DATAPOINT 2200

ASSEMBLER

MANUAL

I. INTRODUCTION:

The Business Terminal Assembler (BTA) provides facilities for generating object coding for the CTC Business Terminal. The initial implementation of the assembler is on a time-sharing facility, but will eventually be implemented on the Business Terminal itself.

Basically, the assembler is a program that assigns numerical values to symbols and outputs these numerical values upon input of the associated symbols. Symbols in certain fields have pre-assigned values such as the opcode mnemonics. The value assigned to an instruction mnemonic is the binary bit configuration recognized by the Business Terminal hardware for that instruction. For example the following instruction mnemonics have the octal values given below:

MNEMONIC	VALUE
ADB	201
LBM	317
INPUT	075

Symbols in fields other than the opcode field may be defined by the user. Pre-defined and user-defined symbols are kept separately by the assembler so the user may define symbols that are the same as the pre-defined symbols without encountering any difficulties. For example:

LOCATION	VALUE	STATEMENT	
000100	207	ADM	ADM
000101	104 000 100	JMP	ADM

2. STATEMENTS:

A BTA statement consists of a label field, any number of instruction fields, and a comment field. The ability to put many instructions on a single line enables more effective use of the Business Terminal and also allows logical grouping of instructions which make a program more readable.

Comments may appear starting anywhere on a line. A comment is indicated by the presence of a period and effects the assembler as if spaces appeared in the columns from the period to the end of the line.

Commas look line spaces to the assembler. All of the fields in a statement must be separated by at least one space.

If the first column of a statement contains a character other than a space, the statement is assumed to be labeled. The label must start with a letter and may contain any number of any combination of letters and digits. Note, however, that only the first six characters in a label are used for distinguishing it from other labels.

A location counter is maintained by the assembler and incremented after every instruction is processed. The current value of the location counter is the memory address of the object value being generated by the current instruction. The label is given the value of the location counter at the start of processing the line. This allows the first address under which

the statement will generate data or instructions to be given a symbolic name which may be referenced by other statements. The value of the location counter is a 16-bit unsigned value.

The instruction field always contains an opcode field and additionally contains an operand field if and only if the opcode requires one. Opcodes are any number of any combination of characters predetermined by the assembler. Only the first three characters of the opcode are significant but more may be used if desired. The opcode field specifies either an instruction mnemonic, which causes the assembler to generate a numeric value which can be interpreted by the Business Terminal hardware; or an assembler directive, which causes the assembler to take some action different from the normal instruction processing. The operand is a symbolic expression which is evaluated at assembly time and used in whatever manner required by the opcode.

A list of the instruction mnemonics and the result of the Business Terminal hardware interpreting them is given in Appendix II. The numerical value of each mnemonic is given in Appendix III.

There are six assembler directives.

END indicates that there is no more input data to be processed and that the assembler should complete generating output.

SKI(P) increments the location counter by the value of the operand expression.

EQU changes the value of the label on the statement from the value of the location counter at the beginning of processing the statement to the value of the operand expression.

BLK (for block justify) sets the location counter to the next address divisably by sixty-four which is useful in complicated programs for speeding up the time of execution by the Business Terminal.

RPT (for repeat) indicates that the assembler should process the following statement the number of times indicated by the value of the operand expression.

DC (for define constant) generates eight-bit object words from expressions following the opcode. This is the one place where commas are differentiated from spaces. If an expression is terminated by a comma, the DC directive generates a word of object code from the expression value and looks for another expression. If the expression is terminated by a space, the DC directive returns control to the main assembler loop which looks for another opcode. Another special exception is made for string items found in DC directives. All the characters of a string item are significant, and as many words as are necessary are generated to accommodate all the characters of the given string item. Again, a comma is looked for after the closing quote in a string item to see if more expressions will follow.

A symbolic expression consists of any number of numbers, strings, or symbols with operators between them. Addition and subtraction are allowed (indicated by + and -) and the operators must follow numbers or symbols immediately. If a space terminates a number or symbol, the expression is

assumed to be ended. Numbers are assumed to be decimal (base 10) unless they have one or more leading zeros in which case they are taken to be octal (base 8). The special symbol consisting of a single dollar sign is the value of the location counter at the beginning of processing of the current instruction field. Note that if an instruction generates more than one addresable quantity (eight bits) the dollar sign value is the location of the first word generated. Expressions yield 16-bit values.

String quantities are delimited (preceeded and followed) by appostrophies ('). In expressions, only the last two characters are used if more than two appear and if only one appears the most significant eight-bits are zero. If no characters appear ('')a zero results.

Note that for opcodes which require only an 8-bit operand, the least significant (right-hand) eight of the sixteen bits in an expression value are used unless the expression is terminated by a pound sign (#) instead of a space, in which case the value is shifted right eight bits with zeros filling the left-hand eight.

APPENDIX I

PRELIMINARY BUSINESS TERMINAL ASSEMBLER (BTA) SYNTAX

<NULL> ::=

<SPACE> ::= ∅

<DIGIT> ::= 1|2|3|4|5|6|7|8|9|0

<LETTER> ::= A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z

<ALPHA STRING> ::= <LETTER>|<ALPHA STRING><LETTER>|<NULL>

<SIGN> ::= +|-

<NUMBER> ::= <DIGIT>|<NUMBER><DIGIT>

<CHARACTER> ::= <SPACE>|<DIGIT>|<LETTER>|<SIGN>|*|,

<LABEL> ::= <LETTER>|<LABEL><LETTER>|<LABEL><DIGIT>

<OP STRING> ::= <LETTER><LETTER><LETTER><ALPHA STRING>|<LETTER><LETTER>
<LETTER>|<LETTER><LETTER><DIGIT><ALPHA STRING>

<SEP> ::= <SPACE>|,|<SEP><SPACE>|<SEP>,

<SYMBOL> ::= <LABEL>|<NUMBER>

<EXP> ::= <SYMBOL>|<EXP><SIGN><SYMBOL>|<SIGN><SYMBOL>

<COMMENT> ::= <CHARACTER>|<COMMENT><CHARACTER>

<LINE ITEM> ::= <OP STRING>|<OP STRING><SEP><EXP>|<OP STRING><SEP>

<LINE> ::= <LABEL><SEP><LINE ITEM><SEP><COMMENT>|<SEP><LINE ITEM>
<COMMENT>|*<COMMENT>|<SEP>*<COMMENT>

APPENDIX III

INSTRUCTION FORMATS

Source and Destination Codes (s and d):

0	A	Register	
1	B	Register	
2	C	Register	
3	D	Register	
4	E	Register	
5	H	Register	Memory data
6	L	Register	Address
7	Memory data		

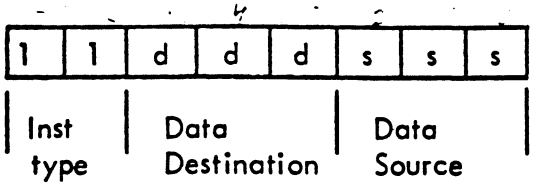
Operand Codes (p):

0	Add	AD
1	Add with carry	AC
2	Subtract	SU
3	Subtract with carry	SB
4	And	ND
5	Exclusive-or	XR
6	Inclusive-or	OR
7	Compare	CP

Flag flip-flop codes (c):

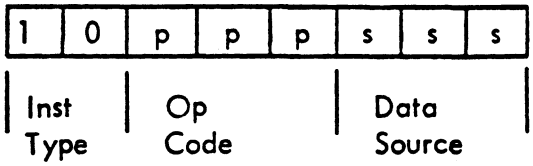
1	carry	C
2	zero	Z
3	sign	S
5	parity	P
~~general~~		~~J~~
~~general~~		~~K~~

Register and Memory Load Instructions



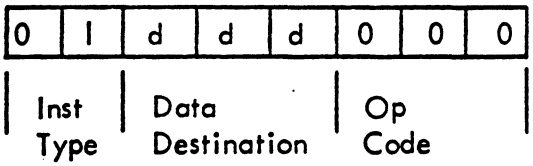
Octal: 3 d s

Register and Memory Arithmetic/Logic Instructions



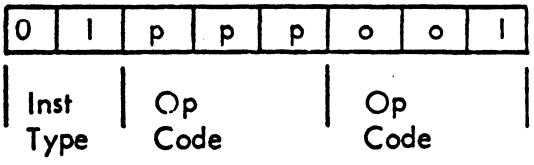
Octal: 2 p s

Load Immediate Instructions



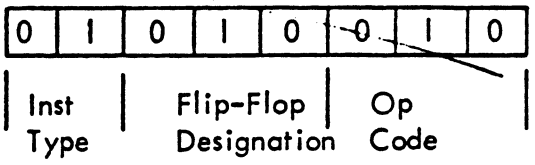
Octal: 1 d o

Arithmatic/Logic Immediate Instructions



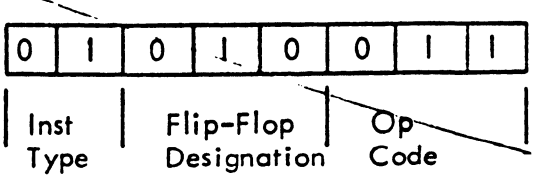
Octal: 1 p 1

Set J Instruction



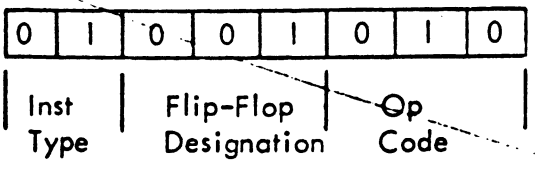
Octal: 122

Reset J Instruction



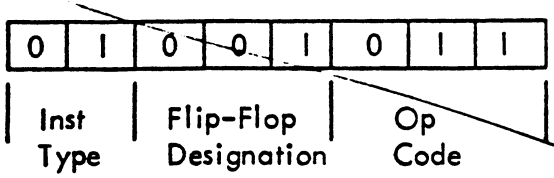
Octal: 123

Set K Instruction



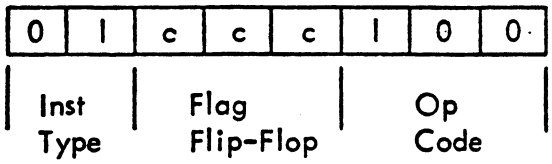
Octal: 112

Reset K Instruction



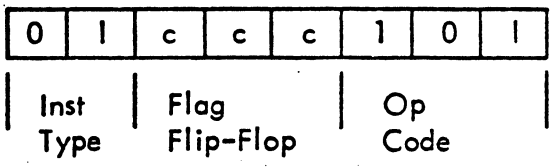
Octal: 1 1 3

Jump if c False



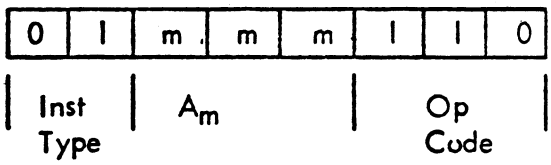
Octal: 1 c 4

Jump if c True Instructions



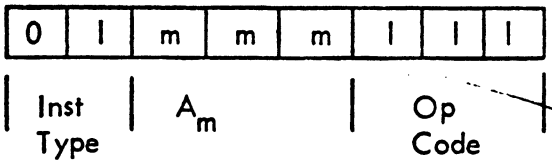
Octal 1 c 5

Jump if A_m False Instructions



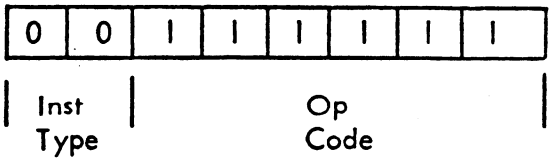
Octal: 1 m 6

Jump if A_m True Instructions



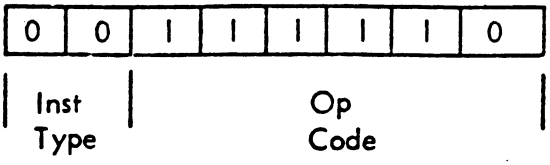
Octal: 1 m 7

Subroutine Call Instruction



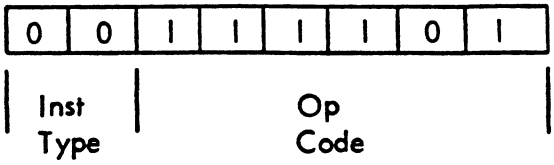
Octal: 077

Subroutine Return Instruction



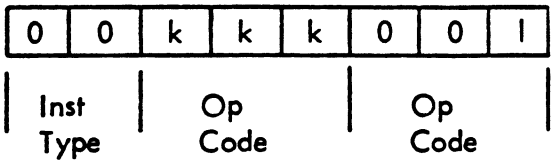
Octal: 076

Read Instruction



Octal: 075

Shift Instructions

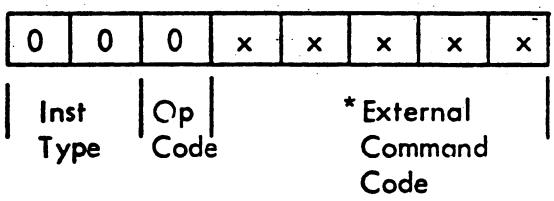


Octal: ~~025~~ 057
061

Shift Codes (k):

5	Shift Right Circular	SRC
6	Shift Left Circular	SLC

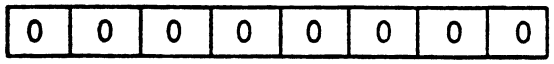
External Commands Instructions



Octal 001-037

*See Appendix IV

Halt Instructions



Octal: 000

Octal: 377

APPENDIX IV

EXTERNAL COMMANDS

CODE	COMMAND	<exp>
001	Address	ADR
002	Sense Status	STATUS
003	Sense Date	DATA
004	Write Strobe	WRITE
005	Command 1	COM 1
006	Command 2	COM 2
007	Command 3	COM 3
010	Beep	BEEP
011	Click	CLICK
012	Slect Deck 1	DECK 1
013	Select Deck 2	DECK 2
014	Read Block	RBK
015	Write Block	WBK
016	Write EOF	WEOF
017	Backspace One Block	BSP
020	Slew Forward	SF
021	Slew Backward	SB
022	Rewind	REWND
023	Stop Tape	TSTOP

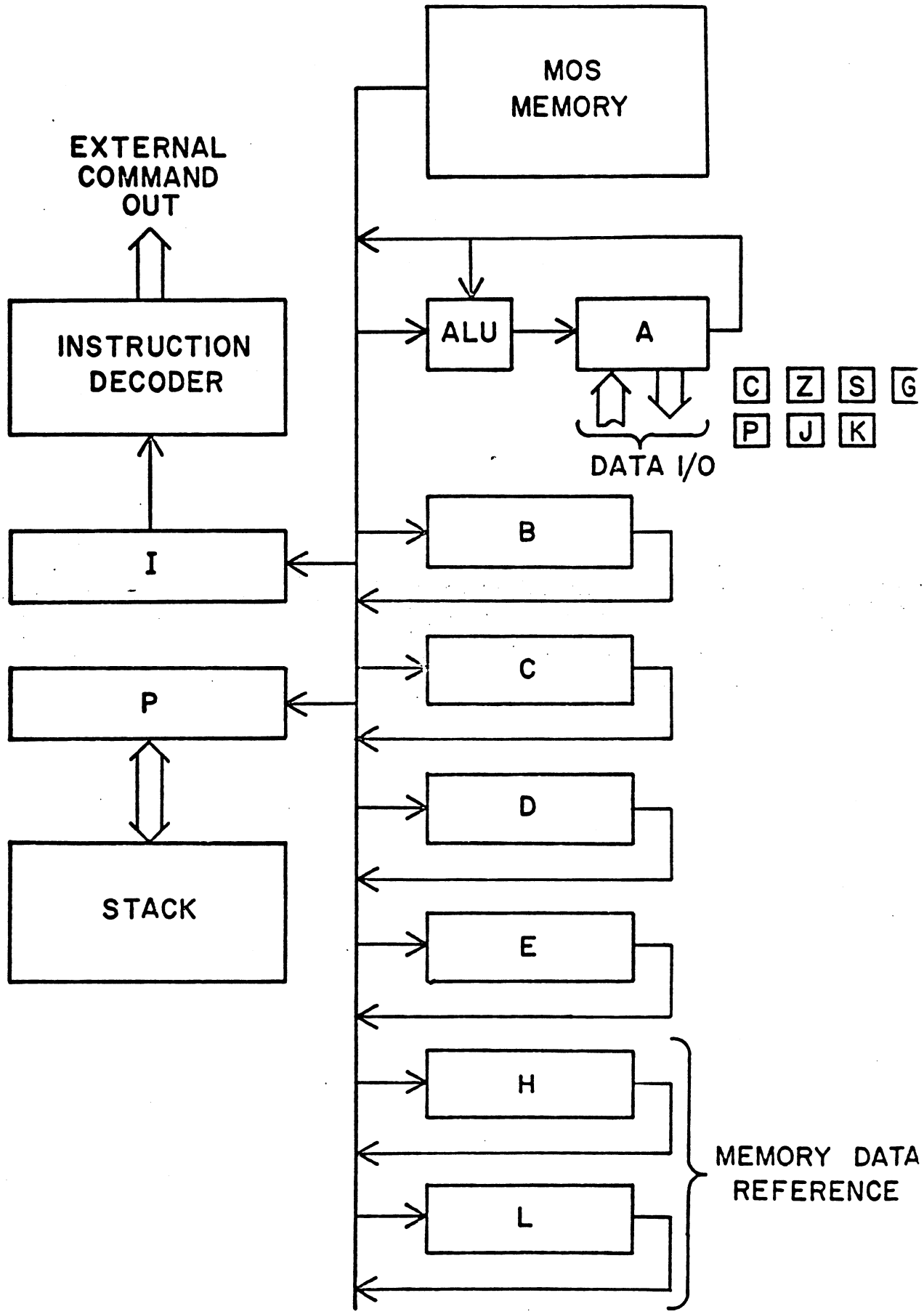
APPENDIX V

TAPE STATUS BITS

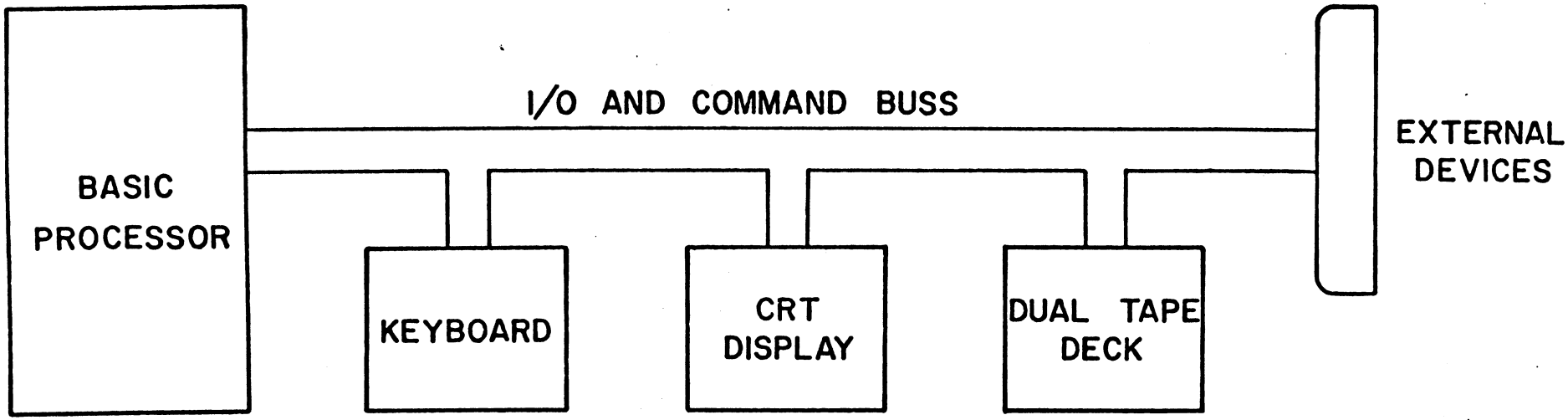
BIT #	TRUE INDICATION
0	Deck Ready
1	End of tape
2	Read Ready
3	Write Ready
4	Interrecord Gap
5	End-of-file gap

CRT/KB STATUS BITS

BIT #	TRUE INDICATION
0	CRT Write Ready
1	KB Read Ready
2	"Keyboard" Switch Depressed
3	"Display" Switch Depressed

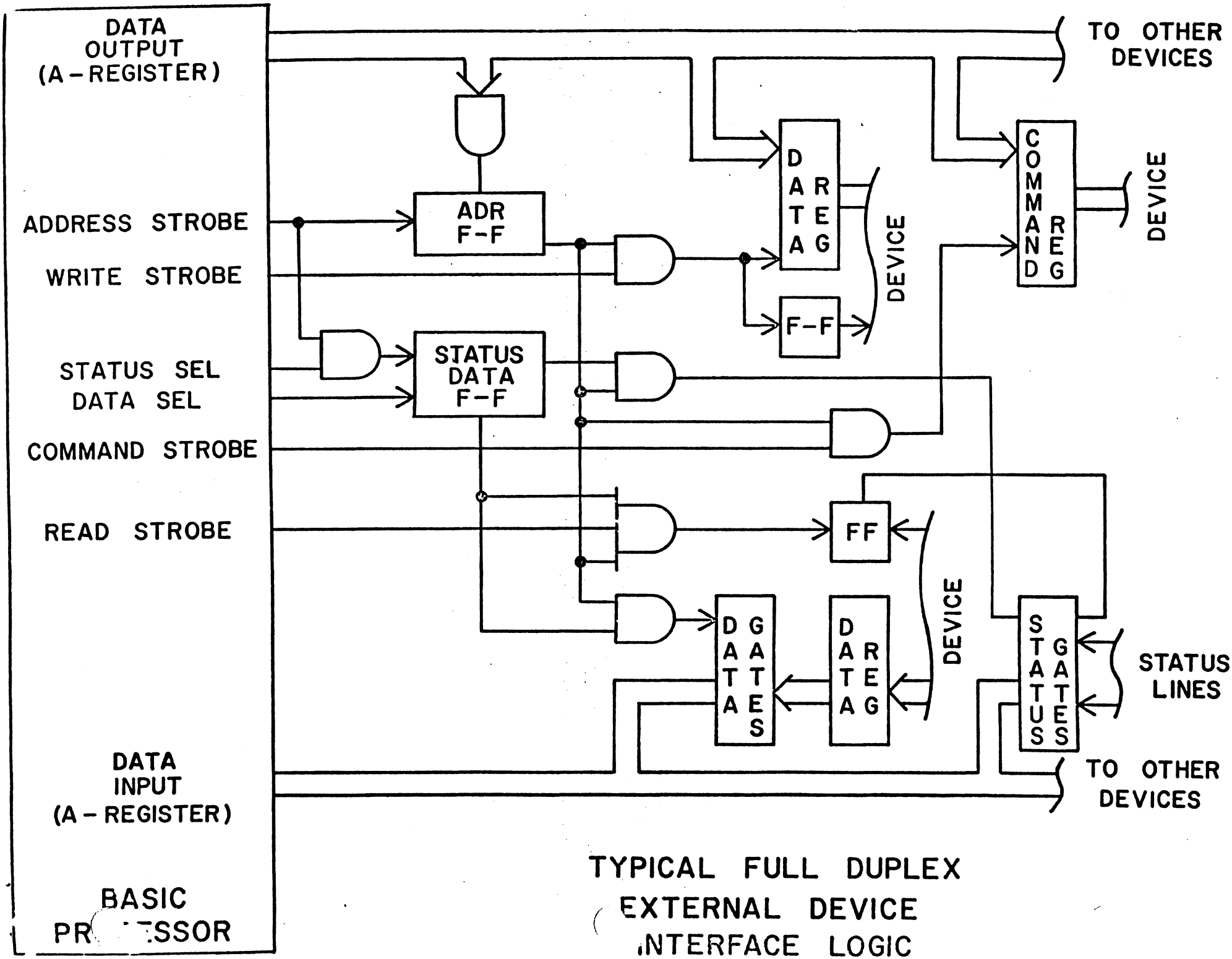


BASIC PROCESSOR



BUSINESS TERMINAL

BLOCK DIAGRAM



TYPICAL FULL DUPLEX
EXTERNAL DEVICE
INTERFACE LOGIC