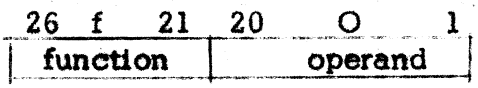
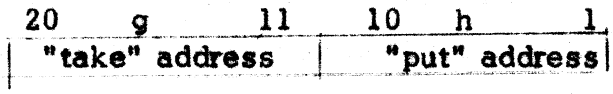
RW-400 NOTES

The RW-400 is a 26 bit word stored program parallel machine. Its basic configuration is a 1K compute module connected to a buffer module containing two additional 1K banks. This connection is made through an exchange unit which permits further buffer modules and peripheral devices to be attached. The buffer modules are used as data sinks and are capable of limited I/O and search instructions. Only one bank of one module can be connected at one time.

The instruction logic is two address with most operations affecting an accumulator or A register. The instruction format is



with the operand field divided as



on most instructions. There is an auxiliary arithmetic or B register used with some operations.

The arithmetic instructions include the usual add, subtract, multiply, and divide as well as square root and absolute subtract (absolute difference). Each of these has result transferral variations which increase the repertoire. The arithmetic used is two's complement on $2^{25}-1$ with the high bit (26) being a true sign bit. There is an independent overflow indicator associated with the accumulator.

Boolean operations are possible, but are somewhat irregular.

Addressing

Three modes of addressing are allowed. These are: 1, direct, referring to locations in the compute module; 2, constant, in which the operand is positive zero; and 3, indirect, which references a location in the connected

buffer module. Indirect addressing utilizes two registers in the buffer module for final location. These are the "read" or R and "write" or W. These registers are incremented by one after each use.

The addressing modes are selected by using 0 in the address field for constant, $1\text{-}1776_8$ for direct and 1777_8 for indirect. Certain instructions prohibit constant or indirect addressing and so these locations in memory can be referenced.

Since there are two address fields per instruction, this results in up to nine addressing modes per instruction.

Result Transmittal

Each basic arithmetic operation (add, subtract, etc.) appears in three Result Transmittal species. Results of operation always appear in accumulator.

1. Store

   Data from the location in g field is combined with contents of the accumulator and result stored at location in h field.

2. Hold

   Data from g location is combined with data from h location and result stays in accumulator.

3. Replace

   Data from g location is combined with data from h location and result is also stored in h location.

RW-400 INSTRUCTION SET

Data Movement

There are only three specific data movement instructions available.

1. LOAD A Machine code: 33

Operands: contents of instruction operand field

Operation: Accumulator is cleared and contents of operand field appear in bits 1-20 of A register. This instruction permits clearing of A or setting it to any 20 bit constant. No memory referral is made.

Mnemonic: LAD

2. STORE A, B Machine Code: 34

Operation: Contents of B register is stored at g address, A contents at h address. Indirect addressing applies via W register in buffer, with B in first location. Constant mode of addressing inhibits A or B store but if both g and h are constant, registers are interchanged with A contents going to B and B contents to A.

Mnemonic: DAB

3. TRANSMIT Machine code: 31

Operation: Contents of g address are stored at h address. Accumulator is not affected. Indirect addressing holds with g field using R register and h using W register. Constant mode does not apply but location zero is referenced.

Mnemonic: TRM

Examples: Machine code appears in octal with f, g, and h considered separately.

Mnemonic	Code			Action
LAD 0	33	0000	0000	A register cleared
LAD 1, 1720	33	0001	1720	A register set to 3720 (2000_{10})
DAB 0, 100	34	0000	0100	A contents stored at location 100
DAB 0, 0	34	0000	0000	A and B contents interchanged
DAB 1777, 20	34	1777	0020	B contents stored in buffer at W address and W register incremented. A contents stored at 20.
TRM 100, 0	31	0100	0000	Contents of 100 stored at 0.100 and A unaffected
TRM 1777,1777	31	1777	1777	Contents of r address in buffer stored at w address in buffer. R and W registers incremented.

ARITHMETIC INSTRUCTIONS

The following instructions operate in the A or A and B registers. The three addressing modes apply. The constant mode in the g field giving an operand of +0 and constant in the h field giving an operand of +0 and inhibiting a memory store of the results if applicable. When indirect addressing is used, the R register is always associated with the g field. Indirect of the "REPLACE" and "HOLD" varieties use the R with the h field as well, incrementing it if the g is also indirect. The "STORE" variety uses the W register with the h field.

1. STORE ADD Machine code: 05

Operation: Operand at g address is added to accumulator contents and result stored at h address. constant mode g causes A contents to be stored at h address. Constant mode h has result of adding g address data to A with no memory change. Indirect g mode extracts data via R register with consequent R incrementation. Indirect h mode stores result via W register with W incrementation.

Mnemonic: SAD

2. HOLD ADD Machine code: 07

Operation: A register is cleared and g and h contents are summed in A. Constant mode g or h results in only one operand coming to A. Indirect addressing in either g or h uses R register with consequent incrementing. If both g and h are indirect, two consecutive buffer locations are referenced with R being incremented twice.

Mnemonic: HAD

3. REPLACE ADD Machine code: 06

Operation: A register is cleared and data at g and h addresses are summed in A with result stored in h location also. Constant mode g or h

fields effectively bring the non-constant contents to A. If both are constant, A is set zero. With indirect mode R register is used with both fields. If both are indirect, sequential locations in the buffer are used with the second location considered the h.

Mnemonic: RAD

4. STORE SUBTRACT Machine code: 01

Operation: Data from g is subtracted from A contents and results stored at h. Constant and indirect addressing apply as in STORE ADD.

Mnemonic: SSB

5. HOLD SUBTRACT Machine code: 03

Operation: A is cleared, complement of g data is brought to A and h data added to it. Constant g mode is not complemented. Other constant and indirect modes operate as HOLD ADD.

Mnemonic: HSB

6. REPLACE SUBTRACT Machine code: 02

Operation: A is cleared, complement of g data is brought to A and h data added to it with result stored at h. Constant g is not complemented. Otherwise addressing operates as REPLACE ADD.

Mnemonic: RSB

7. STORE ABSOLUTE SUBTRACT Machine code: 11

Operation: The absolute value of A replaces A and the absolute value of the g data is subtracted and result stored at h. This is the signed difference. Memory mode applies as in other STORE operations. Double constant mode sets A to 1A1.

Mnemonic: SSA

8. HOLD ABSOLUTE SUBTRACT Machine code: 13

Operation: Complement of absolute g data is brought to A and absolute value of h is added. Constant g is not complemented.

Mnemonic: HSA

9. REPLACE ABSOLUTE SUBTRACT Machine code: 12

Operation: A is cleared and absolute value of g data is subtracted from absolute value of h data in A and result is stored in h.

Mnemonic: RSA

10. STORE MULTIPLY Machine code: 25

Operation: A contents are multiplied by g data with produce formed in A and B registers considered as a 50 bit accumulator. Bit 26 in A and B are the sign of the product. B is joined to the low order end of A and contains the least significant portion of the answer. The A contents (the most significant, high order) are stored in h. Constant g will clear A, B and h. Constant h inhibits store. Indirect addressing holds as in other "STORE's".

Mnemonic: SMU

11. HOLD MULTIPLY Machine code: 27

Operation: A is cleared and product of g and h data is formed in the combined A and B registers. Indirect and constant appl as in other "HOLD" operations.

Mnemonic: HMU

12. REPLACE MULTIPLY Machine code: 26

Operation: A is cleared, product of g and h data is formed in A and B and A contents stored via h. Indirect and constant address modes follow general rules for "REPLACE".

Mnemonic: RMU

13. STORE DIVIDE Machine code: 21

Operation: A contents are used as the divident and g as the divisor. Division by computer is a shift-subtract process and on the RW-400 the circuitry necessitates that the divisor appear larger than the dividend. This involves checking that the first non-zero bit of the divisor be of higher order than that of the dividend. The quotient appears in A with the remainder in B, and the quotient is stored via h. Sign of quotient and remainder will be the same.

Indirect mode applies as in other "STORE" functions.

If the dividend is greater than or equal to the divisor, A and B are unaffected and g data is stored via h. The overflow indicator is set. Constant mode in g gives the same result as would division by zero.

Constant mode h inhibits store.

Mnemonic: SDV

14. HOLD DIVIDE Machine code: 23

Operation: A and B are cleared and h data is divided by g data in A with remainder in B. The quotient (A result) is rounded by adding high order bit (25) to A. Indirect applies as in other "HOLD" functions.

REPLACE DIVIDE

The dividend-divisor inequality holds with same results as HOLD DIVIDE on aborts.

Mnemonic: RDV

16. STORE SQUARE ROOT Machine code: 15

Operation: Sum of A contents and g data is made and a 25 bit square of this is formed in A with any remainder appearing in B. Sign of root and remainder match. The low bit of the square root is always set to 1 and is stored via h in this form.

Indirect and constant addressing modes apply.

Mnemonic: SSQ

17. HOLD SQUARE ROOT Machine code: 17

Operation: A and B are cleared and the g and h data are summed. The square root of this sum appears in A with remainder in B. The square root has low bit set to 1. Constant and indirect addressing hold.

Mnemonic: HSQ

18. REPLACE SQUARE ROOT Machine code: 16

Operation: Same as HSQ but square root is also stored at h. Constant and indirect apply.

Mnemonic: RSQ

The following two arithmetic instructions deal with the A register contents and do not store results.

1. ADD TO ACCUMULATOR Machine code: 04

Operation: Data from g and h addresses are summed with accumulator contents. Constant mode applies and R register is used exclusively for indirect as in "HOLD" operations.

Mnemonic: AAC

2. MULTIPLY ADD Machine code: 24

Operation: The B register is cleared and the product of g and h is formed in A and B as in other multiply operations except the original A contents are added to the high order half of the product.

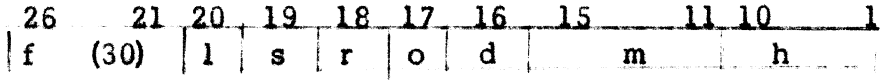
Mnemonic: MAD

SHIFT OPERATIONS

Shifts in the RW-400 are end-off manipulations of the A contents or of the combined A - B contents with the B register considered to the right (i.e., low order) of A. Zeroes are introduced at either end of the shifted register.

These instructions are single address, with the h field being used to store the shifted operand. The g field is used to specify the type of shift to be performed. Indirect and constant addressing applies to the h address and the W register is used for indirect storage.

Instruction Format



1. (m) bits 11-15. This is a count of positions to shift. Counts of $1\text{-}30_{10}$ ($1\text{-}36_8$) are true shifts. A count of 0 yields a rounding operationg (see 5). A shift of 31_{10} (37_8) causes not a true shift but a normalize or float operation. This float shifts the register until a 1 appears in a specified end of the A register or 31 positions have been shifted. The result is stored via h and the count of shifts is transferred to the A register.

2. (d) direction bit 16. This bit indicates if the shift or float is to be to the left or right. If bit is one, operand moves left, if zero, right. The right float halts when a 1 appears as bit 1 of A.

3. (s) species bit 19. When this bit is 1 a logical shift occurs. The logical shift considers the sign (bit 26) as an integral part of the data. When this bit is 0, a magnitude shift is performed which operates only on the low bits (1-25) of the data in a register. On left magnitude floats, shift halts when a 1 reaches position 25; on logical position 26. On magnitude operations bit 26 of both A and B are ignored.

4. (l) length bit 20. When this bit is one, the combined A and B registers are shifted. On logical shifts bits pass between position 1 of A and position 26 of B and 1 of A and 25 of B on magnitude shifts. A zero in this position limits the shift to the A contents. On either condition only the A contents are stored, however.

5. (r) round bit 18. The general shift function (code 30) also performs rounding of A contents separately (shift count 0) or in conjunction with actual manipulation. This adds a round-off bit to A contents after the shift.

On left shifts this round bit is the next bit to enter bit 1 of the A and on right shifts it is the last shifted out.

On single length left shifts this bit is always zero.

On all right shifts of zero count (single or double) the round bit is bit 1 of A.

6. (o) overflow bit 17. The overflow indicator is associated with the left (high order) end of the A register. This is usually set by arithmetical operations, but it can be set by shifting ones past position 25. Rounding will not set the overflow on right shift of 0.

Mnemonic: SHFT

BOOLEAN OPERATIONS

The Boolean operations in the RW-400 are distinguished as INSERT. They are subdivided as HOLD, STORE, and REPLACE since they follow the arithmetic instruction usage for data transmittal.

Three Boolean processes are involved in these operations: I Complementation symbolized " $\sim$ " which changes 1's to 0's and 0's to 1's. II And (logical product) symbolized " $\wedge$ "

$$\begin{array}{r} 1100 \\ \wedge\ 1010 \\ \hline 1000 \end{array}$$

III Or (logical sum) symbolized "v"

$$\begin{array}{r} 1100 \\ \text{v}\,1010 \\ \hline 1110 \end{array}$$

In the Boolean operations the sign bit is not distinguished and operations are performed on all 26 bits of A and B registers.

1. STORE INSERT Machine code: 35

The g data is "anded" with the A register contents and the result stored in h address. Constant and indirect addressing hold as in arithmetic "STORE".

Mnemonic: SIN

2. HOLD INSERT Machine code: 37

Operation: The logical product of the g data and A contents are formed in B. The logical product of the h data and complement of g are formed and this is logically summed in A. The end result of this is to select bits from A positions corresponding to the 1 bits in g location and substitute them for the corresponding bits of the h word. Symbolically result is $(H \wedge \sim G) \text{ v } (A \wedge G)$. Since the result appears only in A and B, it could be considered as clearing bits in A where 0's appear in g and substituting corresponding bits from h.

Constant and indirect addressing modes are used as arithmetic "HOLD".

Mnemonic: HIN

3. REPLACE INSERT Machine code: 36

Operation: The combination is the same as the HOLD INSERT but the A result is also stored via h. Constant and indirect addressing apply as in arithmetic "REPLACE".

Mnemonic: RIN

The RW-400 allows interrupts from peripheral devices. These set alert flip flops which can be treated as a twenty bit Interrupt Register I. The S or sense register is used in conjunction with this to recognize these alerts. (See Interrupt section for bit equivalence details) Bit 20 is the enable bit in S which allows control to pass to location zero when an interrupt occurs.

INSERT S Machine code: 77

Operand: Combined g and h field of instruction.

Operation: This instruction operates as does the REPLACE INSERT with the S register used as the h location and the low 20 bits of the instruction acting as the g data. Addressing modes do not apply but a special case arises when the low 20 bits of the instruction are all zero. In this case A and B are unaffected as are the low 19 bits of S but bit 20 in S is set to 1 thereby enabling any previous allowed interrupt to cause control transfer.

Mnemonic: INS

JUMP INSTRUCTIONS

There is no unconditional jump instruction (branch, transfer) in the RW-400, but certain conditional jumps operate as such by proper choice of parameters.

In all jump instructions the h field address is the jump location. Neither indirect nor constant address mode apply since the jump must have a termination.

Therefore jumps to 0000 and 1777 are possible.

In all jumps if condition is not met next sequential instruction is taken.

The jumps can be roughly divided by g field usage. There are four jump function codes of which three use the g field for memory referral. The fourth (test) divides the g field sufficiently to be dealt with separately.

A. Memory reference jumps

1. Compare jump Machine code: 73

This uses the g field as a reference address to compare with A contents. If the memory contents are greater than or equal to the A contents, the condition is met. The indirect applies to the g field using the R register. The constant mode in g senses for A being positive 0. A contents are unaffected.

Mnemonic:

2. Tally Jump Machine code: 71

This instruction is comparable to index skip instructions on other machines. There are no index registers available but memory

locations are used a s loop counters. Indirect addressing applies using the R register. A constant g mode yields an unconditional jump.

The memory contents are sensed. If they are +0 or -0, they are unaffected. If greater than 0, they are decremented, if less than, incremented.

Jump contion is +0 or = -0. RW-400 arithmetic is two's complement so -2 is one less than -0. This instruction can give a -0 result.

Mnemonic:

3. Link Jump Machine code: 72

This is a return jump or sub-routine entry on the 400. The return address is the next sequential location (P+1). Using the g field address, this address is stored in the h field (i.e., P+1 g 1-10). Control then passes to the h field address. Neither constant nor indirect modes apply to the g field. This allows the storage of addresses in locations 0 and 1777 of the compute module.

Mnemonic:

B. Test Jump Machine Code: 70

This instruction uses the g field to specify the register or registers to reference and also which bit in the register is to be sensed. This sense occurs in the A register and so this could affect the A contents.

Instruction format

26	21	20	19	18	17	16	15	11	10
f (70)		I	S	N	J	T		M	h

Bits 17-20 determine the register or registers referenced. If these bits are all zero, the current A contents are sensed. If any bit is non-zero, then the accumulator is cleared and the called-for data entered.

Bit 16 indicates the condition to be met, zero or non-zero, and bits 11-15 the bit position to check.

If the entire g field is zero, an unconditional jump is made to h address.

1. M field bits 11-15. This count indicates the bit in A to be sensed. If bits 17-20 of instruction are non-zero, the called-for register data is brought to A in the low order positions. If the position count is zero, the condition is assumed not to be met.

   Positions 27-31 are possible and refer to separate indicators. Called-for data will be brought to A as determined by bits 17-20 of the instruction, however.

   These indicators will respond as non-zero if previously set. They are turned off when sensed.

   27 Overflow

   28 Parity Error (memory)

   29 Program Error (illegal function code)

   30 Conditional Tape Read

   31 Control Panel (manually or program set)

2. Special registers

   Four special or pseudo registers can be sensed. They are:

   I Interrupt or Alert register. This is a twenty bit representation of flip-flops set by I/O or internal conditions.

   S Sense register. This is the register set internally to allow automatic recognition of alerts. Also 20 bits.

N A set of sixteen flip-flops giving status of I/O lines connected to exchange unit.

J Jump register. An eight bit representation of the control panel jump or sense switches.

If more than one of these registers are called, they are "or"ed to the accumulator. If this multi call includes both the Interrupt and Sense register, then the S is "and"ed to the others.

e.g.: To sense S, I, and J, the A contents will be (IVJ) S.

I/O AND BUFFER PROGRAMMING

Since th RW-400 is of modular design, various components are connected through the exchange unit IX400 or CX400. This connects the compute module or modules to the buffer modules for additional storage and to most of the I/O devices.

These connections are effected by means of Command Output instructions to the Exchange Unit. This will be considered as the first class of the Command Output instructions.

Connect/Disconnect Command Output Instruction Format

26	21	20	19	18	17	14	13	1
42		W		2		D. T.		S. F.

In absolute form of f, g, h, this is 42 x 4XX XXXX.

The W (bit 20) is the wait for ready or terminate when unready signal. If this is 1, the commanding module will hang until the exchange accepts the command. If zero, program error will be set and next instruction read up.

D.T. Device Type (bits 14-17). This gives the general type of device which is to be connected. They are divided as to the data transfer format.

Code

0 refers to a buffer module connection which transfers word data

1 refers to paper tape reader or mag tape controller which performs an assembly.

2 signals Peripheral Buffers, Drum Controller, or digital clock.

3 is the display Buffer on CRT device

5 is for the printer

6 is the Exchange itself

17 is a disconnect.

S.F. Specific Function (bits 1-13). This varies with the type of exchange in use. The CX400 has a 1024 core memory reference file which contains the actual select codes and locations in it are selected here. The IX400 directly interprets these bits in making connections.

Disconnecting attached devices is accomplished implicitly by making a new connect request or explicitly by making a connect request with all zeroes in the low 13 bits. This explicit disconnect applies to both CX and IX400.

Devices no longer needed should be disconnected explicitly to avoid causing a program error on further connect requests.

Peripheral Devices, i.e., any but Buffer modules. Once the connection has been made then the particular device must be signalled for input-output by another command output instruction. This is an instruction of the following form:

26	21	20	19	18	17	14	13	1
42		W	0		0		Y	

where the W (wait) has the same meaning and Y defines the operation and device. If the device is directly connected to the computer, only this command is necessary as no Exchange connection is possible.

INPUT/OUTPUT

The computer input/output instructions are a combined search and transfer operation. In the first word of the I/O area as defined by the g field of the instruction a search word is stored. Data from the peripheral device will be compared with this before I/O takes place. If this search word is zero, then no comparison is made. The I/O begins with word g+1 in either case.

The h field of the instruction contains a count of words to be transferred. If this is zero, no transfer takes place, but a search is performed. The actual number of words is not needed with input transfers, if the device will send a completion signal (e.g., record gap on mag tape). In this case the word count need be only larger than expected data.

These instructions are used by the buffer as well as the computer. They are also used to transfer data between computer and buffer but no search is allowed. The W register is used as address location in buffer to store data and R as location to extract data.

THE BUFFER MODULE

The BM400 consists of two banks of 1024 core memory locations. There is sufficient circuitry associated for the Buffer to perform independent I/O with associated register setting.

Once the compute module has made connection to the Buffer module it can initiate these operations or use its own data channels to input or output buffer locations.

These instructions also have the function code 42 for command output

Format

26	f	2	20	19	18	17	14	13	1
	42		W	3		0		Y	

or

42 X60Y YYYY

Bit 20 acts as in other Command Output instruction to stall the computer until the device is ready or to set Program Error and continue.

The instructions associated with the buffer can be considered in three groups. 1. Those that only the compute module can give to the buffer. 2. Those that can be given by the compute or buffer module. 3. Those which only the buffer can give to itself.

In the first category are the condition checks. These divide the Y field as follows:

	13	11	10	9	4
Instruc.		1	0	0 0	Y

Bits 4-9 are ignored in the circuitry. The response is in 13 bit form.

Y = 0 Send General Status. This puts a status response which can be input for buffer condition check.

Y = 1 Send L. The L register is a 10 bit count of words input or output.

Y = 2 Send R. The R register is the 10 bit register associated with g field on indirect addressing. It is also used as a program register during independent buffer operations, i.e., address of instruction.

Y = 3 Send W. The W register is the other 10 bit register used in indirect addressing. It is also used for storage referral in self instruction mode.

The other instruction legal only from the compute module is the independent buffer initiation. This instruction is Set Rand start Self Instruction. This is to give the initial location of the instructions since the R register is used as P. The low ten bits of the instruction give the quantity to set in R. Zero is a special case and will cause no change in the R register.

Y Format

13	11	10	1
		XXXX	

Shared instructions deal with setting registers and indicators.

Format Register set

13	11	10	1
Y		XXXX	

Where Y indicates register and the low 10 bits give quantity to store

Y = 2 Set L. L gives count of words in or out

Y = 3 Set W. Write register

Y = 4 Set R. Read register

Associated with the Buffer module are indicators of buffer status. These can be sensed by the buffer for branching. They are affected by two instructions

Format

13	11	10	9	8	5	4	1
1		1	Y	0		X	

Bits 5-8 are not sensed in this instruction

Y = 0 Reset indicators. There are 3 indicators - 1. Branch, 2. Program Error, 3. Parity. This instruction will clear them. Any combination of the three is allowed by setting one's in the X field. Bit 4 for branch, bit 3 for Program Error, and bit 1 for parity. Bit 2 is unused.

Y = 1 Set Branch Indicator. For this the X field is 10.

There is a combination instruction which exchanges the R and W contents and can also set the branch indicator.

Format

13	11	10	9	8	5	4	3	1
1		1	1	0		X	1	

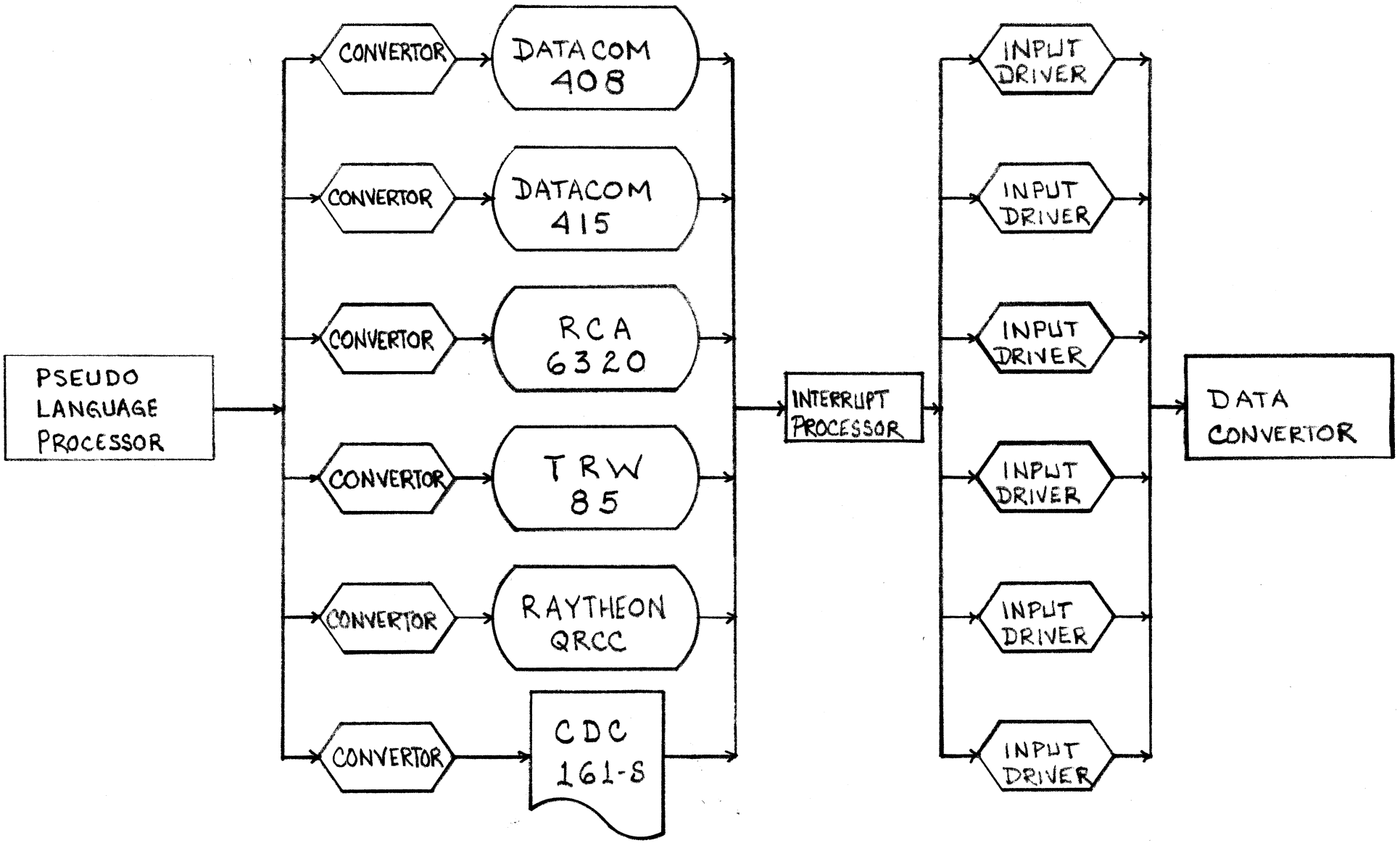
Bit 4 is used to set or not set branch indicator.

From information available at this time, the display system portion of the LCP appears to lend itself to a COGS-like technique. In particular, the need to drive devices with varying characteristics from a single source can be fufilled by providing a pseudo display language capable of encompassing the operating specifications of all available (or known) devices. This language can then be converted to the proper form by a subroutine which is cognizant of both the pseudo and the "real" formats. Choosing the correct subroutine (as there must be one for each device) would be a trivial matter based on the output address of the message. This is assuming there is a difference in addressing each device. The problems of communication, characteristic research, pseudo language definition and conversion techniques have been solved in COGS. This is not to say they can be utilized without modification, but the theory is sound and probably at least 75% of the design would remain unchanged.

The problems associated with input via the display consoles include, in addition to the reverse of the process discussed above, the need for handling interrupts emanating from the consoles. Since interrupt processing is a somewhat cut-and-dried procedure, a central routine would probably suffice for all consoles. Once a request to input has been granted, control could be turned over to the input routine for the requesting console.

A central, bi-directional data handling routine will be required for conversion to and from the universal code as defined in the executive system. It is believed that by using a single routine which is cognizant of which raw data code is involved would provide a substantial savings in core storage space by eliminating duplicate housekeeping chores. It is also recommended that this routine have knowledge ~~to~~ of the existence of the microwave data link in the disposition of a message, so that check sums may be attached during conversion.

The attached block diagram may assist in visualizing the proposed system.


PSEUDO LANGUAGE PROCESSOR
CONVERTOR
DATACOM 408
CONVERTOR
DATACOM 415
CONVERTOR
RCA 6320
CONVERTOR
TRW 85
CONVERTOR
RAYTHEON QRCC
CONVERTOR
CDC 161-S
INTERRUPT PROCESSOR
INPUT DRIVER
INPUT DRIVER
INPUT DRIVER
INPUT DRIVER
INPUT DRIVER
INPUT DRIVER
DATA CONVERTOR

UNIVERSAL CHARACTER SET

1 July

Char	Desc	Octal	Char	Desc	Octal	Char	Desc	Octal	Char	Desc	Octal
0	Zero	000	P		052	:	Colon	130			330
1	One	001	p		053	;	Semicolon	131			331
2	Two	002	Q		054	@	as/at	132			332
3	Three	003	q		055	⊠	Box-X	133			333
4	Four	004	R		056	"	Quotes	134			334
			r		057	'	Apostr	135			335
5	Five	005									336
6	Six	006	S		060	$	Dollar	136	¬		337
7	Seven	007	s		061	¢	Cents	137			
8	Eight	010	T		062	Σ	Summa.	140	±	Pls-Minus	340
9	Nine	011	t		063	1/4	one Qtr	141	≠	Not Equal	341
			U		064	≤	Eq or Less	142	<	Less Than	342
	Blank	012	u		065	1/2	One Half	143	|	Vert-Line	343
	(Space)	013							>	Greater	344
	"Reserved"		V		066	≥	Eq or Gtr	144	~	Similar To	345
			v		067	3/4	Three Frth	145	≡	Identity	346
A		014	W		070	∞	Infinite	146	ψ	PSI	347
a		015	w		071	↓	Arrow Down	147	γ	Gamma	350
B		016	X		072	θ	Theta	150	Γ	UC Gamma	351
b		017	x		073	↑	Arrow Up	151	δ	Delta	352
C		020	Y		074				Δ	Delta	353
c		021	y		075	ϕ	Phi	152	ε	Epsilon	354
			Z		076	→	Arrow Rt	153	∂	Part Deriv	355
D		022	z		077	K	Kappa	154	⌋	R Sl Br	356
d		023				←	Arrow Lft	155	⌊	L Sl Br	357
E		024	(	Lft Paren	100	]	Rt Brack	156	∧	And	360
e		025	!	Exclama	101	[	Lft Brack	157	∨	Or	361
F		026	?	Question	102					Es Cde #3	362
f		027	#	Numbers	103	3	Cubes	160		Es Cde #4	363
			°	Degree	104	2	Squared	161			
G		030	/	Slash	105	▨	Esc Cde #2	162			364
g		031				▣	Esc Cde #1	163			365
H		032	%	Percent	106						366
h		033	&	And	107			164			367
I		034	*	Asterisk	110			165			
i		035	)	Rt Paren	111			166		Clr Shift	370
			.	Period	112			167		Index	371
J		036	,	Comma	113					End Doc	372
j		037					Contr Dot	170		Start Doc	373
K		040	π	Pi	114		Underline	171		Start Graphic	374
k		041	-	Minus	115		Tab	172		Stop Graphic	375
L		042	ω	Omega	116		C.R.	173			376
l		043	+	Plus	117					Conf Char	377
			α	Alpha	120		Backspace	174			
M		044	x	Multiply	121		U.C.	175			
m		045					L.C.	176			
N		046	ß	Beta	122		End Msg	177			
n		047	÷	Divide	123						
O		050	=	Equal	124						
o		051	-	Dash	125						
			√	Sq Root	126						
			∫	Integral	127						

400 Series is Underline Alpha/Numeric
1000 Series is Color Shift