IND 0                     Subroutine linkage
          IND      REPRT1           REPRT 1 is the actual starting
                                    address of the subroutine.
```

9. The last character to be transmitted at the end of transmitting a message must be an all marks character (all 1's).

10. At the end of each program bank, careful consideration should be given to the instructions in the last 2 positions and to those instructions that fell into the succeeding program bank.

11. The following memory allocation has been established as a standard programming convention:

Decimal Location	Contents
0000 - 0007	Program interrupt and controller selector command words
0008 - 0031	Parameters for utility routines and general use
0032 - 0511	Program constants, subroutine linkage
512 - 1023	Scan words (channel tables) and constants
1024 - 7499	Object programs
7500 - 7999	Utility programs and programming tools
8000 - 8191	Loader programs

12. The C-register instructions (PIC. AIC. XCZ. NCZ) will have decimal or symbolic operands which will be assembled as a numerical value rather than a memory address.

13. If an operand referred to by a double length instruction (LDD, STD, BRS, AMD) falls in an odd location, the operand will be stored in the next highest even location and a "no-operation" instruction will be inserted in the vacated odd location.

BUFFER OPERATIONS

Bit Buffer Channel

Data is sent to a buffer via the transmit data drivers. Data is received from a buffer via the receive data lines. Control signals are sent to a buffer via EFD, the external function drivers. Information as to the status of a buffer is tested via ESL, the external status lines.

BIT BUFFER INSTRUCTIONS

Mnemonic	Operand	Word Times
Register Transfer	R,	
TRA	FROM, TO	The bit contained in the receive buffer is transferred to position 18 of R; the receive buffer and flag are reset.
Register Transfer	, T	
TRA	FROM, TO	The low order bit of the Z drivers is transferred to the transmit data buffer. The transmit flag is reset.
SCN	I	1+3N
SCAN		Scan the bit buffer units. The bit buffers are interrogated for data received or to be transmitted. Data is moved to and from the bit buffers.
DEF	I	1

DEF 1. Reset receive flag and receive data buffer.
DEF 2. Reset transmit flag and data buffer.
DEF 3. Set receive mode turn carrier off.
DEF 4. Set transmit mode turn carrier on.
DEF 5. Reset receive clock.
DEF 6-10. Not used.

NES	I	1

NES 1. Receive flag set (data buffer contains a new bit).
NES 2. Transmit flag set (data buffer is ready for a new bit).
NES 3-4. Not used.
NES 5. Interlock on.
NES 6. Carrier on.
NES 7-10. Not used.

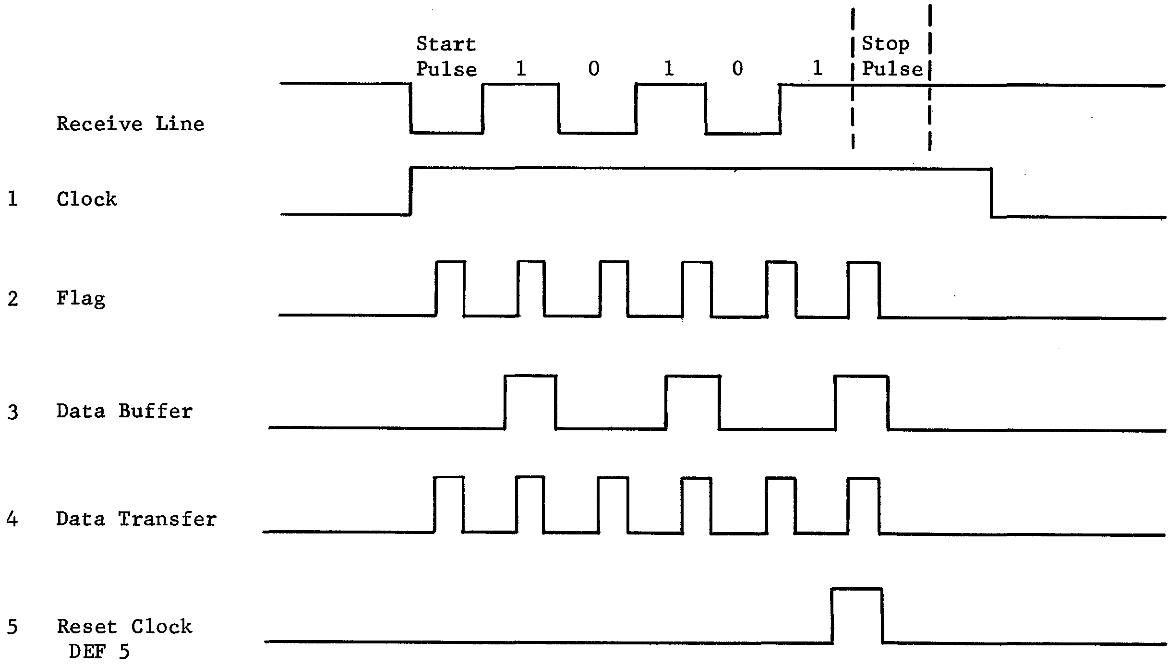
RECEIVE OPERATION

Assume that a remote terminal device is sending out a continuous stream of marks, (the line is in the idle condition). Then the operator at the remote terminal begins transmitting information. When the start bit (a space) is received, a clock is started. The clock is used to time the future sampling of the line. The start bit is transferred into the receive data buffer by the bit buffer channel (BBC), and the receive flag is set. When the clock reaches the proper time, the line is sampled again, the bit on the line is transferred to the receive data buffer, and the receive flag is set. This process of sampling the line at regular intervals, transferring the data on the line to the receive data buffer, and setting the receive flag continues until the clock of the BBC is stopped by the program. Since the BBC will transfer the information from the line into the receive data buffer every bit time, the program must test the receive flag and take away the bit in the receive data buffer before the line is sampled again by the BBC.

Whenever the bit is taken, the receive flag and the receive data buffer are automatically reset. At some point, the program decides that the appropriate number of bits have been received and sends a signal to the BBC which stops the clock. The receive flag will remain reset until another start bit is received. As a protection against noise on the transmission line causing the clock to start running, the BBC circuitry requires the space condition to exist on the line for at least one-half of a bit-time to start the clock. Thus, noise of less duration than one-half of a bit-time will have no effect.

A BBC can be used with a half-duplex line by ignoring the receive section when sending and by ignoring the transmit section when receiving. If a subset is used, control of the carrier is accomplished by activating the appropriate external function driver (with a DEF instruction).

The following timing diagram shows how the character Y would be received by a bit buffer as a 5-level teletype character.



1. When a start pulse is received the clock in the receive unit is started and the line is sampled in the center of each bit period of the character.

2. The receive flag is set when the line is sampled and the bit is sent to the receive data buffer.

3. The data buffer temporarily stores the bit which has just come in from the line.

4. The program tests to see if the flag is set. If it is, the program will transfer the bit to a register. Transferring the bit will automatically reset the receive flag and data buffer by issuing a DEF1 instruction.

5. After the complete character is received the program initiates a DEF5 instruction which resets the clock. The clock will not be set again until another start bit is received.

TRANSMIT OPERATION

Assume that the program is not transmitting and that the transmit flag is set. This means that the BBC is ready to take a new bit from the program. The program sends a bit to the transmit data buffer. This automatically resets the transmit flag. At regular intervals, the BBC transfers the bit in the transmit data buffer to the transmission line. When this happens, the transmit data buffer shifts a bit onto the line, whether or not a new bit has been supplied. The program must test the transmit flag and provide a new bit before this transfer occurs. This process will repeat for each bit in the bit stream. At the end of the bit stream, the last bit will remain in the transmit data buffer and will be transferred to the line regularly. Therefore, the last bit in a bit stream will be a 1, so that the line remains in the mark condition when no information is being transmitted. Note that with a BBC the length of the bit stream is completely under program control.

The next diagram illustrates how the character R would be transmitted to a communications line. The character R would be represented in memory as 11101010, where the right-hand 0 is the start bit and the two left-hand 1's are the stop bits. The 5 bits in between the start bit and stop bits represent the 5-level teletype code for the letter R.

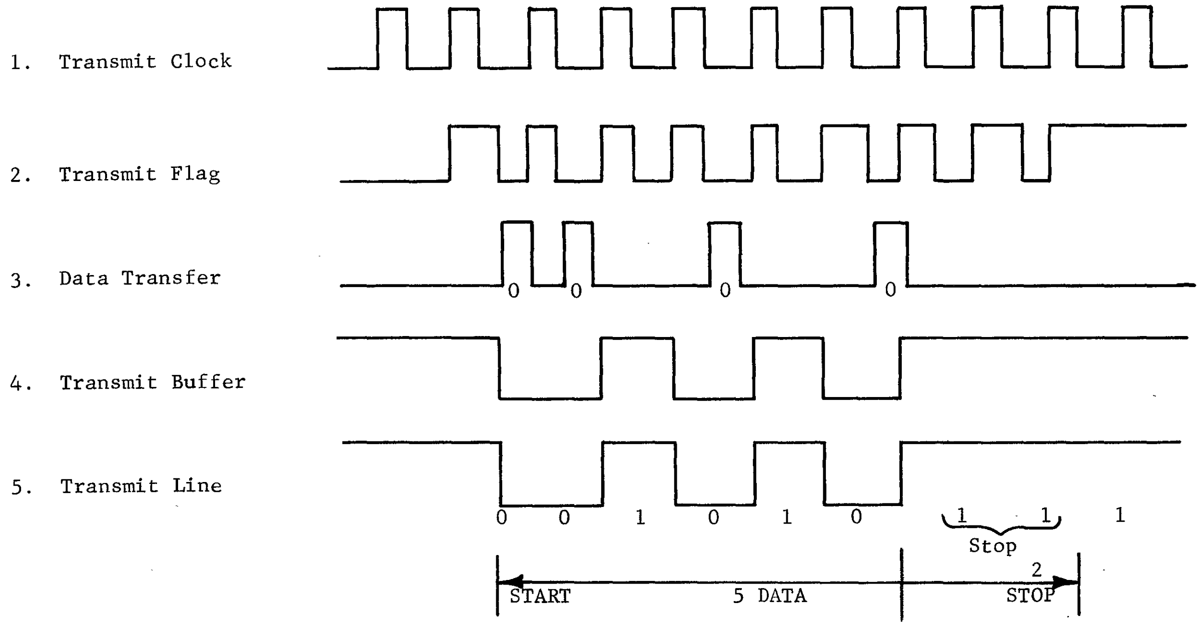
1. The transmit clock occurs every bit period as specified by the data timing unit.

2. The transmit flag is set each time the transmit clock occurs and is reset when the data is transferred to the transmit buffer.

3. When the program finds the transmit flag set, it transfers the next data bit to the BBC, which automatically resets the transmit flag.

4. This shows how the transmit buffer would look over a period of one character time.

5. This shows the signal as it appears on the line.



HARDWARE SCAN

The SCN instruction is for use with the bit channels only. It will not operate properly with any other buffer unit. Therefore, only bit buffers should be among the channels from C_i to C_f. This means that all bit buffer channels should be addressed sequentially.

Bit buffer channel addresses can not be intermixed with character buffer channels or word buffer channels.

The initial channel to be scanned is specified in the instruction. The final channel to be scanned is specified by the scan words or channel 127, whichever occurs first. Channels are scanned sequentially as follows:

$$C_i, \ C_{i+1}, \ C_{i+2}, \ \ldots, \ C_{f-2}, \ C_{f-1}, \ C_f,$$

where

C_i is the initial channel,
C_f is the final channel, and
N = number of channels scanned
= f-i+1.

The time required for SCN is one word time for setup plus three word times for each channel scanned, or:

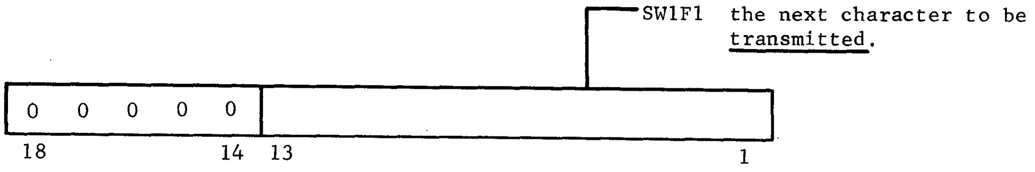
Word Times = 1+3N.

This time is required whether data is transferred or not. Also, this time is required for a simplex, half-duplex, or full-duplex channel.

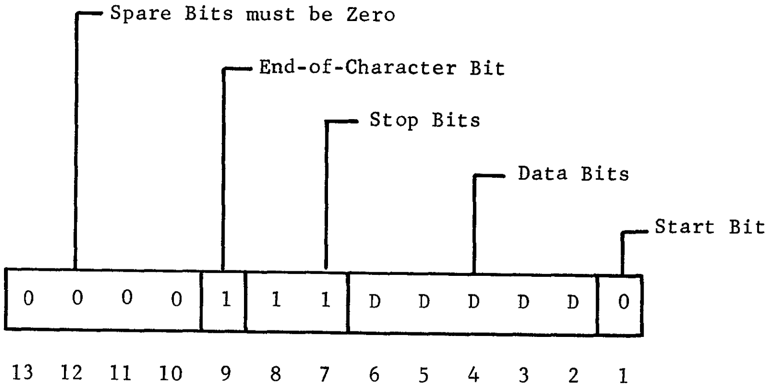
The SCN instruction uses the A and B registers, and the previous contents will be destroyed. Also the C-register will contain C_f after it is completed. At the end of a Transmission, the last word placed in scan word one continues to be transmitted. It is necessary to put a word of all marks in scan word one for idle line condition.

DEF1, DEF2, DEF5, NES1, NES2, and all data transfer is handled automatically by the SCN instruction. The program must, however, give the DEF3 and DEF4 instructions appropriately.

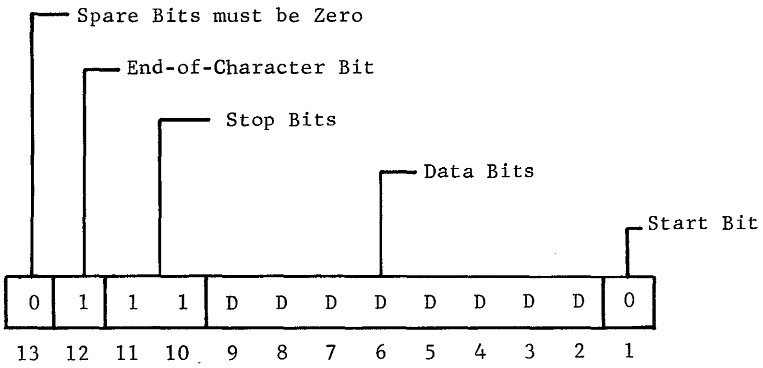
Scan Word 1



It is possible to transmit 5-, 6-, 7-, and 8-level codes of 8, 9, 10, and 11 bits. The format for 5-level, 8-unit codes is:

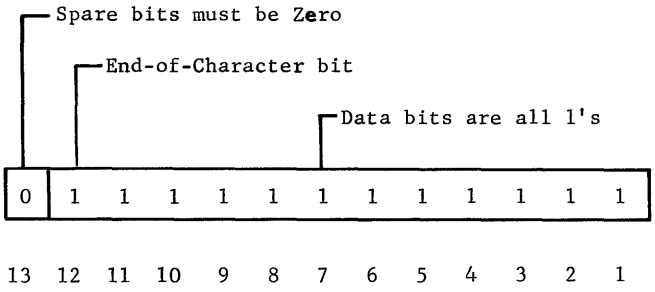


The format for 8-level, 11-unit codes is:



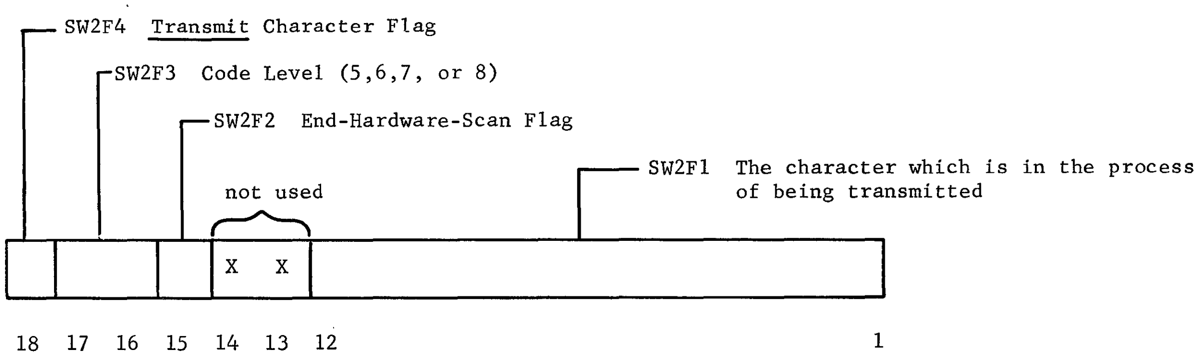
The format for 6- and 7-level codes is similar.

It is sometimes necessary to transmit one or more fill characters. A delay time of one character is a marking condition on the line for one character time. This can be achieved by making the start bits, data bits, and stop bits all 1's. This should also be the last character transmitted at the end of transmitting a message. A one-character delay for 8-level, 11-unit codes is as follows:



The end-of-character bit is defined as the last 1-bit in the field. This must be present. If not, the last 1-bit of data will be interpreted as the end-of-character bit.

Scan Word 2


SW2F4 Transmit Character Flag
SW2F3 Code Level (5,6,7, or 8)
SW2F2 End-Hardware-Scan Flag
SW2F1 The character which is in the process of being transmitted
not used
X X
18 17 16 15 14 13 12
1

SW2F1 is controlled entirely by hardware and requires no detail program control. The bits are shifted right to the bit buffer channel and then to the line until the end-of-character bit is in position 1. This occurs when SW2F1 is (000000000001).

SW2F2 is set to indicate the final bit buffer channel number when the program is initially assembled and thereafter need not be considered. It is necessary to change SW2F2 for the final channel for any change in the number of bit buffer channels:

1 = this is the last bit buffer to be scanned.
0 = continue scanning.

If the final channel is not indicated, the SCN instruction will automatically end at channel 63.

SW2F3 defines for receive purposes the code level of the line (5, 6, 7, or 8) as follows:

SW2F3 (Bits 17 and 16)
17|16
00 = 5-level code
01 = 6-level code
10 = 7-level code
11 = 8-level code.

This is set when the program is initially assembled (or changed octally) and thereafter need not be considered.

SW2F4 is set by the hardware when the new transmit character is transferred from SW1F1 to SW2F1. It is reset by the program after the new character is loaded into SW1F1.

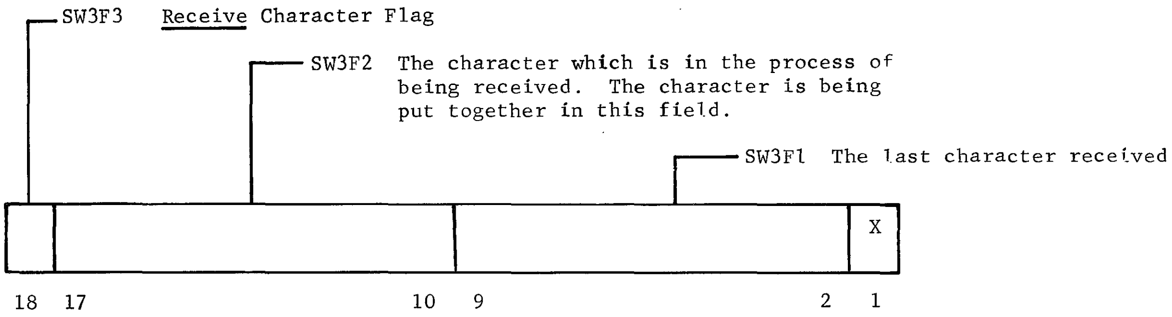
SW2F4

1 = SW1F1 is ready for a new character
0 = SW1F1 is not ready for a new character.

Scan Word 3

SW3F1 is set by the hardware when SW3F2 receives a full character as defined by SW2F3. The data bits will be in the following positions:

5-level code in positions 2-6. Positions 7-9 are 0.
6-level code in positions 2-7. Positions 8-9 are 0.
7-level code in positions 2-8. Position 9 is 0.
8-level code in positions 2-9.


SW3F3 Receive Character Flag
SW3F2 The character which is in the process of being received. The character is being put together in this field.
SW3F1 The last character received
X
18
17
10
9
2
1

SW3F2 is controlled entirely by hardware and requires no program control.

SW3F3 is set by the hardware when the new received character is transferred from SW3F2 to SW3F1. It is reset by the program after the new character is removed from SW3F1. SW3F1 does not have to be changed by the program.

SW3F3

1 = SW3F1 has a new character
0 = SW3F1 does not have a new character.

Scan Word Locations in Memory

The three scan words per line are located in memory as follows.

		Location	
		Decimal	Octal
Scan Word 1	Channel 0	512	1000
	Channel 1	513	1001
	.	.	
	.	.	
	.	.	
	.	.	
	Channel 127	639	1177
Scan Word 2	Channel 0	640	1200
	Channel 1	641	1201
	.	.	
	.	.	
	.	.	
	.	.	
	Channel 127	767	1377
Scan Word 3	Channel 0	768	1400
	Channel 1	769	1401
	.	.	
	.	.	
	.	.	
	.	.	
	Channel 127	895	1577

Any of the 384 locations not used for scanning BBC's, may be used for any other purpose. For example, channel 0 is used for the paper tape reader and the scan instruction does not apply to paper tape. Scan words 1, 2 and 3 for the paper tape reader are wired in hardware.

Receive and Transmit

The Scan instruction accomplishes the following at a rate necessary to check each bit buffer once each bit time.

Receive

When a start bit appears in the bit buffer, the receive flag is set. The SCN instruction transfers the bit to the character-being-received half of scan word 3, and resets the receive flag. When the next bit of the character appears in the bit buffer, the receive flag is set, the SCN instruction shifts the previous bit over 1 position and transfers in the new bit of the character. Prior to each shift and transfer of a bit, the SCN instruction checks for whether or not the bit in the bit buffer is the last bit for the character. When the last bit is in the bit buffer, the character is shifted to the last-character-received side and the last bit is shifted in also. The character must then be shifted out by the program before another character is fully received. New characters are shifted into the last-character-received side whether the preceding one was shifted out or not.

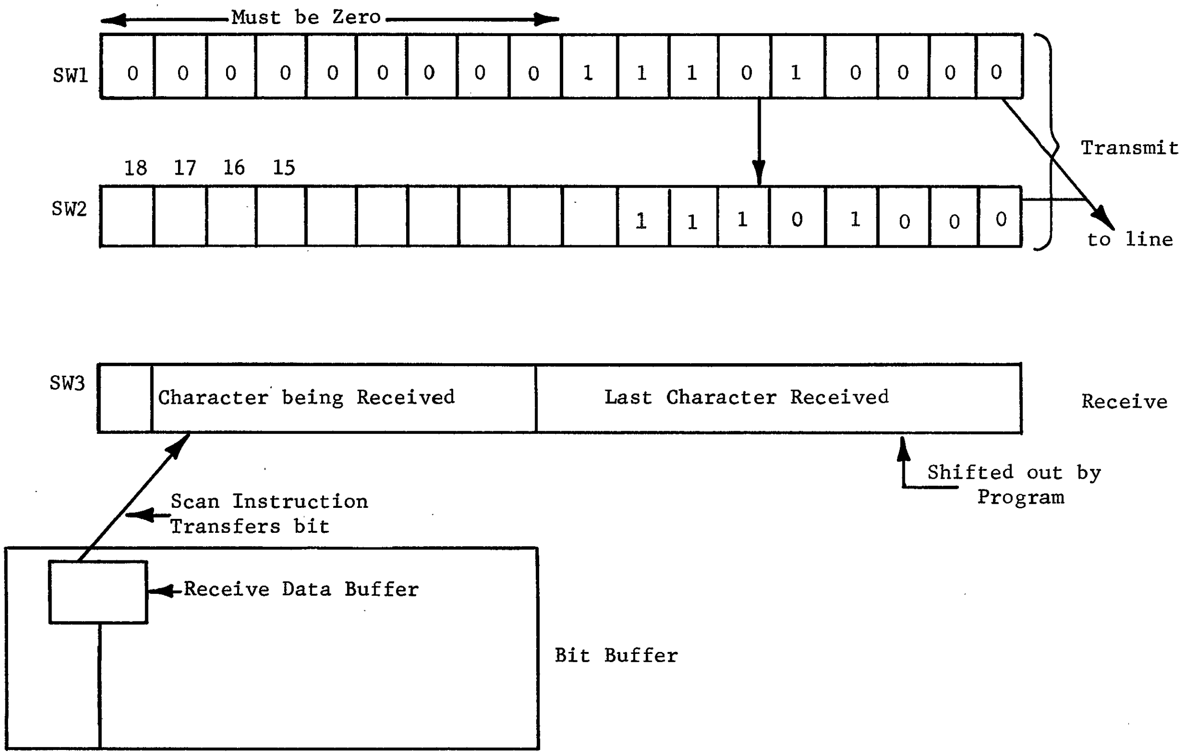


Figure 17. Hardware Scan Block Diagram

Transmit

Assuming that the transmit mode has been set, once each scan cycle a bit will be transmitted from the bit buffer. If nothing is to be transmitted, the line should be in a marking condition (idle). Scan word 2 contains the character being transmitted. Upon the completion of transmitting a character from scan word 2, the character in scan word 1 is transferred into scan word 2, and automatically transmitted. The program loads scan word 1 with the next character to be transmitted.

PROGRAM INTERRUPT

Program interrupt occurs under control of the Q-counter. When Q counts to zero, the following sequence occurs:

1. The instruction being executed is completed. This can take from 1 to 10 (ten word times is the *worst case execution time of the CSR instruction) word times, depending on the instruction.

* See "Instruction Repertoire" for detailed description.

2. If a memory interrupt is requested by the controller selector, 1 word time is taken to service the request.

3. Effectively, a BRS 0 is executed. This operation requires 2 word times plus execution of the program. Interrupt can take from 3 to 13 word times.

If Alpha is the location of the instruction being executed when the program interrupt occurred, then the BRS 0 performs the following:

1. Alpha +1 is stored in location 0.

2. The contents of location 1 is transferred to the P-register and program execution started there.

The Program Interrupt Routine must begin with:

```
* STF     WS1                   Store special flip-flops
  LDQ     Count                 Load Q with new value
  STD     WS2                   Store A and B
  STC     WS3                   Store C
```

The Program Interrupt Routine must end with:

```
* LDC     WS3                   Load C
  LDD     WS2                   Load A and B
  LDF     WS1                   Load special flip-flops
  BRU     0           X         Return to point of interrupt
```

The Program Interrupt Routine will normally include execution of the Scan instruction. Also, the worst case execution of the Program Interrupt Routine will be less than the time period between program interrupts. Thus, a program interrupt cannot occur while a Scan instruction is being executed. A program interrupt during an SCN instruction cannot be successfully done.

PROGRAMMING EXAMPLES, BIT BUFFER CHANNEL

The following example shows one method that might be used to receive one character from a bit buffer. This method does not use the SCN instruction.

```
Location  Instruction  Symbol   OPR   Operand    Remarks

            15530               ORG   7000       ORIGIN LOCATION 7000
 15530      600000     RECVE    LDB   BIT7       BIT NUMBER SEVEN
 15531      022001              NES   1          RECEIVE FLAG SET
 15532      121531              BZE   *-1        NO, GO BACK
 15533      042444              SR1   BR,B       YES, SHIFT NEW BIT TO B-REGISTER
 15534      160000              BEV   RECVE+1    COMPLETE CHARACTER NOT IN, GO BACK
 15535      026020              DEF   5          CHARACTER IN, RESET RECEIVE CLOCK
```

* See “Instruction Repertoire” for detailed description.

1. Initially bit 7 is put into the B-register. This will be used to test whether a whole character has been received.

2. The NES1 command tests to see if the receive flag is set. If the flag is not set, the BZE command branches back to test the flag again.

3. If the flag is set, the bit contained in the data buffer is shifted into position 17 of the B-register.

4. If the B-register is even, control is transferred back to get the next bit. If the B-register is odd, meaning the initial bit set in B has reached position 1, the even test fails and the program continues with the next instruction.

5. The DEF5 instruction resets the receive clock.

The next example is one method which might be used to transmit one character onto a transmission line via a bit buffer without using the SCN instruction.

```
Location  Instruction    Symbol    OPR    Operand              Remarks

            03720                  ORG    2000
            02400        $NCHAR    EQU    1280
 03720     603120                  LDB    $NCHAR      LOAD CHARACTER FROM TABLE
 03721     022002        XMIT      NES    2           TRANSMIT FLAG SET
 03722     121721                  BZE    *-1         NO, GO BACK
 03723     060401                  TRA    B,T         TRANSFER BIT TO TRANSMIT DATA DRIVERS
 03724     042404                  SR1    B,B         SHIFT B-REGISTER RIGHT ONE
 03725     131721                  BNZ    XMIT        WHOLE CHARACTER NOT OUT, GO BACK
```

1. The character to be transmitted is put into the B-register.

2. The transmit flag is tested to see if it is set.

3. When the flag sets the low order bit of B is sent to the transmit buffer.

4. Bits shifted right 1 place and tested for zero. If B is non-zero, control is transferred back to transmit next bit. When B becomes zero, the BNZ test fails and the program goes on to execute the next instruction.

The next two examples show how to receive a character and transmit a character using Hardware Scan (SCN). It should be noted these are examples and do not necessarily show the way they will be written in the operating programs.

Receive - Hardware Scan

```
Location   Instruction    Symbol     OPR      Operand        Remarks

                                     REM                     SAMPLE HARDWARE SCAN RECEIVE
                                                             PROGRAM
              05670                  ORG      3000           ORIGIN 3000
              01400       $SCW3      EQU      768            SCAN WORD STARTING ADDRESS
  05670      377777       NBIT18     OCT      377777         MASK FOR RECEIVE FLAG
  05671      030001       START      SCN      1              SCAN BIT BUFFER
  05672      603060                  LDB      $SCW3          LOAD CHARACTER BEING RECEIVED
  05673      141671                  BPL      *-2            CHARACTER NOT IN, GO BACK
  05674      401670                  LDA      NBIT18         CHARACTER IN, GET MASK CONSTANT
  05675      533060                  NAM      $SCW3          MASK OFF RECEIVE FLAG
```

Transmit - Hardware Scan

```
Location   Instruction    Symbol     OPR      Operand        Remarks

              01750                  ORG      1000           ORIGIN LOCATION 1000
              01000       $SCW1      EQU      513            SCAN WORD ONE
              01200       $SCW2      EQU      641            SCAN WORD TWO
  01750      030001                  SCN      1              SCAN BIT BUFFER
  01751      603050                  LDB      $SCW2          LOAD SCAN WORD TWO
  01752      141750                  BPL      *-2            TRANSMIT FLAG NOT SET, GO BACK
  01753      603070                  LDB      $XWORD         LOAD CHARACTER TO BE TRANSMITTED
  01754      703040                  STB      $SCW1          STORE IN SCAN WORD ONE
  01755      601767                  LDB      BIT18N         LOAD MASK
  01756      733050                  NBM      $SCW2          MASK OFF TRANSMIT FLAG



  01767      377777       BIT18N     OCT      377777         MASK CONSTANT
              01600       $XWORD     EQU      896            TABLE LOCATION, NEXT CHARACTER TO
                                                             XMIT.
```

Next, is a simplified example of a Program Interrupt Executive Routine containing a Scan instruction. At Symbol PIE1 is found the Store Flip-Flops instruction. This saves all the branch and control flip-flops from the last instruction executed. Next, all the registers are stored and the SCN (Scan) instruction is issued. Upon leaving the Scan instruction, the registers and flip-flops are restored and control is transferred back to the program which was interrupted.

If control of mode conditions within the bit buffers is required, it should be noted that the individual channels must be set to their appropriate mode before entering the Scan Operation (Receive or Transmit Mode).

```
Location  Instruction  Symbol  OPR   Operand  X   Remarks

                                REM                 SAMPLE PROGRAM INTERRUPT EXECUTIVE
            00000               ORG   0000          ORIGIN OF SUBROUTINE LINK
 00000     000000      PIE      IND   0             LOCATION ZERO
 00001     017500               IND   PIE1          LOCATION ONE
            17500               ORG   8000          ORIGIN OF PIE SUBROUTINE
 17500     361514      PIE1     STF   PIEF          STORE FLIP-FLOPS
 17501     231515               LDQ   PIEQ          LOAD Q-COUNTER
 17502     301511               STC   PIEC          STORE C-COUNTER
 17503     311512               STD   PIED          STORE A- AND B-REGISTERS
 17504     030001               SCN   1             SCAN BIT BUFFERS
 17505     211512               LDD   PIED          LOAD A- AND B-REGISTERS
 17506     201511               LDC   PIEC          LOAD C-COUNTER
 17507     261514               LDF   PIEF          LOAD FLIP-FLOPS
 17510     106000               BRU   PIE      X    BRANCH BACK TO EXIT POINT
 17511     000000      PIEC     DEC   0             TEMPORARY STORAGE FOR C-COUNTER
 17512     000000      PIED     DEC   0             STORAGE FOR A-REGISTER
 17513     000000               DEC   0             STORAGE FOR B-REGISTER
 17514     000000      PIEF     DEC   0             FLIP-FLOP STORAGE
 17515     003554      PIEQ     DEC   1900          Q-COUNTER STORAGE (CONSTANT)
```

Character Buffer Channel (CBC)

The character buffer channel provides the interface to a half-duplex transmission line. The standard bit stream lengths are 5, 6, 7, and 8 bits. The character buffers should be used on lines operating at 300 bits per second or greater.

CHARACTER BUFFER INSTRUCTIONS

Mnemonic	Operand	Word Times
Register Transfer	,T	(TRA from ______ to T) The least significant 5, 6, 7, or 8 bits of the Z drivers are sent to the transmit data buffer and the transmit flag is reset.
DEF	I	1

DEF 1	Reset receive flag and data register.
DEF 2	Reset transmit flag and data register.
DEF 3	Set receive mode (turn carrier off).
DEF 4	Set transmit mode (turn carrier on).
DEF 5-8	Not used.
DEF 9	Answer incoming call.
DEF 0	Disconnect call.

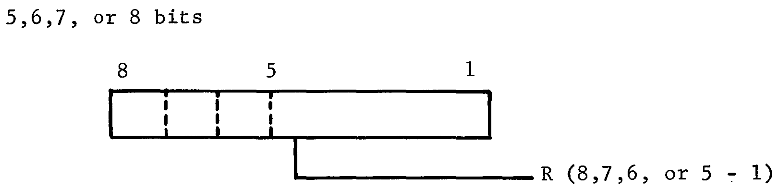
Mnemonic	Operand	Word Times
NES	I	1

NES 1	Receive flag set (data register contains a new character).
NES 2	Transmit flag set (data register is ready for a new character).
NES 3	Call in progress.
NES 4	Request answer.
NES 5	Data mode.
NES 6	Carrier on.
NES 7	Clear to send.
NES 8-10	Not used.

LDT - Do not use.
SCN - Do not use.

Register Transfer R, (TRA from R to ____)

The 5, 6, 7 or 8 bits as specified by the size of the character buffer are transferred from R. The receive data buffer and flag are reset (DEF1).



RECEIVE OPERATION

Assume that the character buffer channel (CBC) has been put in the receive mode by the program, that the receive flag is reset, and that the sending unit is transmitting a continuous stream of marks. (The line is in the idle condition.) The sending unit starts transmitting a character. The character is preceded by a start bit (a space) and followed by a stop bit (a mark). When the start bit is received, a clock is started. The clock is used to time the future sampling of the line. The start bit is shifted into the shift register. At regular intervals, the line is sampled and the bit which is present at sampling time is shifted into the shift register. When the shift register is full, the character bits are automatically transferred into the data register, the receive flag is set, and the clock is stopped. The clock will start again and the above process will repeat when the next start bit is received on the transmission line. As a protection against noise on the transmission line causing the clock to start running, the character buffer circuitry requires that the space condition exist on the line for at least one-half of a bit time to start the clock. Thus, noise of less duration than one-half of a bit time will have no effect. Since the character buffer will transfer a word into the data register whether or not the data register and receive flag are reset, the program must test the receive flag and take the character before another is transferred into the data register. When the program takes the character from the data register, the data register and the receive flag are automatically reset.

The timing diagram (Figure 18) illustrates how an 8-bit word would be received at a CBC.

1. The DEF 3 instruction puts the CBC into the receive mode.
2. The DEF 1 instruction resets the receive flag and data buffer.
3. The receive clock is shown sampling the line every bit period.
4. Line 4 shows that the contents of the receive buffer are transferred to the data register after all the bits are received.
5. Line 5 shows the receive communications line going into the CBC.
6. Line 6 shows what the receive buffer would look like after all bits are received.
7. Line 8 shows the receive flag setting when the receive buffer is transferred to the data register.

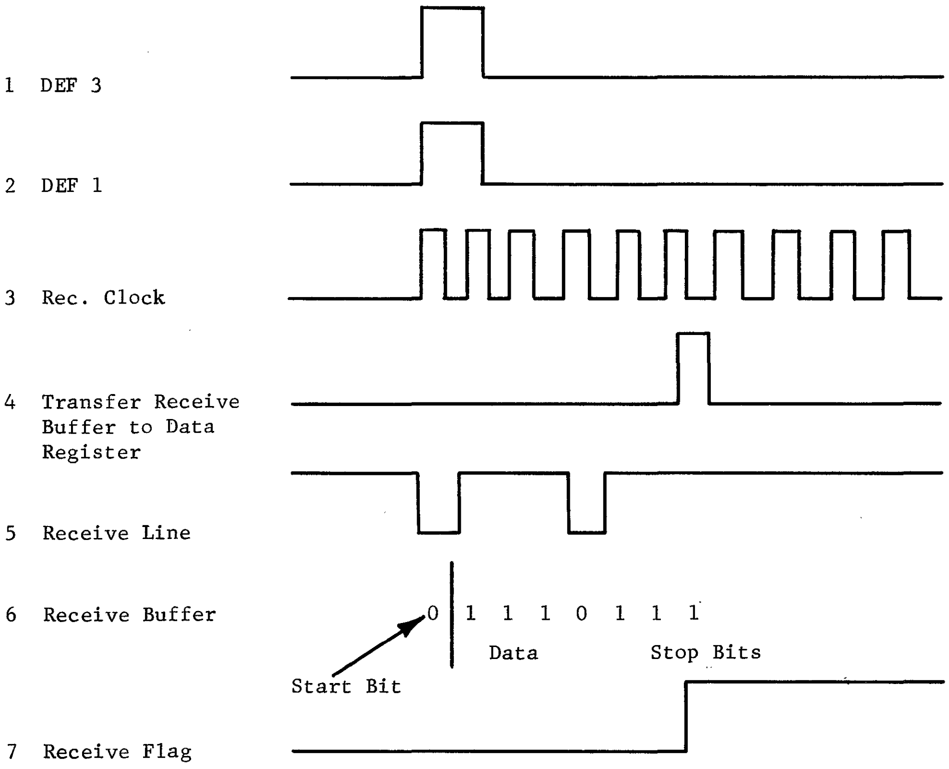


Figure 18. CBC Receive Timing Diagram

TRANSMIT OPERATION

Assume that the program has put the CBC in the transmit mode, the CBC is in theprocess of sending a word out on the line, and a word is waiting in the data register. When the current word has been shifted into the line, the CBC will transfer the word in the data register to the shift register. At this time, the transmit flag will automatically be set. The 5 bits transferred into the shift register will automatically be preceded by a start bit and followed by 2 stop bits when transmitted onto the line for a total of 8 bits. When the shift register is again empty, the CBC will transfer the word in the data register to the shift register and repeat the process if the transmit flag is reset. However, if the transmit flag is still set, indicating that the program has not put a new word into the data register, the CBC will continue to put stop bits (marks) on the line until the transmit flag is reset. When the program transfers a new word into the data register, the transmit flag will be automatically reset and the above process will be repeated. For maximum line utilization, the program must test the transmit flag and supply a new word before the current word has been completely shifted onto the line.

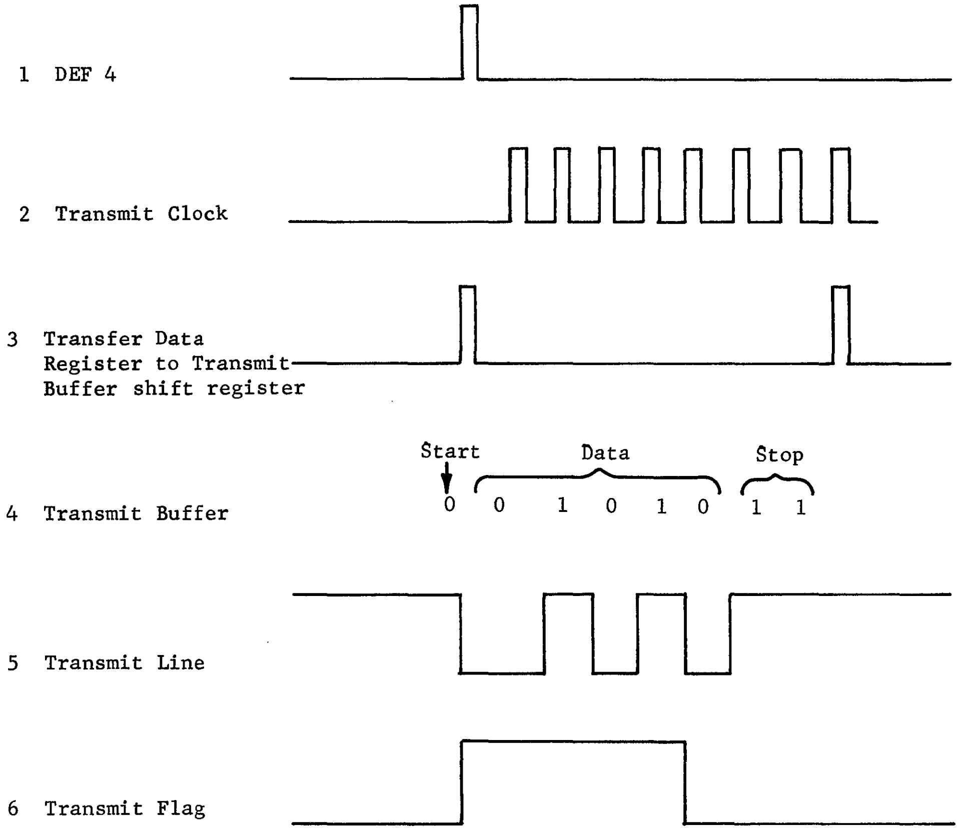


Figure 19. CBC Transmit Timing Diagram

The timing diagram (Figure 19) illustrates graphically what happens when a 5-bit character is transmitted onto a communications line by a character buffer channel.

1. The DEF 4 instruction sets the character buffer to the transmit mode.
2. The transmit clock sends data onto the line at regular intervals.
3. When the transmit buffer shift register becomes empty the data contained in the data register is transferred to the shift register.
4. This is the binary representation of the character in the shift register.
5. Line 5 shows the output of the transmit section of the character buffer.
6. The transmit flag is shown setting when the word is transferred from the data register to the shift register.

The example below shows one method that might be used to receive characters from a character buffer.

```
Symbol     OPR      Operand  X            Remarks

           ORG      7000
           DEF      31          SET RECEIVE MODE, RESET FLAG AND
                                BUFFER
LOOK       NES      1           RECEIVE FLAG SET?
           BZE      *-1         NO, GO BACK
           TRA      R,B         YES, TRANSFER CHARACTER TO B
           STB      INPUT    X  STORE IN MEMORY
           ADO      INPUT       ADD ONE TO INPUT ADDRESS
           XBZ      EOM         IS THIS THE END OF MESSAGE?
           BNZ      LOOK        NO, GO GET ANOTHER CHARACTER

INPUT      IND      1000        INPUT ADDRESS
EOM        OCT      000077      END-OF-MESSAGE CHARACTER
```

1. The DEF 3 1 instruction puts the character buffer into the receive mode and resets the receive flag and data buffer.
2. The NES 1 command tests the receive flag for a set condition.
3. When the flag sets, the BZE test fails and the character is transferred to the B-register.
4. The character is stored in memory and tested to see if it is an end-of-message character.
5. If the character isn't an EOM, control is transferred back to get next character.

Word Buffer Channel (WBC)

The word buffer channel (WBC) provides the interface to a half-duplex transmission line, on a word basis. A WBC buffers a bit stream 20 bits in length, where the length is determined by the wiring in the 20-bit code level connector.

The 20-bit buffer is intended for interconnecting DATANET-30's. Usually system considerations indicate that a WBC should be used on lines operating at more than 300 bits per second. The following rates are selectable with standard speed connectors: 600, 1200, 1800, 2000, 2400, and 3000 bits per second. Two WBC's can be mounted in a buffer module and the speeds of operation may be independently selected. Each buffer selector address of each WBC is independently assigned and is specified by the wiring of the address plug for the module.

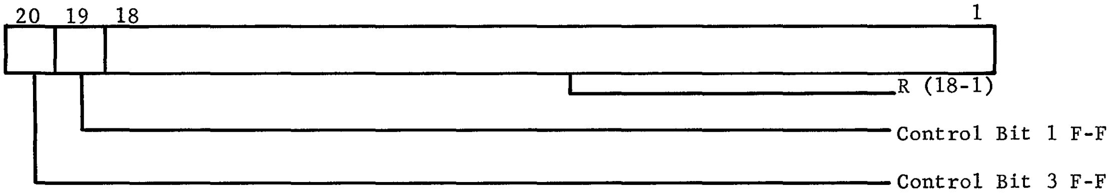
WORD BUFFER INSTRUCTION

Mnemonic	Operand		Word Times
Register Transfer	R,	(TRA from R, to ____)	

The 20 bits in the data register are distributed as follows:

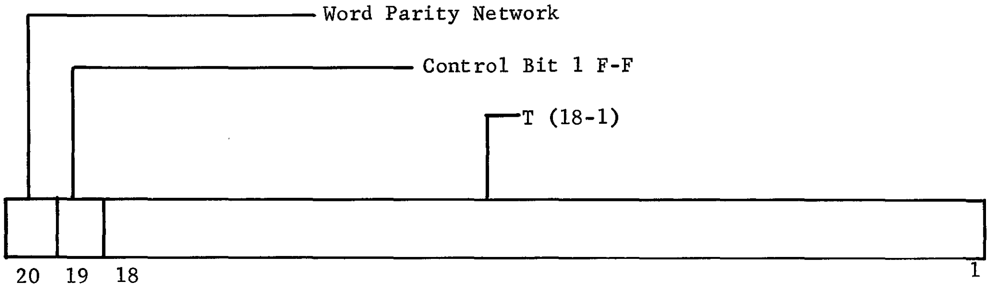
Bits 18-1 go to R(18-1). Bit 19 goes to the control bit 1 flip-flop and bit 20 goes to the control bit 3 flip-flops. The receive flag and data register are reset.

20 bits



Register Transfer	,T	(TRA from ____ to T)	

Bits 18-1 of the B-register are transferred to bits 18-1 of the transmit data register. Bits 19 and 20 of the transmit data register come from control bit 1 and the word parity network.



Mnemonic	Operand	Word Times
DEF	I	1

DEF 1	Reset receive flag and data buffer.
DEF 2	Reset transmit flag and data register.
DEF 3	Set receive mode (turn carrier off).
DEF 4	Set transmit mode (turn carrier on) and initiate transmission.
DEF 5-10	Not used.

NES	I	1

NES 1	Receive flag set (data register contains a new word).
NES 2	Transmit flag set (data register is ready for a new word).
NES 3-10	Not used.

LDT - Do not use.
SCN - Do not use.

RECEIVE OPERATION

Assume that the WBC has been put in the receive mode by the program, that the receive flag is reset, and that the sending unit is transmitting a continuous stream of marks (the line is in the idle condition). The sending unit starts transmitting a 20-bit word. The word is preceded by a start bit (a space) and followed by a stop bit (a mark). When the start bit is received, a clock is started. The clock is used to time the future sampling of the line. The start bit is shifted into the shift register. At regular intervals, the line is sampled and the bit which is present at sampling time is shifted into the shift register. When the shift register is full, the 20-data bits are automatically transferred into the data register, the receive flag is set, and the clock is stopped. The clock will start again and the above process will repeat when the next start bit is received on the transmission line. As a protection against noise on the transmission line causing the clock to start running, the word buffer circuitry requires that the space condition exist on the line for at least one-half of a bit time to start the clock. Thus, noise of less duration than one-half of a bit time will have no effect. Since the word buffer will transfer a word into the data register whether or not the data register and receive flag are reset, the program must test the receive flag and take the word before another is transferred into the data register. When the program takes the word from the data register, the data register and the receive flag are automatically reset.

The timing diagram (Figure 20) illustrates how a 20-bit word would be received at a WBC:

1. The DEF 3 instruction puts the WBC into the receive mode.

2. The DEF 1 instruction resets the receive flag and data buffer.

3. The receive clock is shown sampling the line every bit period.

4. Line 4 shows that the contents of the receive buffer are transferred to the data register after all the bits are received.

5. Line 5 shows the receive communications line going into the WBC.

6. Line 6 shows what the receive buffer would look like after all 22 bits are received.

7. Line 7 shows the receive flag setting when the receive buffer is transferred to the data register.

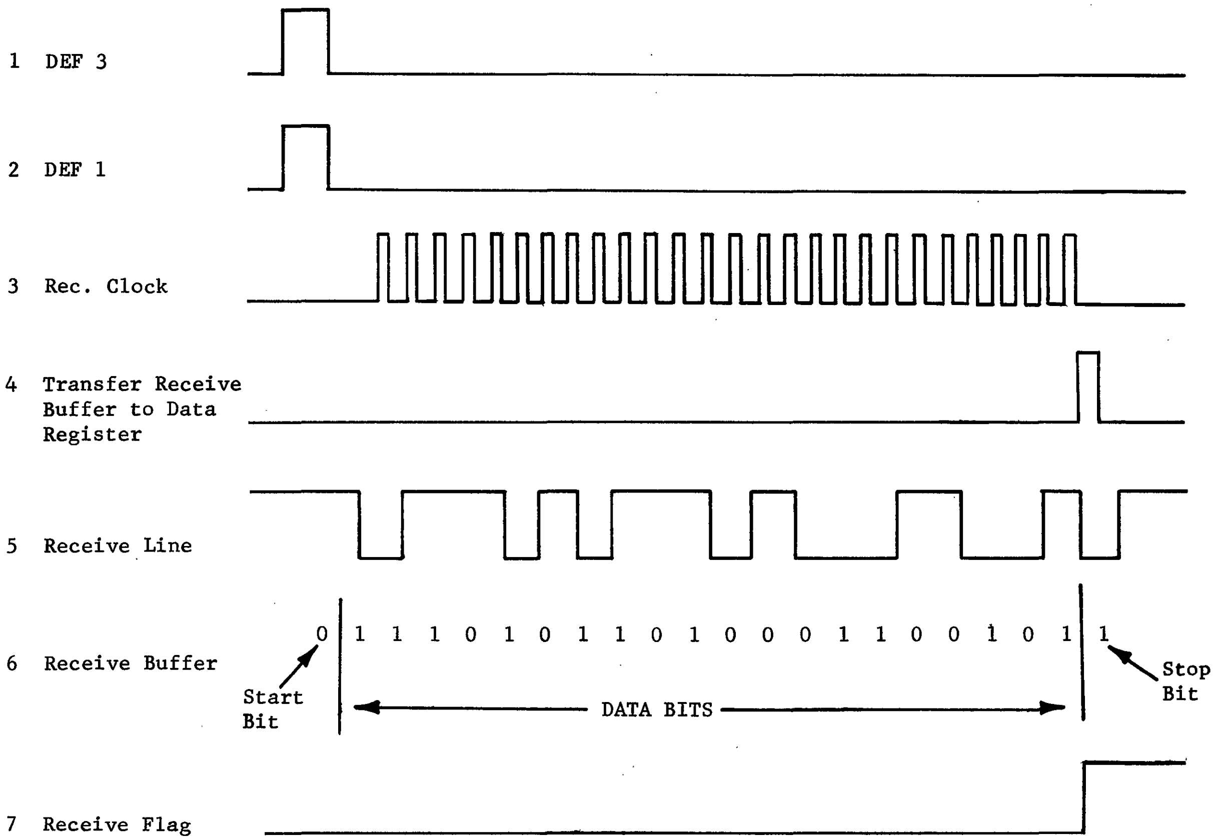


Figure 20. WBC Receive Timing Diagram

TRANSMIT OPERATION

Assume that the program has put the WBC in the transmit mode, the WBC is in the process of sending a word out on the line, and a word is waiting in the data register. When the current word has been shifted into the line, the WBC will transfer the word in the data register to the shift register. At this time, the transmit flag will automatically be set. The 20 bits transferred into the shift register will automatically be preceded by a start bit and followed by a stop bit when transmitted onto the line for a total of 22 bits. When the shift register is again empty, the WBC will transfer the word in the data register to the shift register and repeat the process if the transmit flag is reset. However, if the transmit flag is still set, indicating that the program has not put a new word into the data register, the WBC will continue to put stop bits (marks) on the line until the transmit flag is reset. When the program transfers a new word into the data register, the transmit flag will be automatically reset and the above process will be repeated. For maximum line utilization, the program must test the transmit flag and supply a new word before the current word has been completely shifted onto the line.

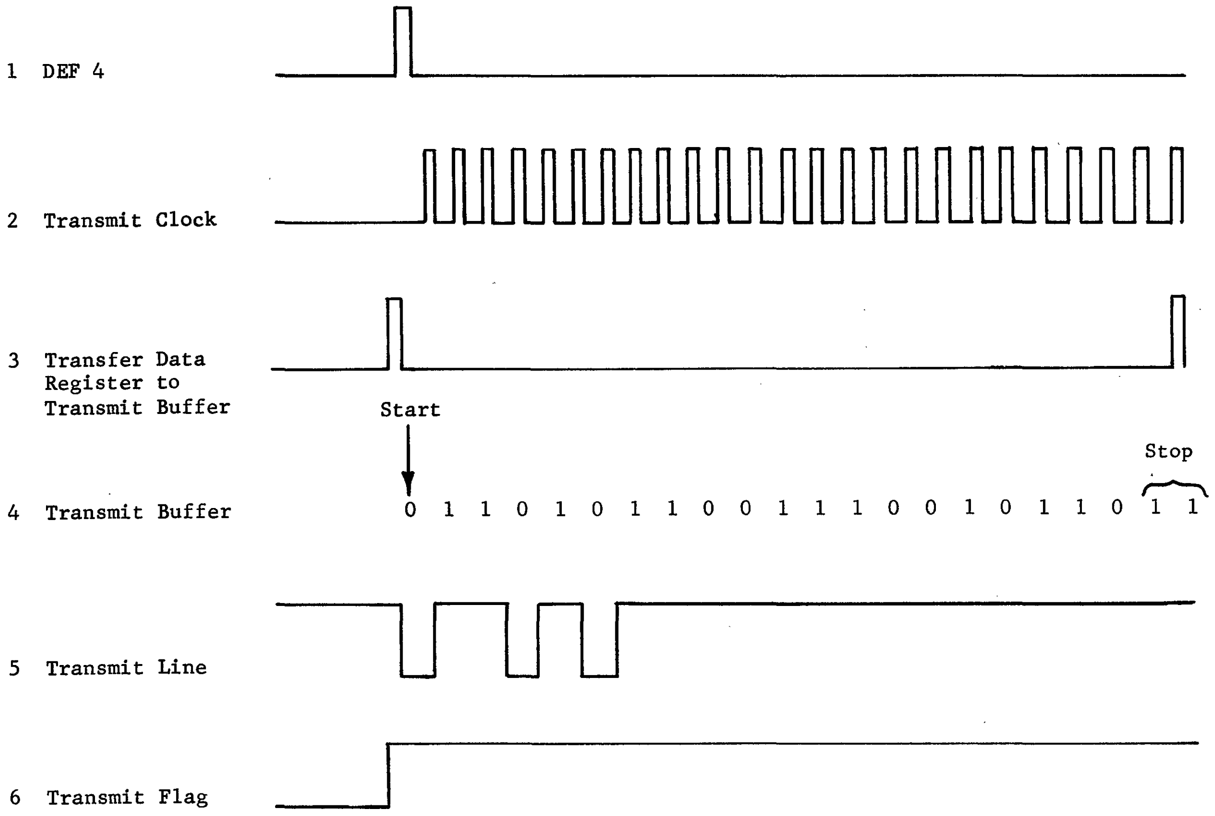


Figure 21. WBC Transmit Timing Diagram

The timing diagram (Figure 21) illustrates what happens when a 20-bit word is transmitted onto a communications line by a word buffer channel:

1. The DEF 4 instruction sets the WBC to the transmit mode.

2. The transmit clock sends data onto the line at regular intervals determined by the baud rate of the line.

3. When the transmit buffer shift register becomes empty the data contained in the data register is transferred to the shift register.

4. This is the binary representation of the binary word in the shift register.

5. Line 5 shows the output of the transmit section of the WBC.

6. The transmit flag is shown setting when the word is transferred from the data register to the shift register.

RECEIVE-WORD BUFFER EXAMPLE

```
Location  Instruction   Symbol   OPR     Operand  X             Remarks

                                 REM                 RECEIVE VIA WORD BUFFER
            03720                ORG     2000        ORIGIN LOCATION 2000
 03720      011017               PIC     15          PLACE BUFFER ADDRESS IN C
 03721      026005               DEF     31          SET RECEIVE MODE, RESET BUFFER
 03722      022001      RECVE    NES     1           TEST FOR FLAG SET
 03723      121722               BZE     *-1         NOT SET, GO BACK
 03724      060044               TRA     R,B         SET, TRANSFER R TO B
 03725      705730               STB     MEMORY   X  STORE WORD IN MEMORY
 03726      341730               ADO     MEMORY      INCREMENT MEMORY ADDRESS
 03727      101722               BRU     RECVE       GO GET NEXT WORD
 03730      005670      MEMORY   IND     3000        INPUT AREA INDIRECT ADDRESS
```

Initially the word buffer address is put into the C-register. The receive mode is set and the buffer is reset by the DEF 3 1 instruction. The flag is tested and the program waits for the flag to set. When the flag sets, the contents of the data buffer are transferred to the B-register, which automatically resets the receive flag and data buffer. The data is stored in memory, and control is transferred back to get next word.

TRANSMIT-WORD BUFFER-EXAMPLE

```
Location  Instruction   Symbol   OPR   Operand  X        Remarks

            07640                ORG   4000        ORIGIN LOCATION 4000
07640       011032               PIC   WBCHN       PUT WORD BUFFER ADDRESS IN C
07641       062004               TRC   0,B         TRANSFER ALL 1's TO B
07642       022002               NES   2           TRANSMIT FLAG SET
07643       121642               BZE   *-1         NO, GO BACK
07644       060401               TRA   B,T         YES, TRANSFER WORD TO BUFFER
07645       026010               DEF   4           SET TRANSMIT MODE
07646       605655      LOOP     LDB   NEXTWD   X  LOAD NEXT WORD TO GO
07647       022002               NES   2           TRANSMIT FLAG SET
07648       121647               BZE   *-1         NO, GO BACK
07649       060401               TRA   B,T         YES, TRANSFER WORD TO BUFFER
07650       341655               ADO   NEXTWD      ADD ONE TO OUTPUT AREA ADDRESS
07651       351654               SBO   WDCNT       SUBTRACT ONE FROM WORD COUNT
07652       131646               BNZ   LOOP        BRANCH TO TRANSMIT NEXT WORD
07653       106000               BRU   0        X  BRANCH LOCATION 0
07654                   WDCNT    DEC   50          NUMBER OF WORDS TO GO
07655                   NEXTWD   IND   6000        OUTPUT AREA INDIRECT ADDRESS
                        WBCHN    EQU   26
```

The Receive Parallel Unit (RPU)

The receive parallel unit (RPU) is a 14-channel, parallel-receive-only unit attached to the buffer selector. Each RPU has an address plug, but no timing plug. Data is received asynchronously at the rate of transmission of the transmitting device.

RECEIVE PARALLEL UNIT INSTRUCTIONS

External Function (DEF) Lines

All 10 DEF lines from the DATANET-30 are brought into the RPU. The line names given apply to the Bell System's DATA-PHONE Data Set 402B. However, these lines may perform other functions for other digital subset interfaces.

Mnemonic	Operand	Word Times
DEF DRIVE EXTERNAL FUNCTION	I	1
DEF 1	Reset Character Ready	1
DEF 2	Reset Answer Back A and B	1
DEF 3	Reset Answer Back Mode	1
DEF 4	Set Answer Back Mode	1
DEF 5	Answer Back A	1
DEF 6	Auxiliary Function (Set Transit Mode)	1
DEF 7	Auxiliary Function (Reset Transmit Mode)	1
DEF 8	Answer Back B	1
DEF 9	Answer Incoming Call	1
DEF 10	Disconnect Call	1

Mnemonic	Operand	Word Times
NES	I	1

AND EXTERNAL STATUS
LINES TO Z

NES 1	Character Ready	1
NES 2	Auxiliary Status Line (Line Turn Around)	1
NES 3	Call in Progress	1
NES 4	Request to Answer Call	1
NES 5	Auxiliary Status Line (Space Detect)	1
NES 6	Auxiliary Status Line	1
NES 7	Auxiliary Status Line	1
NES 8	Auxiliary Status Line	1

RECEIVE OPERATION

The RPU will accept any parallel character occupying up to 14 channels. The number of channels can be reduced to fit the code in use when fewer than 14 channels are used.

Characters are transferred to the least significant bit positions of the DATANET-30 word. The least significant digit of the character will be transferred to Y-Z01, the second least significant digit to Y-Z02, etc.

The RPU is a single buffer device. The character being received will exist in the single buffer for only one character time. As the transmitting device transmits each succeeding character, the new character appears in the buffer immediately. A character must be shifted out of the buffer before the next one is received (transmitted) or the character in the buffer will be lost.

Assume that nothing is being received, and that the RPU is in a state to receive data. When a character is received it is sent to the receive buffer and a flag is set. The program has been periodically interrogating the state of the flag. When the flag is set, and when the program detects this condition, the character is transferred out of the RPU buffer and the flag is reset.

The RPU is capable of utilizing automatic answering and answer-back features of the digital subset. Both hardware automatic answering and program answering can be done. Hardware answering is done by the digital subset. When the digital subset answers, the external status line (ESL 3)-call-in-progress signal will be set until the call is terminated. The program can terminate a call with a Drive External Function (DEF 10)-Disconnect Call instruction.

When the program does the automatic answering, upon the receipt of an incoming call, an External Status Line (ESL 4) signal -- request to answer call -- is set, but the call is not answered until the program can do so. The program can answer with a Drive External Function (DEF 9) instruction - Answer Incoming Call. The call is terminated with the DEF instruction - Disconnect Call.

Normally, the RPU will be in the receive mode. To initiate the answer-back mode, a DEF, Set Answer Back Mode instruction is executed. This puts the RPU in the answer-back mode. There are two answer-back lines that can be set by DEF instructions: Set Answer Back A and Set Answer Back B. The DEF instruction Reset Answer Back A and B will remove the answer-back mode. The digital subset is returned to the receive mode by the DEF instruction Reset Answer Back Mode.

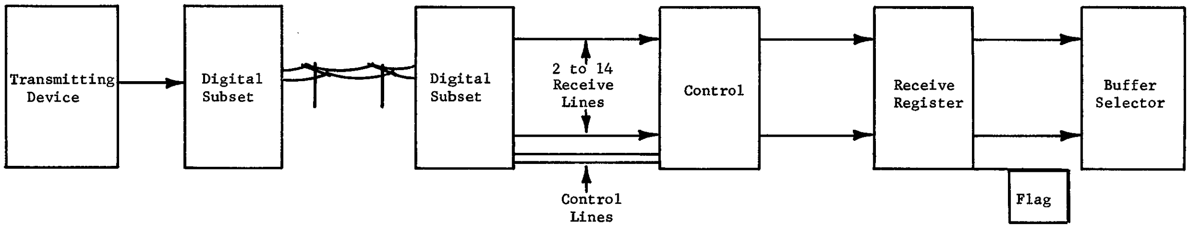


Figure 22. Receive Parallel Unit Block Diagram

PROGRAMMING THE PERIPHERAL EQUIPMENT ON THE CONTROLLER SELECTOR

General Description

The controller selector enables the DATANET-30 to incorporate a variety of peripheral devices. The controller selector is a common control and transfer point for such peripheral units as the magnetic tape system and disc storage units.

Through the use of plug-in connectors, peripheral units can be connected in varying configurations and interchanged according to the requirements of the system.

CONTROLLER SELECTOR INSTRUCTIONS

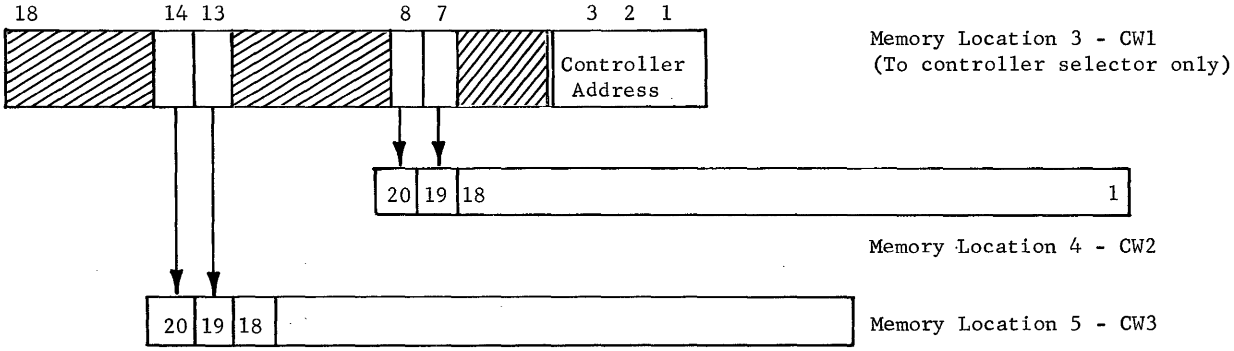
The instructions in this section are broken down into the areas of each peripheral device. There are 3 basic instructions which apply to all peripherals.

Mnemonic	Operand	Word Times
CSR	I	3-10
CONTROLLER STATUS REQUEST.		Loads the B-register with an image of the status lines of the peripheral controller specified by I. I is the plug number of the peripheral on the controller selector.
NIS 7		1
AND INTERNAL STATUS LINE 7.		Interrogates the controller selector to determine if the last controller select command issued has been completed. Sets the branch flip-flops: 1 = controller select is finished. 0 = controller select has not been completed.
SEL		1
SELECT.		Initiates operations as specified by locations 3, 4, and 5 of memory. This instruction is equivalent to a DIF7. (Drive Internal Function)

PERIPHERAL COMMAND WORDS

The peripheral command words are stored in a constants area. In order to transfer the command words to a peripheral, they are moved from the constants area to memory locations 3, 4, and 5; and a select (SEL) instruction is executed.

Memory location 3 contains the address of the selected peripheral in bit positions 1, 2 and 3. Since the commands for the peripherals are 20 bits in length and the DATANET-30 word only contains 18 bits, two bits must be added to the two peripheral command words contained in memory locations 4 and 5. The two extra bits for command word 3 come from positions 13 and 14 of location 3. Two extra bits for command word 2 come from positions 8 and 7 of location 3:



The controller selector stores the bits in positions 7, 8, 13, and 14 from command word 1 and automatically adds the extra bits onto command words 2 and 3 when they are sent to the peripheral unit in order to have the proper length and bit configuration. This pattern is followed for all peripherals on the controller selector.

The Disc Storage Unit (DSU)

This section contains only information special to programming the disc storage unit (DSU) from the DATANET-30. Additional information may be found in the manual for the DSU and other publications.

A DSU consists of 16 storage discs. Information can be recorded or retrieved from both sides of each disc. From one to four 16-disc file units can be connected into one DSU controller.

The maximum bit transfer rate between main memory and a disc file is 500kc (five hundred thousand bits per second). However, information is transferred between memory and the controller or between the controller and the disc file in groups of 18-bit words, or at a rate of 25kc (twenty-five thousand words per second). At this rate, a DSU demands memory access every fifth word time. For this reason, a DSU should be given the highest priority of any of the peripherals connected into the controller selector. The recommended plug address for a DSU is 0, but it could have any plug address from 0 to 7 provided it had the highest priority (lowest address number) of the particular configuration connected into the controller selector.

Each word recorded on a disc consists of 18 information bits plus an odd parity bit which is generated by the DSU controller. The minimum amount of information which can be transferred in either direction by one instruction is 64 words, or one frame. The maximum amount of information which can be transferred in either direction is sixteen 64-word records.

Each 64-word frame is recorded serially in a circular track. There are 256 tracks on each surface of a disc. The 128 outer tracks are each divided into 16 sectors, each sector capable of storing one 64-word record. The transfer rate to or from the 128 outer tracks is 500,000 bits per second. The 128 inner tracks are each divided into 8 sectors, each sector capable of storing one 64-word record. The transfer rate to or from the 128 inner tracks is 250,000 bits per second.

Each disc is served by a positioning arm. Each positioning arm contains eight read-write heads; four heads serve the upper surface of the disc and four serve the lower surface. An actuator for each positioning arm can move the arm parallel to the disc, so that all 256 tracks on each surface can be served. The heads are numbered 0 - 7. Heads 0 - 3 serve the 128 inner tracks (2 for each side of the disc). Heads 4 - 7 serve the 128 outer tracks (2 for each side of the disc). Because there are 4 heads for each side of the disc, the actuator must move the positioning arm a maximum of 63 track positions to serve the 256 tracks on a disc surface.

DISC STORAGE UNIT COMMAND WORD FORMAT

The DSU command word format is shown below.

Operation	Octal Code		File No.	Bits 15 14 13
PRF	00020P	1st command word	0	0 0 1
	5F0000	2nd command word	1	0 1 0
	MMMMMM	3rd command word	2	1 1 1
			3	1 0 0

POSITION One of the DSU controllers, P, (0 - 3) is positioned to receive or transmit a specific record. P is the plug number of the DSU on the controller selector. The line M contains the actual address (octal) of the selected disc file. F is the disc file number on the controller selected by P.

Operation	Octal Code	
RRF	00010P	1st command word
	2F00NN	2nd command word
	0MMMMM	3rd command word
READ	N is the number (1 - 16) of 64-word records to be transmitted from disc storage to core storage. F is the number (0 - 3) of the selected disc file. M is the core memory address into which the first word of the record is stored. P is the plug number of the DSU on the controller selector.	
WRF	00030P	1st command word
	7F00NN	2nd command word
	0MMMMM	3rd command word
WRITE	N is the number (1 - 16) of 64-word records to be transmitted from core storage to disc storage. F is the number (0 - 3) of the selected disc file. M is the memory location of the first word to be transmitted from core storage to disc storage. P is the plug number of the DSU on the controller selector.	

NOTE: The mnemonics are never used in actual coding. The command words must be written in octal form.

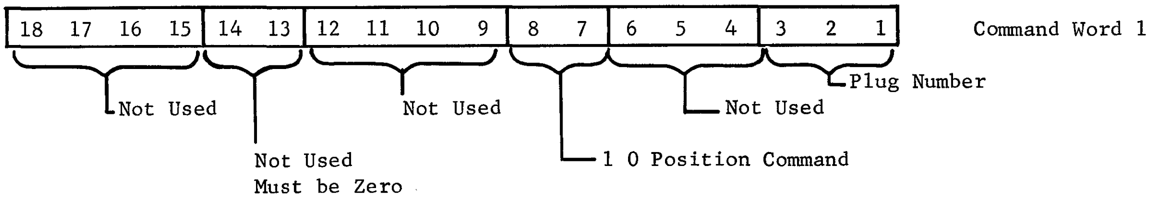
The sequence for addressing the DSU is to select the DSU to be addressed and position one of the arms. The access time varies depending upon the distance the arm must travel to be in position and upon latency time. After the desired arm is in position, the read/write instructions may be executed.

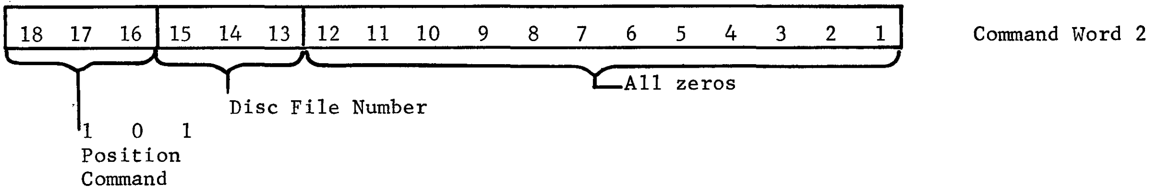
POSITION COMMAND WORDS

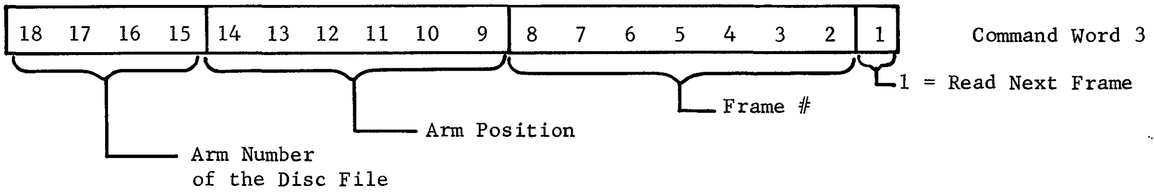
When the command words are to be executed, it is first necessary to store the three command words in memory locations 3, 4, and 5. An SEL instruction executes the positioning instructions (command words).

The third word sent from memory to the controller selects the arm, the arm position, and the address of the frame or frames to be transferred. Bit positions 14 and 13 of command word 1 and bit positions 18 - 15 of command word 2 select the arm (0 - 15) which contains the head or heads to do the writing or reading. Bits 14 - 9 of command word 3 select the arm position (0 - 63) involved in the transfer of information. Bits 8 - 2 select the first frame (0 - 95) to be read or written.

Below is shown the command word format for positioning an arm to a desired track and frame.







Selection of the frame to be transferred also automatically selects the head which is to perform the read or write operation. Each of the eight heads on the positioning arm can read a specified number of frames as follows:

Frame Number Per Arm Position	Head Number
0 - 7	0
8 - 15	1
16 - 23	2
24 - 31	3
32 - 47	4
48 - 63	5
64 - 79	6
80 - 95	7

The seven bits of the frame number (command word 3) designate the head which is to perform the read or write operation as well as the number of the frame. All the 96 frames capable of being read when the positioning arm is in a given position can be addressed by the seven frame number bits whose binary value varies from 0000000 for frame 0 to 1011111 for frame 95, as follows:

Command Word 3 Bits 8 7 6 5 4 3 2		
0 0 0 0 0 0 0 to 0 0 0 0 1 1 1	Inner Tracks 0 - 63 Frames 0 through 7	Top Side
0 0 0 1 0 0 0 to 0 0 0 1 1 1 1	Inner Tracks 0 - 63 Frames 8 through 15	Top Side
0 0 1 0 0 0 0 to 0 0 1 0 1 1 1	Inner Tracks 0 - 63 Frames 16 through 23	Bottom Side
0 0 1 1 0 0 0 to 0 0 1 1 1 1 1	Inner Tracks 0 - 63 Frames 24 through 31	Bottom Side
0 1 0 0 0 0 0 to 0 1 0 1 1 1 1	Outer Tracks 0 - 63 Frames 32 through 47	Top Side
0 1 1 0 0 0 0 to 0 1 1 1 1 1 1	Outer Tracks 0 - 63 Frames 48 through 63	Top Side
1 0 0 0 0 0 0 to 1 0 0 1 1 1 1	Outer Tracks 0 - 63 Frames 64 through 79	Bottom Side
1 0 1 0 0 0 0 to 1 0 1 1 1 1 1	Outer Tracks 0 - 63 Frames 80 through 95	Bottom Side
1 1 0 0 0 0 0 to 1 1 1 1 1 1 1	96 through 127 Invalid Address	

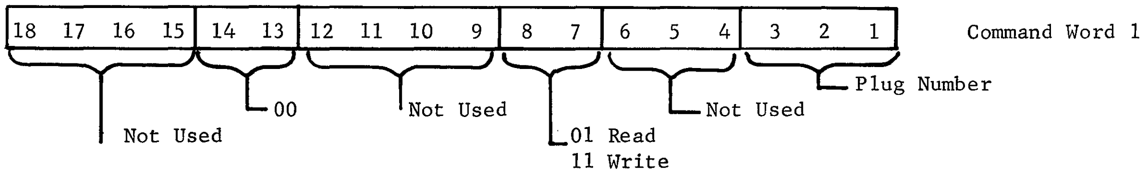
The maximum number of frames which can be transferred by one instruction is 16. It is not necessary that these 16 frames (or any part of 16 frames) all be in the outer tracks or all be in the inner tracks. The transfer of information during the execution of an instruction can start in the inner tracks and continue in the outer tracks. As frames are being transferred, a count is maintained in the DSU controller, so that the read or write operation continues for the specified number of frames. As already explained, the sequential incrementing of the frame address in the controller automatically results in the proper head switching. Because frame 95 is the highest valid address, incrementing the address in the controller beyond 95 causes frame 0 to be the next frame transferred.

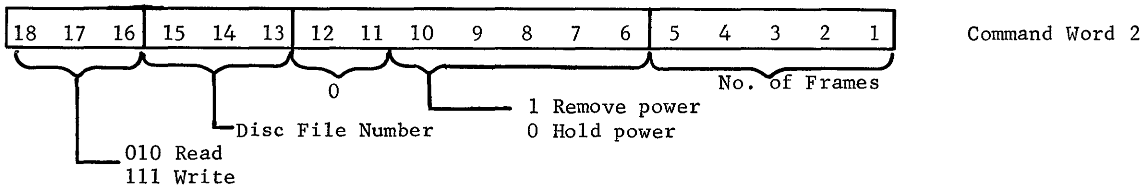
Bit 1 of word 3 is identified by read next frame. When this bit is on (contains a 1) the seven bits of the frame address are ignored and the subsequent reading or writing operation takes place in the next frame. Bit 1 of command word 3 is used when it is desired to sample a frame from any given position of the positioning arm. Rather than search for a specific frame out of the 96 possible, the next frame can be read. This form of addressing can also be used when it is known that every frame in a track is to be transferred and it does not make any difference which is read first.

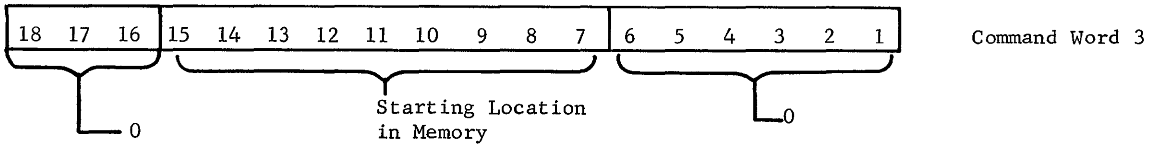
READ/WRITE COMMAND WORDS

After a DSU arm has been positioned the DSU can then be addressed for a read or write operation. It is first necessary to store the three command words for the read/write operation in memory locations 3, 4, and 5. An SEL instruction executes the read or write.

The command word format for read or write operations is as follows:







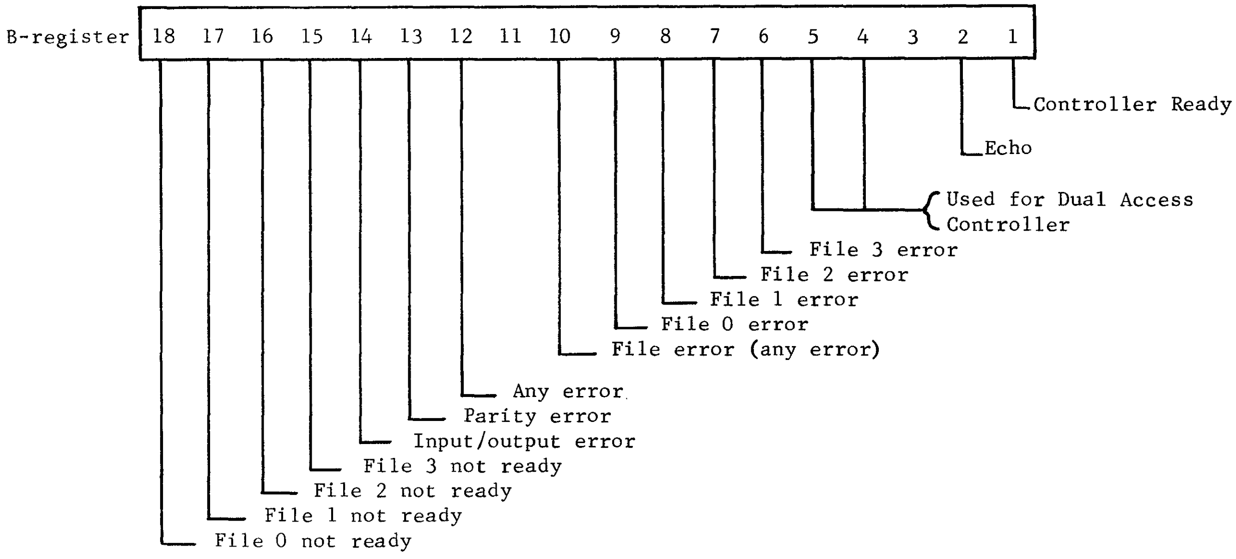
Command word 1 selects the controller selector plug (P) into which the DSU is connected. Once the DSU unit has been selected by the controller selector, the DSU controller goes into the busy state and waits for the next two words from memory. The next two words are sent to the DSU controller and indicates the operation to be performed (read or write), the file which is to perform the operation, the number of frames to be transferred, and the starting address in memory where information is to be sent or retrieved.

Bit positions 8 and 7 of word 1, and 18 - 16 of word 2 cause the file to read (octal 12) or write (octal 37). Bits 15 - 13 of word 2 indicate the file which is to perform the read or write operation. Bits 5 - 1 of word 2 indicates the number of records (0 - 16) which can be transferred. A "1" in bit position 10 of command word 2 of a Read or Write sequence will cause power to be removed from the positioning motor upon completion of that sequence. A "0" in bit position 10 of command word 2 holds power to the positioning motor.

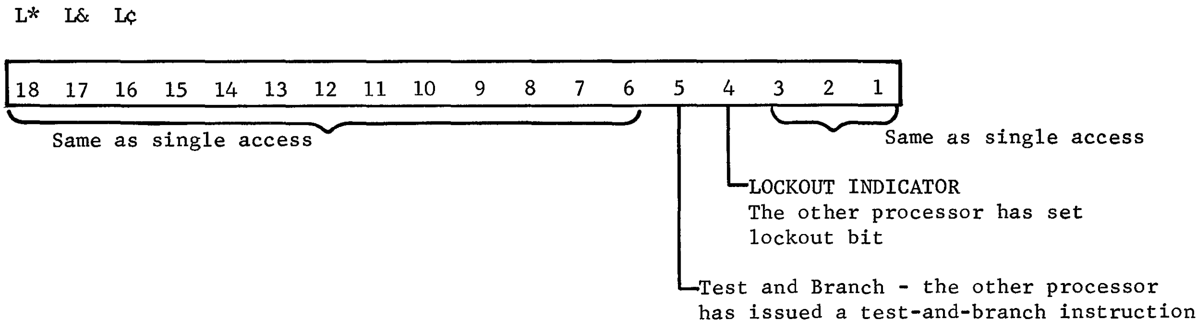
Bits 15 - 7 of word 3 transferred to the controller indicate the starting location in memory of the read or write operation. The nine bits of the starting location address allow this address in the controller to be stepped 1024 times or, in other words, to count the 1024 words of 16 records, the maximum which can be transferred by one instruction. Because bits 6 - 1 of word 3 are not used, the starting location address must be a multiple of 64. Bits 15 - 7 can address memory capacities up to 32,767 words.

BRANCH CONDITIONS

Single-access DSU branch conditions may be tested by looking at the B-register after a CSR command. Bits 15, 16, 17, and 18 are on in the illustration below:



The dual-access DSU may be tested in an identical manner with two additional test positions:


L*
L&
L¢
18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
Same as single access
Same as single access
LOCKOUT INDICATOR
The other processor has set lockout bit
Test and Branch - the other processor has issued a test-and-branch instruction

SAMPLE CODING TO ADDRESS a DSU

Below is given a sample coding for positioning the arm:

```
NIS    7
BZE    *-1
CSR    0
BEV    *-1
BMI    *-2
LDB    1st Word (Command)
STB    3
LDD    2nd and 3rd Word (Command)
STD    4
SEL        (Arm starts seek for Position. DSU goes ready when in position. Now issue
            Read or Write.)
```

Below is given a sample coding for a read or write operation:

```
NIS    7
BZE    *-1
CSR    0
BEV    *-1
BMI    *-2
LDB    1st Word (Command) Read/Write
STB    3
LDD    2nd and 3rd Word (Command) Read/Write
STD    4
SEL        (To Execute Read/Write.)
```

Operating Times

Following are the DSU operating times:

Speed of rotation of discs	1200 rpm
Effective bit transfer rate	
Inner tracks	250 kc
Outer tracks	500 kc
Maximum latency time*	52 ms
Average latency time	26 ms
Average access time (latency time plus positioning time)	199 ms

PHYSICAL CHARACTERISTICS

The physical characteristics of the DSU are shown below:

Number of discs per file	16
Number of recording surfaces	32
Number of positioning arms	16
Number of read/write heads per positioning arm	8
Number of read/write heads per surface	4
Number of tracks per surface	256
Inner zone	128
Outer zone	128
Number of words per frame	64
Number of frames per track	
Inner zone	8
Outer zone	16
Number of frames per surface	3,072
Number of frames per 16-disc file	98,304
Number of words per file	6,291,456
Number of bits per file	
Information bits	132,120,576
Check word bits	2,064,384
Total number of bits per file	134,184,960

* Latency time is the time necessary for a piece of information to reach a read/write head as the disc revolves.

PROGRAMMING THE MAGNETIC TAPE UNITS

This section contains only information special to programming the magnetic tapes from the DATANET-30. More detailed information on magnetic tape and additional programming information may be found in the manual for the magnetic tape units and other publications.

Magnetic tape units can be operated in two different modes: decimal and special binary. During forward movement of the tape, information can be written on or read from tape in both modes. During backward movement, information can be read from tape in both modes.

During the decimal mode of operation the zone bits -- the two most significant bits of each six-bit binary-coded decimal (BCD) character -- are altered during transfer of information between magnetic tape and memory. This alteration of the zone bits takes place automatically in the tape controller as follows:

BCD Character in Memory	BCD Character on Tape
0 0 XXXX	0 0 XXXX
0 1 XXXX	1 1 XXXX
1 0 XXXX	1 0 XXXX
1 1 XXXX	0 1 XXXX

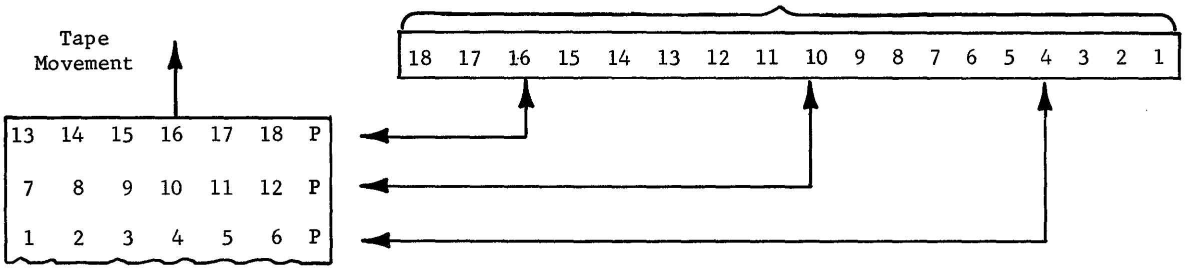
The four least significant bits (XXXX) of each BCD character are the same in memory or on tape with one exception: a BCD 0 in memory is 000000 but on magnetic tape it is 001010. The alteration of information during the decimal mode takes place for any configuration of bits (there are no illegal bit configurations). The alteration of information during the decimal mode of operation makes the DATANET-30 magnetic tapes compatible with magnetic tape formats now in use.

During binary operations of magnetic tapes, information is transferred between magnetic tapes and memory without alteration of bits.

Decimal Mode

In the decimal mode of magnetic tape operations, 18 bits (18 - 1) of a memory word correspond to 3 BCD characters. Each word from memory is checked for parity in the tape controller. When information is read from tape, bits 0 and 1 are made 0 when three BCD characters enter a memory cell. The following illustration shows the relationship between a word in memory and the three BCD characters on magnetic tape.

3 - Six Bit BCD Characters in Memory

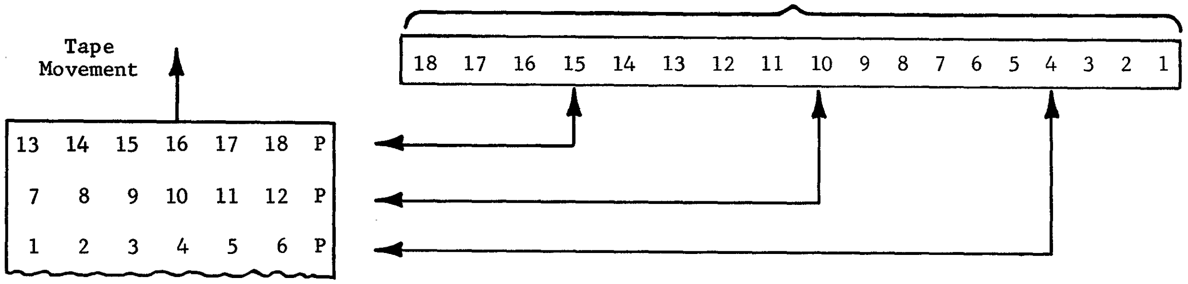


P is a generated even parity bit for each character

Binary Mode

During binary mode operations, 18 bits (1 - 18) are written on tape as three lines of information.

18 - Bit Binary Word in Memory



P is a generated odd parity bit for each line of binary information

The format of information on tape in the binary mode is the same as in the decimal mode. In the binary mode, however, the zone bits and 0 are not altered during the transfer of information. Also, in the binary mode, the parity bit P generated for each line on tape is an odd parity bit.

Record Length

After reading (in binary or decimal mode) N words from magnetic tape into memory starting at location M, memory location M + N will contain zeros if exactly N words were read from a record on tape containing N words. If the number of words contained in the record currently read is less than N, then only the contents of the record will be stored in memory and the 2's complement of the difference (N - record length) will be stored in memory cell M + N with a 1-bit in position 18. If the number of words in the record is greater than N, then only N words will be stored in memory and the increment (record length - N) will be stored in memory cell M + N with a 0 in the sign position. M is not automatically modified. In order to forward space (skip) one record, the RTS, RTD, or RTB command is used with N set equal to 0. This statement also applies to the read tape backward instructions except that M - N will contain zeros if exactly N words were read from a record on tape containing N words. M - N will contain the 2's complement of the difference (N - record length) with a 1 in position 18 if the number of words contained in the record currently read is less than N. M - N will contain the increment (record length - N) if the number of words in the record is greater than N.

Magnetic Tape Instructions

Operation — Octal Code

WTD
WRITE TAPE DECIMAL.

0T000P - CW1
2MMMMM - CW2
TNNNNN - CW3

N decimal words from memory starting at location M are written on handler T. P is the plug number of the tape controller.

RTD
READ TAPE DECIMAL.

0T000P
4MMMMM
TNNNNN

A maximum of N decimal words is read by tape handler T and placed in memory starting at location M.

WTB
WRITE TAPE BINARY.

0T020P
3MMMMM
TNNNNN

N words of information from memory starting at location M are written by tape handler T. Bits 18-1 are written on tape exactly as in memory.

RTB
READ TAPE BINARY.

0T020P
5MMMMM
TNNNNN

A maximum of N words is read by tape handler T and stored in memory starting at location M.

Operation

Octal Code

RBD
READ BACKWARD DECIMAL.

0T010P
4MMMMM
TNNNNN

Decimal information is read from tape moving backwards. A maximum of N words is read into memory, the first word being placed in location M. The second word is placed in M - 1 and so on until N words are read. The tape controller alters the zone bits of characters read so that they conform to GE Compatibles/200 internal BCD characters.

RBB
READ BACKWARD BINARY.

0T030P
5MMMMM
TNNNNN

Information is read from tape moving backwards. Contents of bit positions 2 - 19 of each word read are placed in memory exactly as on tape (zone bits are not altered). A maximum of N words is read into memory, the first word being placed in M. The second word read is placed in M - 1 and so forth until N words are read.

RWD
REWIND.

0T020P
000000
T00000

Rewind tape handler T to leader.

WEF
WRITE END-OF-FILE.

0T000P
200000
T00000

The end-of-file character (0001111) and end-of-file gap are written on tape by tape handler T.

BKW
BACKSPACE AND POSITION WRITE HEAD.

0T010P
600000
T00000

The tape on tape handler T is backspaced one record and the write head is positioned to write.

Command Words

The table below shows the digits used for specifying each tape handler:

	Handler							
Command Word	0	1	2	3	4	5	6	7
1st Word	$0\underline{0}0Y_10P$	$0\underline{0}0Y_10P$	$0\underline{0}0Y_10P$	$0\underline{1}0Y_10P$	$0\underline{2}0Y_10P$	$0\underline{2}0Y_10P$	$0\underline{2}0Y_10P$	$0\underline{3}0Y_10P$
2nd Word	Y_2MMMMM	Y_2MMMMM	Y_2MMMMM	Y_2MMMMM	Y_2MMMMM	Y_2MMMMM	Y_2MMMMM	Y_2MMMMM
3rd Word	$\underline{1}$NNNNN	$\underline{2}$NNNNN	$\underline{4}$NNNNN	$\underline{0}$NNNNN	$\underline{1}$NNNNN	$\underline{2}$NNNNN	$\underline{4}$NNNNN	$\underline{0}$NNNNN

Y_1 Y_2 are the octal digits for the different tape instructions. The numbers underlined are used for specifying the handler number.

P is the plug number of the tape controller.

M is the address being written out of or read into memory.

N is the number of words being read or written - that is, record length. For example:

Instruction	Tape Handler	Plug
RTB	3	2

The 3 command words from "Octal Code" column.

$$\left\{\begin{array}{lcl} \text{0T020P} & = & \text{010202} \\ \text{5MMMMM} & = & \text{500200} \\ \text{TNNNNN} & = & \underbrace{\text{000400}}_{\text{number of words being read or written}} \end{array}\right.$$

From table above

Inst.	Y_1	Y_2
WTD	0	2
WTB	2	3
RTD	0	4
RTB	2	5
RBD	1	4
RBB	3	5
RWD	2	0
WEF	0	2
BKW	1	6

Handler Number	T for Command Word 1	T for Command Word 3
0	0	1
1	0	2
2	0	4
3	1	0
4	2	1
5	2	2
6	2	4
7	3	0

The above tables can be used when setting up the three command words. Before issuing the command words to the tape controller, the command words are first transferred to memory locations 3, 4, and 5. A SEL instruction executes the transfer of the command words to the magnetic tape controller. When the command word instructions are coded, the above octal coding is used as the operand.

Programming Example

The following example shows how to write 64-word records on magnetic tape handler 2 out of location 500:

```
Symbol     OPR     Operand    Remarks

WTB        CSR     1          GET STATUS LINES
           NBZ     READY      TAPE READY?
           BZE     *-2        NO, GO BACK
           LDB     1STWD      GET COMMAND WORD 1
           STB     3          STORE IN LOCATION 3
           LDD     WD2,3      GET WORDS 2 AND 3
           STD     4          STORE IN LOCATIONS 4 AND 5
           SEL                SELECT PERIPHERAL
           NIS     7          SELECT DONE?
           BZE     *-1        NO GO, WAIT
           BRU     WTB

READY      OCT     000001     BIT 1 TO TEST READY
1STWD      OCT     000201     COMMAND WORD 1
WD2,3      OCT     300500     COMMAND WORD 2
           OCT     200100     COMMAND WORD 3
```

In the preceding example, initially the CSR command is executed to test the ready status of the tape controller. When the controller becomes ready, the 3 command words are loaded from their temporary storage locations and put into locations 3, 4, and 5. The SEL command initiates operation of the controller selector unit and the commands are automatically sent to the tape controller. Next, the NIS7 interrogates the controller selector to see if the last controller select is finished. When the select has been finished the program returns to write a new record.

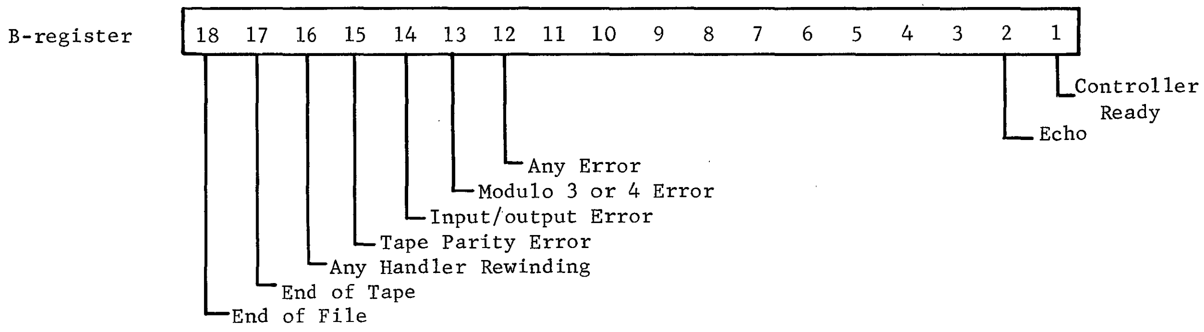
Tape Unit Conditions

Tapes contain a silver spot to signal the physical end of the tape. When detected by a photoelectric cell within the tape unit, an indicator on the tape controller is set. The condition of the indicator should be tested by programmed instructions after reading or writing each record. If the indicator is not set, normal processing will continue. If it is set, an end-of-tape branch will jump into specified subroutines - normally rewinding the current reel and switching to a new reel. The end of file sentinel is the magnetic representation of the binary code 001111 preceded by an erased section of the tape 3-3/4 inches long.

During magnetic tape operations several other exceptional conditions may occur which are secondary to the main processing job. Handling of these exceptional conditions may be conveniently assigned to "executive routines." These conditions are handled as branch conditions.

Branch Conditions

The branch conditions concerned with the tape controller may be tested by examining the bits in the B-register after a CSR instruction. When the particular bit is on, the condition is true, as shown below:



Examples:

Controller Ready (Controller on Plug 4)

```
        CSR     4
        BEV     *-1  (not ready)
```

Echo

```
        CSR     4
        SR1     B,Z
        BEV             No  error
```

Any Error

```
        CSR     4                       CSR     4
        NBZ     004L                    SL6     B,Z
        BZE             No error        BPL         No error
         .
         .
004L    OCT   004000
```

Mod 3 or 4 Error

```
        CSR     4                       CSR     4
        NBZ     01L                     SL5     B,Z
        BZE             No error        BPL         No error
         .
         .
01L     OCT     010000
```

Tape Parity Error

```
        CSR     4                       CSR     4
        NBZ     04L                     SL3     B,Z
        BZE             No error        BPL         No error
         .
         .
04L     OCT     040000
```

Any Handler Rewinding

```
        CSR     4                       CSR     4
        NBZ     1L                      SL2     B,Z
        BZE             No error        BPL         No error
         .
         .
1L      OCT     100000
```

PROGRAMMING THE PAPER TAPE READER

The paper tape reader reads at a continuous rate of 300 characters per second. Tape can be read under program control or hardware control, depending upon the format in which it is punched. Paper tape punched in the hardware load format is always read at the maximum 300-character-per-second rate under automatic control of the DATANET-30 circuitry. The paper tape reader is always on buffer selector address 0.

Paper tape may be read under program control in two modes, continuous mode and step mode. Five- to eight-level tape may be read but normally only eight-level tape will be used. If paper tape is read in continuous mode, the character under the read station must be taken away 500 microseconds after the flag is set. If the 500 microsecond timing restriction is not met, reading must be done in the step mode at a speed of approximately 50 characters per second.

In either mode, when the sprocket hole is detected, the character under the read station causes the receive flag to be set. When the character is taken away, the flag is automatically reset and the reader moves the tape to the next character. This control of the movement of tape is in effect at both 300 and 50 characters-per-second speeds. The sprocket hole serves as a timing source. A sprocket hole only indicates a character and will set the receive flag.

The reader is turned on by the POWER ON switch on the paper tape reader control panel. Normal operation requires that the reader be turned on at all times.

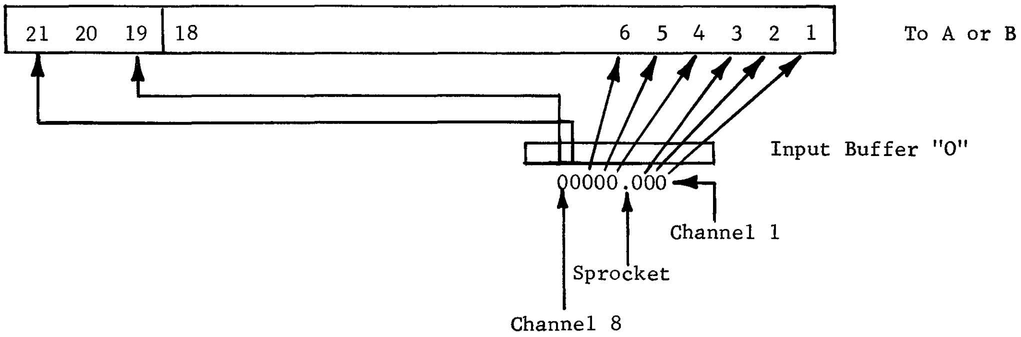
Reading Paper Tape Under Program Control

PAPER TAPE READER INSTRUCTIONS

Following are the paper tape reader instructions:

Register Transfer (From R,____)

The character contained in the buffer is transferred to register A or B, as in the diagram below. The receive flag and data buffer are reset. If stopped, any register transfer instruction from R starts paper moving or allows the movement of paper to continue.



DEF 1 Reset flag and read next character. The reader starts paper moving through the reader or allows the movement of paper to continue.

DEF 2-10 No effect.

NES 1 Read flag set (a new character is ready).

SCN Do not use.

LDT No effect.

Register Transfer ______, T - No effect.

The following example is a few lines of coding which show one way in which paper tape might be read. In this example, paper tape is punched in 6-level code and 3 characters are assembled into one word. Channels 7 and 8 are not punched.. In this example, the 7 and 8 channels are transferred but are not used.

```
Location  Instruction  Symbol   OPR   Operand  X   Remarks

           13560                ORG   6000         ORIGIN LOCATION
 13560     011000               PIC   0            PUT PAPER TAPE READER ADDRESS IN C
 13561     022001      READ     NES   1            CHARACTER PRESENT?
 13562     121561               BZE   *-1          NO, GO BACK
 13563     044044               SL6   R,B          YES, SHIFT TO B-REGISTER
 13564     022001               NES   1            CHARACTER PRESENT?
 13565     121564               BZE   *-1          NO, GO BACK
 13566     044444               SL6   BR,B         YES, SHIFT TO B-REGISTER
 13567     022001               NES   1            CHARACTER PRESENT?
 13570     121567               BZE   *-1          NO, GO BACK
 13571     060444               TRA   BR,B         YES, TRANSFER TO B
 13572     705576               STB   WKSTOR   X   STORE IN MEMORY INPUT AREA
 13573     341576               ADO   WKSTOR       ADD 1 TO INDIRECT MEMORY ADDRESS
 13574     771577               XBZ   STOP         IS THIS A STOP WORD?
 13575     131561               BNZ   READ         NO, GO READ NEW WORD
 13576     001750      WKSTOR   IND   1000         INDIRECT ADDRESS
 13577     777777      STOP     OCT   777777       STOP CONSTANT
```

Initially buffer selector address 0 is put into the C-register. The NES1 command tests the buffer for a character, and status line 1 will remain a 0 until a character is present. When the flag sets, the program falls through the BZE test and shifts the character into the B-register. When three characters have been assembled in the B-register, they are stored away in memory and a test is made to see if the last word was a stop signal. If the word was not a stop signal, control is transferred back to the symbol READ and the reading process continues.

NOTE: When tape is loaded in the reader, the tape will stop with a sprocket hole over the read station. A sprocket hole by itself will set the flag and represents a "blank" character.

PROGRAM LOAD FORMAT

A paper tape generated by General Assembly Program 3 in the program load format can only be loaded into the DATANET-30 by a loader program. It is not hardware loadable.

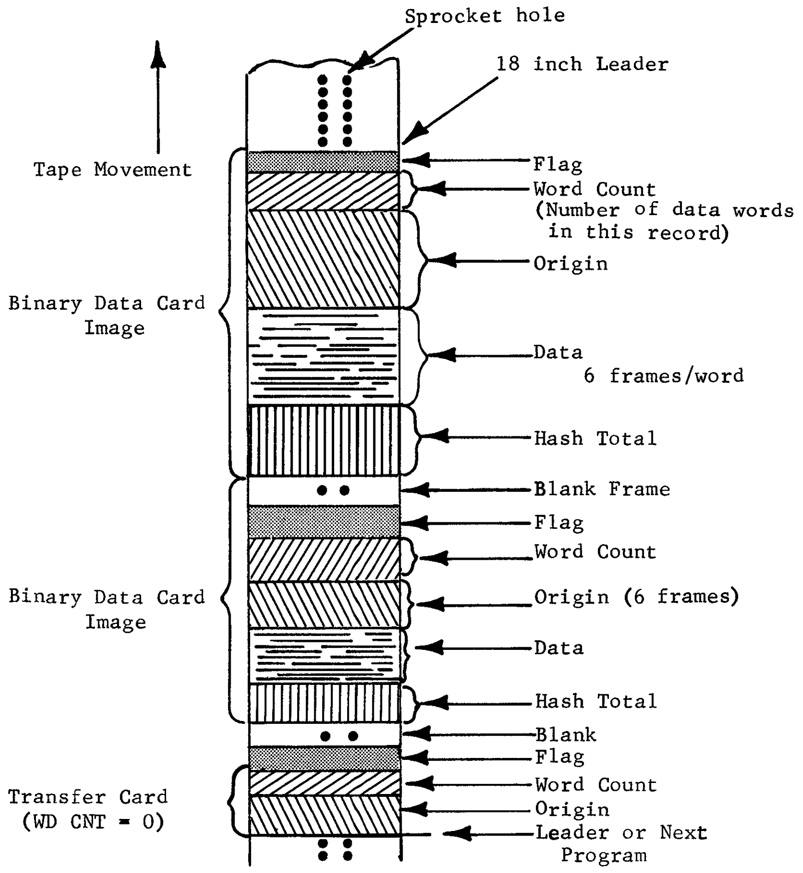
The program load paper tape code is shown below:

		Channel on Tape							
		8	7	6	5	4	3	2	1
	Leader →	0	0	0	1	0	0	0	0
	Flag →	1	0	0	0	0	0	0	0
Digit	0	0	0	1	1	1	0	0	0
	1	0	0	0	0	0	0	0	1
	2	0	0	0	0	0	0	1	0
	-	-	-	-	-	-	-	-	-
	7	0	0	0	0	0	1	1	1

1 = Hole

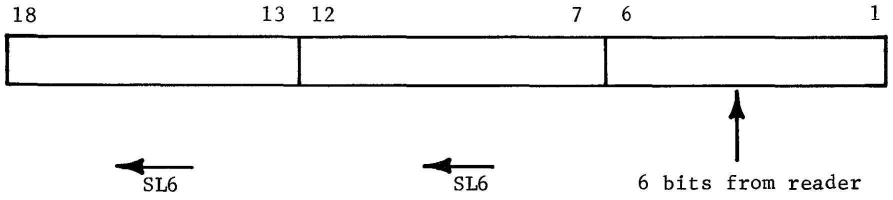
0 = No Hole

EXAMPLE:



Hardware Load and the Paper Tape Reader

Once initiated, the loading of data from the paper tape reader is accomplished entirely under hardware control. A special format (operation code), in channels 7 and 8 (the control channels) controls the shift of data in channels 1 - 6 from the reader to the B-register and then into memory. The characters in channels 1 - 6 are transferred into the B-register and assembled to form a word. Since the DATANET-30 word is 18 bits, two shifts of 6 bits each are required.



When the B-register is filled with the third transfer of data into B, the word is transferred to memory. (Operation code 01XXXXXX.)

Operation code in channels 8 & 7

Operation

8 7 654321

1 0 111111 Begin hardware load. The reader searches for this code before the transfer of data can start.

0 0 XXXXXX SL6 BR, B
Bits 1 - 6 from the paper tape reader are OR-ed into 1 - 6 of Y with the contents of the B-register. Y is shifted left 6 to Z. Z is transferred to the B-register.

0 1 XXXXXX TRA BR, B
Store B in memory location specified by P. Count P up 1. Clear B.

Bits 1 - 6 from the paper tape reader are OR-ed into 1 - 6 of Y with the contents of the B-register. Y is transferred to Z without change. Z is transferred to the B-register. The contents of the B-register are stored in memory as specified by P. P is counted up by 1, and the B-register is cleared.

1 1 XXXXXX TRA BR, B
TRA B, P. Clear B

Bits 1 - 6 from the paper tape reader are OR-ed into 1 - 6 of Y with the contents of the B-register. Y is transferred to Z without change. Z is transferred to B. Then the contents of the B-register are transferred to P, and the B-register is cleared.

Operation code in channels 8 & 7	Operation
1 0 XXXXX0	End hardware load. Control is automatically transferred to the program. The program starts at the address specified by the P-counter.

NOTE: Only begin hardware load and end hardware load use all 8 channels for the operation code. A punch is a 1, a blank is a 0. A blank space (sprocket hole only) causes zeros to be transferred into B.

STRUCTURE TABLE TO HARDWARE LOAD OPERATION

The sequence of operations for hardware load is shown by the following steps:

1. When hardware load is initiated, the C-register is set to zero, the Q-counter is set to -1, the paper starts moving through the reader, and the tape is examined for the begin hardware load character.

2. Read a character.

Character 87 | 654321	This Occurs	Go To Step
10 | 111111 Begin HWL	Sets B-register to Zero	3
XX | XXXXXX any character except hardware load	Nothing happens	2

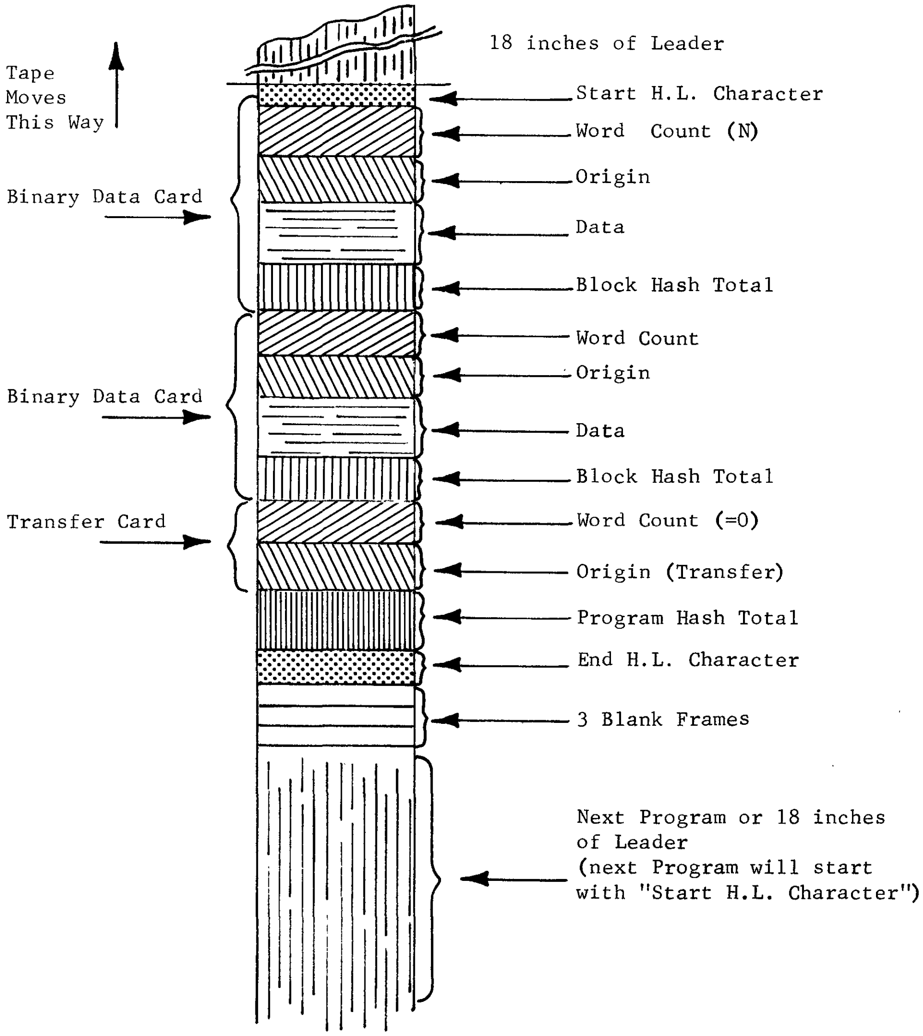
3. Read a character.

Character 87 | 654321		Go To Step
00 | XXXXXX (0XX) 0 | X X	SL6 BR,B	3
01 | XXXXXX (1XX) 1 | X X	TRA BR, B STB "P CTR" Count P (P=P+1) Set B-register to zero	3
11 | XXXXXX (3XX) 3 | X X	TRA BR, P Set B-register to zero	3
10 | 000000 (200) 2 | 0 0	Start the program at location specified by P-counter	Starting Location of Program

Hardware Load Format

The hardware load format output of the General Assembly Program may be loaded into the DATANET-30 by either hardware load or a loader program. When the paper tape is loaded via a loader program, checking is accomplished by the block hash total and program hash total. When the paper tape is loaded via hardware load, no checking by hash total is accomplished.

The block hash total is located at position N + 1 of a block of N words. Program hash total is located after the address of a transfer word, and before the end hardware load character. Block is the equivalent of a binary card or binary tape record. Octal cards will be converted to a block length of one. An example of hardware load paper tape format is shown below:



UTILITY ROUTINES

Since the output from the DATANET-30 General Assembly Program is magnetic tape (switch option) or punched cards and the input to the DATANET-30 is punched paper tape, a conversion program is needed. A utility routine (General Assembly Program 3) on the DATANET-30 General Assembly Program systems tape will accomplish this, producing paper tape in various formats on a free-standing paper tape unit which has the eight-level straight transfer mode. One of the formats is compatible with hardware load, so that self-loading programs can be produced. Other formats are read by paper tape loader programs.

The Paper Tape Conversion (General Assembly Program 3) Utility Routine can be run following the DATANET-30 General Assembly Program by setting the console switches.

SYSTEM CONSIDERATIONS

The Message Switching Center

When operating as a message switching center, the configuration of the overall system must be considered:

1. Number and type of incoming/outgoing lines -- half duplex, full duplex, etc.
2. Number of receive-only remote terminals
3. Number of stations per multipoint line
4. The speed of transmission on each line, if there are transmission speed differences in the system.
5. The handling of priority messages, if any
6. Whether or not another DATANET-30 is included in the system
7. Routing codes: multiple broadcast or single address
8. Remote station identification codes
9. Message format
10. How communication with other networks will be handled
11. Control of the system for beginning of day and end of day
12. The type of remote terminal equipment and all operating characteristics.

The above list only partially covers the considerations necessary. After the characteristics of each system have been determined, the programming can proceed.

Integrated Data Processing

The inclusion of a computer in the overall system permits various methods of handling incoming/outgoing messages.

In one case, incoming data intended for the computer is transferred directly. In another, the incoming data is stored first in a disc storage unit and retrieved by the computer.

A system may also store data in the disc storage unit and transfer it to the computer at a certain time of day for batch processing. Individual operating procedures and program requirements will necessarily be developed for each system.

PROGRAM PREPARATION

General

The principal programming tool is an assembler. Writing programs at the assembler language level is the fastest and most economical way to create the efficient real-time programs needed.

The relative importance assigned to the system factors of operating time, memory utilization, and coding effort strongly affects the relative importance assigned to the software factors of assembler, compiler, and subroutines.

For real-time applications, operating time is of paramount importance, because system capability is strongly dependent upon program efficiency of operating time. Memory utilization is also important, since system performance depends strongly on the amount of memory available for data storage. Coding effort is of much less importance in the overall considerations and life of a program.

In order to minimize the operating time and the memory space needed, the program should be written at the assembly language level.

The DATANET-30 assembly program was written to run on a GE-225 Information Processing System. Programs written for the DATANET-30 must be assembled on a GE-225. If the programs are written at the Computer Department Headquarters, assembly can be done there. If the programs are written in other areas, they can be assembled at one of the many General Electric Information Processing Centers located throughout the country.

For most systems, the system capability will be inversely proportional to the amount of time required to service a line -- that is, if the time per line can be reduced 20 percent with more efficient programming, then the system has the capability to handle 20 percent more lines. Another way of looking at the importance of the operating time used by a program is that if system requirements specify that 10 ms are available in which to service all lines once, then a program which requires 11 ms cannot be used unless the 10 ms specification is changed, the number of lines reduced, or the 11 ms program made more efficient. Because the amount of operating time is so important, several special features have been included in the hardware to reduce the operating time. Writing the actual coding at the assembler level is the best way to utilize these special features and attain the necessary efficiency.

The amount of memory used for the instructions and tables in a program will determine how much memory is left over for data storage. Decreasing the program memory required will increase the data storage memory available, thus improving the store and forward performance. In addition, for those applications which permit giving a busy signal, more data storage memory will delay or possibly eliminate the point at which a busy signal will have to be given because of a full memory. For those applications which do not permit giving a busy signal, decreasing the memory required for the program will decrease the total amount of memory needed in the system.

THE GENERAL ASSEMBLY PROGRAM

General Description

The General Assembly Program is an effort-saving procedure that permits writing programs in specific mnemonics rather than in the absolute computer coding. For example, mnemonic ADO is used to indicate the add 1 operation, mnemonic SBO to indicate the subtract 1 operation, etc. The instruction mnemonics are chosen to be as self-explanatory as possible.

The General Assembly Program examines the mnemonics and translates them into the corresponding absolute code of the computer. The output of the assembly program is the original source program converted to absolute code in machine readable form on punched cards, magnetic tape, or paper tape.

When a program is written, memory addresses may be specified in decimal or symbolic notation. ADO 100 means add 1 to location 100. ADO BETA means add 1 to location BETA, where the General Assembly Program automatically assigns the memory location of BETA. The programmer need only specify the starting address into which the first instruction of the program is stored.

In addition to the mnemonic code for the instructions in the normal list of instructions, the General Assembly Program uses other mnemonic codes called "pseudo-operations." A pseudo-operation is not a computer instruction but is a control instruction to the General Assembly Program. The pseudo-operation has the same form as a computer instruction, and it is listed like a normal instruction in the preparation of a program. For example, ORG is a pseudo-operation which may be used to indicate the starting address in the assignment of a program to memory. Thus, ORG 400 indicates that a program is to enter memory with the first instruction at location 400 decimal. The General Assembly Program automatically assigns succeeding memory locations to the remaining instructions of the program.

In addition to translating the mnemonics into machine language, the General Assembly Program provides the following advantages:

1. Various errors, specifically clerical errors, are detected during program assembly. This effects a substantial saving in program debugging effort, because the errors can be rectified prior to debugging.

2. The assembler generates punched cards and/or a listing on the high-speed printer that includes all error indications, the assembled program, and a complete list of symbols used, with their assigned memory locations. This provides an accurate record of the program plus helpful auxiliary information.

The Coding Sheet

The General Assembly Program coding sheet is divided into six fields: symbol, operation, operand, X, remarks, and sequence. The numbers 1 - 80 in the header information on each sheet correspond to the column numbers of a standard 80-column punched card. When a symbolic program is punched into cards, columns 7 and 21 are not used; these blank columns separate fields used in the program assembly.

SYMBOL FIELD

Columns 1 - 6 constitute the symbol field. Symbols may consist of from 1 to 6 characters. At least, one of the characters in the symbol field must be alphabetic. HOPE and CONST3 are legitimate symbols; 345 is not a legitimate symbol. A symbol may be either to the right or left in the symbol field; that is, the symbol AB in columns 1 and 2 is the same symbol as AB in columns 5 and 6. The plus and minus signs cannot be used in the symbol field, because they are used in the operand field for relative addressing. A blank (space) in the symbol field is ignored by the General Assembly Program assembler.

OPERATION FIELD

Columns 8, 9, and 10 make up the operation field. Any of the mnemonic codes for the normal computer instructions (LDA, BRU, etc.) or for the pseudo-operations (ORG, DEC, etc.) can be placed in this field. An invalid mnemonic causes an error notation during assembly.

OPERAND FIELD

Columns 12 - 19 constitute the operand field. Operands may be alphabetic or alphanumeric symbols up to six characters in length or a decimal number, and can be positioned anywhere in the operand field. A single asterisk may be placed in this field to denote reference to this instruction address. (This is equivalent to writing the same symbolic name in both the symbol and operand fields on one line.) Symbols may also consist of arithmetic combinations not to exceed eight characters of sums and differences of numbers, symbols, and asterisks. Arithmetic expressions permit relative referencing to a specified symbol (for example, *-1 which means self minus one) to reduce the number of symbols used. The plus and minus signs are used only in the operand field and only when expressing a relative address or a signed constant. The subject of relative addressing is discussed later. All numbers appearing in the operand field are considered to be decimal except when following the operation OCT, ALF, LOC, and EQO. Numbers following OCT, LOC, and EQO are assumed to be octal and are converted to their binary equivalent. Digits following ALF are converted to their binary-coded decimal (BCD) equivalents. Blanks (spaces) in the operand field are ignored, unless they follow the operation ALF or NAL.

X FIELD

Indirect addressing is specified by an X in column 20. If a character other than a blank appears in column 20, the General Assembly Program inserts the indirect address bit into the absolute instruction word being assembled. However, if the character in the X field is not an "X" or a blank, an error will be flagged by the assembly program. A blank in column 20 indicates that no indirect addressing is to be performed.

GENERAL ELECTRIC

COMPUTER DEPARTMENT, PHOENIX, ARIZONA

DATANET 30 GENERAL ASSEMBLY PROGRAM CODING SHEET

PROGRAMMER

PROGRAM

DATE

PAGE
OF

	Symbol	Opr	Operand	X	REMARKS	Sequence
	1 2 3 4 5 6	8 9 10	12 13 14 15 16 17 18 19	20	31 75	76 77 78 79 80
1						
2						
3						
4						
5						
6						
7						
8						
9						
10						
11						
12						
13						
14						
15						
16						
17						
18						
19						
20						
21						
22						
23						
24						
25						

REMARKS FIELD

Columns 25 - 75 make up the remarks field. Remarks are written in this field for reference by the programmer. These remarks are punched in the assembly program source deck, but the information is not carried through to the final object program. Thus, information in the remarks field is obtained only on a printed listing.

SEQUENCE FIELD

Columns 76 - 80 constitute the sequence field. Each card is numbered so that a deck can be sorted into proper order should the cards get out of sequence. The sequence field is not pertinent to the General Assembly Program.

Relative Addressing

The General Assembly Program provides facility for the assignment of addresses relative to some starting point (relative addressing). Assume, for example, that the symbol B is equal to memory location 0500. Using the technique of relative addressing, memory location 0510 can now be addressed by simply writing B+10 in the operand field of the coding sheet:

Symbol	Operation	Operand
B	EQU	500
	LDA	B
	.	
	.	
	.	
	.	
	LDA	B+10

The EQU pseudo-operation equates the symbol B to memory location 0500. The instruction LDA (Load Register A) loads the A-register with the contents of memory location 0500. The next LDA instruction, some program steps later, loads register A with the contents of B+10 (location 0500 + 10 = 0510).

Pseudo-Operations

In addition to the machine instructions in the DATANET-30 instruction repertoire, there are a number of pseudo-operations which facilitate programming. A pseudo-operation is not a computer instruction. It is a control instruction to the General Assembly Program in assembling a program, and it is listed the same as a normal instruction in the preparation of a program. Normally, pseudo-operations are never executed by the computer as actual instructions. Pseudo-operations are used to generate constants, to control the assembly process, or to annotate the program listing.

The various pseudo-operations are given below in alphabetical order:

ALF

ALPHANUMERIC. The first three characters in the operand field are converted to a binary-coded decimal word and assigned a memory location. Blanks are considered characters.

*

ASTERISK. If an asterisk (*) is in the first column of the symbol field, the entire card is assumed to be a remarks card and the mnemonic REM need not be specified in columns 8 - 10. This operation will have no effect on the assembled program and is used only to annotate the program listing. The complete card (columns 1 - 80) is reproduced in the program listing.

NOTE: An asterisk in any other symbol field column is illegal, if an asterisk is not in the first column.

ǂ

ASTERISK 12,7,8 ASTERISK. Slew to top of page. Causes the printing of the assembly listing to start at the top of a new page. A card with the characters *, 12-7-8, *, punched in columns 1 - 3, will be treated as a Remarks card and cause the printer to slew to the top of the next page. The character in column 2 is a multiple punch of 12,7,8.

BSS

BLOCK STARTED BY SYMBOL. Increases the memory allocation counter in the General Assembly Program by the number specified in the operand field. It is used to reserve a block of memory locations. The operand may be decimal or symbolic. If decimal, the number is converted to binary. If symbolic, the symbol used must be predefined. The BSS operation may be used as often as desired.

$_ _

DOLLAR SIGN. When the $ character is used as the leading character of a symbol, and the symbol is referenced by an instruction, the General Assembly Program automatically inserts memory addressing mode 3 into the instruction word, divides the address of the $_ _ symbol by 16, and inserts the resultant address into the instruction word. The absolute address of the $_ _ symbol (initially assigned by the programmer) must be less than 8191 and modulo 16. (For further information see Chapter III.)

DDC

DOUBLE LENGTH DECIMAL. Used to enter decimal constants larger than 131,071 or, in other words, a constant larger than can fit into one word. The decimal constant is assigned two sequential memory locations starting with the first available even location, and with the least significant half in the odd location. If no binary scale is specified, the assembly program assumes a binary scale of 35.

DEC

DECIMAL. Used to enter a decimal constant in the object program and to convert it to binary. The constant is assigned one memory location. The operand may be symbolic or decimal. If the operand is symbolic, at least one character must be other than 0 through 9, +, -, ., B, or E. Leading zeros are ignored and the number right justified.

END

END OF PROGRAM. Causes the assembly program to generate an instruction that transfers control to the location specified in the operand field when the object program is executed. The operand may be decimal or symbolic. If decimal, the operand is converted to binary. If symbolic, the symbol must be predefined. In addition, the END operation signifies end-of-program and terminates assembly. This operation may be used only once and must be the last instruction of the source program. If no END operation is used, an error comment will result but assembly will be terminated by the end-of-deck condition. The X field of an END operation is not used by the assembly program.

EQO

EQUALS OCTAL. Performs the same function as the EQU operation, but the content of the operand field is assumed to be an octal number.

EQU

EQUALS. Used to overrule the normal memory assignment performed by the assembly program. The operand may be decimal or symbolic and specifies the memory location to be used by the assembly program. If the operand is decimal, it is converted to binary. If symbolic, the symbol used must be predefined. The EQU operation may be used as often as desired and at any point in the source program. This operation has no effect on the memory allocation register in the assembly program, so that the normal memory assignment by the assembly program continues in sequence.

IND

INDIRECT ADDRESS. Used to generate a constant, where the constant is a memory address. The operand may be symbolic or numeric. If numeric, it is assumed to be a decimal number and is converted to binary. If symbolic, the address of the symbol is used.

INA

INDEX BY A-REGISTER. Similar to IND except that a bit is set in this word so that when it is used as an indirect address, the contents of the A-register will be added to the memory address portion of this word.

INB

INDEX BY B. Same as INA except that the B-register is used instead of the A-register.

INC

INDEX BY C. Same as INA except that the C-register is used instead of the A-register.

LOC

LOCATION IN OCTAL. Performs the same function as the ORG operation but the contents of the operand field are assumed to be in octal form.

NAL

NEGATIVE ALPHANUMERIC. Used to enter the 2's complement of an alphanumeric constant in the object program.

OCT

OCTAL. Used to enter an octal constant in the object program. The octal number in the operand field is converted to binary (right-justified) and assigned one memory location determined by the memory allocation register. The assembly program ignores leading zeros in the operand field. If fewer than six digits are provided for the operand field, the assembly program will right justify the digits. A leading plus or minus sign in the operand field will set the leading bit of the constant to 0 or 1.

ORG

ORIGIN. Establishes the starting location in memory of the program. The assembly program begins assembly of the object program as specified by ORG. One ORG card is required at the beginning of each assembly run. If no ORG card is included, the assembly of the program automatically begins at location 0000. Any number of ORG cards may be used in one assembly. The number following ORG must be in decimal.

REM

REMARKS. Lines identified by REM in the operand field are used to annotate the program listing. These lines are not assigned memory locations in the assembly program. The complete card, columns 1 - 80, is reproduced in the program listing.

TCD

PUNCH TRANSFER CARD. Generates an instruction that transfers control to the location specified in the operand field when the object program is being loaded. The operand may be decimal or symbolic. If decimal, the address is converted to binary. If symbolic, the symbol used must be predefined. TCD may be used as often as desired in the source program. This operation has no effect on the memory allocation register, so that the memory assignment by the assembly program will continue in sequence.

ZXX - The Z is followed by 2 octal digits. These digits become the operation code portion of the generated word (instruction). The operand is computed normally and is assumed to be either symbolic, decimal numeric, or a combination.

Assembly Errors and Suspected Errors

The following codes listed are errors or suspected errors found during assembly by the General Assembly Program. The objective is to convey as much error information as possible to the programmer.

Except for machine malfunctions, the computer will stop only under three circumstances, during assembly:

1. The number of special symbolic operands exceeds the size of the symbol table (symbolic table overflow).

2. The total number of symbols exceeds the size of the symbol table.

3. During the final phase of assembly, a name appearing in the symbol field cannot be found in the symbol table (lost symbol).

When these errors occur, an indicative typeout results and the computer goes into a programmed loop. However, if desired, switch 19 may be manually set, and assembly will continue. The result of forcing the assembly to continue is:

1. The special symbolic operands encountered after the error halt are not entered in symbol table 1. This may result in the improper assignment of a memory address to these symbols in the following phases.

2. The symbols following the error halt and are not entered into symbol table 2. This will result in the detection of undefined symbols during the final phase.

3. Assembly will continue. If the symbolic name was a special operand, the assignment of memory locations to the instructions following the error halt may be out of phase with the numeric assignment performed by the previous phase of the assembly.

Error Codes

Following is a list of the error codes:

Code

Ø Illegal Mnemonic Operation

This becomes a HLT (00).

U Undefined Symbol

A symbol name appearing in the operand field does not appear in the symbol field of any instruction. Constant 0000 is inserted as an operand address.

M Multiply Defined Symbol

Either the symbol field or the operand field contains a symbolic name which appears in the symbol field of two different instruction lines. If the error detected was in the symbol field, assembly will continue with the present setting of the memory allocation register. If the error detected was in the operand field, the value assigned to the symbol the last time it appeared will be used as the operand address in the assembled instruction.

A Error or Suspected Error in the Operand Address

Blank operand field in a line normally requiring an address. An entry in the operand field of a line which normally should be blank. The numeric value of the operand does not meet the requirement of the line in which it was used. The value of the operand address will be logically OR-ed into the instruction.

T Error or Suspected Error in X-Field

The X-field contains an entry in an instruction which does not access memory. The X-field contains any character other than X or is a numeric.

S Scale Factors in DEC

The specified binary and decimal scales are incompatible. Two decimal or binary scales have been specified in the constant line.

$ Channel Table Usage

The $ character in the first position of a symbol indicates to DATANET-30 General Assembly Program that this is to be treated specially. This symbol must be assigned by the programmer to a memory location that is a multiple of 16_{10}. If this error tag appears, it means that either the specified address was not module 16 or less than 8192 or both.

APPENDIX A

DATANET-30 GENERAL ASSEMBLY PROGRAM

DATANET-30 SYSTEMS TAPE

The DATANET-30 systems tape contains the following programs:

1. DATANET-30 General Assembly Program
2. General Assembly Program 3 (paper tape conversion)
3. General Assembly Program 4 (magnetic tape updating).

The programs above are linked together in this order:

1. General Assembly Program 4
2. DATANET-30 General Assembly Program
3. General Assembly Program 3.

To run General Assembly Program 4, it is necessary to call the program from the systems tape with a "call General Assembly Program 4" card.

The DATANET-30 General Assembly Program may be run:

1. After General Assembly Program 4 (with SW1 and SW16 down)
2. By itself (using DATANET-30 General Assembly Program call card).

General Assembly Program 3 may be run:

1. After DATANET-30 General Assembly Program
2. By itself (using General Assembly Program 3 call card).

General Assembly Program 4 is used to update symbolic source programs. It updates magnetic tape by comparing the sequence number of a card to the sequence number of a record on tape and inserts the card in the correct position.

General Assembly Program 3 is used to convert punched cards or magnetic tape to punched paper tape and will punch paper tape in either of two formats, depending on the console switch setting.

DATANET-30 GENERAL ASSEMBLY PROGRAM

DATANET-30 General Assembly Program has operating procedures identical to those for the GE-225 General Assembly Program.

The DATANET-30 General Assembly Program operates either from cards or a systems tape. When the General Assembly Program is loaded from cards, it is referred to as a Card General Assembly Program. When the assembly program is loaded from the systems tape, it is referred to as a Tape General Assembly Program.

Since cards or tape may be used as the input/output medium, there is a separate set of operating instructions for card assembly programs and tape assembly programs.

The General Assembly Program is made up of three separate programs: pass 0, pass 1, and pass 2. The input to pass 0 is the symbolic program. The output from pass 0 plus the output from pass 1 is the input to pass 2. The output from pass 2 is the assembled program.

Cards or tape may be used as the input and output for all passes. Whether cards or tape are used determines the setting of console switch 4. A flow diagram of the three assembly programs is shown in Figure A-1.

The minimum hardware requirements for the operation of the DATANET-30 General Assembly Program are:

1. Card reader
2. Card punch or magnetic tape
3. Typewriter
4. 8192 words of memory.

Options and Console Switches

The console switches (A-register input switches) of the GE-225 are used to indicate the peripheral configuration available while using the General Assembly Program. All switches, with the exception of switch 19, should be set initially and remain the same through all passes of the assembly program.

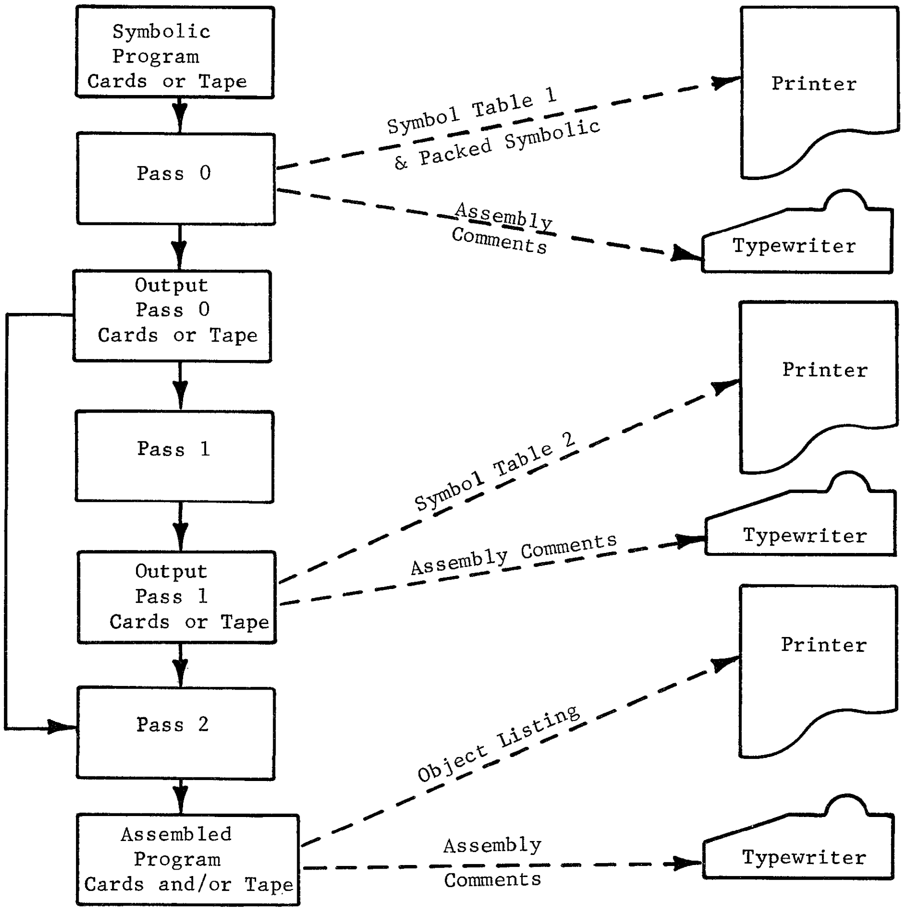

Symbolic Program Cards or Tape
Pass 0
Output Pass 0 Cards or Tape
Pass 1
Output Pass 1 Cards or Tape
Pass 2
Assembled Program Cards and/or Tape
Symbol Table 1 & Packed Symbolic
Printer
Assembly Comments
Typewriter
Symbol Table 2
Printer
Assembly Comments
Typewriter
Object Listing
Printer
Assembly Comments
Typewriter

Figure A-1. Flow Diagram of the DATANET-30 General Assembly Program

Switch 2

Normal: Printer is on line.

Down: No on-line printer. An octal program deck is punched instead of a binary program deck.

Switch 3

Normal: Tape 3 is used to print comments on the assembly program pass 2 program listing.

Down: Comments are omitted from the assembly program pass 2 program listing.

Switch 4

Normal: Tapes 4 and 5 are used as output/input to assembly program passes 0, 1, and 2, respectively.

Down: Cards instead of tapes are used as output/input to passes 0, 1, and 2, respectively. (Switch 4 down overrides switches 3 and 6.)

Switch 6

Normal: The binary program output is not written on tape 6.

Down: The binary program output from pass 2 is written on tape 6.

Switch 7

Normal: Ignored

Down: Go To Assembly 3 upon completion of General Assembly.

Switch 9

Normal: Card punch on line.

Down: No card punch on line.

Switch 14

Normal: No packed symbolic listing.

Down: Packed symbolic listing.

Switch 15

Normal: Ignored by the assembly program.

Down: Symbolic program deck is written on tape 3 before any processing is done by the assembly program.

Switch 16

Normal: Input to pass 0 is the symbolic program card deck.

Down: Input to pass 0 is on tape 3. Tape 3, the comments tape, may be changed instead of the symbolic card deck through an updating routine before making a second assembly.

Switch 18

Normal: Types or prints "no reference symbols" after pass 0.

Down: Suppresses the typing or printing of "no reference symbols" after pass 0.

Switch 19

Toggling of switch 19 bypasses "symbol table overflow" stop during passes 0 and 1, and "symbol lost" stop during pass 2.

Switch Combinations and Requirements

The following table shows different switch combinations and their requirements:

Programs	Switch Down	Comment Tape #3	Working Tape #4 & #5	Binary Program Tape #6	Punch On-Line	Printer On-Line
CARD	2 & 4	No	No	No	Yes	No
General Assembly	4	No	No	No	Yes	Yes
Program						
TAPE	None	Yes	Yes	No	Yes	Yes
General Assembly	2	Yes	Yes	No	Yes	No
Program	3	No	Yes	No	Yes	Yes
	6	Yes	Yes	Yes	Yes	Yes
	6 & 9*	Yes	Yes	Yes	No	Yes
	3 & 6	No	Yes	Yes	Yes	Yes
	3,6 & 9*					

* When switch 9 is used, switch 6 also must be set, because the punch is off line and the output must be written on tape.

When switch 16 is used, and switch 3 is in normal position, the input to pass 0, the original symbolic deck, is read from tape 3 instead of cards.

The other switches pertain to format and may be used at the discretion of the programmer, providing the hardware is available.

Card General Assembly Program Operating Instructions

GENERAL ASSEMBLY PROGRAM PASS 0

The procedure for pass 0 is as follows:

1. Set up input deck starting with the pass 0 binary deck, followed by symbolic program, followed by one blank card.

2. Set console switches as desired.

3. Load cards in card reader; depress LOAD CARD, RESET ALARMS, and RESET P; place processor in AUTO mode; and depress START.

4. If switch 4 is down, the output from pass 0 will be punched cards. These cards must be arranged in the order described on a later page. This output will also be listed if the printer is on line. In addition, a packed list of special symbols (which may be suppressed by a switch setting), a list of undefined symbols, a list of multiple symbols, and the symbolic names which are not referenced in the program are printed. If no high-speed printer is available on line, the above lists will be typed on the typewriter.

5. Messages:

NO END CARD — Indicates the symbolic deck does not terminate with an end card. Assembly will continue to the normal end of job.

END OF PASS 0 — Signifies the end of assembly program run 0.

SYMBOL TABLE OVERFLOW 1 — Indicates the number of special symbolic operands exceeds 250. Program goes into loop which may be overridden by setting switch 19. This causes pass 0 to continue but all special symbols following this loop are not placed in the table.

SYMBOL TABLE OVERFLOW 2 — Indicates the total number of symbols exceeds 1000. The program goes into a loop which may be overridden by setting switch 19. Pass 0 then continues but symbols following the loop are not analyzed as undefined, multiple, or no reference symbols.

CARD READ ERROR

Action Required:

a. Place computer in manual mode.

b. Backspace card reader by removing the cards from the hopper and the card from the read platform. Place these cards in front of the deck, replace the deck in the card hopper, and place the last card read in the read platform.

* Special symbolic operands which are referred to by double length instruction.

	c. Press A—▶I button. d. Place processor in AUTO mode and depress START.
UNDEFINED SYMBOLS	A symbol or symbols were referenced but were not defined.
MULTIPLE SYMBOLS	A multiply-defined symbol in the symbol or operand fields.
NO REFERENCE	No reference was made to the symbols following this message.
XXX ERRORS TAPE 3	If tape 3 is used for comments, this typeout signifies the number of bad spots on tape 3. (Switch 3 in normal position.)
XXX ERRORS TAPE 4	Signifies the number of bad spots on tape 4. (Switch 4 in normal position.)

6. Action required for all other stops:

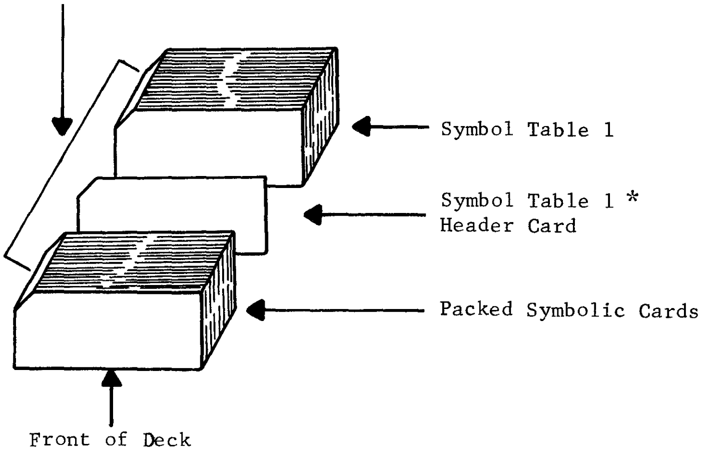
   a. Place processor in MANUAL mode.

   b. Check the CARD PUNCH READY indicator. If punch is not in ready status, place in ready status and depress START.

   c. Check the N-REGISTER READY light. If not in ready status, manually type spaces until N-register becomes ready and depress START. If the CARD READER alarm indicator is on, check the card deck for damaged cards. Replace if necessary and reload the program from the beginning.

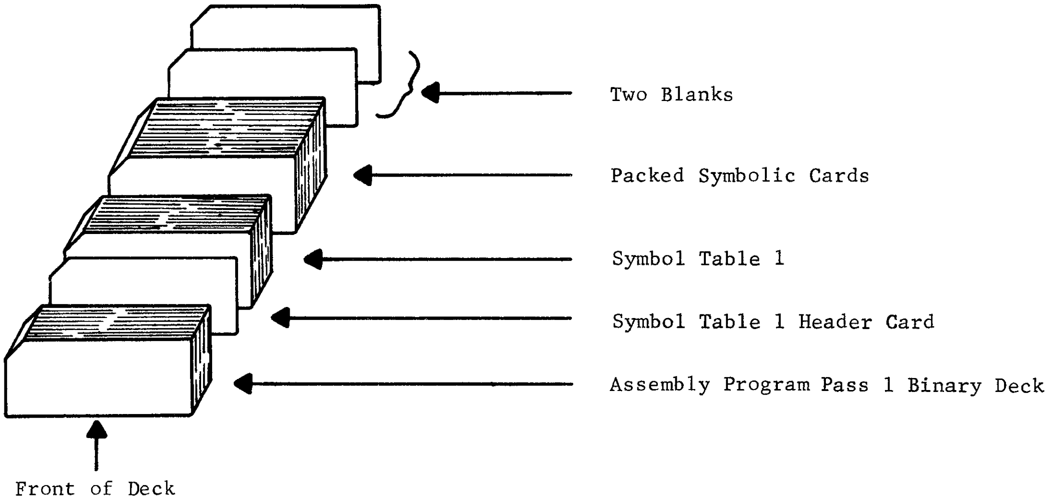
OUTPUT FROM GENERAL ASSEMBLY PROGRAM PASS 0

The output from pass 0 is shown below:



* The symbol table 1 header card may be recognized by the Hollerith character ST1 punched in columns 1, 2, and 3. "S" in Hollerith code is a 0-2 punch. "T" is a 0-3 punch. The code for "1" is a 1 punch.

Rearrange the output from assembly program pass 0 as shown below:


Two Blanks
Packed Symbolic Cards
Symbol Table 1
Symbol Table 1 Header Card
Assembly Program Pass 1 Binary Deck
Front of Deck

GENERAL ASSEMBLY PROGRAM PASS 1

The procedure for pass 1 is as follows:

1. If console switch 4 is down, the output from pass 0 is a packed program with sequence numbers starting with 20000 (columns 74 - 78) followed by a table of special symbolic operands with sequence numbers starting at 10000 (columns 74 - 78). The cards that have sequence numbers beginning with 1XXXX (columns 74 - 78) should be placed in front of those cards starting with 2XXXX (columns 74 - 78) prior to combining them with the pass 1 binary deck followed by the rearranged output from pass 0, followed by two blank cards. (See Figure A-2.)

   If console switch 4 is in normal position the output from pass 0 is written on tapes 4 and 5. In this case only the pass 1 program and two blanks are loaded into the card reader.

2. Load cards. (Same as #3 page A-6)

3. If switch 4 is down, the output from pass 1 is a sorted table of symbols and equivalent locations, which is punched out. If the printer is on line, these are listed on the high-speed printer. In addition, a list of all multiply-defined symbols, together with all of the equivalent values associated with each symbol, is printed (or typed if no printer is available).

   If switch 4 is in normal position, the output from pass 1 is written on tapes 4 and 5, in which case, only the pass 2 program and two blanks are loaded into the card reader.

Errors or possible errors detected in the operand field of a BSS, EQU, or ORG instruction are printed or typed with the present setting of the memory allocation register, the card type, and the error code. The error codes are:

U - an undefined symbol
A - a possible error in an address

4. Messages:

   NO END CARD
   Indicates the symbolic deck does not terminate with an end card. Assembly continues to the normal end of job.

   MULTIPLE SYMBOLS
   Indicates a multiply-defined symbol in the symbol or operand fields.

   END OF PASS 1
   Signifies the end of the assembly program pass 1 run.

   SYMBOL TABLE OVERFLOW
   Messages (and action to be taken) are the same as for the General Assembly Program 0 run.

5. For all other errors, repeat previous load procedure described for pass 1.

GENERAL ASSEMBLY PROGRAM PASS 2

The procedure for pass 2 is as follows:

1. The input for pass 2 is the output from pass 0 and pass 1. If console switch 4 is down, set up the input deck as follows:

   Assembly program pass 2 binary deck followed by the output of pass 1 followed by the rearranged output from pass 0. (See Figure A-2.)

2. Load cards.

3. Messages:

   ERRORS
   Indicates presence of a real or suspected source program error.

   NO ERRORS
   Indicates no errors were found.

   END OF PASS 2
   Signifies the end of the pass 2 run.

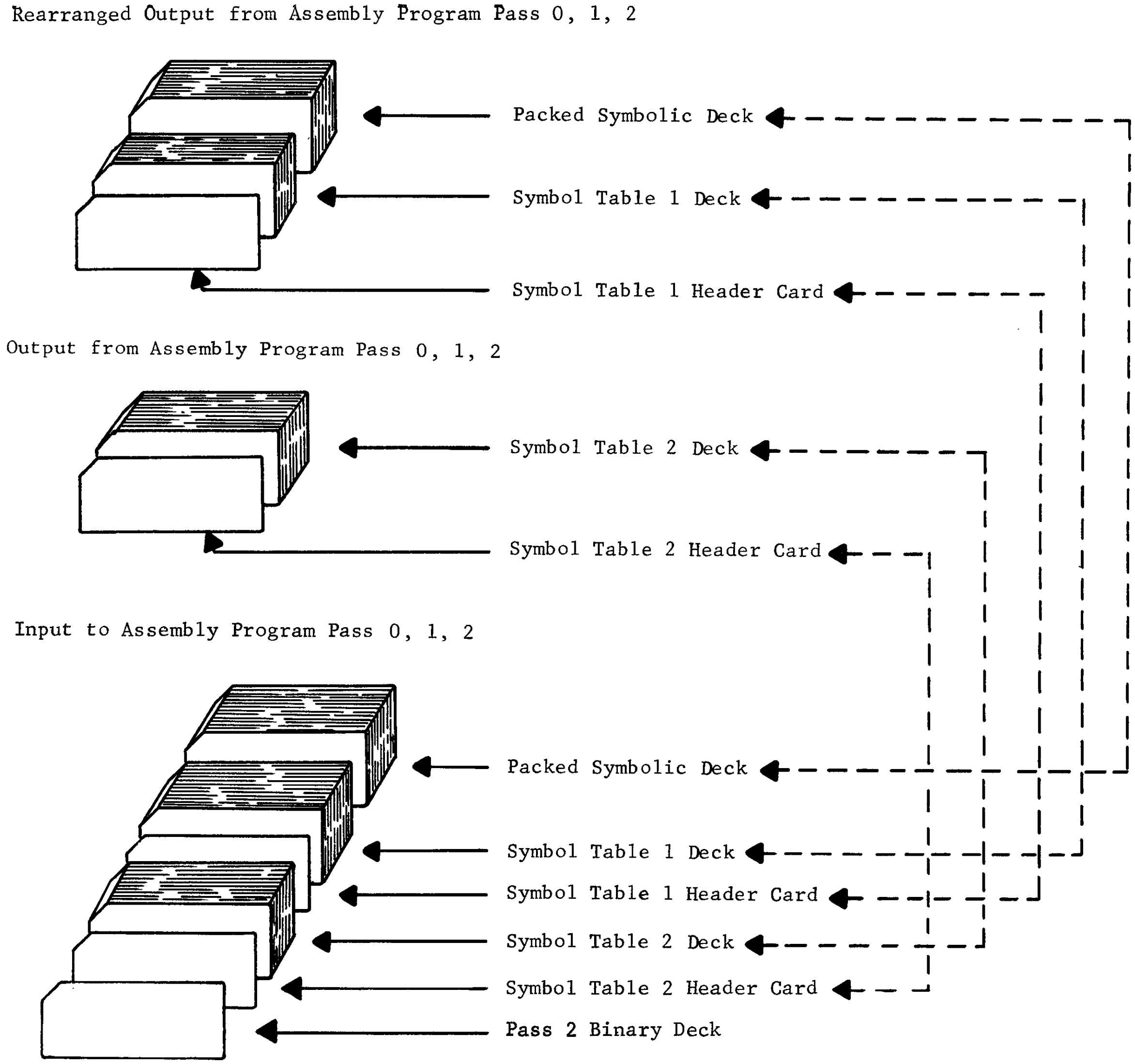
   SYMBOL LOST
   Is typed with the setting of the memory allocation register and the symbol in question when a symbol appearing in the symbol field cannot be found in the symbol table. This is caused by a machine error and may necessitate a reassembly. Action required for SYMBOL LOST:

   a. List all output from pass 0.

b. Correct cards as necessary.

c. Restart assembly at assembly program 0, 1, or 2, as required.

4. The output from pass 2 is an octal punched card deck, if no printer is on-line, or a printer listing, if a high-speed printer is available, and binary cards. The listing (or octal cards) contains the octal memory location assigned to the instruction in octal, the symbolic instruction, and the codes for real or suspected errors in the instruction.



Rearranged Output from Assembly Program Pass 0, 1, 2
Packed Symbolic Deck
Symbol Table 1 Deck
Symbol Table 1 Header Card
Output from Assembly Program Pass 0, 1, 2
Symbol Table 2 Deck
Symbol Table 2 Header Card
Input to Assembly Program Pass 0, 1, 2
Packed Symbolic Deck
Symbol Table 1 Deck
Symbol Table 1 Header Card
Symbol Table 2 Deck
Symbol Table 2 Header Card
Pass 2 Binary Deck

Figure A-2. Arrangement of Input for Pass 2

DATANET-30 General Assembly Program may be modified for 4k memory GE-225's to accept up to 500 symbols.

These binary corrections are listed below and should be inserted before assembly program 0:

	LOCATIONS	CONTENTS
Assembly Program 0	00053_8	764_8
	00054_8	7776_8

This binary correction card should be inserted before the assembly program 1 transfer card.

	LOCATIONS	CONTENTS
Assembly Program 1	01110_8	764_8

Tape General Assembly Program Operating Instructions

To run the Tape General Assembly Program:

1. Mount the DATANET-30 General Assembly Program systems tape on handler 1. Mount working tapes on handlers 3, 4 and 5, with the write-permit rings in place. If console switch 6 is set, mount a working tape on handler 6.

2. Load the input deck into the card reader. It should be set up as follows:

   a. DATANET-30 General Assembly Program call card.

   b. Symbolic program to be assembled. (If symbolic program is on magnetic tape, mount the tape on handler 3 and set console switch 16.)

   c. Two blank cards.

3. Depress RESET ALARM and RESET A, LOAD CARD, and RESET P. Depress AUTO and START.

The assembly program will be called in and will run from start to completion. Error messages are the same as described in preceding pages.

GENERAL ASSEMBLY PROGRAM 3 - PAPER TAPE CONVERSION

Assembly program 3 (see flow chart in Figure A-3) is a magnetic tape or cards-to-paper-tape conversion program. It may be run from the systems tape or by loading the assembly program 3 program from punched cards. The minimum hardware requirements are as follows.

1. Card reader or magnetic tape
2. Paper tape punch
3. 4096 words of memory
4. Typewriter.

Processing

The input to assembly program 3 may be magnetic tape, binary cards, or octal cards. Type of input (cards or tape) is determined by console switch setting. The format of paper tape output is determined by switch setting.

Assembly program 3 examines the console switch settings and types a message to the operator instructing him to set the paper tape punch to the mode specified by the console switches. After acknowledgment by the operator of correct paper tape mode, the program punches a leader of 18 inches.

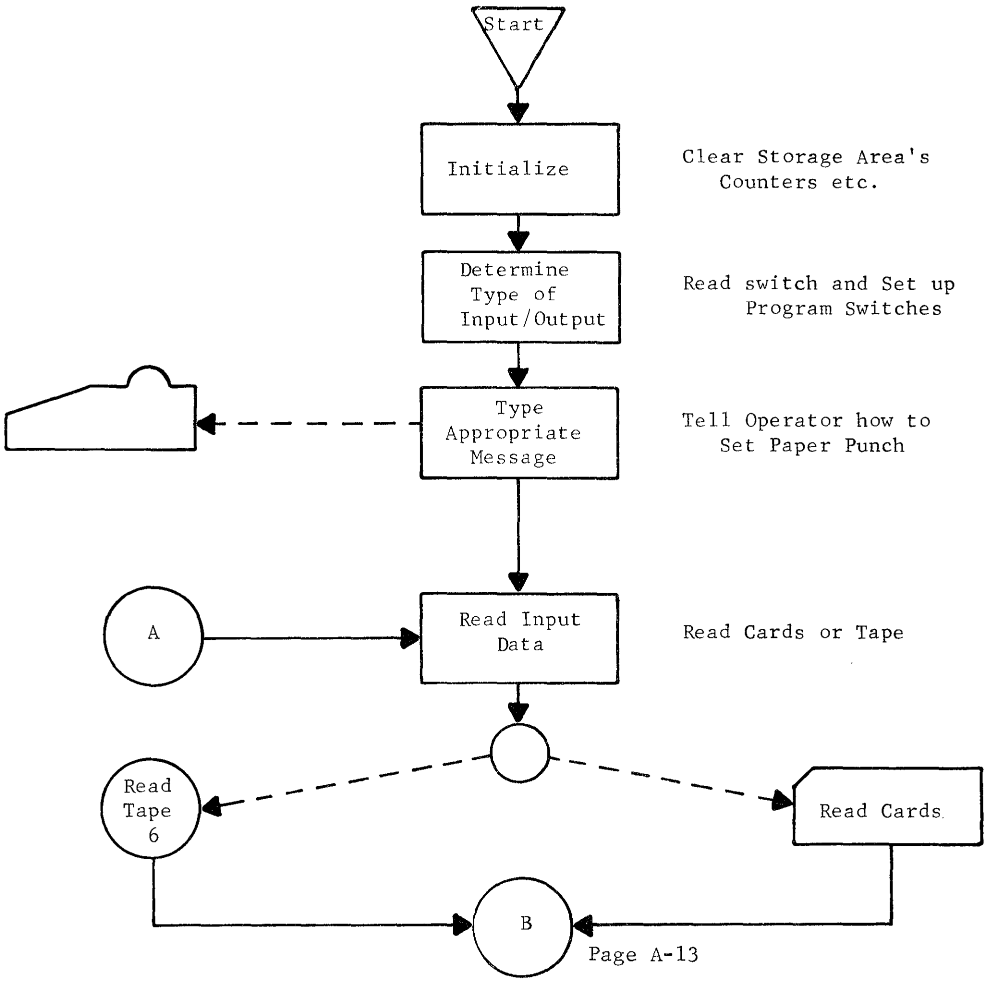

Start
Initialize
Clear Storage Area's Counters etc.
Determine Type of Input/Output
Read switch and Set up Program Switches
Type Appropriate Message
Tell Operator how to Set Paper Punch
A
Read Input Data
Read Cards or Tape
Read Tape 6
Read Cards
B
Page A-13

Figure A-3. Flow Chart for Assembly Program 3

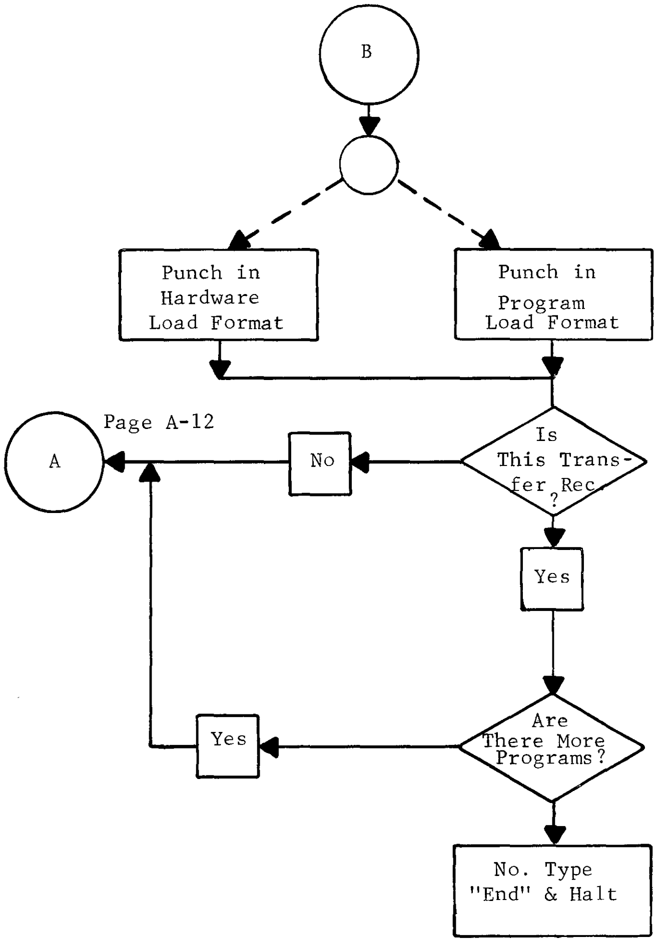


Figure A-3. Flow Chart for Assembly Program 3

Then assembly program 3 reads in the input to be punched, and punches it in the desired mode. Punching is continuous until a transfer cardor tape end-of-file is detected at which time assembly program 3 terminates the punching of data.

The program then tests to see if there is more data to be punched in either the card reader or tape unit; if so, it is punched in the specified format. If no more data is to be punched, the program punches 18 inches of trailer, types "end" messages, and terminates.

For paper tape punches in hardware load format or program load format, it is necessary for the system to have a free-standing paper tape unit with the 8-level straight transfer mode feature (Model 4WGA652).

Assembly Program 3 Console Switch Settings

The switch settings for assembly program 3 are as follows:

Switch 6

Normal: Input to assembly program 3 is on cards.

Down: Input to assembly program 3 is on tape 6, plug 1.

Switch 7

Normal: Ignored.

Down: Read in assembly program 3 from systems tape program after completion of DATANET-30 General Assembly Program.

Console switches 10, 11, and 12 define the mode in which the output is to be punched. No other modes exist at this time.

FORMAT	SWITCHES 10	11	12
Hardware Load Format	Norm	Norm	Norm
Program Load Format	Norm	Norm	Down

Operating Instructions

If a DATANET-30 Systems Tape is available, mount the systems tape on handler 1, controller 1, and assemble deck as follows:

1. Place assembly program 3 call card in the card reader followed by deck to be punched. If input is on tape, mount the tape on handler 6 and set console switch 6.
2. Two blank cards.
3. Set console switch as desired for mode.
4. Depress LOAD CARD, RESET P, AUTO, and START.

If a DATANET-30 Systems Tape is not available, assemble deck as follows:

1. Place the assembly program 3 program on cards on the card reader followed by the deck to be punched. If input is on tape mount the tape on handler 6 and set console switch 6.

2. Two blanks.

3. Set console switches as desired.

4. Depress LOAD CARD, RESET P, AUTO, and START.

If assembly program 3 is to be run following a DATANET-30 General Assembly Program, the DATANET-30 Systems Tape must be on tape 1:

1. Set console switch 7. This causes assembly program 3 to be read in after completion of the DATANET-30 General Assembly Program.

2. Set console switches as desired for proper mode.

3. Set console switch 6. This writes the ouput of DATANET-30 General Assembly Program on tape 6.

4. Run the General Assembly Program as previously described.

GENERAL ASSEMBLY PROGRAM 4

General

Assembly program 4 is a magnetic tape generating and updating routine. It may be used to make the symbolic source tapes input to the DATANET-30 General Assembly Program. Assembly program 4 is included on the DATANET-30 Systems Tape and may be called in and executed with the assembly program 4 call card, or it may be loaded from punched cards.

The minimum systems configuration required is:

1. 8192 words of memory
2. Typewriter
3. High-speed printer
4. Magnetic tape controller with two handlers
5. Card reader.

Control Cards

The following control cards are used:

NEW	Characters N - E - W punched in columns 8, 9, and 10.
FIN	Characters F - I - N punched in columns 8, 9, and 10.
DEL	Characters D - E - L punched in columns 8, 9, and 10.

There are two types of DEL control cards:

1. Range delete

Columns 8 - 10	Columns 12 - 16	Columns 76 - 80
DEL	(TO)	(FROM)

Records on the old master starting with sequence number (FROM) to record starting with sequence number (TO) are deleted.

2. Single delete

Columns 8 - 10	Columns 76 - 80
DEL	(THIS)

Record on old master with sequence number (THIS) is deleted.

Action on Detecting Control Cards

The action below takes place upon detection of control cards:

1. NEW - The remaining cards in the card reader are written on tape 3 (new master) until a FIN card is detected.

2. FIN - Signifies to assembly program 4 that there are no more cards to read. If any records exist on the old master they are copied to the new master with new sequenced numbers inserted. Old master and new master tapes are closed and rewound (new master with end-of-file record). Program is terminated.

3. DEL - Range - Old master is copied to new master with new sequence numbers until columns 76 - 80 of DEL card are equal to columns 76 - 80 of old master. Old master is then searched until columns 76 - 80 of old master are greater than columns 12 - 16 of DEL cards. Another card is read in and processing continues.

   DEL - Single - Old master is copies to new master with new sequence numbers until columns 76 - 80 of DEL card are equal to columns 76 - 80 of old master. Another card is read in and the old master is advanced to the next record. Processing continues.

4. All other cards are assumed to be updating cards. They are inserted according to their sequence number. If the sequence number of an input card is equal to the sequence number of a record on the old master, the input card will replace the old master record.

NOTE: All input cards including control cards must be in sequence columns 76 - 80. Any card out of sequence will be ignored and error flagged. All input decks must end with a FIN card (no sequence number needed) and two blanks.

Operating Procedure

The operating procedure for assembly program 4 is as follows:

1. Mount the DATANET-30 Systems Tape on handler 1, plug 1. If General Assembly program 4 is to be run from cards, place General Assembly Program 4 program in the card reader.

2. Place old master tape to be updated on handler 2, plug 1. If a new tape is to be generated, place a working tape on handler 2, plug 1.

3. Place a good tape with a write-permit ring on handler 3, plug 1. This is the new master.

4. If General Assembly Program 4 is to be run from systems tape, place a General Assembly Program 4 call card followed by the updating deck in the card reader. If General Assembly Program 4 is to be run from cards, place the updating deck behind the General Assembly Program 4 program deck.

5. Depress LOAD CARD, RESET ALARM, RESET P, and START.

6. If DATANET-30 General Assembly Program is to be run following General Assembly Program 4, place the following console switches down:

   Switch 1. This calls in DATANET-30 General Assembly Program after completion of General Assembly Program 4.

   Switch 16. This switch is pertinent to DATANET-30 General Assembly Program only. It indicates that the source program is on tape 3.

For other switch settings see DATANET-30 General Assembly Program operating instructions.

Memory Addressing Using the General Assembly Program

The previous discussion has centered on describing the memory addressing features built into the DATANET-30. This section will describe the memory addressing features built into the General Assembly Program.

The General Assembly Program instruction mnemonics and pseudo-operations provide a technique for program preparation. This is particularly true with respect to memory addressing, since the General Assembly Program does a great deal of the generation and validity checking of addresses.

The General Assembly Program will interpret an asterisk (*) in the operand field on input data to mean the address of that instruction.

Location	Instruction
05000	LDA *+ 10

In this example, * = 05000 and the relative address *+10 will be 05010.

The * serves as a flag to the General Assembly Program and causes the performance of a special calculation to generate the desired address.

The assembly program is also flagged by the character X in the "X" column. This indicates that indirect addressing is desired on that instruction. The assembly program generates the desired address according to the standard rules and then adds a 1-bit in I (12). One other special requirement must be flagged to the assembly program by the programmer. When it is desired to use channel table addressing, a symbolic operand <u>must</u> be used and the symbol must start with the character $ (dollar sign). The assembly program, upon finding this condition, will assign addressing mode 3 (channel table addressing) by making I (10-11) = 11. It then checks the location of the symbol, verifies that it is less than 8192 and that it is a multiple of 16 (that the low order 4 bits are all zero), divides the location by 16 and inserts the remaining 9 significant bits in the instruction.

To use this mode properly the programmer must do what is done when using any other symbolic address except that the symbol must start with a $ sign, and must be in a modulo 16 address in the first 8192 words of memory.

The two remaining techniques for specifying the desired address are pure symbolic and decimal. Examples of these are:

```
LDA      CONST3
LDA      WS1
LDA      5
LDA      511
LDA      8000
```

CONST 3 and WS1 are symbolic addresses; and 5, 511, and 8000 are decimal addresses. The General Assembly Program checks the desired address, if it is in the same program bank as the instruction being assembled. If it is, address modification mode 0 or 1 (program bank addressing) is assigned along with the correct partial address. If it is not in the same program bank, it is checked for being in the common data bank. If it is, address modification mode 2 (common data bank addressing) is assigned along with the correct partial address. If neither case applies, it is not possible to generate the address directly. The assembly program flags this condition with an A on the assembly program output listing. This indicates an invalid address and must be corrected.

With program banks of 1,024 words, most desired addresses will be either in the common data bank or in the same program bank. The first assembly by the General Assembly Program will indicate the addresses which need to be changed to indirect addressing.

APPENDIX B

CHARACTERISTICS SUMMARY

COMMUNICATIONS PROCESSOR

Single address
Stored program
Read/compute/write cycle
Binary
18 bit word length
Parallel
128 buffer selector channels
Automatic program reload
Memory interrupt feature
Automatic bit buffer scan command
Elapsed time program interrupt counter
78 basic instructions
Indirect addressing
Indexing
6.94 microsecond word time

MEMORY

6.94 microsecond memory cycle
Memory size (words):

4,096
8,192
16,384

HARDWARE SCAN

Bit buffer units only
5-, 6-, 7-, or 8-level codes
Scan time: 21 microseconds per simplex, half-duplex, or full-duplex channel.

INSTRUCTION SUMMARY

		Time in Microseconds
Load	Single and double word	14 and 21 *
Store	Single and double word	14 and 21
Arithmetic	18 bit parallel addition	14
Logical	AND, OR and EXCLUSIVE OR	14
Branch	Conditional and unconditional	7
	To subroutine	21
Register Transfer		7

BUFFER SELECTOR BUFFER UNITS

Bit Buffer Unit

10 simplex channels input and 10 simplex channels output/module
10 half-duplex channels/module
10 full-duplex (or echoplex) channels/module

Module data rates (bits/sec)

45
50
56.25
75
110
150

Code level: 5, 6, 7, or 8 bits/character

Character format: start/stop bit asychronous;
one stop bit (minimum).
Compatible digital subsets: 103A; 103B.
20 ma d-c loop or bipolar voltage interface.

Character Buffer Unit

2 simplex channels/module
2 half-duplex channels/module
~~1 full-duplex channel/module~~

* For ease of computation the 6.94 u sec memory cycle is rounded to 7.0 u sec.

Channel data rates

300 bits/sec to 3000 bits/sec

Code level: 5, 6, 7, or 8 bits/character

Character format: start/stop bit asychronous;
one stop bit (minimum).
Compatible digital subsets: 202A; 202B.
Bipolar voltage interface.

Word Buffer Unit

2 simplex channels/module
2 half-duplex channels/module
1 full-duplex channel/module

Channel data rates (bits/sec)

1200
1800
2000
2400
3000

Code level: 20 bits

Character format: start/stop bit asychronous;
one stop bit (minimum).
Compatible digital subsets: 202A; 202B.
Bipolar voltage interface.

Receive Character Parallel Buffer Unit

2 units/module
Up to 14 bits parallel
Receive only
Answer back capability
Up to 13,000 characters/second
Compatible digital subsets: 401B; 401F; 402B.

CONTROLLER SELECTOR UNIT

Maximum transfer rate	28,800 words/sec
Data transfer cycle time	17.34 microseconds
DATANET-30 memory interrupt time	7 microseconds/word
Execute status request	28 - 70 microseconds

Peripheral Combination Chart:

Peripheral	Possible Address	Load Factor Per Peripheral
Single access DSU	0, 1	.55
Dual access DSU	0, 1	.55
15 kc tape controller	2, 3, 4, 5	.1
41.5 kc tape controller	2, 3, 4, 5	.28
Computer interface unit	2, 3, 4, 5	#

The load factor represents the index for peripherals that may be run concurrently if sum of load factors does not exceed 1.00.

APPENDIX C

CIU-930 COMPUTER INTERFACE UNIT

GENERAL

The CIU-930 provides the interface for the DATANET-30 and a Compatibles/200 information processing system and is used to transfer 21-bit words between them. The words are transferred in parallel. The CIU-930 connects into any channel of the DATANET-30 buffer selector in the same manner as any other DATANET-30 buffer. On the processor side, the CIU-930 connects into any priority control channel. The buffer selector address of the CIU is specified by the wiring of the buffer selector address plug for the CIU module. There is no DATANET-30 hardware restriction on the number of CIU's which may be used, other than the physical space occupied. Each CIU-930 occupies one module. The CIU is asynchronous. It has no service rate and is program controlled.

CIU-930 INSTRUCTIONS

Following are the CIU-930 instructions:

Register Transfer	TRAR, B

Five things are accomplished (see illustration below):

1. The data word contained in the CIU data register is transferred to B: CIU (18 - 1) to B (18 - 1), CIU (19) to control bit 1 flip-flop, CIU (20) to control bit 2 flip-flop and CIU (21) to control bit 3 flip-flop.

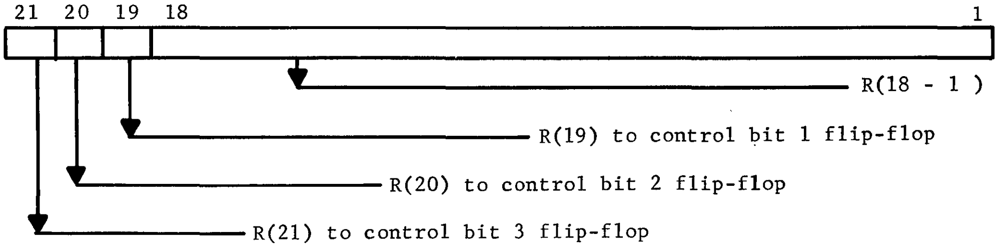
2. The CIU data register is reset.

3. The address register is increased by 1 and transmit mode is set (DEF2).

4. The transfer of another word is initiated.

5. The CIU is put in the busy condition.

CIU-930 to DATANET-30


21 20 19 18
1
R(18 - 1)
R(19) to control bit 1 flip-flop
R(20) to control bit 2 flip-flop
R(21) to control bit 3 flip-flop

Register Transfer **, T**

Four things are accomplished (see illustration below):

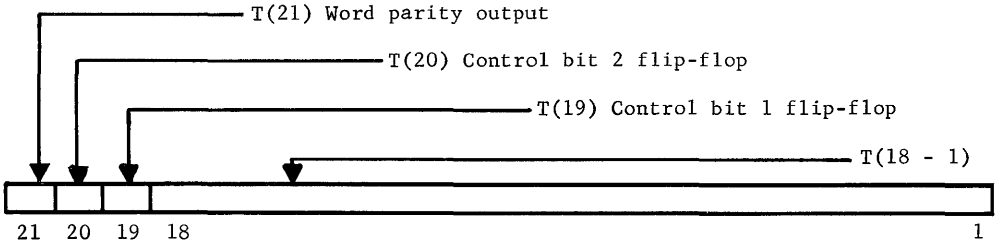
1. The 18-bit word contained in the B-register is sent to the transmit buffer positions 18 - 1. Control bit 1 flip-flop goes to position 19, control bit 2 flip-flop to position 20, and word parity output to position 21.

2. The transfer of the word from the CIU to the Compatibles/200 system is initiated.

3. The address register is increased by 1 and transmit mode is set (DEF2).

4. The CIU is put in the busy condition.

DATANET-30 to CIU-930



Mnemonic	Operand	Word Times
LDT	M	1

The contents of the specified memory address M are sent to the address register of the CIU.

Mnemonic	Operand	Word Times
DEF	I	1

DEF 1 Resets data register, increases address register by 1, and puts CIU in receive mode. Puts the CIU in busy condition, which initiates the transfer of another word from the computer.

DEF 2 Resets data register (before data comes from a register during a register transfer instruction). Puts the CIU in the transmit mode, increases address register by 1.

DEF 3 - 8 Not used.

DEF 9 Sends an automatic program interrupt signal to the computer.

DEF 0 Resets the address register (before the address comes from the program during a Load T (LDT) instruction.

EXTERNAL STATUS LINES

External status line indications are as follows:

NES 1 The CIU is not busy.

NES 2 - 10 Not used.

RECEIVE OPERATION

Assume that nothing is happening as far as the CIU is concerned. At some point, the program in the DATANET-30 initiates taking a block of words from the central processor memory. The program puts a number equal to one less than the initial memory address of the block in the address register of the CIU by means of an LDT instruction. Then the program sends a control signal to the CIU, via the external function drivers, which increases the address register by 1, puts the CIU in the receive mode, resets the data register, and initiates the transfer of the word from the specified central processor memory location to the data register in the CIU. After th word is in the data register, the CIU is no longer busy.

This condition can be tested via external status line 1 (NES 1). The program now executes a register transfer instruction to take the word out of the data register of the CIU and into the DATANET-30. This register transfer instruction also increases the address in the CIU address register by 1, puts the CIU in the receive mode, resets the data register, and initiates the transfer of another word from the central processor memory. This process repeats until the DATANET-30 program has received a sufficient number of words.

An example of receive operation is shown below:

```
Location   Instruction   Symbol    OPR    Operand  X    Remarks

            05670                  ORG    3000          ORIGIN LOCATION
 05670      011031                 PIC    25            PLACE CIU ADDRESS IN C COUNTER
 05671      024001                 DIF    1             RESET CB1,2 AND 3
 05672      251763                 LDT    ADRESS        LOAD CIU WITH 225 MEMORY (225 ADD. -1)
 05673      026002                 DEF    1             SETS REC. MODE
 05674      022001       GETWD     NES    1             CIU BUSY
 05675      121673                 BZE    *-1           YES, GO BACK
 05676      060044                 TRA    R,B           NO, TRANSFER WORD TO B
 05677      020002                 NIS    2             CHECK OUTPUT OF WORD PARITY NETWORK
 05700      135710                 BZE    ERROR    X    IF OUTPUT IS ZERO EXIT TO ERROR
 05701      705706                 STB    DATAIN   X    PARITY OK STORE IN MEMORY
 05702      771705                 XBZ    END           IS THIS THE LAST WORD?
 05703      125711                 BZE    EXIT     X    YES, EXIT
 05704      341706                 ADO    DATAIN        NO, ADD ONE TO MEMORY ADDRESS
 05705      101673                 BRU    GETWD         GO BACK GET NEXT WORD
 05706      777777       END       OCT    777777        END CONSTANT
 05707      015530       DATAIN    IND    7000          INPUT ADDRESS
 05710      000763       ADRESS    IND    499           225 ADDRESS-1
 05711      005752       ERROR     IND    3050          ERROR ADDRESS
 05712      005757       EXIT      IND    3055          NORMAL EXIT ADDRESS
```

Initially the CIU address is put into the C-register. The address register of the CIU is loaded with the desired central processor memory address. The address must be less than the desired starting address, because the DEF 1 instruction which puts the CIU into the receive mode, also increments the address counter by 1. The CIU is tested for a busy condition by the NES 1 command and the program stays in a loop until the CIU becomes ready. When the CIU becomes ready, the word is transferred to the B-register and the address counter is automatically counted up 1. The word is stored in memory, then tested for end-of-block condition. If the end-of-block condition is not found, control is transferred back to get another word.

Transmit Operation

Assume that nothing is happening as far as the CIU is concerned. At some point, the program in the DATANET-30 decides to put a block of words into the Compatibles/200 system. The program puts a number equal to one less than the initial memory address into the address register of the CIU with an LDT instruction.

Then the program transfers a word into the CIU data register with a register transfer instruction. This register transfer instruction also puts the CIU in the busy condition mode, increases the address in the address register by 1, and initiates the transfer of the word from the data register into the central processor memory. After the word has been written into memory, the CIU is no longer busy. This condition can be tested via external status line 1 (NES 1). The DATANET-30 program can now put another word in the data register and send it to the central processor. This process repeats until the DATANET-30 program decides that sufficient words have been transferred to the Compatibles/200 system.

The transmit example works just the reverse of receive with the exception of the DEF 2 instruction to set the CIU to the transmit mode and the DIF 1 to reset the CB 1, CB 2, and parity flip-flops.

```
Location  Instruction   Symbol   OPR    Operand  X        Remarks

                                 REM                TRANSMIT TO 225 VIA CIU
            07640                ORG    4000        ORIGIN LOCATION 4000
 07640      011031               PIC    25          PLACE CIU ADDRESS IN C
 07641      026001               DIF    1           RESET CB1, 2 AND 3
 07642      251747               LDT    ADRESS      LOAD 225 ADDRESS INTO CIU & SET TRANS
 07643      022001      SENDWD   NES    1           CIU BUSY                        MODE
 07644      121643               BZE    *-1         YES TRY AGAIN
 07645      601653               LDB    DATOUT   X  NO LOAD WORD TO BE TRANSFERRED
 07646      060401               TRA    B,T         TRANSFER TO CIU DATA BUFFER
 07647      771654               XBZ    ENDWD       IS THIS THE LAST WORD?
 07650      121655               BZE    TEXIT       YES EXIT
 07651      341653               ADO    DATOUT      NO ADD ONE TO MEMORY ADDRESS
 07652      101643               BRU    SENDWD      GO BACK TRANSMIT NEXT WORD
 07653      013560      DATOUT   IND    6000        DATANET-30 OUTPUT ADDRESS
 07654      777777      ENDWD    OCT    777777      END WORD CONSTANT
 07655      007722      TEXIT    IND    4050        EXIT ADDRESS
 07656      001747      ADRESS   IND    999         225 ADDRESS-1
```

APPENDIX D

CIU-931

This information will be issued at a later date.

APPENDIX E

INSTRUCTION SUMMARY

CONVERSION TABLE, 5-LEVEL BAUDOT TO OCTAL

MACRO COMMANDS

The DATANET-30 General Assembly Program recognizes various macro commands, and will assemble them as follows:

CL2	F,T	CL1	F,T
		CL1	T,T
CL3	F,T	CL1	F,T
		CL1	T,T
		CL1	T,T
CL4	F,T	CL6	F,T
		CR1	T,T
		CR1	T,T
CL5	F,T	CL6	F,T
		CR1	T,T
CL7	F,T	CL6	F,T
		CL1	T,T
CL8	F,T	CL6	F,T
		CL1	T,T
		CL1	T,T
CL9	F,T	CL6	F,T
		CL1	T,T
		CL1	T,T
		CL1	T,T
CR2	F,T	CR1	F,T
		CR1	T,T
CR3	F,T	CR1	F,T
		CR1	T,T
		CR1	T,T
CR4	F,T	CR6	F,T
		CL1	T,T
		CL1	T,T
CR5	F,T	CR6	F,T
		CL1	T,T
CR7	F,T	CR6	F,T
		CR1	T,T
CR8	F,T	CR6	F,T
		CR1	T,T
		CR1	T,T
CR9	F,T	CR6	F,T
		CR1	T,T
		CR1	T,T
		CR1	T,T
SL2	F,T	SL1	F,T
		SL1	T,T
SL3	F,T	SL1	F,T
		SL1	T,T
		SL1	T,T

SL4	F, T	. . . . . .	SL1	F, T
			SL1	T, T
			SL1	T, T
			SL1	T, T
SL5	F, T	. . . . . .	SL1	F, T
			SL1	T, T
			SL1	T, T
			SL1	T, T
			SL1	T, T
SL7	F, T	. . . . . .	SL6	F, T
			SL1	T, T
SL8	F, T	. . . . . .	SL6	F, T
			SL1	T, T
			SL1	T, T
SL9	F, T	. . . . . .	SL6	F, T
			SL1	T, T
			SL1	T, T
			SL1	T, T
SR2	F, T	. . . . . .	SR1	F, T
			SR1	T, T
SR3	F, T	. . . . . .	SR1	F, T
			SR1	T, T
			SR1	T, T
SR4	F, T	. . . . . .	SR1	F, T
			SR1	T, T
			SR1	T, T
			SR1	T, T
SR5	F, T	. . . . . .	SR1	F, T
			SR1	T, T
			SR1	T, T
			SR1	T, T
			SR1	T, T

F is the Register FROM
T is the Register TO

SR7	F, T	. . . . . .	SR6	F, T
			SR1	T, T
SR8	F, T	. . . . . .	SR6	F, T
			SR1	T, T
			SR1	T, T
SR9	F, T	. . . . . .	SR6	F, T
			SR1	T, T
			SR1	T, T
			SR1	T, T
SAM	α	. . . . . .	CMM	α
			AAM	α
			CMM	α
SBM	α	. . . . . .	CMM	α
			ABM	α
			CMM	α
SMA	α	. . . . . .	TRC	A, A
			AMA	α
			TRC	A, A
SMB	α	. . . . . .	TRC	B, B
			AMB	α
			TRC	B, B
SLD	I	. . . . . .	(SLS	A, A
			(SL1	B, B
		I times	.	.
			.	.
			(SLS	A, A
			(SL1	B, B
SRD	I	. . . . . .	(SRS	B, B
			(SRI	A, A
		I times	.	.
			.	.
			(SRS	B, B
			(SR1	A, A

The macro commands that are register transfer commands (with the exception of the double shifts) have the same error checks as a non-macro register transfer command, plus some additional checks. An error will be flagged when the user attempts to:

```
Register Transfer MACRO   0,       anything
Register Transfer MACRO anything,   Z
Register Transfer MACRO anything,   T
```

The from-to bits in the instruction will <u>not</u> be deleted on any of the above errors. The error tag only signifies that the instruction should be examined to see if it is correct.

The macro commands SMA, SMB, SAM and SBM, will have the same error checks and same addressing capabilities as non-macro commands requiring a memory address.

No error checks are perfomed on the macro double shift commands SLD and SRD. The operand must be decimal and must be left-justified in the operand field.

5-LEVEL MOD 28 TELETYPE

BAUDOT TO OCTAL CONVERSION TABLE

(ALPHABETICAL SEQUENCE)

LETTERS	FIGURES	LEFT JUSTIFIED	RIGHT JUSTIFIED
A	-	06	03
B	?	62	31
C	:	34	16
D	$	22	11
E	3	02	01
F	! *)	32	15
G	+	64	32
H	#	50	24
I	8	14	06
J	' *)	26	13
K	(	36	17
L	)	44	22
M	.	70	34
N	,	30	14
Ø	9	60	30
P	0	54	26
Q	1	56	27
R	4	24	12
S	BELL *)	12	05
T	5	40	20
U	7	16	07
V	; *)	74	36
W	2	46	23
X	/	72	35
Y	6	52	25
Z	" *)	42	21
	0	54	26
	1	56	27
	2	46	23
	3	02	01
	4	24	12
	5	40	20
	6	52	25
	7	16	07
	8	14	06
	9	60	30
BLANK	BLANK	00	00
LTRS.	LTRS.	76	37
FIGS.	FIGS.	66	33
L.FEED	L.FEED	04	02
SPACE	SPACE	10	04
CR.RET.	CR.RET.	20	10

(NUMERICAL SEQUENCE)

LETTERS	FIGURES	LEFT JUSTIFIED	RIGHT JUSTIFIED
Blank	Blank	00	00
E	3	02	01
Line Feed	Line Feed	04	02
A	—	06	03
Space	Space	10	04
S	BELL *)	12	05
I	8	14	06
U	7	16	07
Carr.Ret.	Carr.Ret.	20	10
D	$	22	11
R	4	24	12
J	' *)	26	13
N	,	30	14
F	! *)	32	15
C	:	34	16
K	(	36	17
T	5	40	20
Z	" *)	42	21
L	)	44	22
W	2	46	23
H	#	50	24
Y	6	52	25
P	0	54	26
Q	1	56	27
Ø	9	60	30
B	?	62	31
G	+	64	32
Figs.	Figs.	66	33
M	.	70	34
X	/	72	35
V	; *)	74	36
Ltrs.	Ltrs.	76	37

*) NOTE: These symbols are not on printer; for convenience, however, they are printed on this form.

DATANET-30 DATA COMMUNICATIONS PROCESSOR

ABBREVIATED INSTRUCTION REPERTOIRE

FIRST OCTAL DIGIT	SECOND OCTAL DIGIT								THIRD OCTAL DIGIT	
	0	1	2	3	4	5	6	7		
0	HLT	AIC	NIS	SCN	SL1	CL1	TRA	SLS	0	GENERAL
0		PIC							1	
0	HLT	NCZ	NES	CSR	SR1	CR1	TRC	SRS	2	
0	HLT	XCZ	DIF		SL6	CL6	BC0		4	
0	HLT		DEF		SR6	CR6	BC1		6	
1	BRU	BRS	BZE	BNZ	BPL	BM1	BEV	BOD		
2	LDC	LDD	LDZ	LDQ		LDT	LDF			NON-GENERAL
3	STC	STD	STZ	CMM	ADO	SBO	STF	AMD		
4	LDA	CMA	AMA	NMA	RMA	XMA	AAZ			
5	STA	CAM	AAM	NAM	RAM	XAM	NAZ	XAZ		
6	LDB	CMB	AMB	NMB	RMB	XMB	ABZ			
7	STB	CBM	ABM	NBM	RBM	XBM	NBZ	XBZ		

DATANET-30 DATA COMMUNICATIONS PROCESSOR

INSTRUCTION REPERTOIRE

WORD TIMES	CODE OCTAL		OPERAND	FUNCTIONAL DESCRIPTION
*****	LOAD INSTRUCTIONS			
2	40	LDA	M	LOAD A FROM M
2	60	LDB	M	LOAD B FROM M
2	20	LDC	M	LOAD C FROM M
3	21	LDD	M	LOAD DOUBLE -- A FROM M, B FROM M+1
2	26	LDF	M	LOAD SPECIAL FLIP-FLOPS FROM M
2	23	LDQ	M	LOAD Q FROM M
2	25	LDT	M	LOAD T -- SEND M TO TRANSMIT DATA DRIVERS
2	22	LDZ	M	LOAD Z -- SEND M TO Z DRIVERS (NO FURTHER)
2	41	CMA	M	LOAD A WITH M-NOT (COMPLEMENT M TO A)
2	61	CMB	M	LOAD B WITH M-NOT (COMPLEMENT M TO B)
1	011	PIC	I	PLACE I IN C
*****	STORE INSTRUCTIONS			
2	50	STA	M	STORE A IN M
2	70	STB	M	STORE B IN M
2	30	STC	M	STORE C IN M
3	31	STD	M	STORE DOUBLE -- A IN M, B IN M+1
2	36	STF	M	STORE SPECIAL FLIP-FLOPS IN M
2	32	STZ	M	STORE ZERO IN M
2	51	CAM	M	STORE A-NOT IN M (COMPLEMENT A TO M)
2	71	CBM	M	STORE B-NOT IN M (COMPLEMENT B TO M)
2	33	CMM	M	STORE M-NOT IN M (COMPLEMENT M TO M)
*****	ARITHMETIC INSTRUCTIONS			
2	42	AMA	M	ADD M TO A
3	52	AAM	M	ADD A TO M
2	46	AAZ	M	ADD A, M - RESULT TO Z DRIVERS
2	62	AMB	M	ADD M TO B
3	72	ABM	M	ADD B TO M
2	66	ABZ	M	ADD B, M - RESULT TO Z DRIVERS
3	37	AMD	M	ADD DOUBLE LENGTH WORD M-(M+1) TO A-B
3	34	ADO	M	ADD ONE TO M
3	35	SBO	M	SUBTRACT ONE FROM M
1	010	AIC	I	ADD I TO C

WORD TIMES	CODE OCTAL		OPERAND	FUNCTIONAL DESCRIPTION
*****		BRANCH INSTRUCTIONS		
1	10	BRU	M	BRANCH UNCONDITIONALLY
3	11	BRS	M	BRANCH TO SUBROUTINE
1	12	BZE	M	BRANCH IF ZERO FF IS ZERO
1	13	BNZ	M	BRANCH IF ZERO FF IS NON-ZERO
1	14	BPL	M	BRANCH IF PLUS FF IS PLUS
1	15	BMI	M	BRANCH IF PLUS FF IS MINUS
1	16	BEV	M	BRANCH IF EVEN FF IS EVEN
1	17	BOD	M	BRANCH IF EVEN FF IS ODD
*****		LOGICAL OPERATION INSTRUCTIONS		
2	43	NMA	M	M AND A TO A
2	53	NAM	M	M AND A TO M
2	63	NMB	M	M AND B TO B
2	73	NBM	M	M AND B TO M
2	56	NAZ	M	M AND A TO Z ONLY
2	76	NBZ	M	M AND B TO Z ONLY
1	012	NCZ	I	I AND C TO Z ONLY
1	020	NIS	I	I AND INTERNAL STATUS LINES TO Z ONLY
1	022	NES	I	I AND EXTERNAL STATUS LINES TO Z ONLY
2	44	BMA	M	M OR A TO A
2	54	RAM	M	M OR A TO M
2	64	RMB	M	M OR B TO B
2	74	RBM	M	M OR B TO M
2	45	XMA	M	M XOR A TO A
2	55	XAM	M	M XOR A TO M
2	65	XMB	M	M XOR B TO B
2	75	XBM	M	M XOR B TO M
2	57	XAZ	M	M XOR A TO Z ONLY
2	77	XBZ	M	M XOR B TO Z ONLY
1	014	XCZ	I	I XOR C TO Z ONLY
*****		REGISTER TRANSFER INSTRUCTIONS		FROM ABCQRS - TO ABCTZ
1	060	TRA	FROM, TO	TRANSFER
1	062	TRC	FROM, TO	TRANSFER COMPLEMENT
1	040	SL1	FROM, TO	SHIFT LEFT ONE
1	042	SR1	FROM, TO	SHIFT RIGHT ONE
1	044	SL6	FROM, TO	SHIFT LEFT SIX
1	046	SR6	FROM, TO	SHIFT RIGHT SIX
1	070	SLS	FROM, TO	SHIFT LEFT SPECIAL
1	072	SRS	FROM, TO	SHIFT RIGHT SPECIAL
1	050	CL1	FROM, TO	CIRCULATE LEFT ONE
1	052	CR1	FROM, TO	CIRCULATE RIGHT ONE
1	054	CL6	FROM, TO	CIRCULATE LEFT SIX
1	056	CR6	FROM, TO	CIRCULATE RIGHT SIX
1	064	BC0	FROM, TO	BIT CHANGE ZERO (8-LEVEL LINE TO 6-BIT)
1	066	BC1	FROM, TO	BIT CHANGE ONE (6-BIT TO 8-LEVEL LINE)

WORD TIMES	CODE OCTAL		OPERAND	FUNCTIONAL DESCRIPTION
*****		SPECIAL INSTRUCTIONS		
1	00	HLT		CONDITIONAL HALT
1	024	DIF	I	DRIVE INTERNAL FUNCTION LINES
1	026	DEF	I	DRIVE EXTERNAL FUNCTION LINES
	030	SCN	I	SCAN BIT BUFFERS
	032	CSR	I	CONTROLLER STATUS REQUEST

ALPHANUMERIC LISTING
DATANET 30
COMMUNICATIONS PROCESSOR
INSTRUCTION REPERTOIRE

OPR	OPERAND	OCTAL	GROUP	DESCRIPTION	W.T.
GROUP	MNEMONIC			INTERNAL INSTRUCTIONS	
GROUP	MNEMONIC		BBC	BIT BUFFER CHANNEL INSTRUCTIONS	
GROUP	MNEMONIC		BSU	BUFFER SELECTOR UNIT INSTRUCTIONS	
GROUP	MNEMONIC		CBC	CHARACTER BUFFER CHANNEL INSTRUCTIONS	
GROUP	MNEMONIC		CIU	COMPUTER INTERFACE UNIT INSTRUCTIONS	
GROUP	MNEMONIC		CSU	CONTROLLER SELECTOR UNIT INSTRUCTIONS	
GROUP	MNEMONIC		DSU	DISC STORAGE UNIT INSTRUCTIONS	
GROUP	MNEMONIC		HSP	HIGH SPEED PRINTER INSTRUCTIONS	
GROUP	MNEMONIC		MACRO	GENERAL ASSEMBLY PROGRAM MACRO INSTRUCTIONS	
GROUP	MNEMONIC		MTS	MAGNETIC TAPE SYSTEM INSTRUCTIONS	
GROUP	MNEMONIC		PTR	PAPER TAPE READER INSTRUCTIONS	
GROUP	MNEMONIC		RPU	RECEIVE PARALLEL UNIT INSTRUCTIONS	
GROUP	MNEMONIC		WBC	WORD BUFFER CHANNEL INSTRUCTIONS	
AAM	M	520000		ADD A TO M	3
AAZ	M	460000		ADD A,M - RESULT TO Z DRIVERS	2
ABM	M	720000		ADD B TO M	3
ABZ	M	660000		ADD B,M - RESULT TO Z DRIVERS	2
ADO	M	340000		ADD ONE TO M	3
AIC	I	010000		ADD I TO C	1
AMA	M	420000		ADD M TO A	2
AMB	M	620000		ADD M TO B	2
AMD	M	240000		ADD DOUBLE -- ADD M,M+1 TO A,B	3
BC0	FROM,TO	064000		BIT CHANGE ZERO (8-LEVEL LINE TO 6-BIT)	1
BC1	FROM,TO	066000		BIT CHANGE ONE (6-BIT TO 8-LEVEL LINE)	1
BEV	M	160000		BRANCH IF EVEN FF IS EVEN	1
BKW			MTS	BACKSPACE AND POSITION WRITE HEAD	1+3
BMI	M	150000		BRANCH IF PLUS FF IS MINUS	1
BNZ	M	130000		BRANCH IF ZERO FF IS NON-ZERO	1
BOD	M	170000		BRANCH IF EVEN FF IS ODD	1
BPL	M	140000		BRANCH IF PLUS FF IS PLUS	1
BRS	M	110000		BRANCH TO SUBROUTINE	3
BRU	M	100000		BRANCH UNCONDITIONALLY	1
BZE	M	120000		BRANCH IF ZERO FF IS ZERO	1
CAM	M	510000		STORE A-NOT IN M (COMPLEMENT A TO M)	2
CBM	M	710000		STORE B-NOT IN M (COMPLEMENT B TO M)	2
CL1	FROM,TO	050000		CIRCULATE LEFT 1	1
CL2	FROM,TO		MACRO	CIRCULATE LEFT 2	2
CL3	FROM,TO		MACRO	CIRCULATE LEFT 3	3
CL4	FROM,TO		MACRO	CIRCULATE LEFT 4	3
CL5	FROM,TO		MACRO	CIRCULATE LEFT 5	2
CL6	FROM,TO	054000		CIRCULATE LEFT 6	1
CL7	FROM,TO		MACRO	CIRCULATE LEFT 7	2
CL8	FROM,TO		MACRO	CIRCULATE LEFT 8	3

OPR	OPERAND	OCTAL	GROUP	DESCRIPTION	W.T.
CL9	FROM,TO		MACRO	CIRCULATE LEFT 9	4
CMA	M	410000		LOAD A WITH M-NOT (COMPLEMENT M TO A)	2
CMB	M	610000		LOAD B WITH M-NOT (COMPLEMENT M TO B)	2
CMM	M	330000		STORE M-NOT IN M (COMPLEMENT M TO M)	2
CR1	FROM,TO	052000		CIRCULATE RIGHT 1	1
CR2	FROM,TO		MACRO	CIRCULATE RIGHT 2	2
CR3	FROM,TO		MACRO	CIRCULATE RIGHT 3	3
CR4	FROM,TO		MACRO	CIRCULATE RIGHT 4	3
CR5	FROM,TO		MACRO	CIRCULATE RIGHT 5	2
CR6	FROM,TO	056000		CIRCULATE RIGHT 6	1
CR7	FROM,TO		MACRO	CIRCULATE RIGHT 7	2
CR8	FROM,TO		MACRO	CIRCULATE RIGHT 8	3
CR9	FROM,TO		MACRO	CIRCULATE RIGHT 9	4
CSR	I	032000	CSU	CONTROLLER STATUS REQUEST	3-10
DEF	I	026000	BSU	DRIVE EXTERNAL FUNCTION	1
DEF	1	026001	BBC	RESET RECEIVE FLAG AND DATA BUFFER	1
DEF	2	026002	BBC	RESET TRANSMIT FLAG AND DATA BUFFER	1
DEF	3	026004	BBC	TURN CARRIER OFF	1
DEF	4	026010	BBC	TURN CARRIER ON	1
DEF	5	026020	BBC	RESET RECEIVE CLOCK	1
DEF	6	026040	BBC	SET ECHO MODE	1
DEF	7	026100	BBC	RESET ECHO MODE	1
DEF	1	026001	CBC	RESET RECEIVE FLAG AND DATA BUFFER	1
DEF	2	026002	CBC	RESET TRANSMIT FLAG AND DATA BUFFER	1
DEF	3	026004	CBC	TURN CARRIER OFF	1
DEF	4	026010	CBC	TURN CARRIER ON	1
DEF	9	026400	CBC	ANSWER INCOMING CALL	1
DEF	0	027000	CBC	DISCONNECT CALL	1
DEF	1	026001	CIU	RESET FLAG AND BUFFER, SET RECEIVE MODE	1
DEF	2	026002	CIU	RESET FLAG AND BUFFER, SET TRANSMIT MODE	1
DEF	9	026400	CIU	AUTOMATIC PRIORITY INTERRUPT THE 225	1
DEF	0	027000	CIU	RESET THE ADDRESS REGISTER	
DEF	1	026001	PTR	RESET FLAG AND READ NEXT CHARACTER	1
DEF	1	026001	RPU	RESET CHARACTER READY	1
DEF	2	026002	RPU	RESET ANSWERBACK A AND B	1
DEF	3	026004	RPU	RESET ANSWERBACK MODE	1
DEF	4	026010	RPU	SET ANSWERBACK MODE	1
DEF	5	026020	RPU	ANSWERBACK A	1
DEF	6	026040	RPU	AUX FUNCTION SET TRANSMIT MODE	1
DEF	7	026100	RPU	AUX FUNCTION RESET TRANSMIT MODE	1
DEF	8	026200	RPU	ANSWERBACK B	1
DEF	9	026400	RPU	ANSWER INCOMING CALL	1
DEF	0	027000	RPU	DISCONNECT CALL	1

OPR	OPERAND	OCTAL	GROUP	DESCRIPTION	W.T.
DEF	1	026001	WBC	RESET RECEIVE FLAG AND DATA BUFFER	1
DEF	2	026002	WBC	RESET TRANSMIT FLAG AND DATA BUFFER	1
DEF	3	026004	WBC	TURN CARRIER OFF	1
DEF	4	026010	WBC	TURN CARRIER ON	1
DIF	I	024000		DRIVE INTERNAL FUNCTION	1
DIF	1	024001		RESET CB 1 AND 2, AND RESET PARITY BIT FF	1
DIF	2	024002		RESET BUZZER FLIP-FLOP	1
DIF	3	024004		SET BUZZER FLIP-FLOP	1
DIF	4	024010		INITIATE HARDWARE LOAD PROCESS	1
DIF	7	024100	CSU	SELECT PERIPHERAL CONTROLLER	1+3
DIF	8	024200		SET CONTROL BIT FLIP-FLOP 1	1
DIF	9	024400		SET CONTROL BIT FLIP-FLOP 2	1
DIF	0	025000		SET THE PARITY BIT FLIP-FLOP	1
HLT	I	000000		CONDITIONAL HALT	1
LDA	M	400000		LOAD A FROM M	2
LDB	M	600000		LOAD B FROM M	2
LDC	M	200000		LOAD C FROM M	2
LDD	M	210000		LOAD DOUBLE -- A,B FROM M,M+1	3
LDF	M	260000		LOAD SPECIAL FLIP-FLOPS FROM M	2
LDQ	M	230000		LOAD Q FROM M	2
LDT	M	250000	BSU	LOAD T (TRANSMIT DATA DRIVERS) FROM M	2
LDZ	M	220000		LOAD Z (BRANCH FLIP-FLOPS) FROM M	2
NAM	M	530000		M AND A TO M	2
NAZ	M	560000		M AND A TO Z ONLY	2
NBM	M	730000		M AND B TO M	2
NBZ	M	760000		M AND B TO Z ONLY	2
NCZ	I	012000		I AND C TO Z ONLY	1
NES	I	022000	BSU	I AND EXTERNAL STATUS LINES TO Z ONLY	1
NES	1	022001	BBC	RC FLAG SET (BUFFER CONTAINS A NEW BIT)	1
NES	2	022002	BBC	TX FLAG SET (BUFFER READY FOR A NEW BIT)	1
NES	5	022020	BBC	INTERLOCK ON	1
NES	6	022040	BBC	CARRIER ON	1
NES	1	022001	CBC	RC FLAG SET (BUFFER CONTAINS A NEW CHAR.)	1
NES	2	022002	CBC	TX FLAG SET (BUFFER READY FOR A NEW CHAR.)	1
NES	3	022004	CBC	CALL IN PROGRESS	1
NES	4	022010	CBC	REQUEST ANSWER	1
NES	5	022020	CBC	DATA MODE	1
NES	6	022040	CBC	CARRIER ON	
NES	7	022100	CBC	CLEAR TO SEND	1
NES	1	022001	CIU	FLAG SET (BUFFER READY)	1
NES	1	022001	PTR	READ FLAG SET (BUFFER CONTAINS A NEW CHAR.)	1
NES	1	022001	RPU	CHARACTER READY	1
NES	2	022002	RPU	LINE TURN AROUND	1
NES	3	022004	RPU	CALL IN PROGRESS	1
NES	4	022010	RPU	REQUEST TO ANSWER CALL	1

OPR	OPERAND	OCTAL	GROUP	DESCRIPTION	W.T.
NES	5	022020	RPU	SPACE DETECT	
NES	6	022040	RPU	AUX STATUS LINE	
NES	7	022100	RPU	AUX STATUS LINE	
NES	8	022200	RPU	AUX STATUS LINE	
NES	1	022001	WBC	RC FLAG SET (BUFFER CONTAINS A NEW WORD)	1
NES	2	022002	WBC	TX FLAG SET (BUFFER READY FOR A NEW WORD)	1
NIS	I	020000		I AND INTERNAL STATUS LINES TO Z ONLY	1
NIS	1	020001		CHARACTER PARITY OUTPUT	1
NIS	2	020002		WORD PARITY OUTPUT	1
NIS	3	020004		CB FF 2 AND WORD PARITY OUTPUT ARE EQUAL	1
NIS	7	020100	CSU	SELECT COMMAND IS COMPLETED	1
NIS	4	020010		SWITCH IS IN THE MAINTENANCE MODE	1
NIS	8	020200		CB FF 1	1
NIS	9	020400		CB FF 2	1
NIS	0	021000		PARITY FF	
NMA	M	430000		M AND A TO A	2
NMB	M	630000		M AND B TO B	2
PIC	I	011000		PLACE I IN C	1
PRF			DSU	POSITION DISC STORAGE UNIT	
RAM	M	540000		M OR A TO M	2
RBD			MTS	READ BACKWARD DECIMAL	1+3
RBM	M	740000		M OR B TO M	2
RBS			MTS	READ BACKWARD BINARY	1+3
RMA	M	440000		M OR A TO A	2
RMB	M	640000		M OR B TO B	2
RRF			DSU	READ DSU	
RTB			MTS	READ TAPE BINAPY	1+3
RTD			MTS	READ TAPE DECIMAL	1+3
RWD			MTS	REWIND	1+3
SAM	M		MACRO	SUBTRACT A FROM M	7
SBM	M		MACRO	SUBTRACT B FROM M	7
SBO	M	350000		SUBTRACT ONE FROM M	3
SCN	I	030000	BBC	SCAN BIT BUFFER UNITS	1+3N
SEL		024100	CSU	SELECT PERIPHERAL CONTROLLER	1+3
SL1	FROM,TO	040000		SHIFT LEFT 1	1
SL2	FROM,TO		MACRO	SHIFT LEFT 2	2
SL3	FROM,TO		MACRO	SHIFT LEFT 3	3
SL4	FROM,TO		MACRO	SHIFT LEFT 4	4
SL5	FROM,TO		MACRO	SHIFT LEFT 5	5
SL6	FROM,TO	044000		SHIFT LEFT 6	1
SL7	FROM,TO		MACRO	SHIFT LEFT 7	2
SL8	FROM,TO		MACRO	SHIFT LEFT 8	3
SL9	FROM,TO		MACRO	SHIFT LEFT 9	4
SLD	I		MACRO	SHIFT A,B LEFT I BITS	2 I

OPR	OPERAND	OCTAL	GROUP	DESCRIPTION	W.T.
SLS	FROM,TO	070000		SHIFT LEFT SPECIAL	1
SLT			HSP	SLEW PAPER TO TAPE PUNCH	1+3
SLW			HSP	SLEWING OF PAPER	1+3
SMA	M		MACRO	SUBTRACT M FROM A	4
SMB	M		MACRO	SUBTRACT M FROM B	4
SMD	M		MACRO	SUBTRACT M,M+1 FROM A,B	7
SR1	FROM,TO	042000		SHIFT RIGHT 1	1
SR2	FROM,TO		MACRO	SHIFT RIGHT 2	2
SR3	FROM,TO		MACRO	SHIFT RIGHT 3	3
SR4	FROM,TO		MACRO	SHIFT RIGHT 4	4
SR5	FROM,TO		MACRO	SHIFT RIGHT 5	5
SR6	FROM,TO	046000		SHIFT RIGHT 6	1
SR7	FROM,TO		MACRO	SHIFT RIGHT 7	2
SR8	FROM,TO		MACRO	SHIFT RIGHT 8	3
SR9	FROM,TO		MACRO	SHIFT RIGHT 9	4
SRD	I		MACRO	SHIFT A,B RIGHT I BITS	2 I
SRS	FROM,TO	072000		CIRCULATE RIGHT SPECIAL	1
STA	M	500000		STORE A IN M	2
STB	M	700000		STORE B IN M	2
STC	M	300000		STORE C IN M	2
STD	M	310000		STORE DOUBLE -- A,B IN M,M+1	3
STF	M	360000		STORE SPECIAL FLIP-FLOPS	2
STZ	M	320000		STOPE ZERO IN M	2
TRA	FROM,TO	060000		TRANSFER	1
TRC	FROM,TO	062000		TRANSFER COMPLEMENT	1
WEF			MTS	WRITE END OF FILE	1+3
WFL			HSP	WRITE FORMAT LINE	1+3
WPL			HSP	WRITE PRINT LINE	1+3
WRF			DSU	WRITE DSU	
WTB			MTS	WRITE TAPE BINARY	1+3
WTD			MTS	WRITE TAPE DECIMAL	1+3
XAM	M	550000		M XOR A TO M	2
XAZ	M	570000		M XOR A TO Z ONLY	2
XBM	M	750000		M XOR B TO M	2
XBZ	M	770000		M XOR B TO Z ONLY	2
XCZ	I	014000		I XOR C TO Z ONLY	1
XMA	M	450000		M XOR A TO A	2
XMB	M	650000		M XOR B TO B	2

OCTAL LISTING
DATANET 30
COMMUNICATIONS PROCESSOR
INSTRUCTION REPERTOIRE

OPR	OPERAND	OCTAL	GROUP	DESCRIPTION	W.T.
GROUP	MNEMONIC			INTERNAL INSTRUCTIONS	
GROUP	MNEMONIC		BBU	BIT BUFFER UNIT INSTRUCTIONS	
GROUP	MNEMONIC		BSU	BUFFER SELECTOR UNIT INSTRUCTIONS	
GROUP	MNEMONIC		CBU	CHARACTER BUFFER UNIT INSTRUCTIONS	
GROUP	MNEMONIC		CIU	COMPUTER INTERFACE UNIT INSTRUCTIONS	
GROUP	MNEMONIC		CSU	CONTROLLER SELECTOR UNIT INSTRUCTIONS	
GROUP	MNEMONIC		HSP	HIGH SPEED PRINTER INSTRUCTIONS	
GROUP	MNEMONIC		MACRO	GENERAL ASSEMBLY PROGRAM MACRO INSTRUCTIONS	
GROUP	MNEMONIC		MRADS	MASS RANDOM ACCESS DATA STORAGE INSTRUCTIONS	
GROUP	MNEMONIC		MTS	MAGENTIC TAPE SYSTEM INSTRUCTIONS	
GROUP	MNEMONIC		PTR	PAPER TAPE READER INSTRUCTIONS	
GROUP	MNEMONIC		WBU	WORD BUFFER UNIT INSTRUCTIONS	
HLT	I	000000		CONDITIONAL HALT	1
AIC	I	010000		ADD I TO C	1
PIC	I	011000		PLACE I IN C	1
NCZ	I	012000		I AND C TO Z ONLY	1
XCZ	I	014000		I XOR C TO Z ONLY	1
NIS	I	020000		I AND INTERNAL STATUS LINES TO Z ONLY	1
NIS	1	020001		CHARACTER PARITY OUTPUT	1
NIS	2	020002		WORD PARITY OUTPUT	1
NIS	3	020004		CB FF 2 AND WORD PARITY OUTPUT ARE EOUAL	1
NIS	4	020010		SWITCH IS IN THE MAINTENANCE MODE	1
NIS	7	020100	CSU	SELECT COMMAND IS COMPLETED	1
NIS	8	020200		CB FF 1	1
NIS	9	020400		CB FF 2	1
NIS	0	021000		PARITY FF	1
NES	I	022000	BSU	I AND EXTERNAL STATUS LINES TO Z ONLY	1
NES	1	022001	BBC	RC FLAG SET (BUFFER CONTAINS A NEW BIT)	1
NES	1	022001	CBC	RC FLAG SET (BUFFER CONTAINS A NEW CHAR.)	1
NES	1	022001	CIU	FLAG SET (BUFFER READY)	1
NES	1	022001	PTR	READ FLAG SET (BUFFER CONTAINS A NEW CHAR.)	1
NES	1	022001	RPU	CHARACTER READY	
NES	1	022001	WBC	RC FLAG SET (BUFFER CONTAINS A NEW WORD)	1
NES	2	022002	BBC	TX FLAG SET (BUFFER READY FOR A NEW BIT)	1
NES	2	022002	CBC	TX FLAG SET (BUFFER READY FOR A NEW CHAR.)	1
NES	2	022002	RPU	LINE TURN AROUND	
NES	2	022002	WBC	WBC FLAG SET (BUFFER READY FOR A NEW WORD)	1
NES	3	022004	CBC	CALL IN PROGRESS	1
NES	3	022004	RPU	CALL IN PROGRESS	
NES	4	022010	CBC	REOUEST ANSWER	1
NES	4	022010	RPU	REOUEST TO ANSWER CALL	
NES	5	022020	CBC	DATA MODE	1
NES	5	022020	RPU	SPACE DETECT	

OPR	OPERAND	OCTAL	GROUP	DESCRIPTION	W.T.
NES	6	022040	CBC	CARRIER ON	1
NES	6	022040	RPU	AUX STATUS LINE	1
NES	7	022100	CBC	CLEAR TO SEND	1
NES	7	022100	RPU	AUX STATUS LINE	1
NES	8	022200	RPU	AUX STATUS LINE	1
DIF	I	024000		DRIVE INTERNAL FUNCTION	1
DIF	1	024001		RESET CB 1 AND 2, AND RESET PARITY BIT FF	1
DIF	2	024002		RESET BUZZER FLIP-FLOP	1
DIF	3	024004		SET BUZZER FLIP-FLOP	1
DIF	4	024010		INITIATE HARDWARE LOAD PROCESS	1
DIF	7	024100	CSU	SELECT PERIPHERAL CONTROLLER	1+3
SEL		024100	CSU	SELECT PERIPHERAL CONTROLLER	1+3
DIF	8	024200		SET CONTROL BIT FLIP-FLOP 1	1
DIF	9	024400		SET CONTROL BIT FLIP-FLOP 2	1
DIF	0	025000		SET THE PARITY BIT FLIP-FLOP	1
DEF	I	026000	BSU	DRIVE EXTERNAL FUNCTION	1
DEF	1	026001	BBC	RESET RECEIVE FLAG AND DATA BUFFER	1
DEF	1	026001	CBC	RESET RECEIVE FLAG AND DATA BUFFER	1
DEF	1	026001	CIU	RESET FLAG AND BUFFER, SET RECEIVE MODE	1
DEF	1	026001	PTR	RESET FLAG AND READ NEXT CHARACTER	1
DEF	1	026001	RPU	RESET CHARACTER READY	
DEF	1	026001	WBC	RESET RECEIVE FLAG AND DATA BUFFER	1
DEF	2	026002	BBC	RESET TRANSMIT FLAG AND DATA BUFFER	1
DEF	2	026002	CBC	RESET TRANSMIT FLAG AND DATA BUFFER	1
DEF	2	026002	CIU	RESET FLAG AND BUFFER, SET TRANSMIT MODE	1
DEF	2	026002	WBC	RESET TRANSMIT FLAG AND DATA BUFFER	1
DEF	2	026002	RPU	RESET ANSWERBACK A AND B	
DEF	3	026004	BBC	TURN CARRIER OFF	1
DEF	3	026004	CBC	TURN CARRIER OFF	1
DEF	3	026004	RPU	RESET ANSWERBACK MODE	
DEF	3	026004	WBC	TURN CARRIER OFF	1
DEF	4	026010	BBC	TURN CARRIER ON	1
DEF	4	026010	CBC	TURN CARRIER ON	1
DEF	4	026010	RPU	SET ANSWERBACK MODE	
DEF	4	026010	WBC	TURN CARRIER ON	1
DEF	5	026020	BBC	RESET RECEIVE CLOCK	1
DEF	5	026020	RPU	ANSWERBACK A	
DEF	6	026040	BBC	SET ECHO MODE	1
DEF	6	026040	RPU	AUX FUNCTION SET TRANSMIT MODE	

OPR	OPERAND	OCTAL	GROUP	DESCRIPTION	W.T.
DEF	7	026100	BBC	RESET ECHO MODE	1
DEF	7	026100	RPU	AUX FUNCTION RESET TRANSMIT MODE	1
DEF	8	026200	RPU	ANSWERBACK B	1
DEF	9	026400	CBC	ANSWER INCOMING CALL	1
DEF	9	026400	CIU	AUTOMATIC PRIORITY INTERRUPT THE 225	1
DEF	9	026400	RPU	ANSWER INCOMING CALL	1
DEF	0	027000	CBC	DISCONNECT CALL	1
DEF	0	027000	CIU	RESET THE ADDRESS REGISTER	1
DEF	0	027000	RPU	DISCONNECT CALL	1
SCN	I	030000	BBC	SCAN BIT BUFFER UNITS	1+3N
CSR	I	032000	CSU	CONTROLLER STATUS REQUEST	3-10
SL1	FROM,TO	040000		SHIFT LEFT 1	1
SR1	FROM,TO	042000		SHIFT RIGHT 1	1
SL6	FROM,TO	044000		SHIFT LEFT 6	1
SR6	FROM,TO	046000		SHIFT RIGHT 6	1
CL1	FROM,TO	050000		CIRCULATE LEFT 1	1
CR1	FROM,TO	052000		CIRCULATE RIGHT 1	1
CL6	FROM,TO	054000		CIRCULATE LEFT 6	1
CR6	FROM,TO	056000		CIRCULATE RIGHT 6	1
TRA	FROM,TO	060000		TRANSFER	1
TRC	FROM,TO	062000		TRANSFER COMPLEMENT	1
BC0	FROM,TO	064000		BIT CHANGE ZERO (8-LEVEL LINE TO 6-BIT)	1
BC1	FROM,TO	066000		BIT CHANGE ONE (6-BIT TO 8-LEVEL LINE)	1
SLS	FROM,TO	070000		SHIFT LEFT SPECIAL	1
SRS	FROM,TO	072000		CIRCULATE RIGHT SPECIAL	1
BRU	M	100000		BRANCH UNCONDITIONALLY	1
BRS	M	110000		BRANCH TO SUBROUTINE	3
BZE	M	120000		BRANCH IF ZERO FF IS ZERO	1
BNZ	M	130000		BRANCH IF ZERO FF IS NON-ZERO	1
BPL	M	140000		BRANCH IF PLUS FF IS PLUS	1
BMI	M	150000		BRANCH IF PLUS FF IS MINUS	1
BEV	M	160000		BRANCH IF EVEN FF IS EVEN	1
BOD	M	170000		BRANCH IF EVEN FF IS ODD	1
LDC	M	200000		LOAD C FROM M	2
LDD	M	210000		LOAD DOUBLE -- A,B FROM M,M+1	3
LDZ	M	220000		LOAD Z (BRANCH FLIP-FLOPS) FROM M	2
LDQ	M	230000		LOAD Q FROM M	2
AMD	M	240000		ADD DOUBLE -- ADD M,M+1 TO A,B	3
LDT	M	250000	BSU	LOAD T (TRANSMIT DATA DRIVERS) FROM M	2
LDF	M	260000		LOAD SPECIAL FLIP-FLOPS FROM M	2

OPR	OPERAND	OCTAL	GROUP	DESCPIPTION	W.T.
STC	M	300000		STORE C IN M	2
STD	M	310000		STORE DOUBLE -- A,B IN M,M+1	3
STZ	M	320000		STORE ZERO IN M	2
CMM	M	330000		STORE M-NOT IN M (COMPLEMENT M TO M)	2
ADO	M	340000		ADD ONE TO M	3
SBO	M	350000		SUBTRACT ONE FROM M	3
STF	M	360000		STORE SPECIAL FLIP-FLOPS	2
LDA	M	400000		LOAD A FROM M	2
CMA	M	410000		LOAD A WITH M-NOT (COMPLEMENT M TO A)	2
AMA	M	420000		ADD M TO A	2
NMA	M	430000		M AND A TO A	2
RMA	M	440000		M OR A TO A	2
XMA	M	450000		M XOR A TO A	2
AAZ	M	460000		ADD A,M - RESULT TO Z DRIVERS	2
STA	M	500000		STORE A IN M	2
CAM	M	510000		STORE A-NOT IN M (COMPLEMENT A TO M)	2
AAM	M	520000		ADD A TO M	3
NAM	M	530000		M AND A TO M	2
RAM	M	540000		M OR A TO M	2
XAM	M	550000		M XOR A TO M	2
NAZ	M	560000		M AND A TO Z ONLY	2
XAZ	M	570000		M XOR A TO Z ONLY	2
LDB	M	600000		LOAD B FROM M	2
CMB	M	610000		LOAD B WITH M-NOT (COMPLEMENT M TO B)	2
AMB	M	620000		ADD M TO B	2
NMB	M	630000		M AND B TO B	2
RMB	M	640000		M OR B TO B	2
XMB	M	650000		M XOR B TO B	2
ABZ	M	660000		ADD B,M - RESULT TO Z DRIVERS	2
STB	M	700000		STORE B IN M	2
CBM	M	710000		STORE B-NOT IN M (COMPLEMENT B TO M)	2
ABM	M	720000		ADD B TO M	3
NBM	M	730000		M AND B TO M	2
RBM	M	740000		M OR B TO M	2
XBM	M	750000		M XOR B TO M	2
NBZ	M	760000		M AND B TO Z ONLY	2
XBZ	M	770000		M XOR B TO Z ONLY	2
GROUP	MNEMONIC		HSP	HIGH SPEED PRINTER INSTRUCTIONS	
SLT			HSP	SLEW PAPER TO TAPE PUNCH	1+3
SLW			HSP	SLEWING OF PAPER	1+3
WFL			HSP	WRITE FORMAT LINE	1+3
WPL			HSP	WRITE PRINT LINE	1+3
GROUP	MNEMONIC		DSU	DISC STORAGE UNIT INSTRUCTIONS	
PRF			DSU	POSITION DISC STORAGE UNIT	
RRF			DSU	READ DSU	
WRF			DSU	WRITE DSU	

OPR	OPERAND	OCTAL	GROUP	DESCRIPTION	W.T.
GROUP	MNEMONIC		MTS	MAGNETIC TAPE SYSTEM INSTRUCTIONS	
BKW			MTS	BACKSPACE AND POSITION WRITE HEAD	1+3
RBD			MTS	READ BACKWARD DECIMAL	1+3
RBS			MTS	READ BACKWARD BINARY	1+3
RTB			MTS	READ TAPE BINARY	1+3
RTD			MTS	READ TAPE DECIMAL	1+3
RWD			MTS	REWIND	1+3
WEF			MTS	WRITE END OF FILE	1+3
WTB			MTS	WRITE TAPE BINARY	1+3
WTD			MTS	WRITE TAPE DECIMAL	1+3
GROUP	MNEMONIC		MACRO	GENERAL ASSEMBLY PROGRAM MACRO INSTRUCTIONS	
CL2	FROM,TO		MACRO	CIRCULATE LEFT 2	2
CL3	FROM,TO		MACRO	CIRCULATE LEFT 3	3
CL4	FROM,TO		MACRO	CIRCULATE LEFT 4	3
CL5	FROM,TO		MACRO	CIRCULATE LEFT 5	2
CL7	FROM,TO		MACPO	CIRCULATE LEFT 7	2
CL8	FROM,TO		MACRO	CIRCULATE LEFT 8	3
CL9	FROM,TO		MACRO	CIRCULATE LEFT 9	4
CR2	FROM,TO		MACRO	CIRCULATE RIGHT 2	2
CR3	FROM,TO		MACRO	CIRCULATE RIGHT 3	3
CR4	FROM,TO		MACRO	CIRCULATE RIGHT 4	3
CR5	FROM,TO		MACRO	CIRCULATE RIGHT 5	2
CR7	FROM,TO		MACRO	CIRCULATE RIGHT 7	2
CR8	FROM,TO		MACRO	CIRCULATE RIGHT 8	3
CR9	FROM,TO		MACRO	CIRCULATE RIGHT 9	4
SAM	M		MACRO	SUBTRACT A FROM M	7
SBM	M		MACRO	SUBTRACT B FROM M	7
SL2	FROM,TO		MACRO	SHIFT LEFT 2	2
SL3	FROM,TO		MACRO	SHIFT LEFT 3	3
SL4	FROM,TO		MACRO	SHIFT LEFT 4	4
SL5	FROM,TO		MACRO	SHIFT LEFT 5	5
SL7	FROM,TO		MACRO	SHIFT LEFT 7	2
SL8	FROM,TO		MACRO	SHIFT LEFT 8	3
SL9	FROM,TO		MACRO	SHIFT LEFT 9	4
SLD	I		MACRO	SHIFT A,B LEFT I BITS	2 I
SMA	M		MACRO	SUBTRACT M FROM A	4
SMB	M		MACRO	SUBTRACT M FROM B	4
SMD	M		MACRO	SUBTRACT M,M+1 FROM A,B	7
SR2	FROM,TO		MACRO	SHIFT RIGHT 2	2
SR3	FROM,TO		MACRO	SHIFT RIGHT 3	3
SR4	FROM,TO		MACRO	SHIFT RIGHT 4	4
SR5	FROM,TO		MACRO	SHIFT RIGHT 5	5
SR7	FROM,TO		MACRO	SHIFT RIGHT 7	2
SR8	FROM,TO		MACRO	SHIFT RIGHT 8	3
SR9	FROM,TO		MACRO	SHIFT RIGHT 9	4
SRD	I		MACRO	SHIFT A,B RIGHT I BITS	2 I

Progress Is Our Most Important Product

GENERAL ELECTRIC

INFORMATION SYSTEMS DIVISION

LITHO U.S.A.