GENERAL ELECTRIC
COMPUTERS

DATANET-30 Programming Reference Manual

DATANET-30
PROGRAMMING

REFERENCE MANUAL

JANUARY 1964
Rev. July 1965

GENERAL ELECTRIC
COMPUTER DEPARTMENT

PREFACE

This manual covers 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 these documents is important before proceeding into the actual programming of the DATANET-30.

This manual supersedes CPB-1019, dated January 1964.

Comments on this publication may be addressed to Technical Publications, Computer Department, General Electric Company, P. O. Box 2961, Phoenix, Arizona, 85002.

CONTENTS

CPB Number	Appendix	Title
1250	A	DATANET-30 General Assembly Program (Run on the GE-225 computer)
-	B*	Characteristics Summary
-	C*	Instruction Summary
1251	D	Computer Interface Unit (CIU931)
1252	E	Computer Interface Unit (CIU930)
1253	F	Dialing Adapter Unit (DAU930)
1254	G	Common Peripheral Channel (CPC930)
1255	H	Processor Interrupt Unit (PIU930)
1256	I	Dual Access Controller for DSU (DAC)
1257	J	Punch/Reader Unit (PRU930)
1258	K	Card Reader Unit (CRF930)
1259	L	Controller Selector Unit (CSU930) (CSU931)
1260	M	Character Word Unit (CWU931)

*This manual includes only Appendix B (Characteristics Summary) and Appendix C (Instruction Summary).

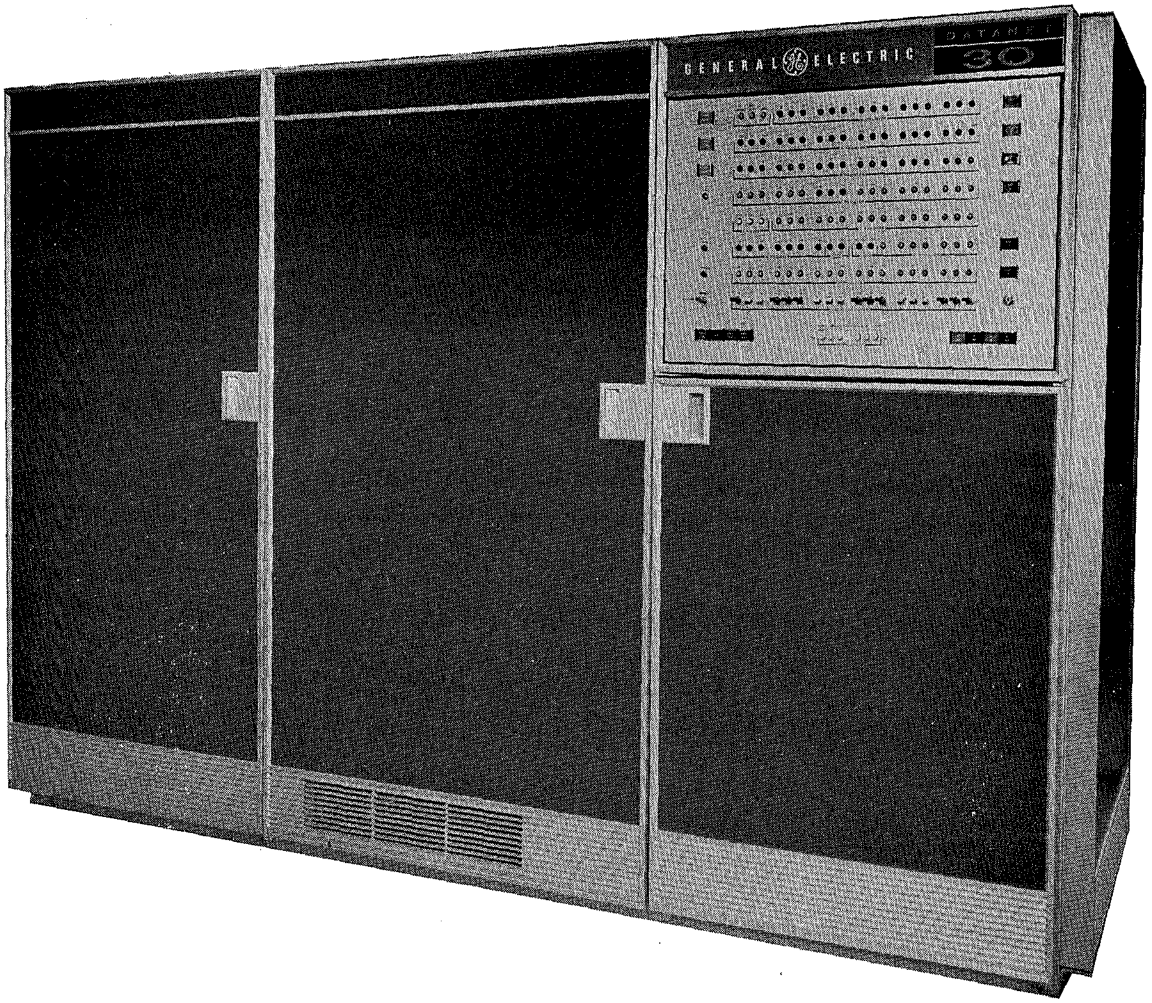
Appendices A, D, E, F,...,M are bound separately and may be ordered as they apply to individual customer equipment configurations.

The above listed manuals may be obtained from:

Publications - Distribution
Computer Department
General Electric Company
P.O. Box 2961
Phoenix, Arizona 85002

ILLUSTRATIONS

Figure Page



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. The hardware is capable of executing up to 144, 000 instructions per second.

Figure 3 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 (BBU)

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 (BBC)

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 (CWU930)

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 (CBC930)

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 (WBC930)

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 CHARACTER/WORD BUFFER UNIT (CWU931)

General

The character/word buffer unit module can contain two character buffer channels (CBC). Each CWU occupies one module space.

The Character Buffer Channel (CBC931)

The function of a character buffer is to transmit data to and receive data from a UNIVAC 1004 remote terminal or similar equipment on a character basis. Transmission to and from the remote terminal is on a bit serial synchronous basis. One character buffer provides the interface between the DATANET-30 and a half-duplex transmission line. The character buffer channel is synchronized by the digital subset connected to the transmission line.

The character buffer control unit contains hardware adaptable to the character length. The character buffers in a module may be operating at different character lengths and speeds. Speed is determined by the subset and remote terminal. It is necessary to have a synchronous digital subset on each end of the transmission line. The standard bit rates are 2000, 2400 bits per second.

The code level may be from 5 to 16 bit codes with character synchronization (no start/stop bits). The code level (character length) is selected or changed by a code level connector for each CBC. By changing code level connectors, the code level may be changed to meet changing remote terminal operations or programming techniques.

The code level connector defines the bit configuration of the synchronizing character, the number of bits per character and where the receive lines will enter the data bits into the working register (A- or B-register), that is, the high- or low-order position of the A- or B-register. The code level connector can be arranged to accept two characters (8 level) before setting the receive flag. Also, the code level connector can be arranged to mask off a bit.

THE CIU930 COMPUTER INTERFACE UNIT

For those systems requiring a combination data communication-information processing system, a CIU930 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 GE-200 Series 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 CIU930 to the processing system prior to the data transfer.

The CIU allows addressing any location in the central processor memory. The CIU930 connects into any channel of the DATANET-30 buffer selector in the same manner as any other buffer. The buffer selector address of the CIU930 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 CIU930 can connect into any GE-200 Series priority control channel. See Figure 1.

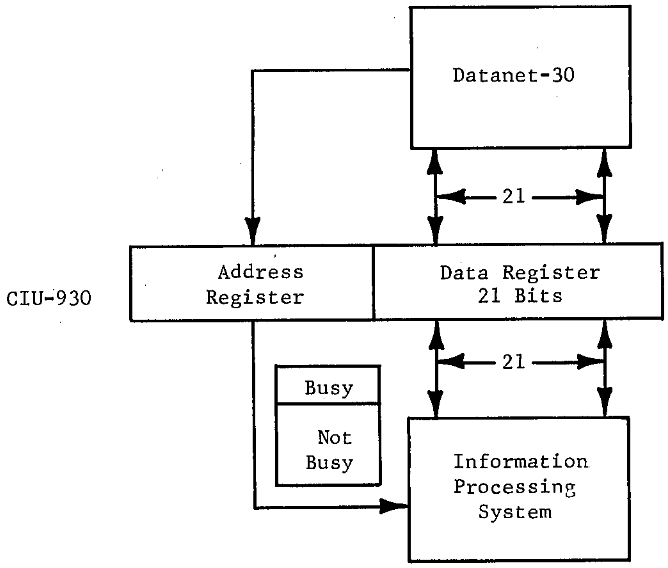


Figure 1. Computer Interface Block Diagram

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 CIU930, and whether or not it can take data from the data register.

The DATANET-30 communicates with the GE-200 Series 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. For additional information, refer to Appendix E of this manual.

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 CIU931 COMPUTER INTERFACE UNIT

The CIU931 Computer Interface Unit is an 18-bit buffer within the DATANET-30 that provides the connecting link between the DATANET-30 and a General Electric Compatibles/400 or 600 system. The CIU connects into the buffer selector of the DATANET-30 and one standard input/output channel of a GE-400 or -600 Series system. The channel may be either a word channel or a character channel. Direction of data flow is under program control.

The transfer rate is up to 39,000 characters per second or 13,000 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 GE-400 or -600 Series computer to execute programs concurrently with the transfer of data in either direction. When the CIU accepts data from a GE-400 or -600 Series computer, a signal is generated to indicate to the DATANET-30 program that service is required. The data 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 GE-400 or -600 Series computer and store the request until the latter can respond.

All data transferred through the CIU931 is parity checked for accuracy. In the event of a parity error, an appropriate signal is generated by the CIU. For additional information, refer to Appendix D of this manual.

THE CONTROLLER SELECTOR UNIT (CSU931)

The Controller Selector Unit permits connecting GE-200 Series peripherals to the DATANET-30.

Eight peripheral equipment controllers 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 controllers, numbered 0-7, operate on a priority basis, with each controller assigned a channel plug number. The controller on channel 0 has the highest priority and channel 7 the lowest. Any controller except the printer controller may be assigned to any channel plug.

The following controller selector channel priority assignment is made, assuming that all types of controllers need to be connected.

Channels 0-1 — Single-access disc storage unit controller. Dual-access disc storage unit controller. Each controller may have 4 disc storage units.

Channels 2,3,4,5 — Magnetic tape controller. Each controller may have 8 tape units.

Channels 6,7 — High-speed printer controller. Each controller may have 1 printer.

DATA COMMUNICATIONS PROCESSOR

Data Flow

The DATANET-30 is organized on an 18-bit parallel, bus logic arrangement. Figure 2 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, nonzero and odd, the branch conditions would be plus, nonzero, 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 3, the buffer selector and controller selector have been added to Figure 2. 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

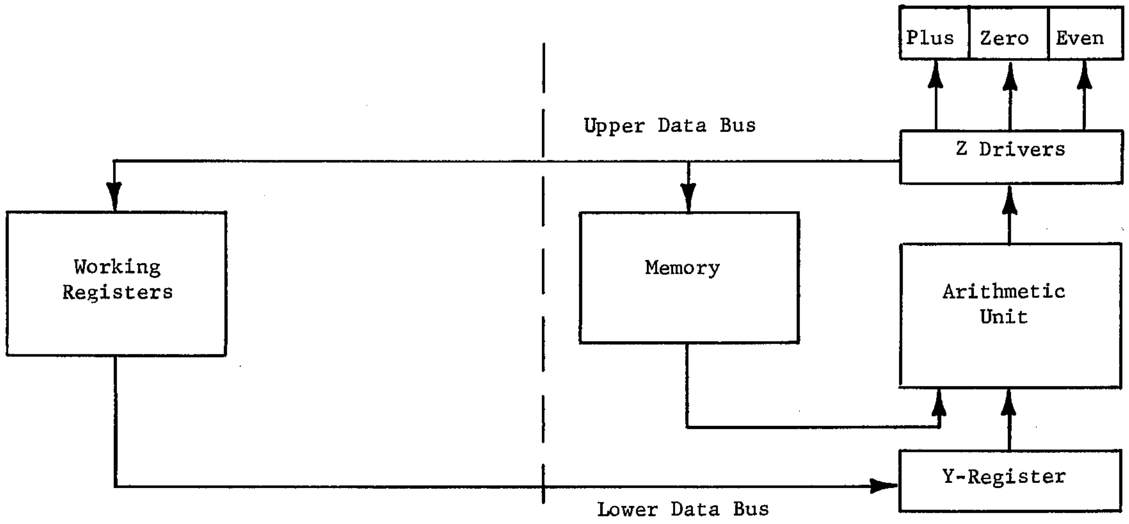


Figure 2. Basic Block Diagram

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.

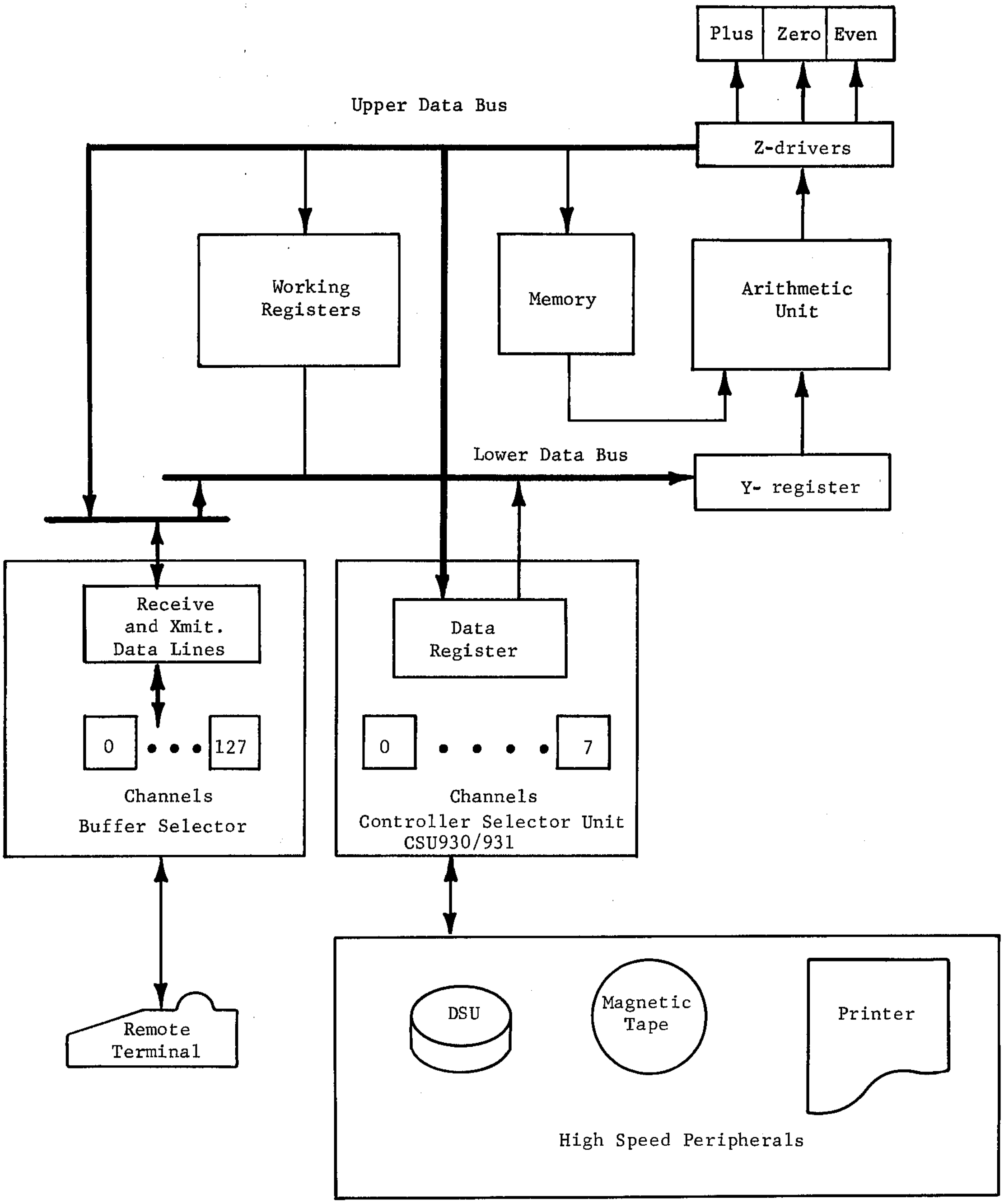


Figure 3. Basic Block Diagram

Detailed Block Diagram

The detailed block diagram (Figure 4) 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 combined in the Y-register, to be manipulated in some selected manner, and then have the result put in 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 the B-register only.

C-Register (7 bits, C, A)

The C-register is used to specify a particular input/output channel of the buffer selector. In addition the C-register 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 single-cycle mode, the register will contain the operand address of the instruction last executed.

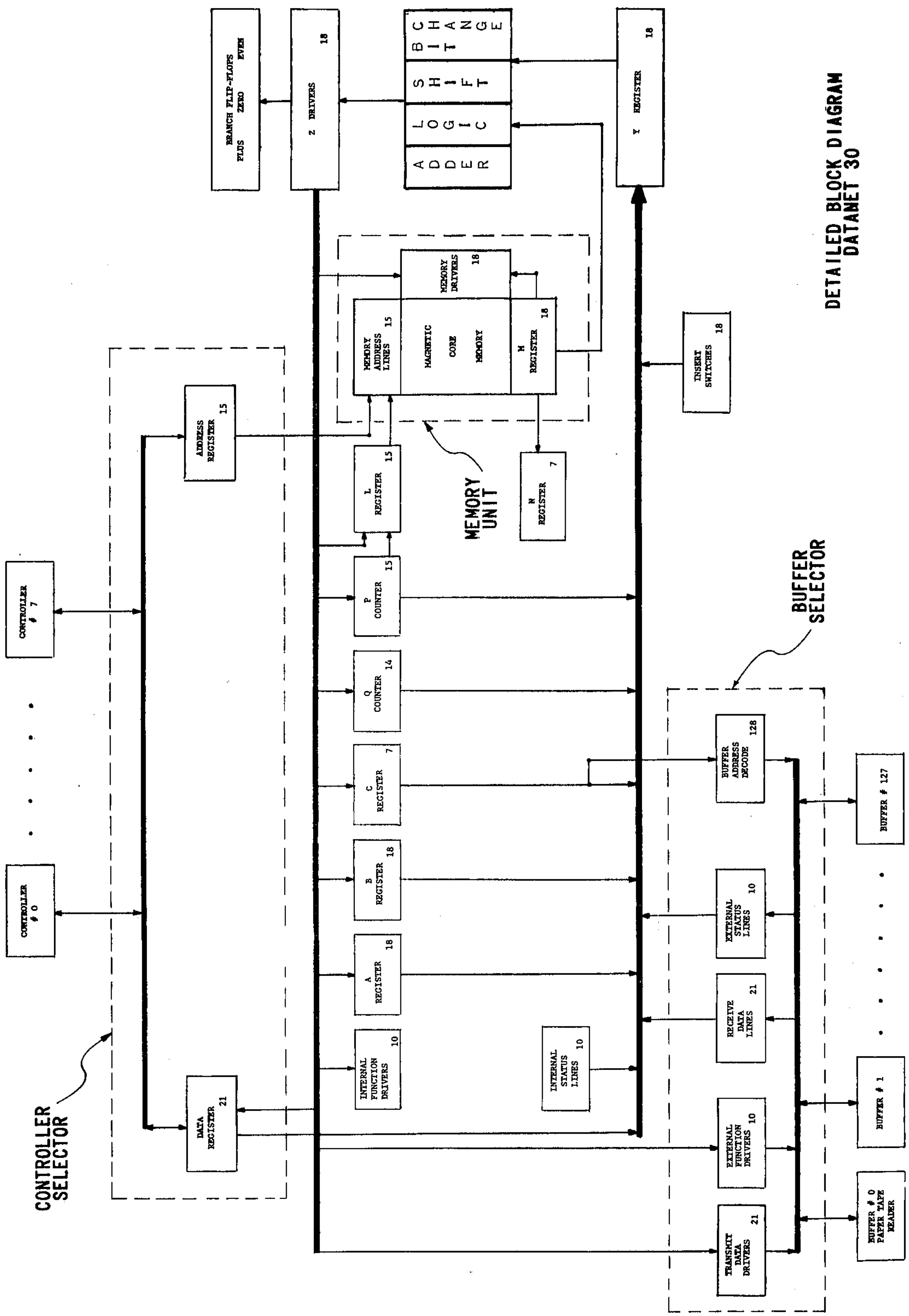


Figure 4. Detailed Block Diagram DATANET-30

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 single-cycle 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 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 the Y- and/or M-registers 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 nonbranch 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 result 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.

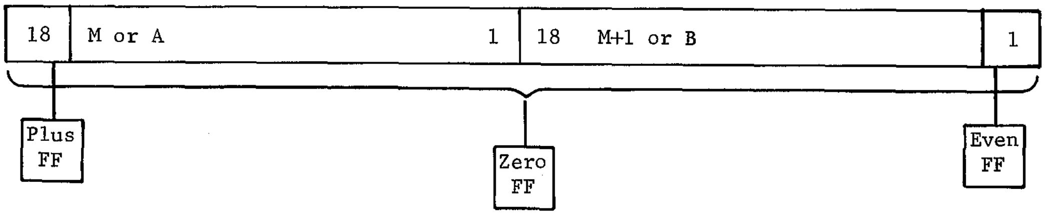
Zero Flip-Flop (1 bit, ZFF, A)

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 nonzero, the ZFF would be nonzero.

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, Chapter 4. They can be grated 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 single-cycle 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.

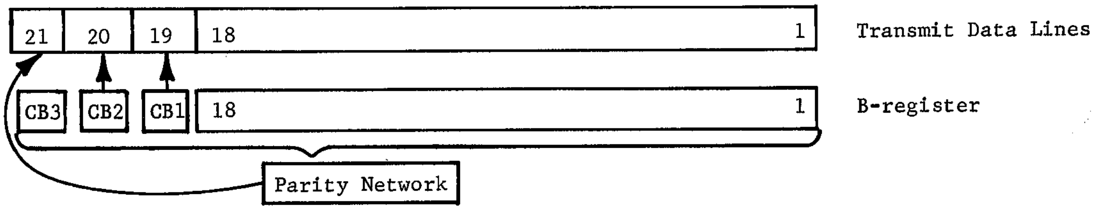
CONTROL BITS 1, 2 and 3

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
DIF1	Resets all 3 control bits		

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, 20, and 21 as required (DIF instructions) and when a Register Transfer instruction is executed, the proper parity will go to line 21.

Set CB3 for even parity in transmitted word. Reset CB3 for odd parity in transmitted word. If only 18 bits are used, reset CB1, 2, and 3 before transmitting.

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 5) 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.

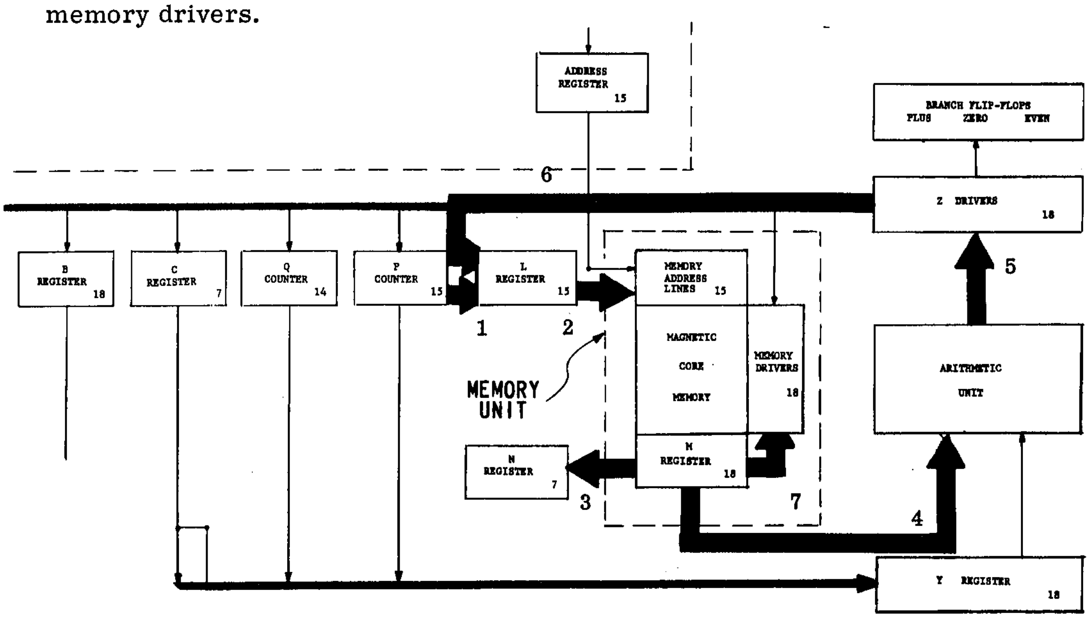


Figure 5. Detailed Block Diagram DATANET-30
Instruction Cycle

LOAD A-REGISTER (LDA) EXECUTION CYCLE. This instruction performs the function of transferring information from M to A (Figure 6):

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.

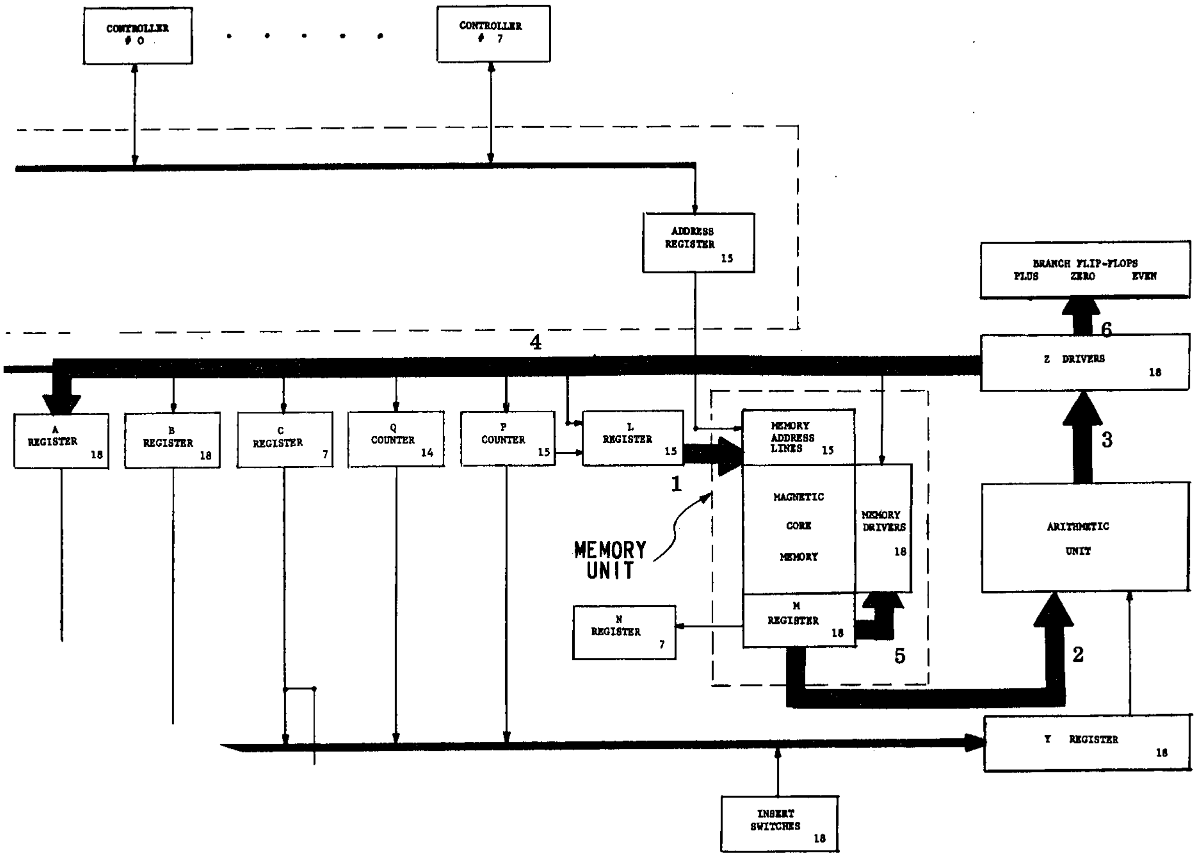


Figure 6. Detailed Block Diagram DATANET-30
Load A (LDA)

STORE B-REGISTER (STB) EXECUTION CYCLE. Information is again transferred from B to the memory (Figure 7):

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.

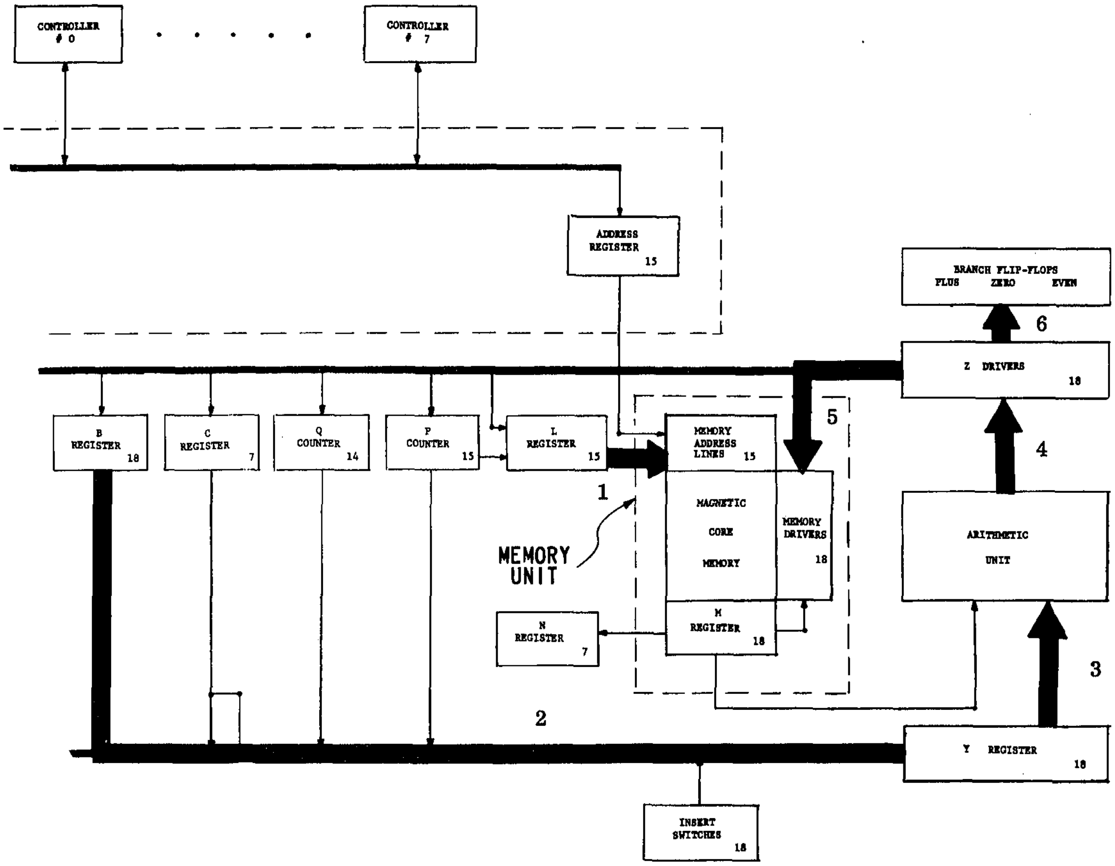


Figure 7. Detailed Block Diagram DATANET-30
Store B (STB)

ADD MEMORY TO A-REGISTER (AMA) EXECUTION CYCLE. This instruction replaces A with the sum of A and M, and regenerates M (Figure 8):

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 and A 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.

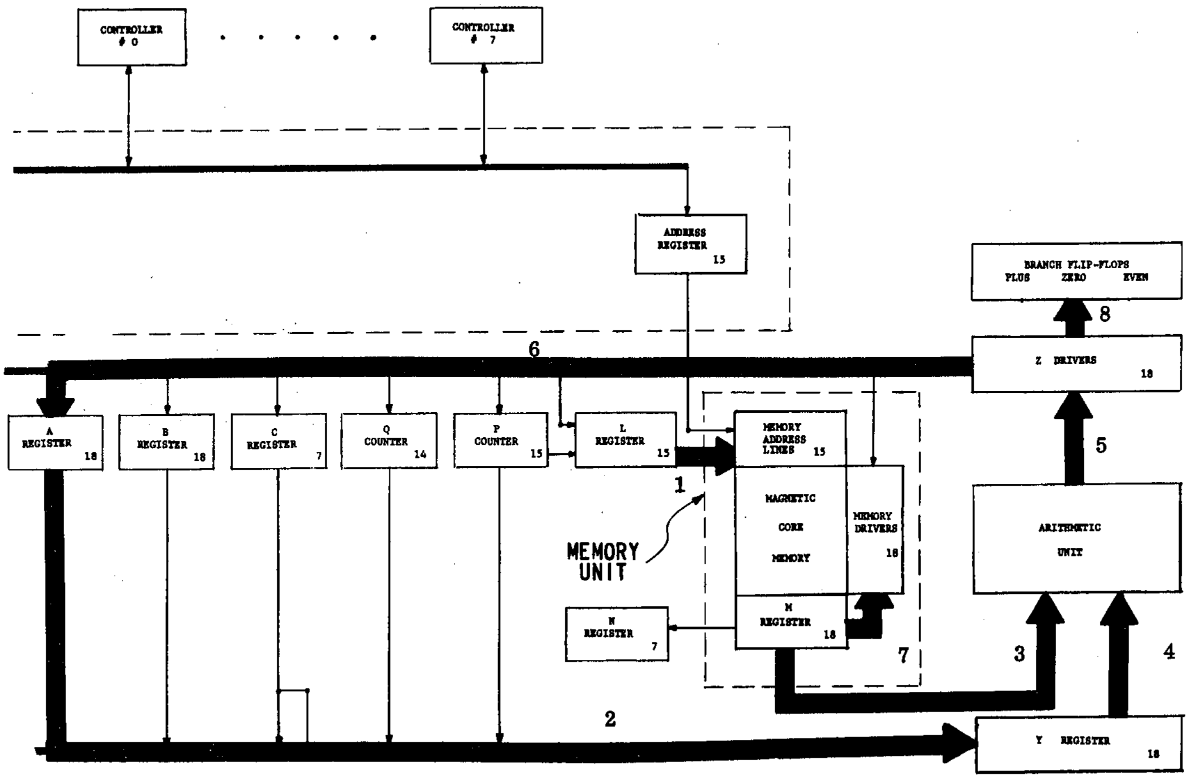


Figure 8. Detailed Block Diagram DATANET-30
Add Memory to A (AMA)

SHIFT RIGHT ONE (SR1) BR,B CYCLE. This instruction performs the Shift Right One (SR1) function in one word time (Figure 9):

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.

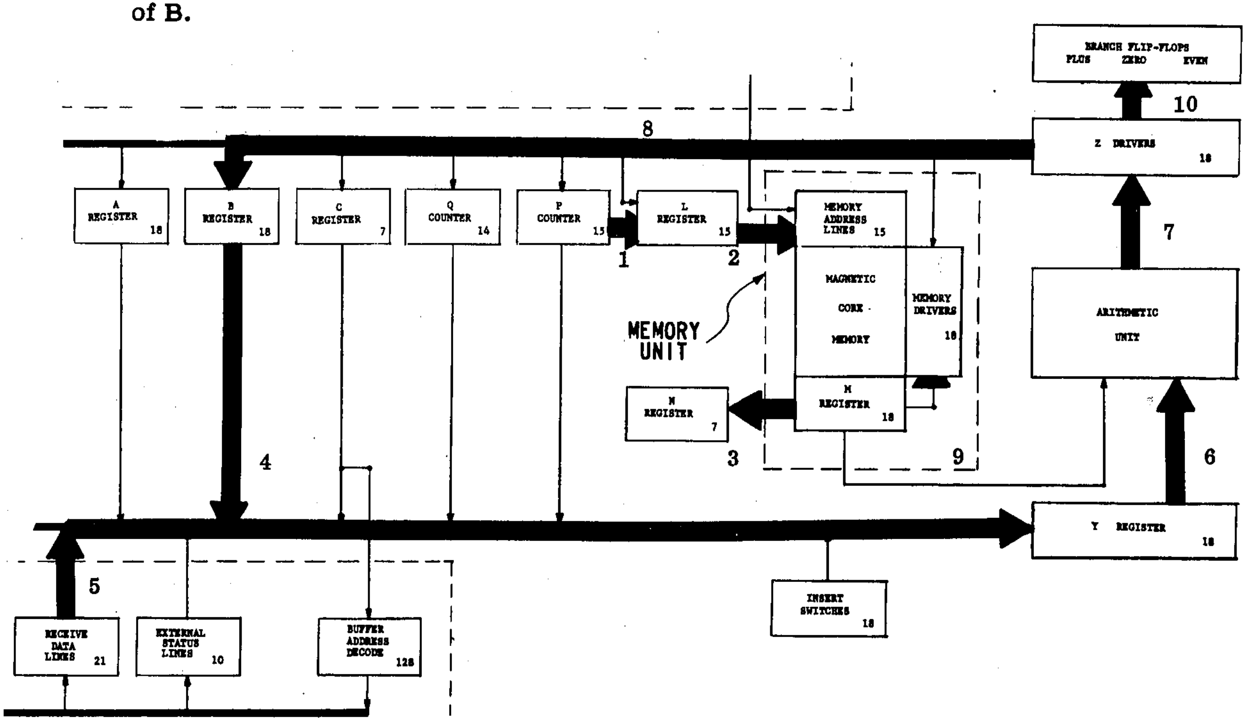


Figure 9. Detailed Block Diagram DATANET-30 Shift Right 1 Receive Lines to B-register (SR1 BR, B)

Perforated Tape Reader

The perforated 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. See Figure 10.

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 perforated 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 perforated 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 perforated 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.

Hardware Load

Hardware Load is a process whereby data is transferred from the perforated 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.

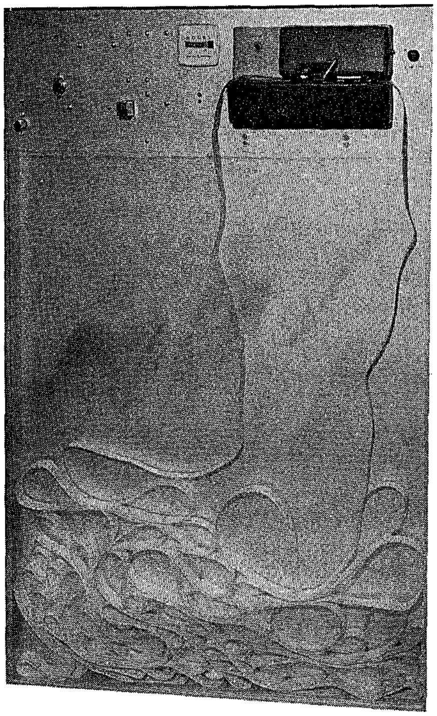


Figure 10. Perforated Tape Reader

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 is a 14-position straight binary counter. If the Q-counter is loaded with a number between 16,383 and 16,351, a Hardware Load will occur before a program interrupt.

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

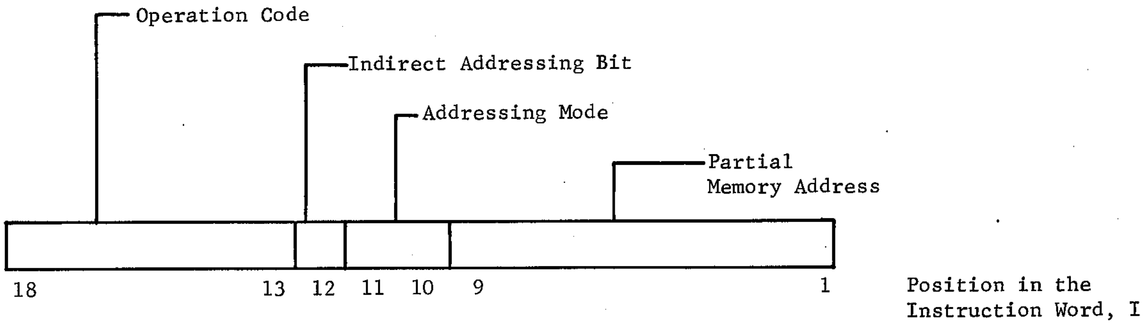
2. General instructions - Those for which the low-order bits contain information to be used by the instruction.

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 nongeneral instructions and three for general instructions (register transfer, status line and function driver, and C-register instructions).

Nongeneral Instructions

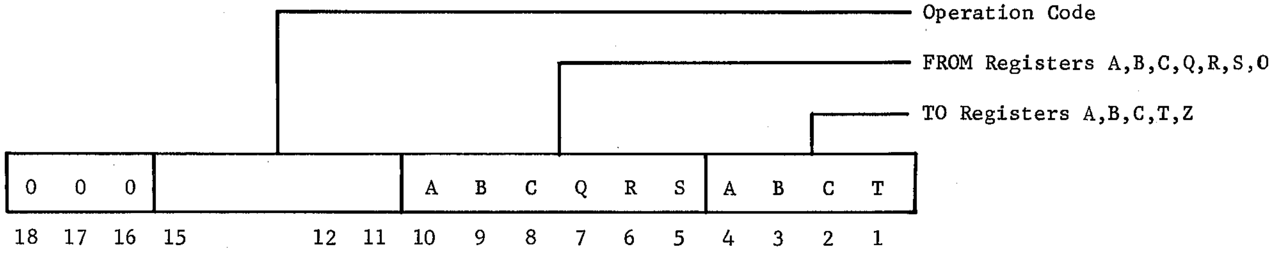
The nongeneral or memory reference, instructions have four fields:



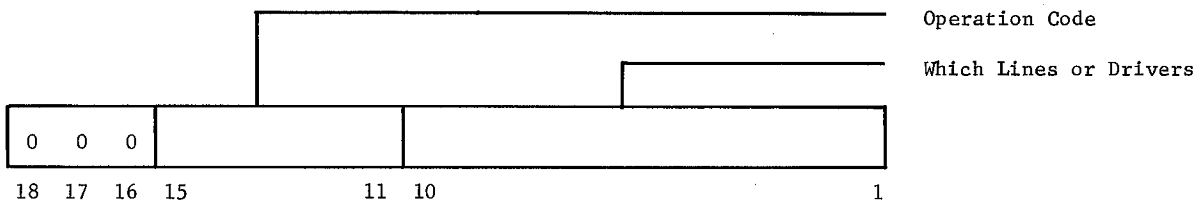
General Instructions

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

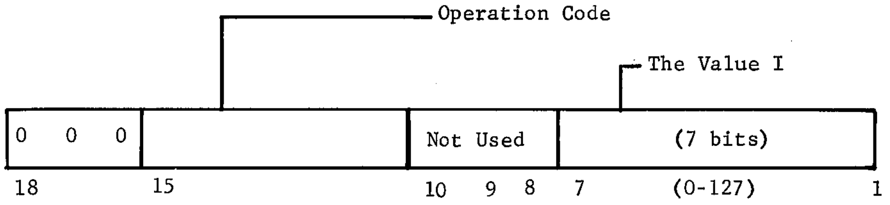
1. The register transfer instructions have three fields:



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



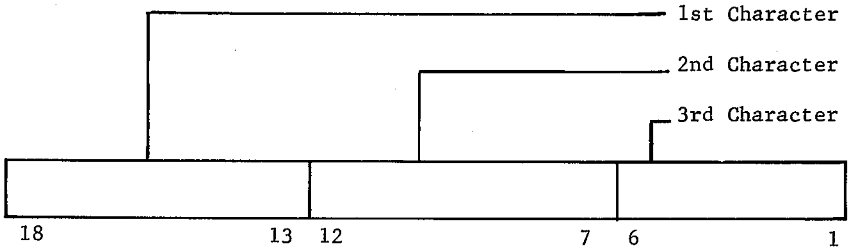
3. The C-register instructions have two pertinent fields:



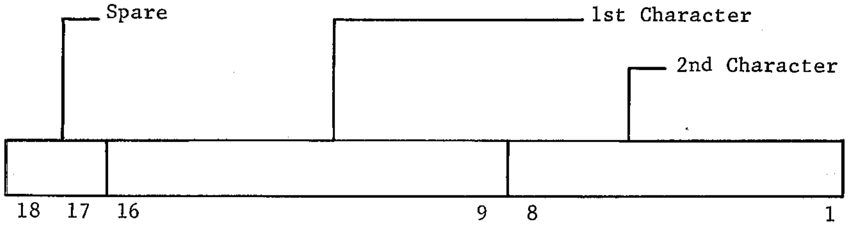
REPRESENTATION OF INFORMATION IN MEMORY

Alphanumeric Data

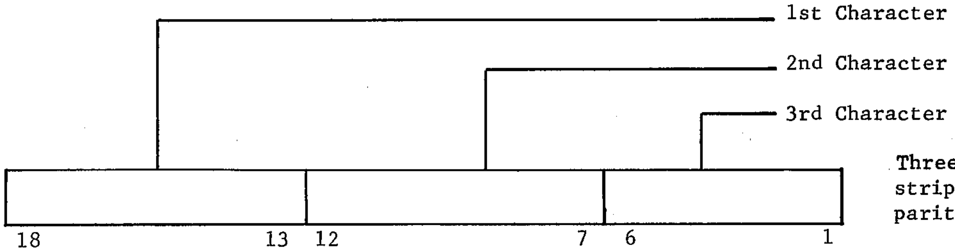
Each DATANET-30 word can contain three six-bit alphanumeric characters. The 64 possible bit combinations can be assigned to 64 symbols in any manner desired, because the DATANET-30 does not use alphanumeric data as a unique code. Therefore, other system conditions will determine the actual bit-pattern-to-symbol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



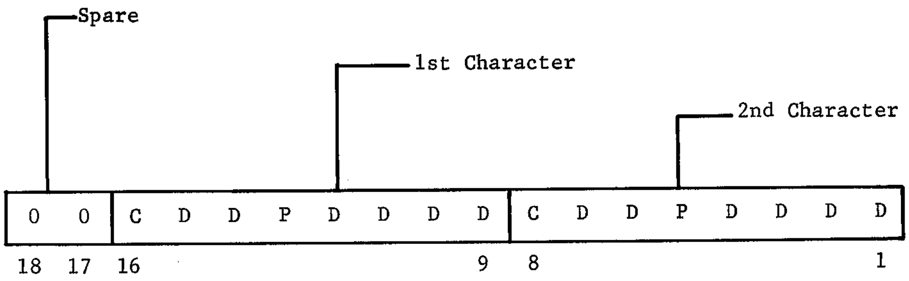
Each DATANET-30 word can contain two eight-bit alphanumeric characters. The particular code set used is dependent primarily on the remote terminals. This word might appear as follows:



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.



Three 8-level characters stripped of control and parity bits.

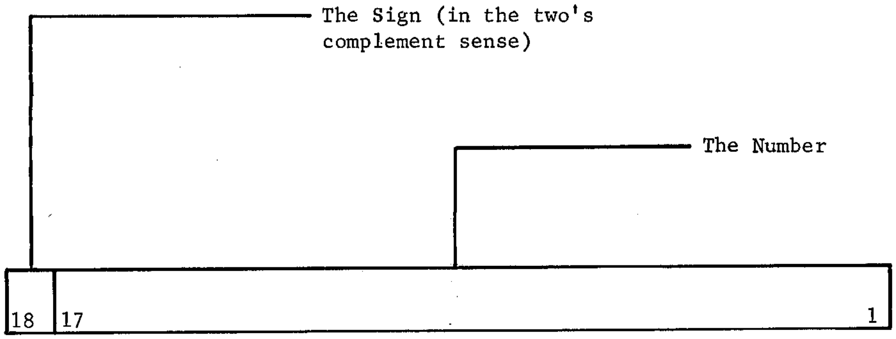


Two 8-level characters still containing parity and control bits, where:

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

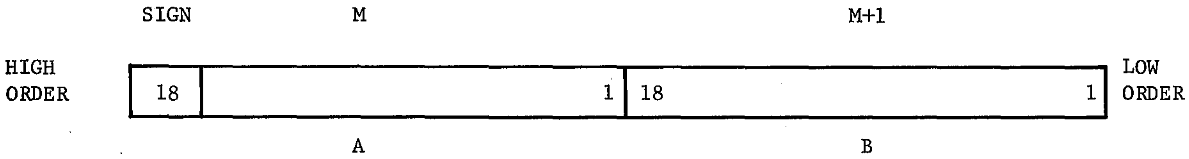
2^{16} 2^{15} 2^{14} 2^{13} 2^{12} 2^{11} 2^{10} 2^{9} 2^{8} 2^{7} 2^{6} 2^{5} 2^{4} 2^{3} 2^{2} 2^{1} 2^{0}

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	
0	00	000	000	000	000	000	-131,072 is not a valid 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 (-34,359, 738, 367 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. Option module instruction

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.

Pseudo-Operations

In addition to the machine instructions in the DATANET-30 instruction repertoire, there are a number of pseudo-operations which facilitate programming. Apseudo-operation is not a computer instruction. It is a control instruction to the 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.

Load Instructions

Mnemonic	Operand	Word Times
LDA LOAD A.	M	2 The contents of M replace the contents of A. The contents of M are unchanged.
LDB LOAD B.	M	2 The contents of M replace the contents of B. The contents of M are unchanged.
LDC LOAD C.	M	2 The contents of M (1-7) replace the contents of C. The high-order bits of M are ignored and M is unchanged.
LDD LOAD DOUBLE.	M	3 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 LOAD Q.	M	2 The contents of M replace the contents of Q. The contents of M are unchanged.
LDZ LOAD Z.	M	2 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 COMPLEMENT MEMORY TO A.	M	2 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.
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	M	Z
(A,B)		(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.
RMB	M	2
OR MEMORY TO B.		A logical OR is performed with the contents of M and the contents of B. The result is placed in B.
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	M	Z
(A,B,C)	(M,I)	(A,B,M)
0	0	0
0	1	1
1	0	1
1	1	0

Mnemonic	Operand	Word Times
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.

Mnemonic	Operand	Word Times
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.

Mnemonic	Operand	Word Times
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.

Mnemonic	Operand	Word Times
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 A and M. The result in the Z-drivers is placed only in the branch flip-flops. A 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.

TRUTH TABLES

X_1	X_2	AND	OR	XOR
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

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 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). The Q-counter is not counted down when a TRA Q instruction is executed.

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

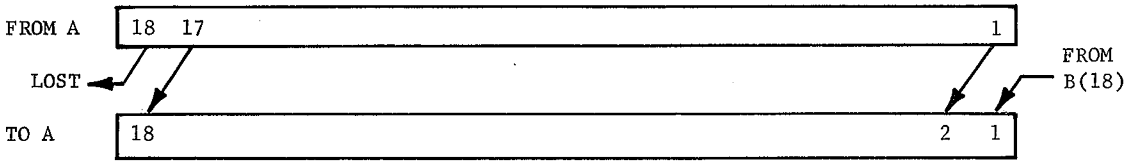
CIRCULATE RIGHT SIX.

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.

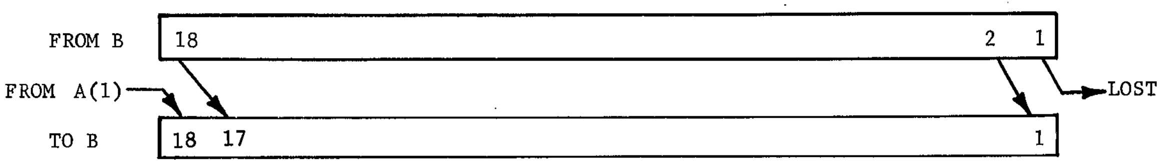
This instruction is a SL1 instruction with one added function. Bit B (18) is shifted into Bit A (1).



SRS	FROM B, TO B	1

SHIFT RIGHT SPECIAL.

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



Mnemonic	Operand	Word Times
BC0	FROM, TO	1

BIT CHANGE ZERO.

This is a special instruction for use with eight-level Friden 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 BC0. 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.

Mnemonic	Operand	Word Times
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.

Mnemonic	Operand	Word Times
BZE	M	1

BRANCH ON ZERO.

If the ZFF is zero, control is transferred to M.

Mnemonic	Operand	Word Times
BNZ	M	1

BRANCH ON NON-ZERO.

If the ZFF is nonzero control is transferred to M.

Mnemonic	Operand	Word Times
BPL	M	1

BRANCH ON PLUS.

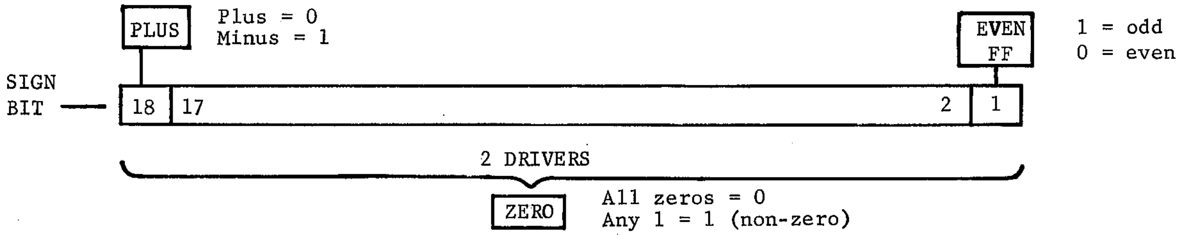
If the plus flip-flop is plus, control is transferred to M.

Mnemonic	Operand	Word Times
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	6
SUBTRACT A FROM MEMORY.	The contents of the A-register are subtracted from the specified memory location M. The result is placed in M.	
SBM	M	4
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	M	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. DIF 0 is bit position 10 in the instruction.

	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

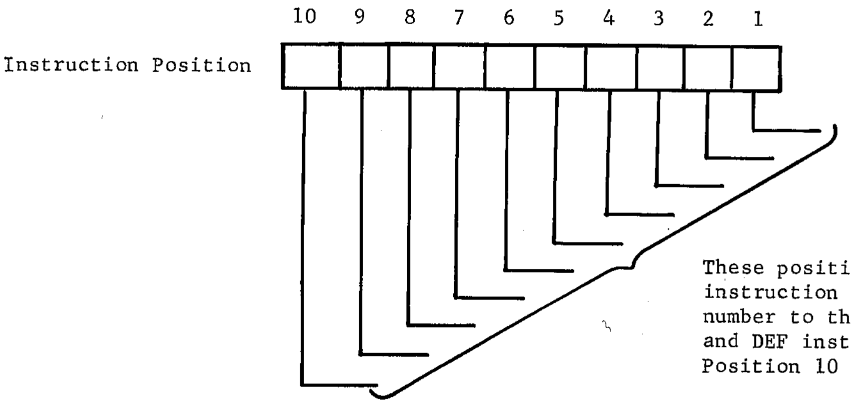
AND INTERNAL STATUS LINES TO Z.

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. NIS 0 is bit position 10 in the instruction.

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. 1 = odd parity, 0 = even parity.
NIS 2	Will be true if		The word parity output of the parity network is a 1. 1 = odd parity, 0 = even parity.
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 MANUAL/PROGRAM switch is in the MANUAL position.
NIS 5	Will be true if		Not assigned.
NIS 6	Will be true if		Status is present in memory location 6. Used only with the CSU931.
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.



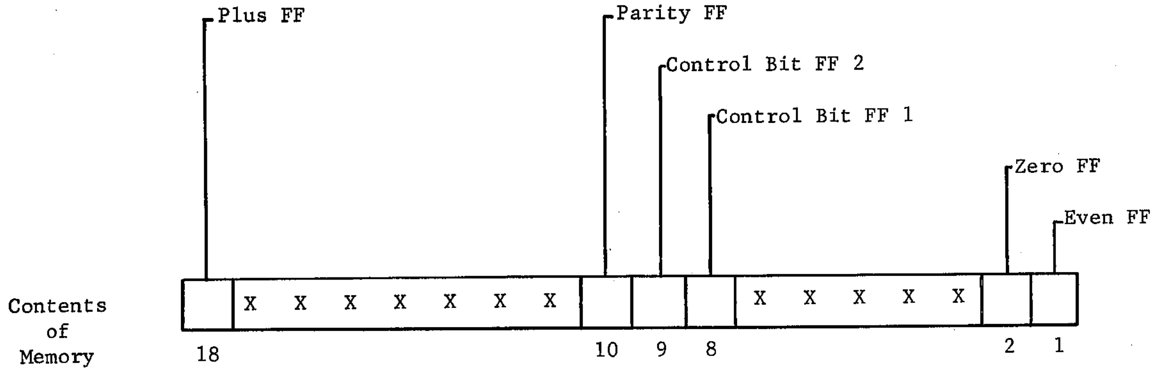
These positions in the instruction correspond by number to the NIS, DIF, NES, and DEF instruction number. Position 10 is represented by 0.

Bit Positions of NIS, NES, DEF, and DIF Instructions

Mnemonic	Operand	Word Times
LDF	M	2

LOAD SPECIAL FLIP-FLOPS.

Selected bits from the contents of M are used to restore the conditions (saved by an 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).

Mnemonic	Operand	Word Times
HLT	I	1

CONDITIONAL HALT.

If the MANUAL/PROGRAM switch is in the MANUAL position and the HALT DISABLE light is off, the DATANET-30 will halt. If the key switch is in the PROGRAM position, Hardware Load is initiated. If the key switch is in MANUAL position and the HALT DISABLE light is on, a Halt instruction is ignored.

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. Use of this instruction is restricted to certain buffers on the buffer selector.

EXTERNAL FUNCTION DRIVERS

Mnemonic	Operand	Word Times
DEF	I (1-10)	1

DRIVE EXTERNAL FUNCTION. (Bit positions 1-10 in the instruction correspond directly to the DEF instruction number. (DEF 0 is used for line 10.)

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 DEF instruction varies with the particular input/output buffer and associated equipment. Refer to the section on a particular buffer for additional information.

EXTERNAL STATUS LINES

Mnemonic	Operand	Word Times
NES	I (1-10)	1

AND EXTERNAL STATUS LINES TO Z. (Bit positions 1-10 in the instruction correspond directly to the NES instruction number. (NES 0 is used for line 10.)

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 NES instruction varies with the particular input/output buffer and associated equipment. Refer to the section on a particular buffer for additional information.

Mnemonic	Operand	Word Times
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 flag in scan word 2, field 2.

The Option Module Instructions

The option module instructions are covered in the appendix for the particular option module.

When computer type peripheral equipment is used, the instructions for the peripheral equipment are included in the appendix for the option module interfacing with the peripheral equipment.

The table of contents shows the appendix applicable to an individual option module.

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	PARTIAL 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	0	First Half Program Bank-Direct
0	0	1	Second Half Program Bank-Direct
0	1	0	Common Data Bank addressing
0	1	1	Channel Table Address
1	0	0	First Half Program Bank-Indirect
1	0	1	Second Half Program Bank-Indirect
1	1	0	Common Data Bank-Indirect
1	1	1	Channel Table Address-Indirect

DETAILED DESCRIPTION

The following descriptions of the hardware aspects of memory addressing are given for use when debugging programs. The 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 sixteen 1024-word program banks for a 16,384-word memory are listed in the table below:

Program Bank	Memory Locations				
	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
9	8192	20000	to	9215	21777
10	9216	22000	to	10239	23777
11	10240	24000	to	11263	25777
12	11264	26000	to	12287	27777
13	12288	30000	to	13311	31777
14	13312	32000	to	14335	33777
15	14336	34000	to	15359	35777
16	15360	36000	to	16383	37777

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.
```

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 common data bank addressing by the assembly program, unless the instruction itself is in the first program bank.

Location	Instruction	OPR	Operand	Remarks
	11610	ORG	5000	
11610	402024	LDA	20	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 must be symbolic and start 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
1-16	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.

If the format of the assembly program run on a GE-225 system is used, indirect addressing is specified in an instruction when an X is placed in the index column (column 20) of the coding sheet. If the format of the assembly program run on the DATANET-30 is used, indirect addressing is specified by a comma immediately following the operand.

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   Symbol   OPR   Operand   X        Remarks
              *                   ORG    2048
  4000     404030                 LDA   POINT     X     Load Register A with alpha
              *                    .
              *                    .
  4030     007760        POINT    IND   ALPHA
                                   .
                                   .
  7760     000174        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 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   107400               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 one 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. The program can branch to the subroutine and return to the point of the branch, or elsewhere, depending on the purpose of the subroutine.

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

6. 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 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; that is, common data bank, same program bank.

MEMORY ADDRESSING USING THE 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 assembly program.

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

The 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 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 = 0512).

The assembly program will interpret an asterisk (*) in the operand field on input data to mean the address of that instruction. The * serves as a flag to the assembly program and causes the performance of a special calculation to generate the desired address.

Location	Instruction
05000	LDA *+ 10

In this example, * = 05000 and the relative address *+10 will be 05012.

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 must 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 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 assembly program checks the desired address, to determine 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 1024 words, most desired addresses will be either in the common data bank or in the same program bank. The first assembly by the assembly program will indicate the addresses which need to be changed to indirect addressing.

IV. CONTROL CONSOLE

The control console (Figure 11) serves both operator and maintenance functions. The control exercised by the console is not usually 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. Repeat 4 and 5 for consecutive positions; repeat 1 through 5 for nonconsecutive positions.
7. Do 1 and 2 to set the P-counter to starting location.
8. 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.
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 <u>not</u> indicate an alert halt and the program will continue to run.

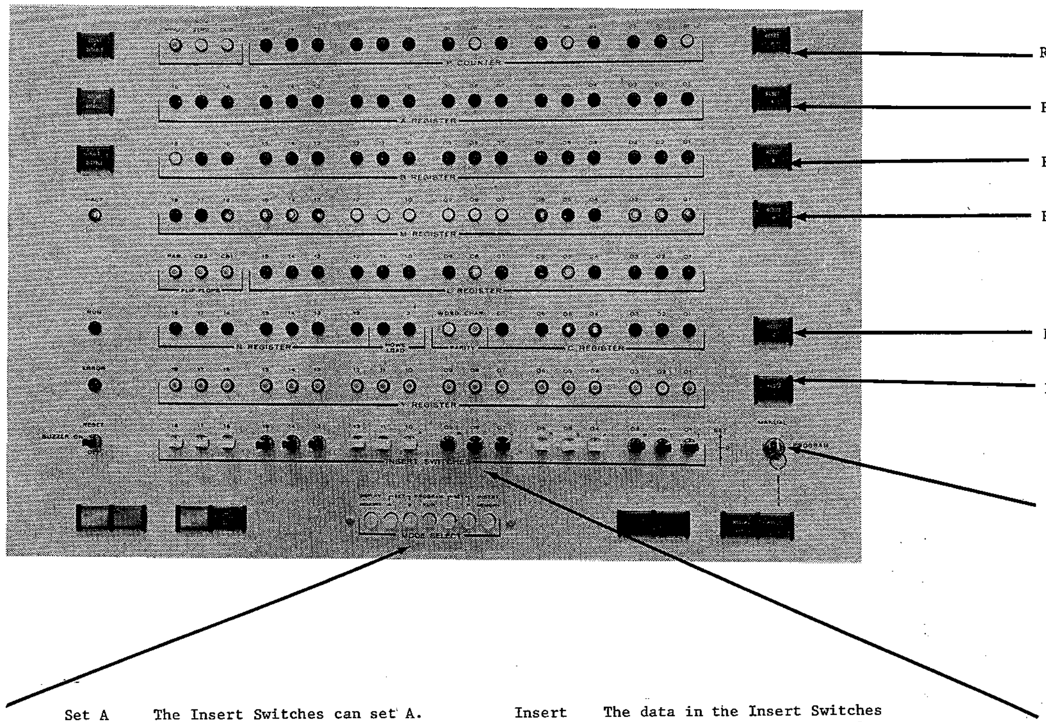
The error light and buzzer only work in either the DISPLAY MEMORY or INSERT MEMORY mode. The error light does not refer to an error in an operating program. The error light and buzzer are both turned on and off with the DIF 3 and DIF 2 instructions.

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:

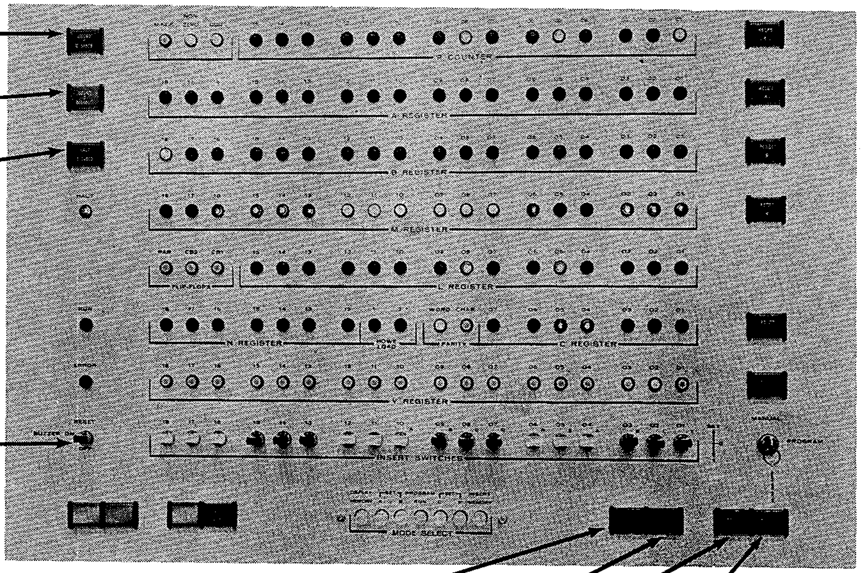
1. Turn on main circuit breaker located behind the front panel of rack 3.
2. Press AC ON button
3. Wait 10 seconds, then press DC ON button.
4. Press MANUAL RESET.



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 11. 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.
SINGLE CYCLE	Pressed	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 pressed. The action depends on the position of the mode selected.
RUN	Pressed	Starts the DATANET-30 program running continuously, if in the Program Run, Display Memory, or Insert Memory mode.
MANUAL RESET	Pressed	Resets all registers, counters, flip-flops, and optional interface units.
MANUAL LOAD	Pressed	Initiates the Hardware Load process, if the Program Run button has been pressed.

Figure 11. Control Console Switches (con't)

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

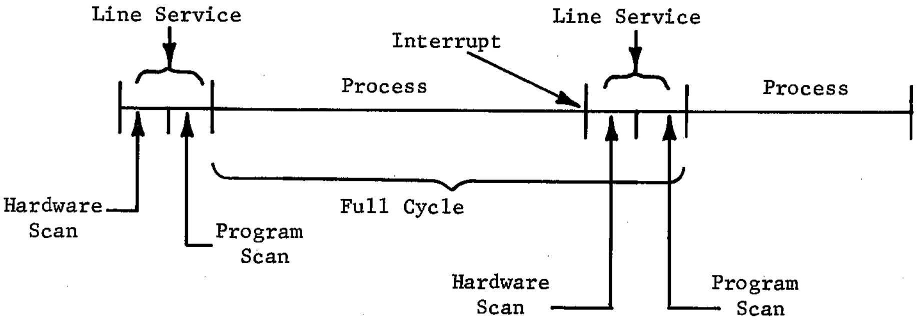
3. Assemble characters into words
4. Assemble words into blocks
5. Assemble blocks into messages
6. Assign message routing
7. Disassemble blocks into words for transmission
8. Disassemble words into characters
9. Put the character in the buffer for transmission.

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.

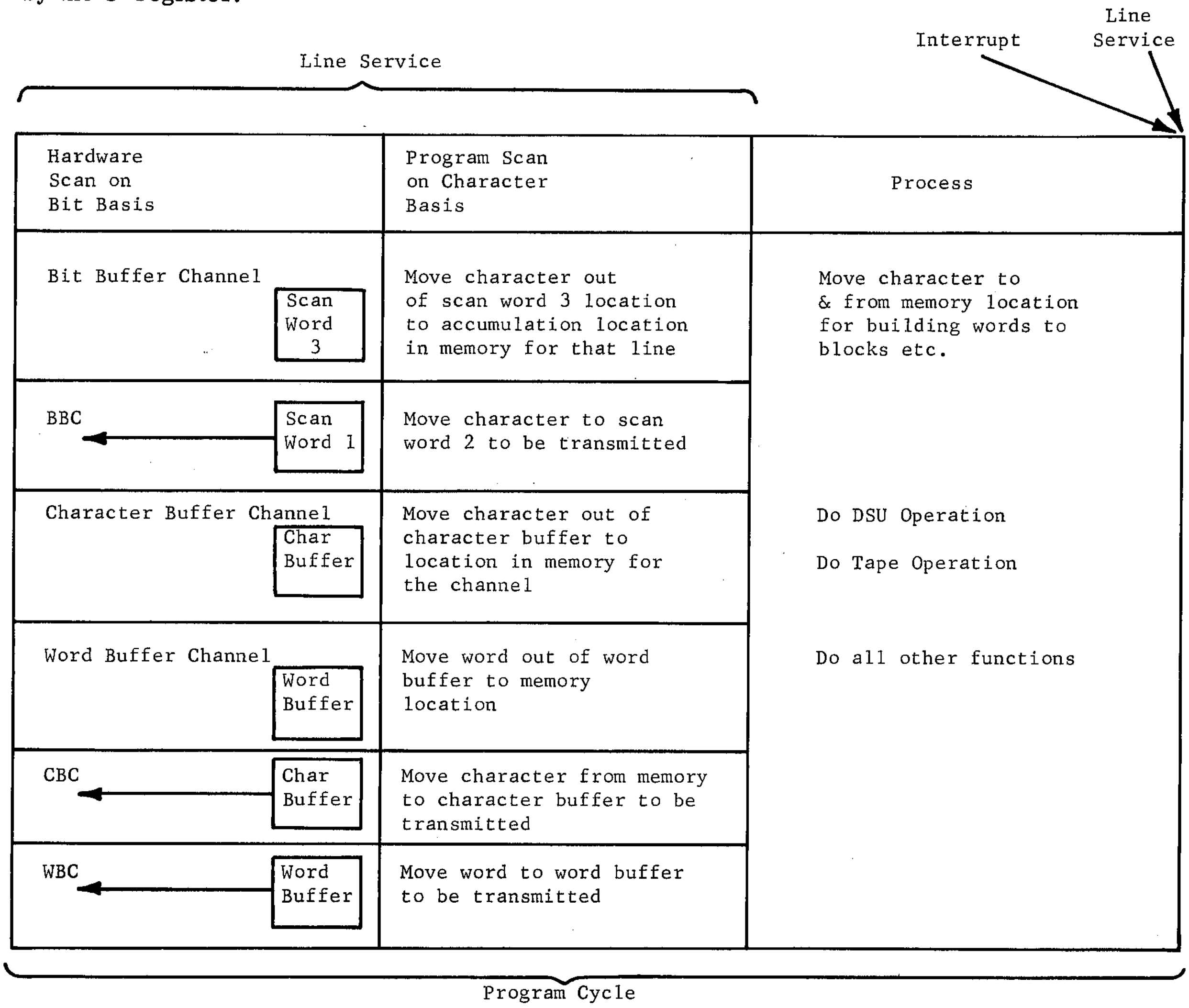


The time will vary with the line service rate required by the remote terminals. One full cycle must therefore be completed at a rate slightly faster than the fastest service rate. In order to do this, processing must be interrupted to allow the hardware scan instruction to service the lines (3 word times 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.
BBC ← Scan Word 1	Move character to scan word 2 to be transmitted	
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
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 12. General Timing Diagram

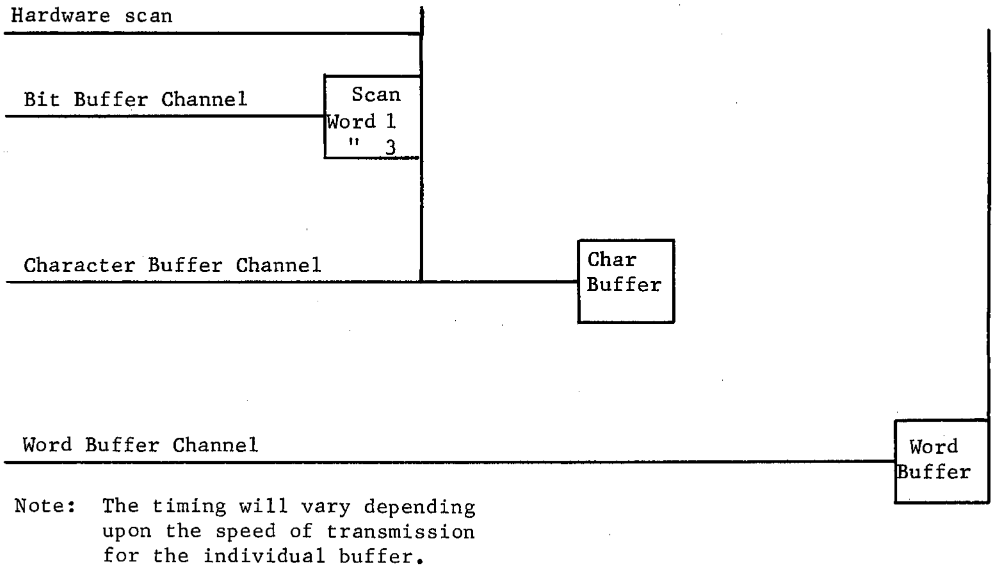


Figure 13. Relative Timing for Scanning Buffers

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.

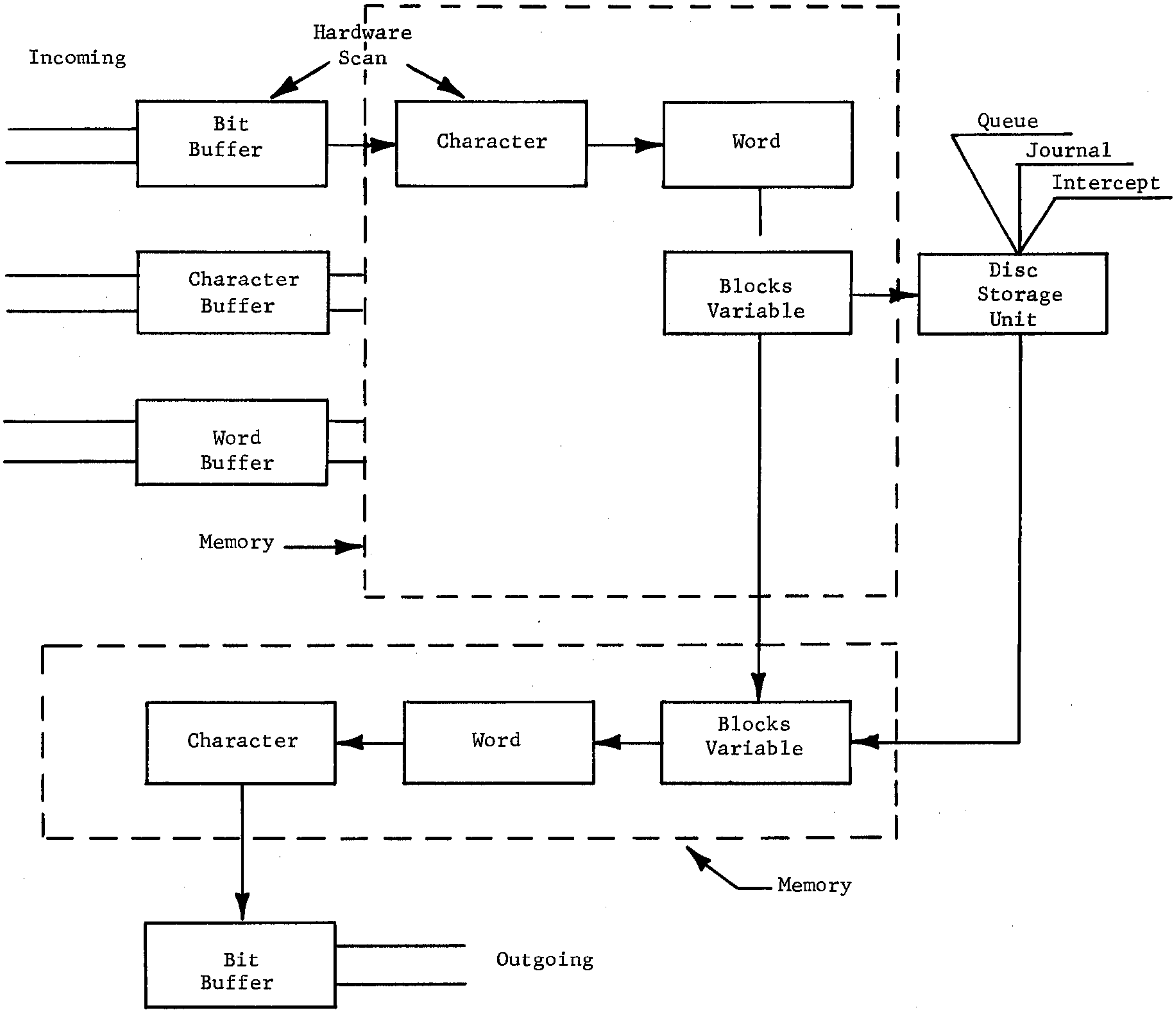


Figure 14. Data Flow Functional Block Diagram

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 a symbol has been referred to by a double-length instruction before the symbol is defined, the symbol will be forced to an even location. Zeros are 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 by the DEF instructions. Information as to the status of a buffer is tested by the NES instructions.

The bit buffers are available in two models: BBC930G1 and BBC931G4. Each model can interface with a Voltage Current Adaptor (VCA), a 103A or 103F data set. The DEF and NES instructions for a bit buffer are different, depending on the model used and the type of line interface.

DEF Instructions BBC930G1,G2

1 - Reset receive flag and buffer
2 - Reset transmit flag and buffer
3 - Reset request to send (103F)
Reset data terminal ready (103A)
4 - Set request to send (103F)
Set data terminal ready (103A)
5 - Reset receive clock
6 - Set Echoplex mode
7 - Reset Echoplex mode
8-0 - Not used

NES Instructions

1 - Receive data flag set
2 - Transmit data flag set
3-8 - Not used
9 - Test receive line
0 - Not used

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 to position 18 of Y and then according to the register transfer instructions. The receive buffer and flag are reset.
Register Transfer	, T	
TRA	FROM, TO	Bit 1 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. I is the number of the starting channel. N is the number of bit buffers scanned.

DEF Instructions BBC931G4, G5

1 - Reset receive flag and buffer
2 - Reset transmit flag and buffer
3 - Reset request to send (103F)
Reset data terminal ready (103A)
4 - Set request to send (103F)
Set data terminal ready (103A)
5 - Reset receive clock
6 - Set restraint (103A)
Set originate mode (103F)
7 - Reset restraint (103A)
Reset originate mode (103F)
8-0 - Not used

NES Instructions

1 - Receive data flag set
2 - Transmit flag set
3-4 - Not used
5 - Data set ready (interlock on)
6 - Carrier off (103F)
7 - Not used
8 - Supervisory receive data
9 - Test receive data
0 - Not used

The DEF and NES instructions for other bit buffers not shown here are furnished on an individual basis.

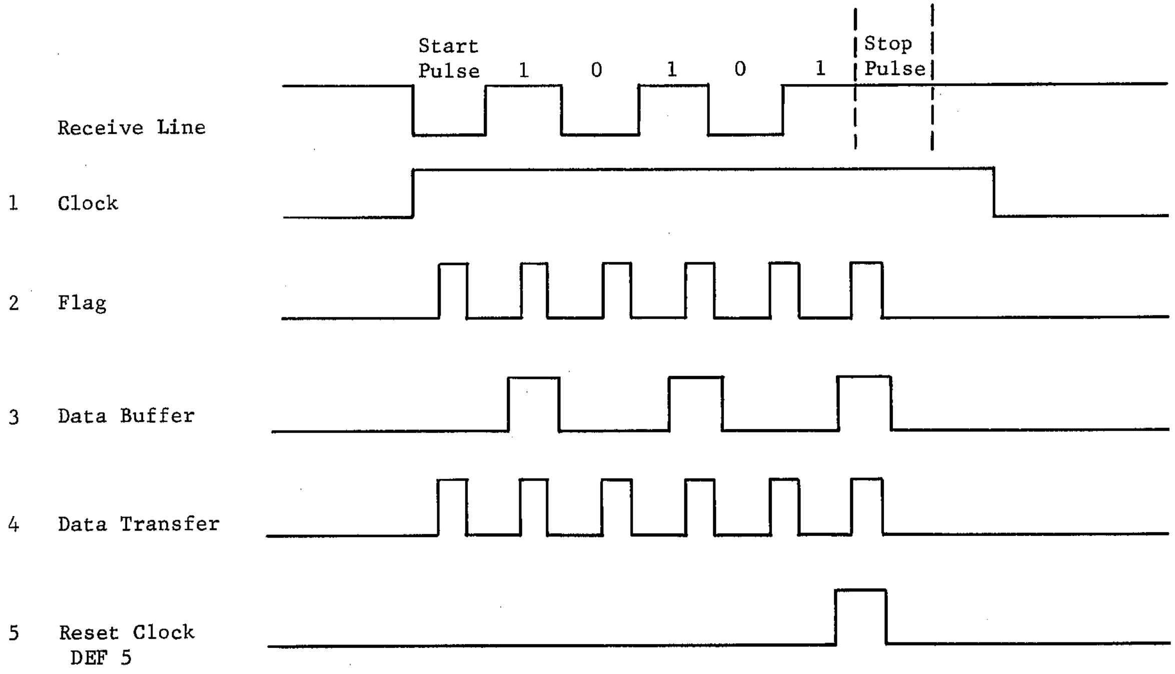
RECEIVE OPERATION

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.

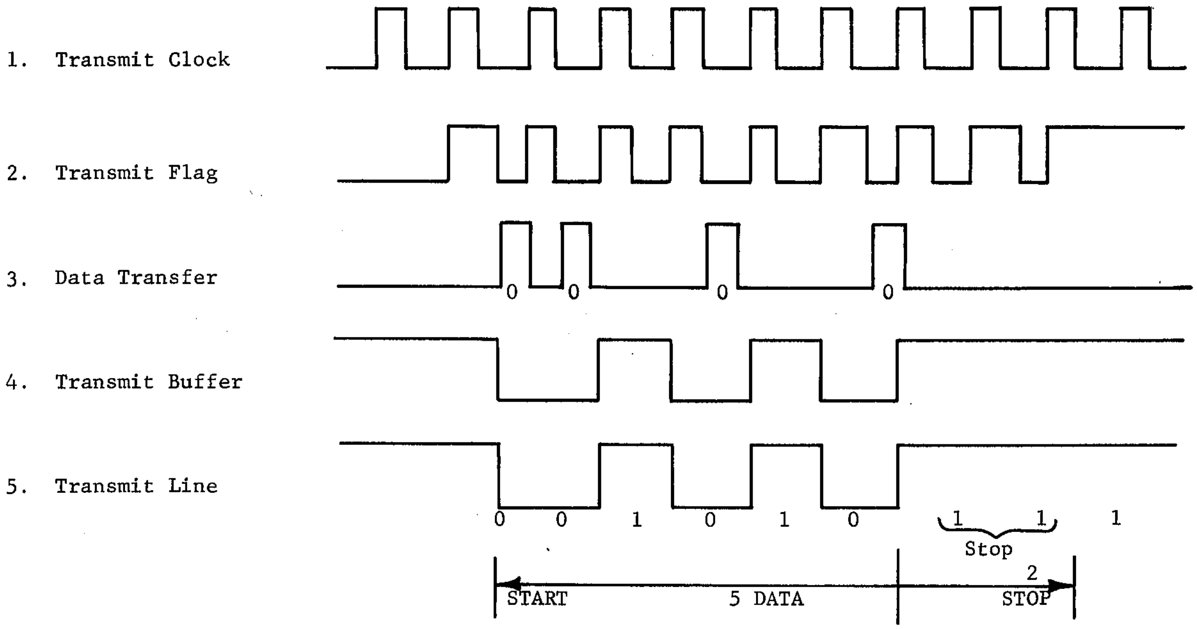
1. The transmit clock occurs every bit period as specified by the data timing unit.

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:

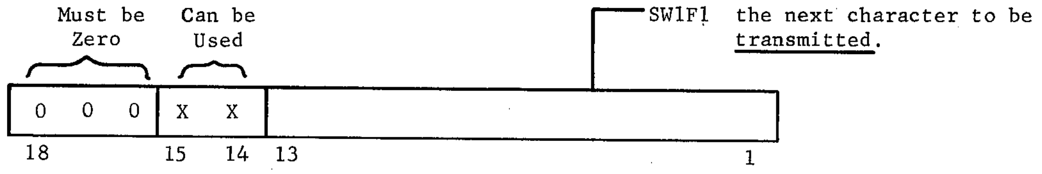
Word Times = $1+3N$.

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

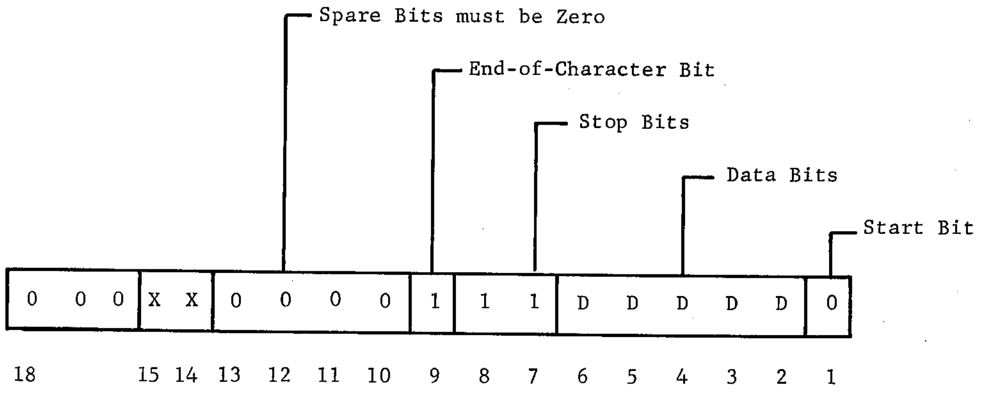
The SCN instruction uses the A and B registers, and the previous contents will be destroyed. Also, the C-register will contain C_f after it is completed. At the end of a Transmission, the last word placed in scan word one continues to be transmitted. It is necessary to put a word of all marks in scan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

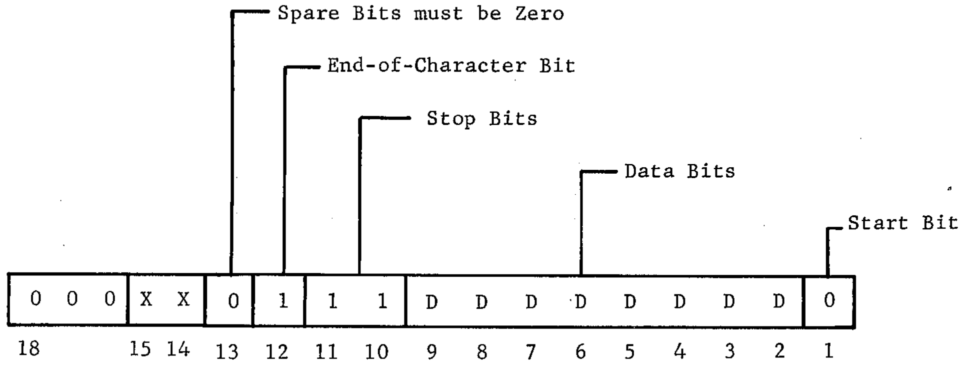
Scan Word 1



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:

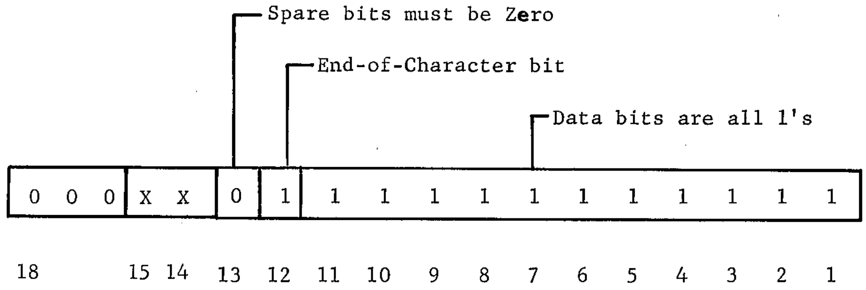


The format for 8-level, 11-unit codes is:



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

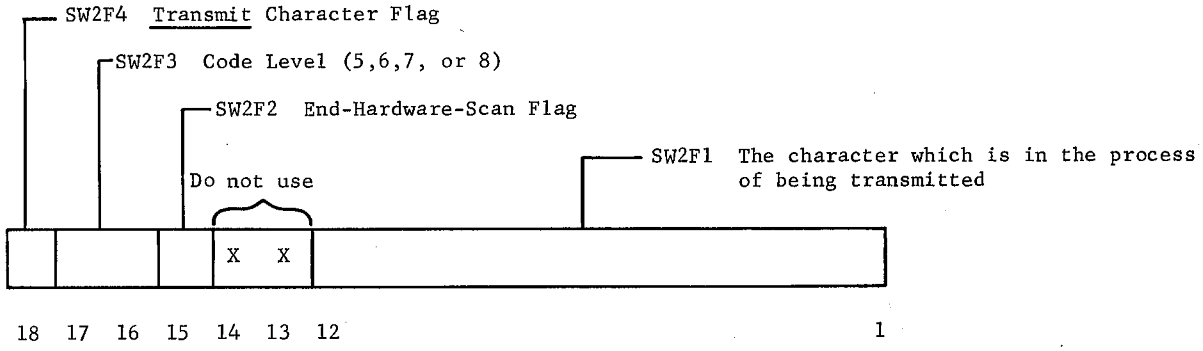
It is sometimes necessary to transmit one or more fill characters. A delay time of one character is a marking condition on the line for one character time. This can be achieved by making the start bits, data bits, and stop bits all 1's. This should also be the last character transmitted at the end of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

To initialize scan word 1, the start, data, stop, and end-of-character bits must be set to all ones and the rest must be zero filled.

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
Do not use
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 in Model 1 and at channel 127 in Models 2 and 3.

SW2F3 defines for receive purposes the code level of the line (5, 6, 7 or 8). This is set when the program is initially assembled (or changed octally) and thereafter need not be considered.

SW2F3 (Bits 17 and 16)

17	16
0	0 = 5-level code
0	1 = 6-level code
1	0 = 7-level code
1	1 = 8-level code

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.

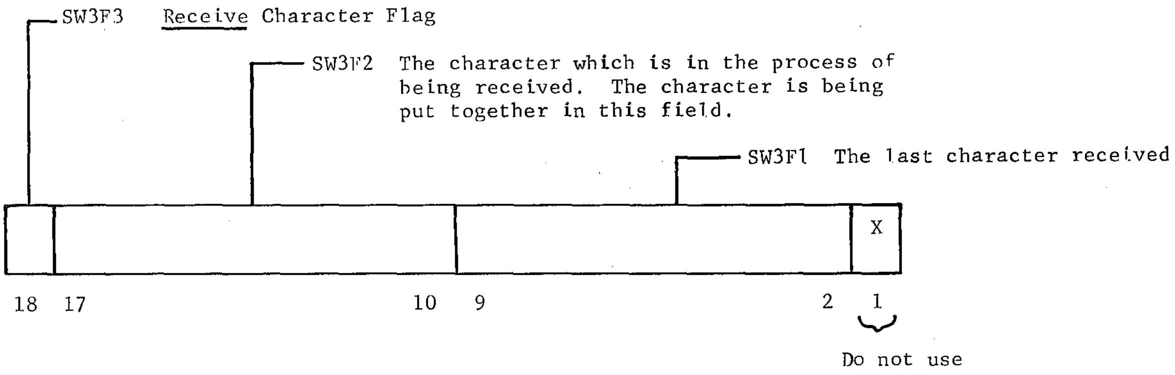
1 = SW1F1 is ready for a new character
0 = SW1F1 is not ready for a new character.

To initialize scan word 2, bit positions 1-12 are set the same as in scan word 1. Bit position 15 is set in the last line to be scanned. Bits 17 and 16 must be set in each scan word in accordance with the code level for that line and bit 18 is always zero.

Scan Word 3

SW3F1 is set by the hardware when SW3F2 receives a full character as defined by SW2F3. The data bits of SW3F1 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
Do not use

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.

Initialize scan word 3 as all zeros.

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.

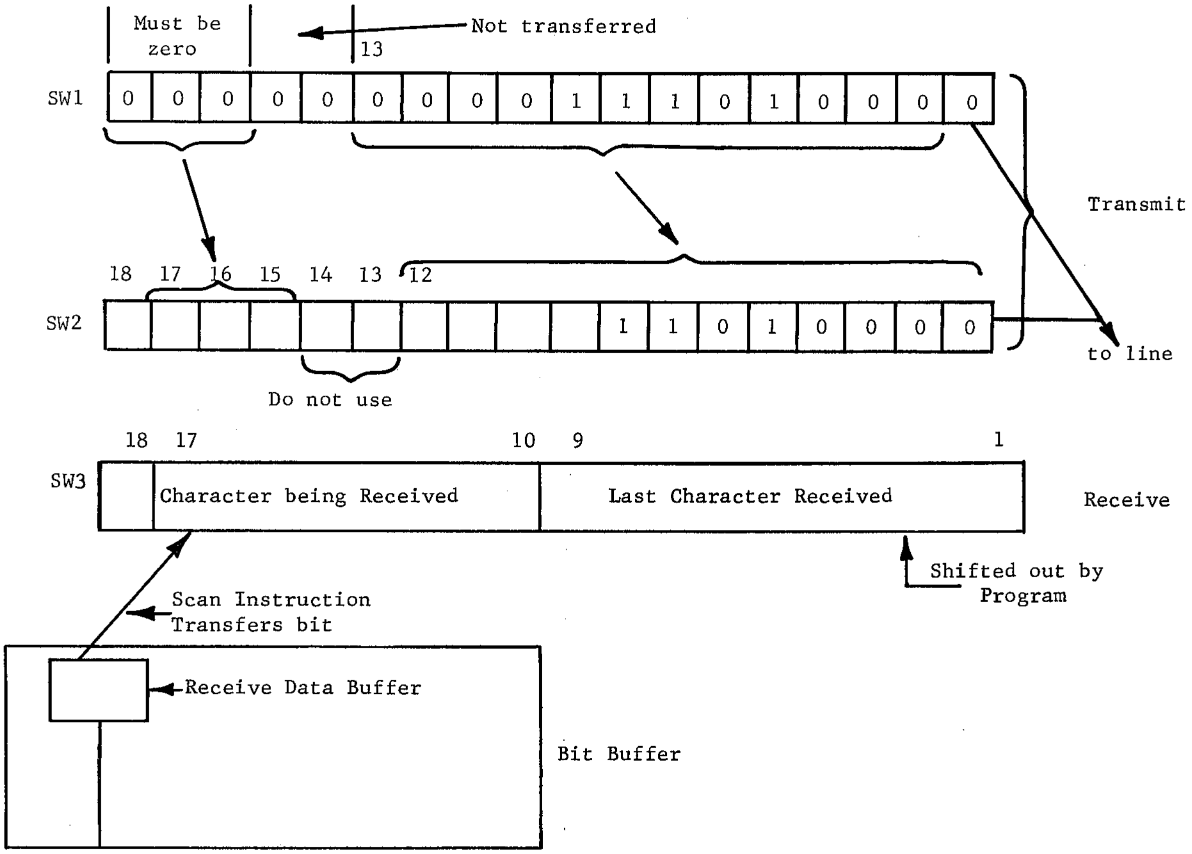


Figure 15. Hardware Scan Block Diagram

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.

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
  STD      WS2                  Store A and B
  STC      WS3                  Store C
  LDQ      Count                Load Q with new value
```

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, and therefore is rarely used.

```
Location   Instruction     Symbol    OPR    Operand     Remarks

             15530                   ORG    7000        ORIGIN LOCATION 7000
 15530      011006                   PIC    6           ADDRESS OF BIT BUFFER
 15531      600000         RECVE     LDB    BIT7        BIT NUMBER SEVEN
 15532      022001                   NES    1           RECEIVE FLAG SET
 15533      121531                   BZE    *-1         NO, GO BACK
 15534      042444                   SR1    BR,B        YES, SHIFT NEW BIT TO B-REGISTER
 15535      160000                   BEV    RECVE+1     COMPLETE CHARACTER NOT IN, GO BACK
 15536      026020                   DEF    5           CHARACTER IN, RESET RECEIVE CLOCK
```

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      011006                 PIC    6         ADDRESS OF BUFFER
 03722      022002       XMIT      NES    2         TRANSMIT FLAG SET
 03723      121721                 BZE    *-1       NO, GO BACK
 03724      060401                 TRA    B,T       TRANSFER BIT TO TRANSMIT DATA DRIVERS
 03725      042404                 SR1    B,B       SHIFT B-REGISTER RIGHT ONE
 03726      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

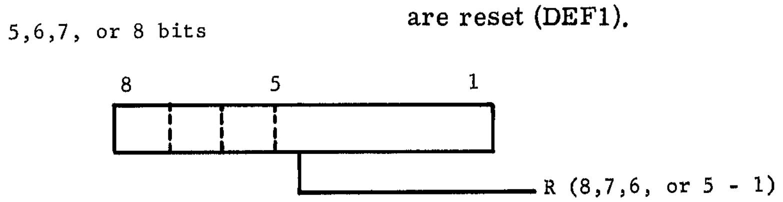
Mnemonic — **Operand** — **Word Times**

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.

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 to the least significant positions in Y and then in accordance with the register transfer instructions. The receive data buffer and flag are reset (DEF1).



```
Mnemonic                                          Operand                       Word Times

DEF                                                  I                         \     1
  1 - Reset receive flag and data register    \                                |
  2 - Reset transmit flag and data register   } VCA     \ 103A/F               |
  3 - Reset request to send                   /         } 201A/B               } 202C/D
  4 - Set request to send                              /                       |
  5 - Set supervisor transmit data                                             |
  6 - Reset supervisor transmit data                                           /
7-8 - Not used
  9 - Set data mode        \ 103A
  0 - Reset data mode      / 202C/D

NES                                                  I                               1

  1 - Receive flag set (data register contains a new character   \ 103F    \ 103A  \ 202C/D
  2 - Transmit flag set (data register is ready for a new character } 201A/B |      |
                                                                 / VCA      |      |
  3 - Call in progress                                                     /       |
  4 - Request answer                                                               /
  5 - Data mode        \ 103A/F
  6 - Carrier on       } 201A/B
  7 - Clear to send    / 202C/D
  8 - Supervisory receive data--202C/D only
9-0 - Not used
```

The two models of this character buffer are CBC930 and CBC930G2. The character buffer channel can interface with a Voltage Current Adaptor or telephone company data sets 103A/F, 201A/B, or 202C/D. The DEF and NES instructions for a CBC vary depending on the model used and the line interface. Model CBC930G2 has all the DEF and NES instructions shown above. Model CBC930 has all instructions shown except DEF 5, DEF 6, and NES 9. The variations according to the subset are indicated.

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 16) 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.

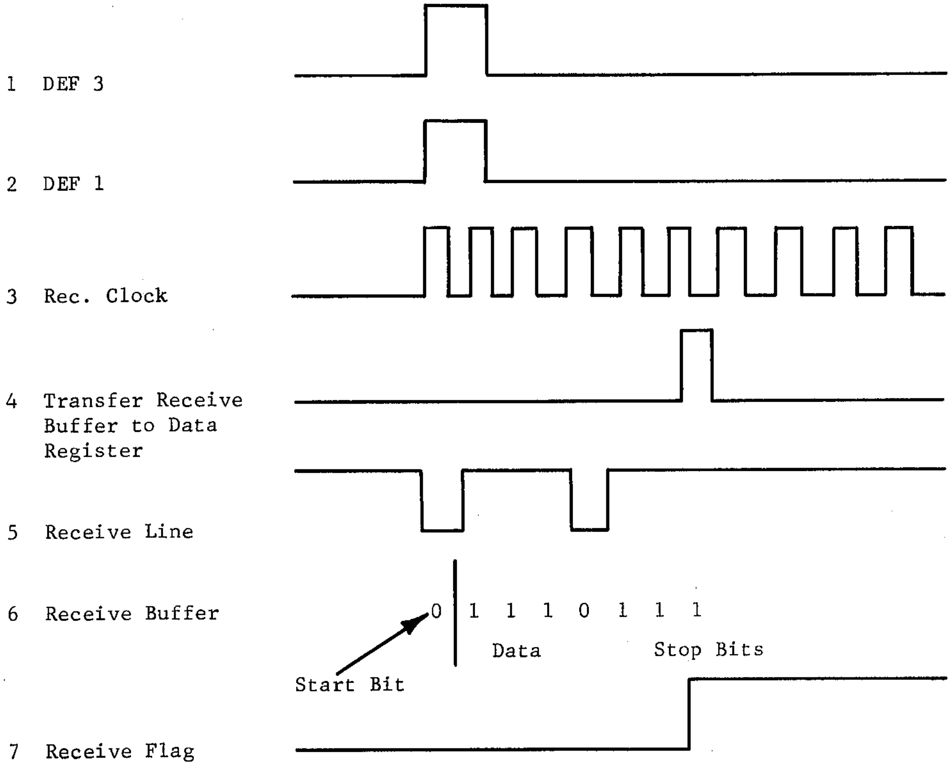


Figure 16. CBC Receive Timing Diagram

TRANSMIT OPERATION

Assume that the program has put the CBC in the transmit mode, the CBC 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 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.

The timing diagram (Figure 17) illustrates graphically what happens when a 5-bit character is transmitted onto a communications line by a character buffer channel.

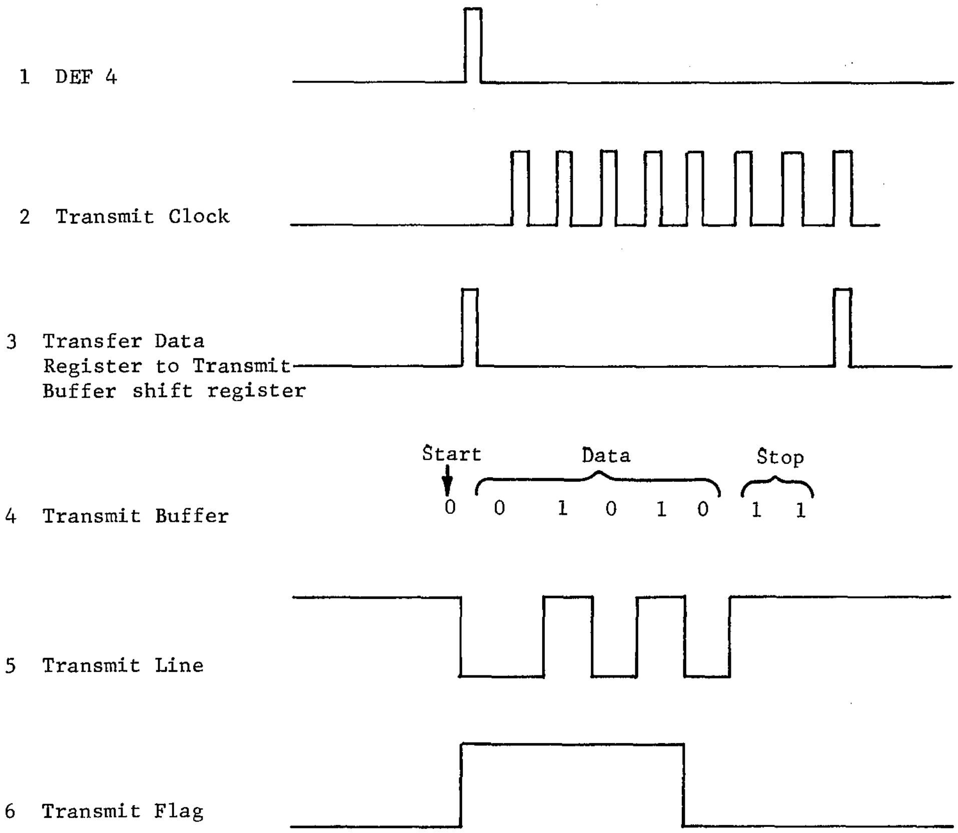


Figure 17. CBC Transmit Timing Diagram

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 testfails 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.

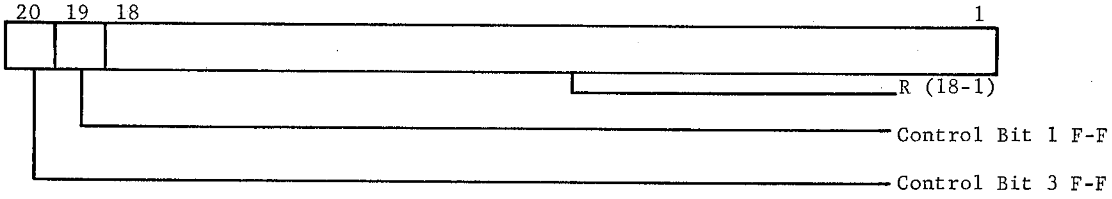
The 20-bit buffer is intended for interconnecting DATANET-30's. Usually system considerations indicate that a WBC should be used on lines operating at more than 300 bits per second. The following rates are selectable with standard speed connectors: 600, 1200, 1800, 2000, 2400, and 3000 bits per second. 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 ____)	

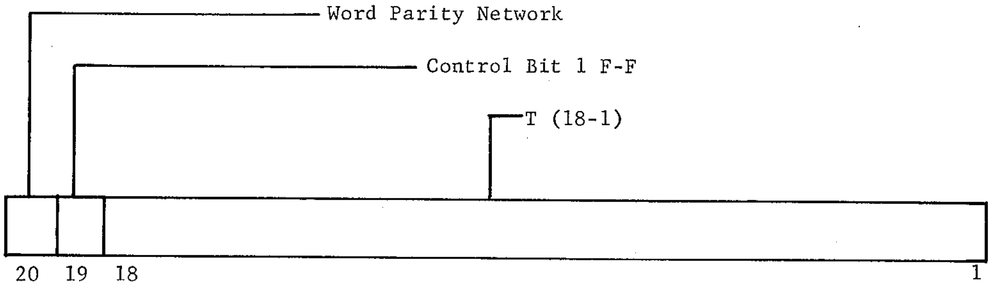
The 20 bits in the data register are distributed as follows:

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.



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

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

NES	I	1

1	Receive flag set (data register contains a new word).
2	Transmit flag set (data register is ready for a new word).
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 18) 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.

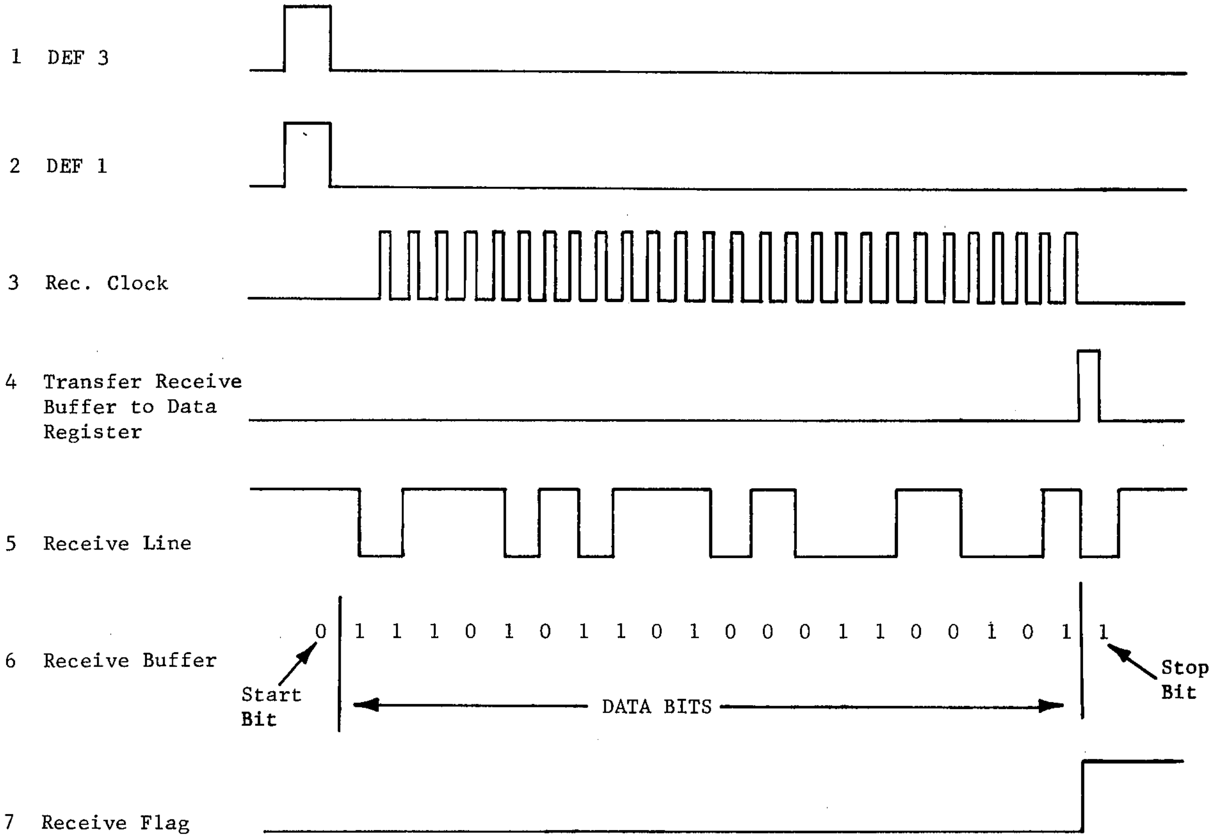


Figure 18. WBC Receive Timing Diagram

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.

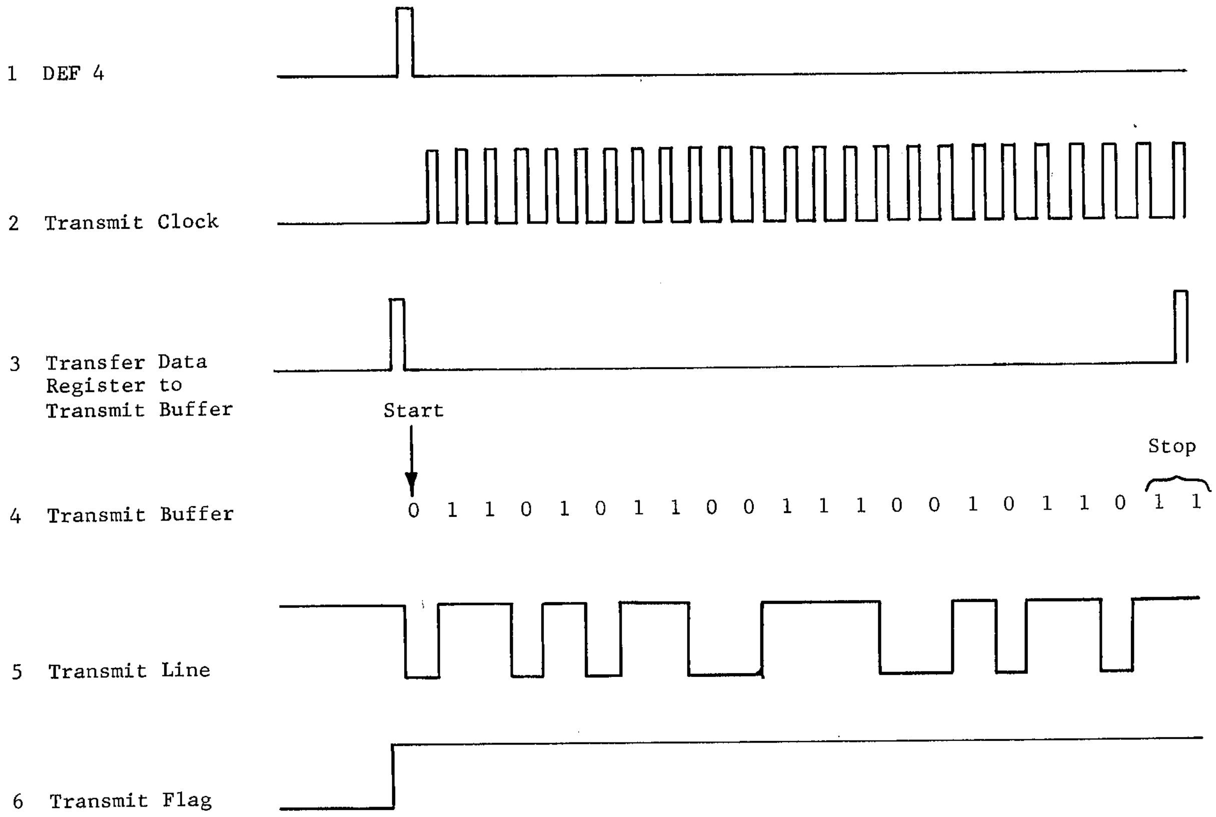


Figure 19. WBC Transmit Timing Diagram

The timing diagram (Figure 19) 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
```

PROGRAMMING THE PERFORATED TAPE READER

The perforated 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. Perforated 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 perforated tape reader is always on buffer selector address 0.

Perforated 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 perforated 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 perforated tape reader control panel. Normal operation requires that the reader be turned on at all times.

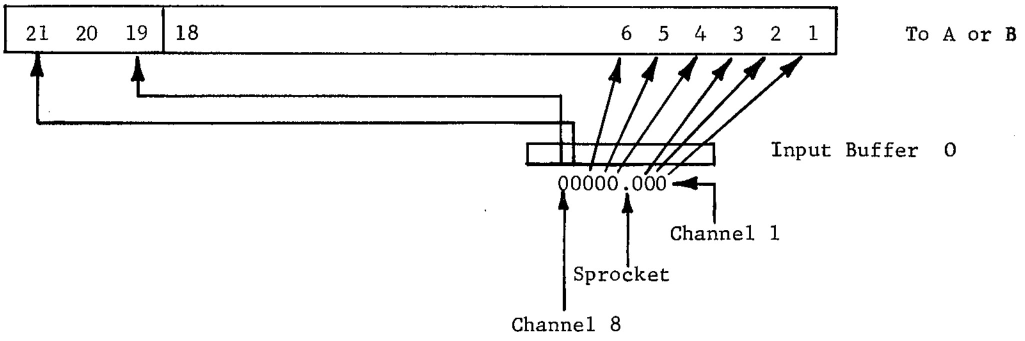
Reading Perforated Tape Under Program Control

PERFORATED TAPE READER INSTRUCTIONS

Following are the perforated 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 tape moving or allows the movement of tape to continue.



DEF 1 Reset flag and read next character. The reader starts tape moving through the reader or allows the movement of tape 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 perforated tape might be read. In this example, 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 perforated tape generated by General Assembly Program 3 (run on a GE-225 computer) in the program load format can only be loaded into the DATANET-30 by a loader program. It is not hardware loadable.

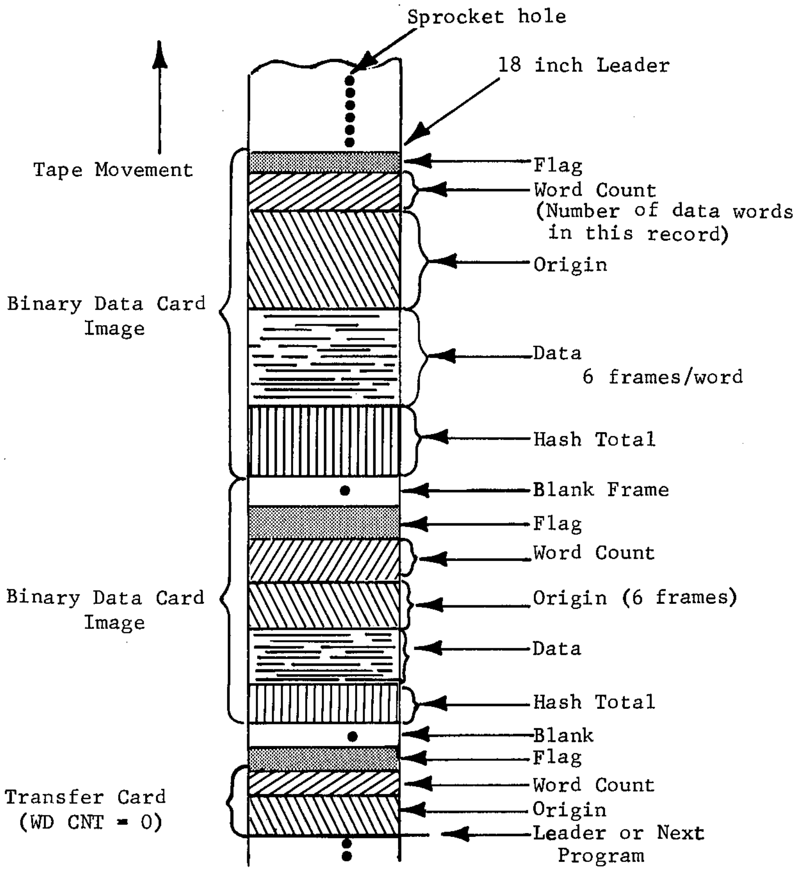
The program load perforated tape code is shown below:

		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

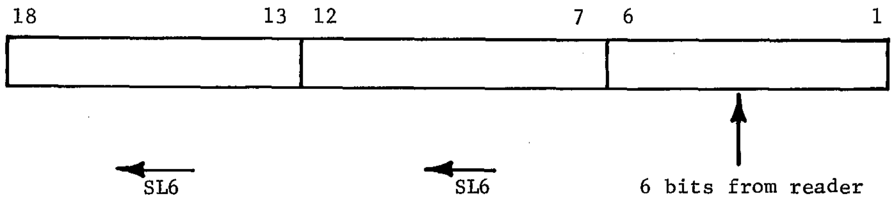
0 = No Hole

EXAMPLE:



Hardware Load and the Perforated Tape Reader

Once initiated, the loading of data from the perforated 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

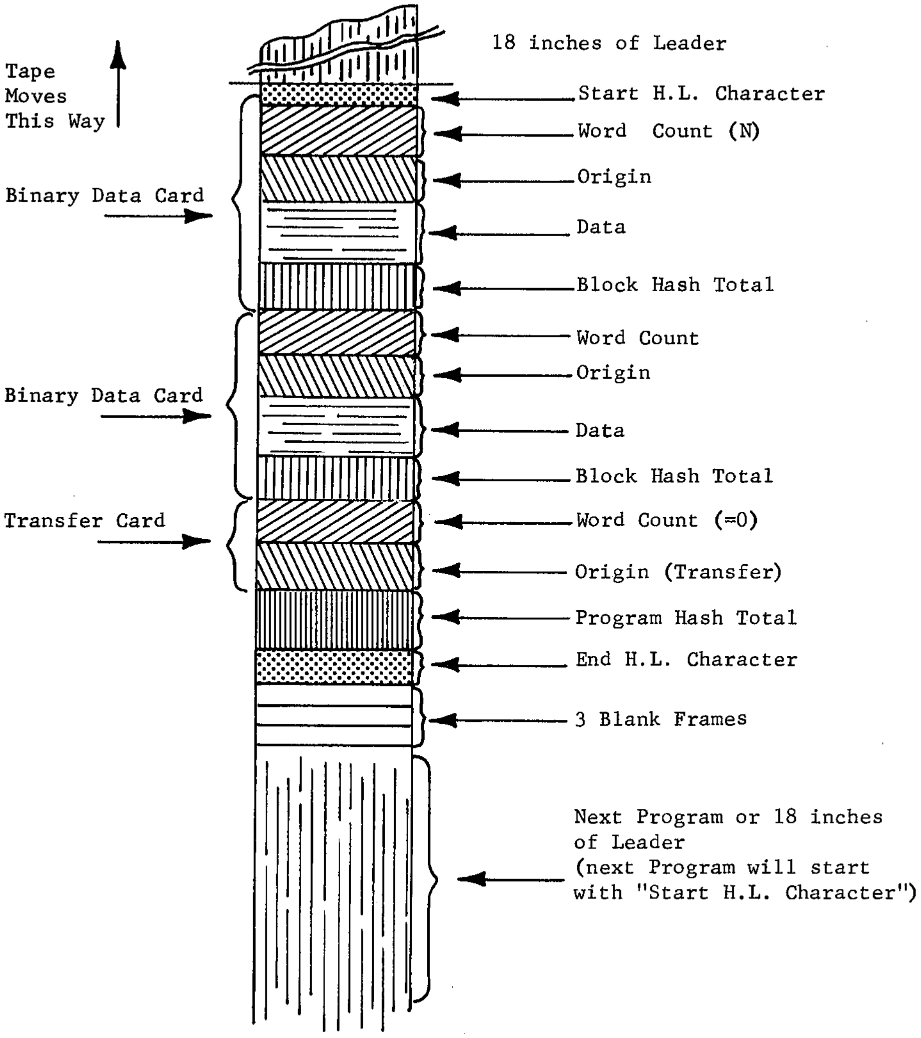
3. Read a character.

Character 87 | 654321		Go To Step
$\underbrace{00}_{0}\,\vert\,\underbrace{XXX}_{X}\underbrace{XXX}_{X}$ (0XX)	SL6 BR,B	3
$\underbrace{01}_{1}\,\vert\,\underbrace{XXX}_{X}\underbrace{XXX}_{X}$ (1XX)	TRA BR,B STB "P CTR" Count P (P=P+1) Set B-register to zero	3
$\underbrace{11}_{3}\,\vert\,\underbrace{XXX}_{X}\underbrace{XXX}_{X}$ (3XX)	TRA BR,P Set B-register to zero	3
$\underbrace{10}_{2}\,\vert\,\underbrace{000}_{0}\underbrace{000}_{0}$ (200)	Start the program at location specified by P-counter	Starting Location of Program

Hardware Load Format

The Hardware Load format output of the assembly program may be loaded into the DATANET-30 by either Hardware Load or a Loader program. When the perforated tape is loaded via a Loader program, checking is accomplished by the block hash total and program hash total. When the perforated 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 perforated tape format is shown below:



ASSEMBLY PROGRAMS

DATANET-30 source programs can be assembled either on a GE-225/235 computer or on a DATANET-30.

The DATANET-30 assembly program run on the GE-225/235 is CD225F2.001/2. Information on this assembly program is included in Appendix A of this manual.

The DATANET-30 assembly program run on the DATANET-30 is CDD30F1.001/2. Information on this assembly program is contained in publication CPB-1074.

The assembly program run on the DATANET-30 will accept programs written for the DATANET-30 assembly program run on the GE-225 computer, thus providing compatibility for assembling DATANET-30 source programs.

UTILITY ROUTINES

Since the output from the DATANET-30 assembly program run on the GE-225 is magnetic tape (switch option) or punched cards and the input to the DATANET-30 is perforated 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 perforated tape in various formats on a free-standing perforated 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 tape loader programs. The Paper Tape Conversion (General Assembly Program 3) Utility Routine for perforated tape can be run following the DATANET-30 General Assembly Program by setting the console switches.

PROGRAMMING AIDS

Aids for program debugging are abailable. The following is a list of these and other software that have been developed. Information regarding these and other software aids as they are developed can be obtained from the Computer Program Library, Computer Department, General Electric Company, P.O. Box 2961, Phoenix, Arizona, 85001.

LIBRARY NUMBER	DESCRIPTION
CDD30B1.001	Card Loader--Loads binary and octal cards into memory. This is a paper tape loop for the tape reader of the DATANET-30.
CDD30B1.002	Bootstrap Tape Loader--Loads programs from magnetic tape.
CDD30B2.001	Edited Memory Dump--Dumps all of memory on the printer except for the memory dump program. Prints 8 memory locations per line. The starting address of the 8 memory locations is to the left of each line printed. (8 or 16k.)
CDD30B2.002	CORE Dump Mnemonics--Dumps memory on the printer. Dumps a line in octal and on the next line prints mnemonic op code.
CDD30B2.003	Memory Dump--Octal/Baudot--Dumps on the printer 8 words per line followed on the same line by the Baudot Equivalent of the 8 words.
CDD30B2.005	Trace--Prints every instruction of a non-real time program and the state of the working registers after each instruction is executed.
CDD30B2.006	Memory Lookup--Searches for specified bit patterns entered through the switches and lists on the printer all locations where the bit pattern appears. Will also list all references to a specific memory location.
CDD30B3.001	Magnetic Tape Dump--Dump to printer binary or BCD tape in octal format.
CDD30B3.002	Tape or Card to Tape--Writes BCD tapes to tape or Hollerith cards to tape.
CDD30B3.003	Decimal Tape to Printer--Dumps BCD tape to printer.
CDD30B3.004	Mixed Binary Octal Cards to Magnetic Tape--Writes DATANET-30 object programs on magnetic tape.
CDD30D1.001	Multiply--Multiplies two 17 bit words to produce a 35 bit product.
CDD30D1.002	Divide--Divides 35 bit word by 17 bit word to produce up to a 17 bit quotient and 16 bit remainder.

LIBRARY NUMBER	DESCRIPTION
CDD30E1.001	Card Read--Reads a card and stores in memory. No conversion.
CDD30E8.001	Disc Storage Unit I/O--Simplifies the use of the DSU by performing the necessary preparatory and error checking functions associated with reading and writing on the DSU.
CDD30E8.002	Data to Disc Storage Unit--Loads card data on DSU in Octal, Baudot or BCD.
CDD30E8.003	Dynamic DSU Dump with I/O--Permits dumping the DSU transfers in real-time on the printer. Also permits DSU updates and moves.
CDD30E8.004	DSU Dump to Printer--Printout in octal and Baudot of selected sequential DSU records.
CDD30E8.005	Zero DSU--Zero selected areas of the DSU under console switch control.
CDD30F2.001	DATANET-30 Assembler on the DATANET-30 (Cards).
CDD30F2.002	DATANET-30 Assembler on the DATANET-30 (Tape).
CDD225F2.001	DATANET-30 Assembler on the GE-225 (Cards).
CDD225F2.002	DATANET-30 Assembler on the GE-225 (Tape).

APPENDIX A

ASSEMBLY PROGRAMS

Appendix A to this manual covers the DATANET-30 General Assembly Program run on the GE-225 computer.

Publication CPB-1074 covers the DATANET-30 Assembly Program run on the DATANET-30.

Both Appendix A to this manual and CPB-1074 are available from:

Marketing Distribution Center
Computer Department
General Electric Company
P. O. Box 2961
Phoenix, Arizona, 85002