APPENDIX D

FLOATING POINT PROCESSOR

D.1 GENERAL

The Floating Point Processor (FPP) performs floating-point arithmetic operations by providing an extension to the CPU arithmetic section. The FPP comprises a single card located in a dedicated slot of the main chassis. The FPP operates with the CPU and the memory via the standard GP Bus, however, some dedicated wiring between the FPP and the CPU is used to increase operating speed.

D.2 LOGIC LAYOUT

The floating-point operations are done between the floating-point accumulator (in the FPP) which contains the first operand, and the memory effective address which contain the second operand. The result is stored into the FPP accumulator or into the memory address, depending on the instruction Store indicator. Floating-point conversions are done between the FPP accumulator and the CPU internal registers A1, A2.

D.3 The FPP logic (Figure D-1) performs complete parallel operation on all data. Data are transferred in and out of the FPP one word (16 bits) at a time. The single-word exponent is loaded into the Exponent-Logic section and the double-word mantissa (or double-precision integer) is loaded into the Mantissa-Logic section. Processing is then done on the 48 bits in parallel. The FPP logic is completely microprogram controlled. The Command/Timing logic section contains the FPP sequensor, instruction decode and control logic, status register, and most FPP input/output control signals.

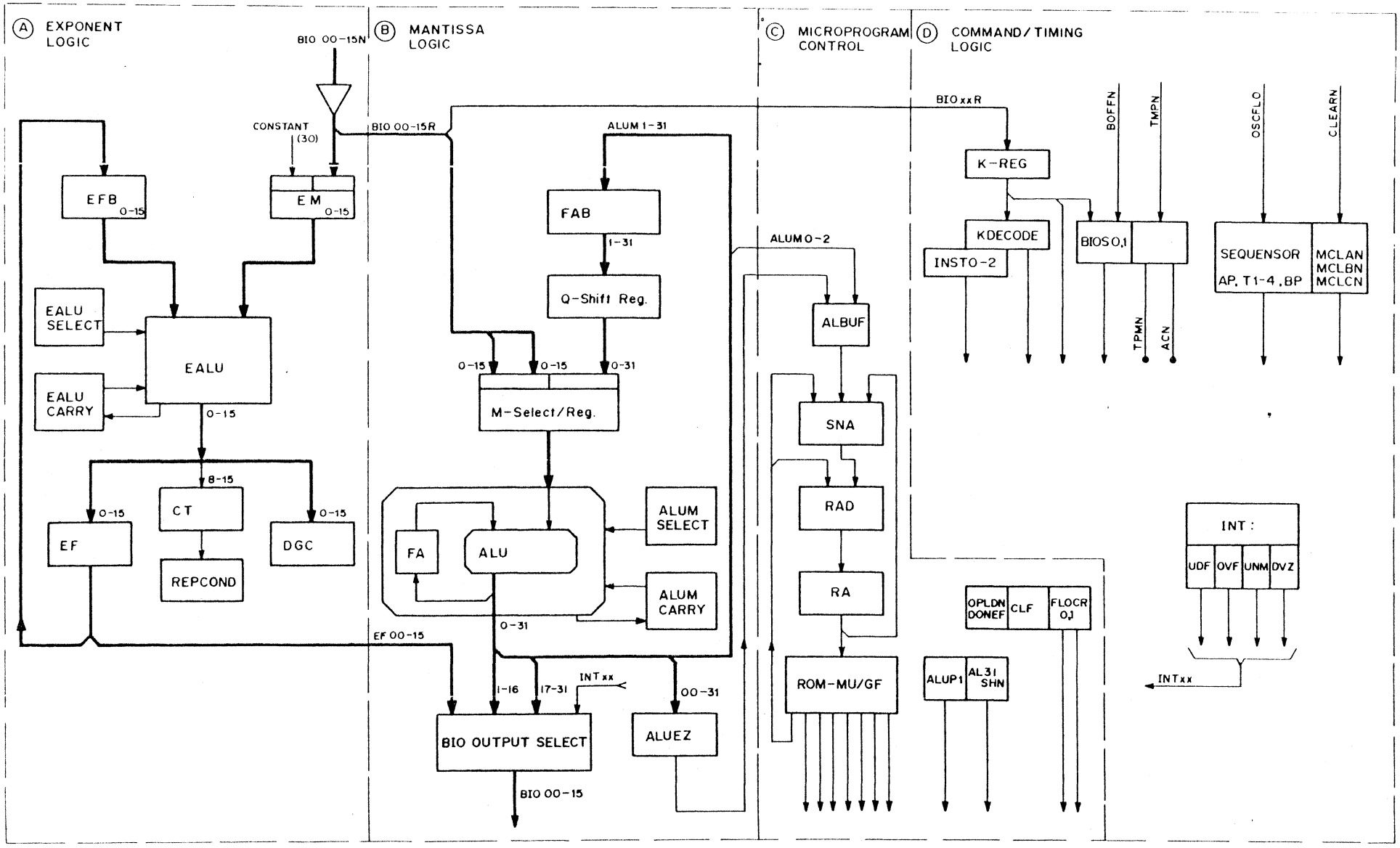


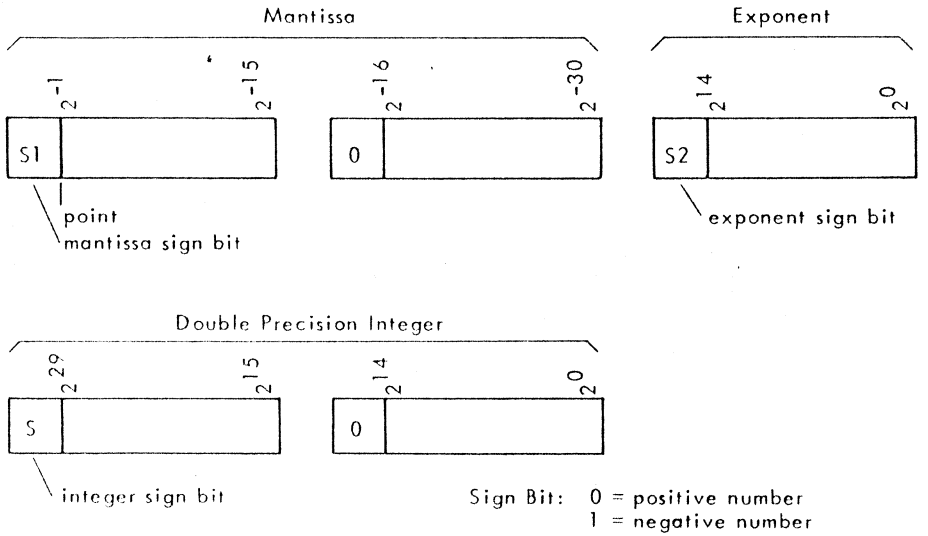
Figure D-1 FPP Block Diagram

D.4 INSTRUCTIONS

The FPP is operated by special floating-point instructions during the normal CPU programmed operation. The floating-point instruction is controlled by the CPU which is the Master of the dialogue and allows the FPP to execute the arithmetic operations. If the FPP is not installed in the system, an attempted floating-point instruction is trapped and a software subroutine may be initiated (if the subroutine is not directly called by a Call Function instruction). The command codes and functions of all FPP instructions are shown in Figure D-2.

D.5 DATA FORMAT

The two types of data handled by the FPP are floating-point data and double-precision integer data (following diagram). The FPP performs arithmetic operations on floating-point data only, and performs conversions between floating-point and integer data.



D.6 Floating-Point Data

Floating-point data are real numbers contained in three 16-bit words: a double-word mantissa and a single-word exponent.

D.7 The mantissa is a fraction (<1), with the point considered to be at the extreme left of the data quantity bits. The mantissa must be left normalized, thus, the left data bit is a 1 for positive numbers (sign bit 0) and a 0 for negative numbers (sign bit 1). Input data are checked for normalization (by comparing the left-most data bit with the sign bit) and the result of any arithmetic operation is normalized. The numerical range of the mantissa is given in Figure D-3.

D.8 The exponent is a single-precision integer with a range of -2^{15} to $+2^{15}-1$. The range of the exponent and the range of the complete floating-point number (mantissa with exponent) is shown on Figure D-3.

D.9 A floating-point data Zero is represented by three null words (mantissa and exponent). The FPP recognizes as Zero any data with a null mantissa, regardless of the value of the exponent. The FPP produces a null result under the following conditions:

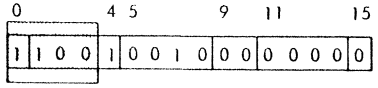
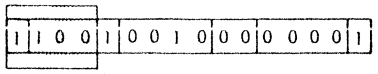
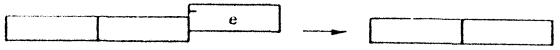
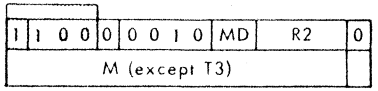
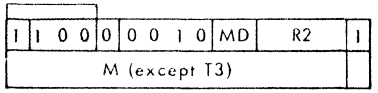
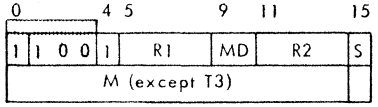
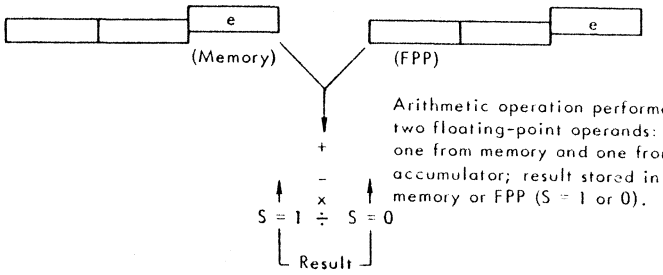
- during FAD/FSU if both received and stored operands are null.
- during FMU if either received or stored operand is null.
- during FLD if the received operand is null.
- during FDV if the stored operand (dividend) is null.

If the received operand (divisor) for an FDV command is null, the FPP sets Divide-by-Zero error status and does not change the stored operand.

D.10 The mantissa of all floating-point operands, other than null, must be normalized. If any unnormalized mantissa is received, the FPP sets Unnormalized-Operand error status and does not change the stored operand. All FPP operation results other than null are normalized.

D.11 Integer Data

The integer (Figure D-3) is a double-precision whole number with a sign bit and 30 quantity bits. The range is from -2^{30} to $+2^{30}-1$.

Instruction (Addressing Mode)	Format	Function	Explanation
FFL - Integer to F.P. (T1)		(A1), (A2) → FPA	(CPU) → FPP  Double-precision integer in CPU A1, A2 sent to FPP, converted to floating-point data.
FFX - F.P. to Integer (T1)		(FPA) → A1, A2	(FPP) → CPU  Floating-point data in FPP accumulator converted to double-precision integer, sent to CPU A1, A2. If exponent > 30 conversion not done (Overflow)
FLD,R - F.P. Load (T4-T7, T3)		(EA) → FPA	(Memory) (FPP)  Floating-point operand from memory loaded into FPP accumulator.
FST,R - F.P. Store (T4-T7, T3)		(FPA) → EA	(FPP) (Memory)  Floating-point data from FPP accumulator stored in memory.
FAD,R - F.P. Add FSU,R - F.P. Subtract FMU,R - F.P. Multiply FDV,R - F.P. Divide (T4-T7, T3)	 R1: 1000, 1010, 1100, 1110	S = 0: (EA) * (FPA) → FPA S = 1: (EA) * (FPA) → EA + − × ÷	 Arithmetic operation performed on two floating-point operands: one from memory and one from FPP accumulator; result stored in memory or FPP (S = 1 or 0).

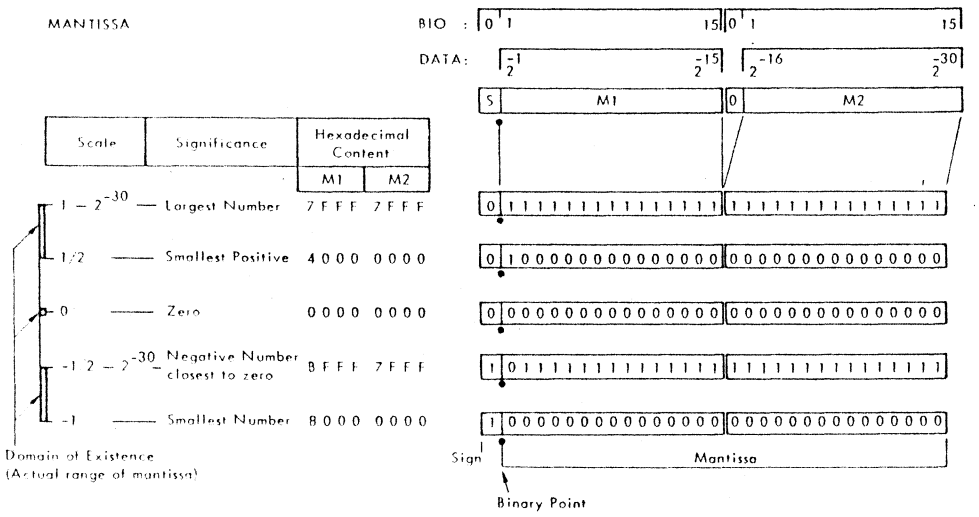
FPA: Floating Point Accumulator in FPP
EA : Memory Effective Address — points to first of three consecutive words in memory.
F.P.: Floating Point

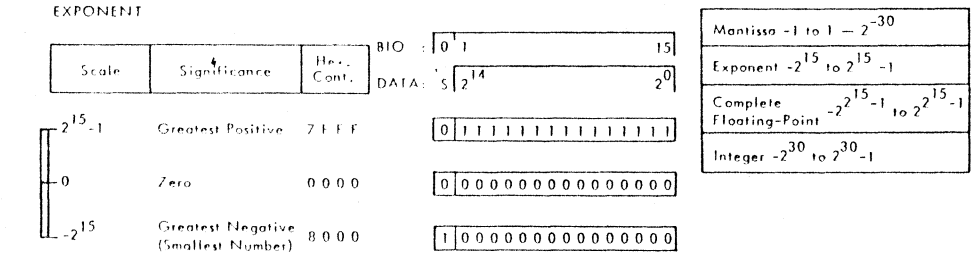
Shaded part [ ] not sent to FPP

Type	MD	R2	Effective Address
T3	0 1	≠0	in CPU register R2
T4	1 0	=0	in next word
T5	1 0	≠0	indexed
T6	1 1	=0	indirect
T7	1 1	≠0	indirect indexed

CPU Condition Register
CR0: result is zero
CR1: result is positive
CR2: result is negative
CR3: abnormal condition detected.

Figure D-2 FPP Instructions





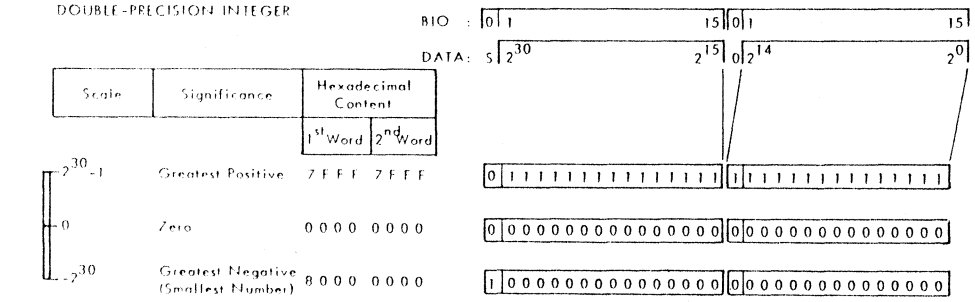


Figure D-3 Format and Range of Numbers

D.12 OPERATIONAL FLOW

The FPP logic is microprogram controlled. Figure D-4 is a flow diagram of all FPP operations. Each microinstruction is represented by a single block, with the microprogram address shown (in hexadecimal) in the upper-left corner of each block.

D.13 The paths from one microinstruction to another are controlled by the microprogram address control (paragraph D.25). Each microinstruction specifies the address of the next microinstruction, either directly (explicit address) or according to certain branch conditions: IDLE, TEFL, OPER, FIX, RESULT. These branch conditions are shown in Table D-1. The contents of the Address ROM are listed in Table D-2.

D.14 Start of Commands

The FPP goes to the Wait (/00) microinstruction upon the completion of any previous command or when the system Master Clear (MCL) is pressed. The FPP stays at Wait until a command is received from the CPU.

D.15 FFL Command

For the FFL Command, the first part of the mantissa is loaded into the M-register left half during microinstruction Wait (/00). A constant of 30 is loaded into the exponent register EM, and then cycled through EALU to EFB.

D.16 The flow branches via IDLE to microinstruction RDM2 (/01) where the second half of the mantissa is loaded into M-register right half. The complete M register is then loaded into the ALU. The flow now branches via TEFL to microinstruction RESALU (/03). The complete mantissa is loaded from the ALU into floating-point accumulator FA. The exponent of 30 is loaded into register EF, while EALU is reset to zero. The flow branches via RESULT to analyze the result of the operation.

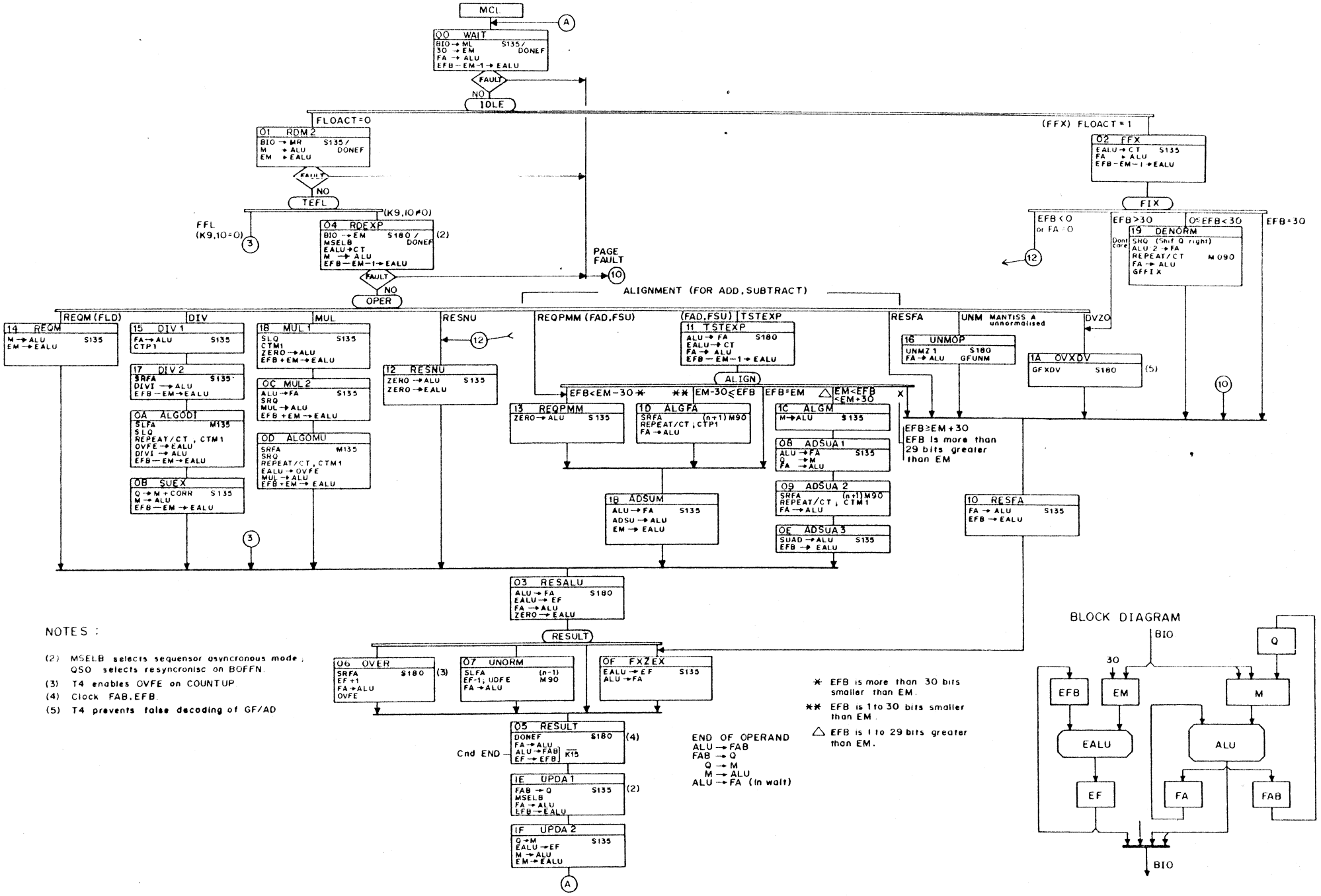


Figure D-4 FPP Flow Diagram

Table D-1 Next Address Branch Conditions (A)

NEXT ADDRESS GENERATION (NAGN) CONTROL BITS

Type		SEL Code μSNA0-1-2	Address Source (active low) SNA3	SNA4	SNA5	SNA6	SNA7	T4 SEQ
EXPLICIT 0		0 0 0	RA0	RA1	RA2	RA3	RA4	-
EXPLICIT 1		1 0 0						
BRANCH	IDLE	0 0 1	0	0	0	0	FLOACT	-
	TEFL	0 1 0	0	0	0	K09	K10	-
	RESULT	0 1 1	0	ALBUFZ	ALBUF0	ALBUF1	ALBUF2	*
	FIX	1 0 1	EFB00	ALBUFZ	EALU00	EQM1	0	-
	ALIGN	1 1 0	EFB00	EM00	EALU00	EQM1	DGC	*
	OPER	1 1 1	Q31D	ALBUFZ	KDVFLD	K06	ALBUF2Z	*

FLOACT = 1 : FFX instruction
ALBUFZ = 1 : (M) = 0; received mantissa is zero
ALBUF0 : overflow bit
ALBUF1 : mantissa sign bit
ALBUF2 : first data bit (2^{-1}) verifies if mantissa is normalized
ALBUF2Z = 1: (FA) = 0; stored mantissa is zero
Q31D = 1 : sign-bit and 1st-bit un-alike; operand is normalized
K06 = 0 : Add/Subtract; K06 = 1: multiply/divide
DGC = 1 : Data Greater than Constant (30)
EQM1 = 1 : Both operands have the same exponent

IDLE

Control FLOACT (Bus Timing from CPU)	Next Address
0	01 - RDM2
1	02 - FFX

TEFL

Control K09, 10	Next Address
0 0	03-RESALU (if FFL)
0 1	04-RDEXP
1 0	
1 1	

Table D-1 Next Address Branch Conditions (B)

OPER

Control (M0×M1) Q31D	ALBUFZ (mant. A=0)	Command Decode KDVFLD	K06	ALBUF2Z (mant. B=0)	Operation		Next Address
0	0	X	X	X	Mantissa unnormalized ERROR Q31D=M0×M1=0		16-UNMOP
0	1	0	0	0	FADSU	(Operand Null)	10-RESFA
0	1	0	0	1			
0	1	0	1	0	FMU		12-RESNU
0	1	0	1	1			
0	1	1	0	0	FLD		
0	1	1	0	1			
0	1	1	1	0	FDV		1A-OVXDV
0	1	1	1	1			
1	0	0	0	0	FADSU		11-TSTEXP
1	0	0	0	1			13-REQPMM
1	0	0	1	0	FMU		18-MUL1
1	0	0	1	1			12-RESNU
1	0	1	0	0	FLD		14-REQM
1	0	1	0	1			
1	0	1	1	0	FDV		15-DIV1
1	0	1	1	1			12-RESNU
1	1	X	X	X			

Q31D = 1 : sign-bit and 1st-bit un-alike; operand is normalized.
ALBUFZ = 1 : (M) = 0; received mantissa-A is zero.
ALBUF2Z = 1: (FA) = 0; stored mantissa-B is zero.
K06 = 0 : Add/Subtract; K06 = 1: Multiply/Divide

FIX

Control EFB0	ALBUFZ	EALU00	EAEB (EQM1)	SNA7	Operation	Next Address
0	0	0	0	0	EFB > 30	1A-OVXDV (EFB-30-1>0)
0	0	0	1	0	EAEB=1 => EALU0≠1	
0	0	1	0	0	0≤EFB < 30	19-DENORM
0	0	1	1	0	EFB=30	10-RESFA
0	1	X	0	0	Mantissa = 0	12-RESNU
0	1	0	1	0		
0	1	1	1	0	EFB=30	12-RESNU
X	X	X	X	1		
1	X	X	X	X		12-RESNU

Exponent < 0 (negative); number < 1 absolute value; may not be changed to fixed-point number.

Table D-1 Nest Address Branch Conditions (C)

ALIGN

Control					Compare Exponent	Next Address
EFB0	EM00	EALU00	EAEB	DGC		
0	0	0	0	0	EM < EFB < EM+30	1C-ALGM
X	1	0	0	0		
0	X	0	0	1	EM+30 ≤ EFB	10-RESFA
0	1	1	0	X	EM ≤ EFB-2^{15}	
1	1	0	0	1	EM+30 ≥ EFB ≥ EM	
0	0	0	1	X		if EALU00 = 0 ⇒ EAEB = 0
0	0	1	0	0	EFB < EM-30	13-REQPMM
1	0	1	0	0		
1	1	1	0	0		
1	0	0	0	X	EFB < EM-2^{15}	
0	0	1	0	1	EM-30 ≤ EFB < EM	1D-ALGFA
1	0	1	0	1		
1	1	1	0	1		
0	0	1	1	0		if EAEB = 1 ⇒ DGC = 0
0	0	1	1	1	0 ≤ EFB = EM	1B-ADSUM
1	1	1	1	1	EFB = EM < 0	
0	1	1	1	1		EF00 ≠ EM00 ⇒ EAEB = 0
X	1	0	1	X		
X	1	1	1	0		
1	0	X	1	X		

DGC = 1 : Data Greater than Constant (30).
EAEB = 1 : Exponents are the same.
EALU00 = 1: Output of EALU ≤ Constant (30).

RESULT

Control		Operation	Next Address
ALBUFZ	ALBUF0, 1, 2		
0	0 0 0		07-UNNORM
0	1 1 1		
0	0 0 1		05-RESULT
0	1 1 0		
0	0 1 X	Overflow	06-OVF
0	1 0 X		
1	0 0 0	Result = zero	OF-FXZEX
1	(≠0)		

Overflow occurs when the two most-significant bits are different after the 32 bits are adjusted right, for operands between -2^{30} and $+2^{30}-1$ inclusive (limit of fixed-point numbers).

ALBUFZ = 1: mantissa is zero
ALBUF0 : overflow bit
ALBUF1 : mantissa sign bit
ALBUF2 : first data bit (2^{-1}) verifies if mantissa is normalized

Table D-2 Address ROM (RAD) Listing

FLOATING POINT ADDRESS STORE

FPAS FORM 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8			16 4-BIT WORDS
FAPIP	DATA	/00FF LEADING FF	
FNAXOL	FPAS	0 0 0 8 3 0 0 C A 0 0 3 0 3 0 F	EXPLICIT ZERO
FNAXOH	FPAS	5 3 3 D 3 B E 9 5 5 0 0 0 0 0 0	
FNAIDL	FPAS	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	IDLE
FNAIDH	FPAS	0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 1	
FNARML	FPAS	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	RDM2
FNARMH	FPAS	0 0 0 0 0 0 0 0 0 0 0 0 4 4 4 3	
FNAREL	FPAS	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	RESULT
FNAREH	FPAS	0 0 0 0 0 0 0 F 7 5 6 6 6 5 7	
FNAXIL	FPAS	0 F B 0 0 0 0 0 0 0 7 0 B 0 0 0	EXPLICIT ONE
FNAXIH	FPAS	0 0 0 0 0 0 0 0 0 0 E 0 0 0 0	
FNAFXL	FPAS	0 2 0 2 0 2 0 2 0 0 0 2 0 0 0 2	FIX
FNAFXH	FPAS	0 2 0 2 0 2 0 2 0 0 0 9 0 0 0 A	
FNAAGL	FPAS	B 0 D 3 0 0 0 C 0 0 D 3 0 0 3 3	ALIGNMENT
FNAAGH	FPAS	0 0 0 0 0 0 0 C B 0 D 3 0 0 0 C	
FNAOPL	FPAS	0 0 0 0 0 0 0 0 2 5 4 4 2 8 3 1	OPERATION
FNAOPH	FPAS	A A 2 2 2 2 2 0 6 6 6 6 6 6 6 6	
		0 1 2 3 4 5 6 7 8 9 A B C D E F	

D.17 FFX Command

For the FFX command, no data is loaded during the Wait microinstruction (/00). A constant of 30 is loaded into exponent register EM. This constant, plus one, is then subtracted from the exponent in EFB and the difference is placed in EALU. The CPU command signal FLOACT is received as part of the FFX command, and the flow branches via IDLE to microinstruction FFX (/02). The exponent difference from 31 is loaded into the CT counter, and the mantissa is loaded into accumulator FA.

D.18 The branch condition FIX tests the exponent to determine if and how the mantissa must be modified for conversion to a double-precision integer:

- If either the mantissa (FA) is zero or the exponent (EFB) is negative, the flow branches to RESNU (/12) where both registers are reset to zero. This Reset-Null operation is completed in the RESALU (/03) microinstruction as the flow moves to the RESULT branch.
- If the exponent (EFB) is precisely 30, the mantissa can be used directly, without any change. The flow branches directly to microinstruction RESFA (/10).
- If the exponent (EFB) is greater than 30, the conversion to an integer is not done. The flow branches to microinstruction RESFA (/10), via microinstruction OVXDV (/1A) where the Overflow interrupt is set.
- If the exponent (EFB) is between 1 and 29 inclusive, the DENORM (/19) microinstruction is performed to convert the mantissa to an integer with the number of significant digits specified by the exponent. The mantissa is shifted right, end-off; the number of shifts is controlled by the CT counter which is loaded by the Wait microinstruction at the start of the FFX command.

The two FFX operations, other than the Null and the Overflow, pass via microinstruction RESFA (/10) to the FXZEX (/0F) branch of RESULT. The sequensor is resynchronized during the cycles UPDA by signals BOFFN and BSYCPUAN/BN to the logic for T3D.

D.19 FLD Command

The FLD command is complete when the two-word mantissa and the exponent have been loaded into the FPP from memory. The mantissa is loaded in microinstructions WAIT (/00) and RDM2 (/01). The exponent is loaded in microinstruction RDEXP (/04). The flow then branches via OPER to microinstruction REQM (/14). Between REQM (/14) and the following RESALU (/03), the loaded data are cycled through the ALU/EALU to the floating-point accumulator (FA/EF). If the mantissa received from memory equals Zero, the OPER branch is via RESNU (/12) to load both the mantissa and exponent registers with zero.

D.20 FST Command

The FST command is controlled by the CPU; no FPP microinstructions are used and the FST is not shown on the flow diagram. CPU Bus-control timing signals BSYCPUAN and BOFF gate the FPP accumulator out onto the BIO lines (logic b). The BSYCPUAN signal also operates the Bus Selection Counter (logic d) to switch the three 16-bit parts of the floating-point data onto the BIO lines.

D.21 FAD, FSU Commands

The double-word mantissa and the exponent are loaded into M/EM by microinstructions WAIT (/00), RDM2 (/01), and RDEXP (/04). This operand is added to, or subtracted from, the operand in FA/EF. The difference between the add and subtract operations is in the logic control of the ALU during microinstruction ADSUM (/1B) or ADSUA3 (/0E). The branch at OPER depends on the conditions of the mantissa (Table D-1):

- RESNU (/12) if both mantissa are zero.
- RESFA (/10) if only the new mantissa (M) is zero. The result is thus already in the accumulator (FA).
- REQPMM (/13) if the mantissa in FA only is zero. The received mantissa is added to zero by microinstructions REQPMM (/13) and ADSUM (/1B).
- TSTEXP (/11) to test and align the exponents if both mantissa are other than zero.

D.22 FMU Command

A double-word mantissa and an exponent are loaded into M/EM by microinstructions WAIT (/00), RDM2 (/01), and RDEXP (/04). The flow then branches via OPER to microinstruction MUL1 (/18). The mantissa of the first operand, already loaded in Q, is shifted left one bit (MUL1, /18) and then right one bit (MUL2, /0C); this operation removes the unused bit-0 of the right word. The actual multiplication

is performed during repeated sequensor cycles during microinstruction ALGOMU (0D).

D.23 FDV Command

The divisor operand is loaded into M/EM by microinstructions WAIT (/00), RDM2 (/01), and RDEXP (/04). The flow then branches via OPER to microinstruction DIV1 (/15). The dividend in FA is shifted right (SRFA in DIV2, /17) to ensure that the dividend is smaller than the divisor. The division is performed during repeated sequensor cycles during the microinstruction ALGODI (/0A).

D.24 LOGIC AND TIMING

The timings for the different FPP operations are provided together on Figure D-5. The control codes for the different logic operations are given on Figure D-6. Detailed logic diagrams are provided by Figure D-7 at the end of this section; specific sheets (a to d) of the logic are referenced by the letter suffix only, in the following manner: Figure D-7, sheet b is referenced as "logic b".

D.25 Microprogram Control (logic c)

The FPP operations are controlled by a selection of microinstruction control words stored in a read only memory (Control ROM). At time AP of a microinstruction cycle, a ROM address is selected and the ROM contents of that address are available to control the FPP logic for that cycle at time BP. Each microinstruction is shown as a separate block on the flow diagram, Figure D-4, and the hexadecimal ROM address is given in each block. The complete contents of the microprogram control ROM are shown in the ROM listing, Table D-3.

D.26 The microcommand bits from the ROM are grouped into functional fields that provide control for the different logic sections. The fields are listed and explained at the beginning of Table D-3.

D.27 Microinstruction Addressing is provided by the Next Address Selection field of each microinstruction word; this field sets up the address for the following microinstruction. The three control bits from the ROM (μSNA0-2) control the SNA multiplexer (logic c) for selecting an explicit address or one of the addresses selected by the operation Branch Conditions (Table D-1). The address-control conditions selected via SNA, and ROM bits μSNA0-2, select an address code from the ROM Address Logic (RAD). At the completion of the microinstruction, the rising-edge of AP which starts the next microinstruction clocks the new address into register RA, and the contents of another ROM word are present at the ROM output. The Next-Address control code is shown in Figure D-6.

```
*       DEFINITION OF THE MICROINSTRUCTION FORMAT

*       FIELD LENGTH DEFINITION
R       FORM    3,1 = 0,3,1,4,2,2,3,3,2,2,2,3,1 = 0
*       FIELD   0       1 2 3 4 5 6 7 8 9 10 11
*       FIELD   0     NEXT ADDRESS SELECTION

EXPO    EQU     0     EXPLICIT WITH ADDRESS < /10
IDLE    EQU     1     IDLE TEST
TEFL    EQU     2     READ EXPONENT OR FFL TEST
REST    EQU     3     RESULT TEST
EXPI    EQU     4     EXPLICIT WITH ADDRESS > /0F
IFIX    EQU     5     FIX TEST
ALGN    EQU     6     ALIGNMENT TEST
OPER    EQU     7     OPERATION TEST

*       FIELD   1     SEQUENSOR

S180    EQU     0     SINGLE BP;  AP IN 180 NS
M090    EQU     1     MULTIPLE BP EVERY 90 NS
S135    EQU     2     SINGLE BP;  AP IN 135 NS
M135    EQU     5     MULTIPLE BP EVERY 135 NS

*       FIELD   2     END

S       EQU     1     END OF PROCESS

*       FIELD   3     MANTISSA ARITHMETIC UNIT SELECTION

ALFA    EQU     0     ALUM = FA
ALUZ    EQU     1     ALUM = ZERO
ALFM    EQU     2     ALUM = M
MULT    EQU     4     ALUM = FA (+OR-) M ACCORDING MULTI ALGORITHM
ADSU    EQU     8     ALUM = FA (+OR-) M ACCORDING K07
SUAD    EQU     10    ALUM = M (+OR-) FA ACCORDING K07
DIVI    EQU     12    ALUM = FA (+OR-) M ACCORDING DIVI ALGORITHM

*       FIELD   4     MANTISSA ACCUMULATOR CONTROL

NFA     EQU     0     FA UNCHANGED
RFA     EQU     1     FA = ALUM RIGHT SHIFTED
LFA     EQU     2     FA = ALUM LEFT SHIFTED
AFA     EQU     3     FA = ALUM

*       FIELD   5     Q REGISTER CONTROL

NCQ     EQU     0     Q UNCHANGED
SLQ     EQU     1     Q SHIFTED LEFT
SRQ     EQU     2     Q SHIFTED RIGHT
LDQ     EQU     3     Q = FAR

*       FIELD   6     M REGISTER CONTROL

MNC     EQU     0     M UNCHANGED
MYQ     EQU     3     M = 0
MSB     EQU     4     M SELECT BIO BUT NO CLOCK (LOADING EXPON.)
MBR     EQU     5     M RIGHT = BIO (LOADING 2ND M.)
MBL     EQU     6     M LEFT = BIO (LOADING 1ST M.)

*       FIELD   7     EXPONENT ARITHMETIC UNIT SELECTION

EAMB    EQU     0     EALU = EFB-EM
EACB    EQU     1     EALU = EFB-EM-1
EQEM    EQU     2     EALU = EM
EALZ    EQU     4     EALU = ZERO
EQFB    EQU     6     EALU = EFB
EAPB    EQU     7     EALU = EFB+EM

*       FIELD   8     EM REGISTER CONTROL

EM30    EQU     1     EM = 30
EMNC    EQU     2     EM UNCHANGED
EMLB    EQU     3     EM = BIO

*       FIELD   9     EF REGISTER CONTROL

EFM1    EQU     0     EF = EF-1
EFP1    EQU     1     EF = EF+1
EFNC    EQU     2     EF UNCHANGED
EFLD    EQU     3     EF = EALU

*       FIELD   10    CT COUNTER CONTROL

CTM1    EQU     0     CT = CT-1
CTP1    EQU     1     CT = CT+1
CTNC    EQU     2     CT UNCHANGED
CTLD    EQU     3     CT = EALU

*       FIELD   11    MISCELLANEOUS COMMAND

GFUNM   EQU     1     SET UNNORMALISED FLAG
GFXDV   EQU     2     SET DVZO OR OVF FIX
GFEVF   EQU     3     SET OVERFLOW ON EXPONENT ARITHMETIC
GFEVP   EQU     4     SET OVERFLOW ON EXPONENT COUNT UP
GFEVN   EQU     5     SET UNDERFLOW ON EXPONENT COUNT DOWN
GFFIX   EQU     6     VALID. OF NEG. FFX CORRECTION
A       EQU     /A
B       EQU     /B
C       EQU     /C
D       EQU     /D
E       EQU     /E
F       EQU     /F
```

Table D-3 Microprogram (ROM) Listing (B)

	NAME		SNA	SEQ	E	ALU	FA	Q	M	EALU	EM	EF	CT	MISC.
00	WAIT	R	IDLE,	S135,	0,	ALFA,	AFA,	NCQ,	MBI,	EACB,	EM30,	EFNC,	CTNC,	0
01	RDM2	R	TEFL,	S135,	0,	ALEM,	NFA,	NCQ,	MBR,	EQEM,	EMNC,	EFNC,	CTNC,	0
02	FFX	R	IFIX,	S135,	0,	ALFA,	NFA,	NCQ,	MNC,	EACB,	EMNC,	EFNC,	CTLD,	0
03	RESALU	R	REST,	S180,	0,	ALFA,	AFA,	NCQ,	MNC,	EALZ,	EMNC,	EFLD,	CTNC,	0
04	RDEXP	R	OPER,	S180,	0,	ALEM,	NFA,	NCQ,	MSB,	EACB,	EMLB,	EFNC,	CTLD,	0
05	RESULT	R	EXP1,	S180,	S,	ALFA,	NFA,	NCQ,	MNC,	EQEM,	EMNC,	EFNC,	CTNC,	0
06	OVER	R	EXP0,	S180,	0,	ALFA,	RFA,	NCQ,	MNC,	EQEM,	EMNC,	EFP1,	CTNC,	GFEVP
07	UNORM	R	EXP0,	M090,	0,	ALFA,	LFA,	NCQ,	MNC,	EQEM,	EMNC,	EFM1,	CTNC,	GFEVN
08	ADSUA1	R	EXP0,	S135,	0,	ALFA,	AFA,	NCQ,	MYQ,	EQEM,	EMNC,	EFNC,	CTNC,	0
09	ADSUA2	R	EXP0,	M090,	0,	ALFA,	RFA,	NCQ,	MNC,	EQEM,	EMNC,	EFNC,	CTM1,	0
0A	ALGOD1	R	EXP0,	M135,	0,	DIVI,	LFA,	SLQ,	MNC,	EAMB,	EMNC,	EFNC,	CTM1,	GFEVF
0B	SUEX	R	EXP0,	S135,	0,	ALEM,	NFA,	NCQ,	MYQ,	EAMB,	EMNC,	EFNC,	CTNC,	0
0C	MUL2	R	EXP0,	S135,	0,	MULT,	AFA,	SRQ,	MNC,	EAPB,	EMNC,	EFNC,	CTNC,	0
0D	ALGOMU	R	EXP0,	M135,	0,	MULT,	RFA,	SRQ,	MNC,	EAPB,	EMNC,	EFNC,	CTM1,	GFEVF
0E	ADSUA3	R	EXP0,	S135,	0,	SUAD,	NFA,	NCQ,	MNC,	EQFB,	EMNC,	EFNC,	CTNC,	0
0F	FXZEX	R	EXP0,	S135,	0,	ALFA,	AFA,	NCQ,	MNC,	EQEM,	EMNC,	EFLD,	CTNC,	0
AD	NAME		SNA	SEQ	E	ALU	FA	Q	M	EALU	EM	EF	CT	MISC.
10	RESFA	R	EXP0	S135	0	ALFA	NFA	NCQ	MNC	EQFB	EMNC	EFNC	CTNC	0
11	TSTEXP	R	ALGN	S180	0	ALFA	NFA	NCQ	MNC	EACB	EMNC	EFNC	CTLD	0
12	RESNU	R	EXP0	S135	0	ALUZ	NFA	NCQ	MNC	EALZ	EMNC	EFNC	CTNC	0
13	REQPMM	R	EXP1	S135	0	ALUZ	NFA	NCQ	MNC	EQEM	EMNC	EFNC	CTNC	0
14	REQM	R	EXP0	S135	0	ALEM	NFA	NCQ	MNC	EQEM	EMNC	EFNC	CTNC	0
15	DIV1	R	EXP1	S135	0	ALFA	NFA	NCQ	MNC	EQEM	EMNC	EFNC	CTP1	0
16	UNMOP	R	EXP1	S180	0	ALFA	NFA	NCQ	MNC	EQEM	EMNC	EFNC	CTNC	GFUNM
17	DIV2	R	EXP0	S135	0	DIVI	RFA	NCQ	MNC	EAMB	EMNC	EFNC	CTNC	0
18	MUL1	R	EXP0	S135	0	ALUZ	NFA	SLQ	MNC	EAPB	EMNC	EFNC	CTM1	0
19	DENORM	R	EXP1	M090	0	ALFA	RFA	SRQ	MNC	EQEM	EMNC	EFNC	CTP1	GFFIX
1A	OVXDV	R	EXP1	S180	0	ALFA	NFA	NCQ	MNC	EQEM	EMNC	EFNC	CTNC	GFXDV
1B	ADSUM	R	EXP0	S135	0	ADSU	AFA	NCQ	MNC	EQEM	EMNC	EFNC	CTNC	0
1C	ALGM	R	EXP0	S135	0	ALEM	NFA	NCQ	MNC	EQEM	EMNC	EFNC	CTNC	0
1D	ALGFA	R	EXP1	M090	0	ALFA	RFA	NCQ	MNC	EQEM	EMNC	EFNC	CTP1	0
1E	UPDA1	R	EXP1	S135	0	ALFA	NFA	LDQ	MSB	EQFB	EMNC	EFNC	CTNC	0
1F	UPDA2	R	EXP0	S135	0	ALEM	NFA	NCQ	MYQ	EQEM	EMNC	EFLD	CTNC	0

Table D-3 Microprogram (ROM) Listing (C)

ADDRESS	FPCS0												FPCS1												FPCS2												FPCS3											
	0 - SNA								1 - SEQ							2 - END	3 - ALU								4 - FA			5 - Q				6 - M							7 - EALU							8 - EM				9 - EF				10 - CT				11 - GF			
0	0	0	1	0	0	1	0	0					0	0	0	0	1	1	0	0					1	1	0	0	0	1	0	1					1	0	1	0	0	0	0					1	0	1	0	0	0	0	
1	0	1	0	0	0	1	0	0					0	0	1	0	0	0	0	0					1	0	1	0	1	0	1	0					1	0	1	0	0	0	0					1	0	1	0	0	0	0	
2	1	0	1	0	0	1	0	0					0	0	0	0	0	0	0	0					0	0	0	0	0	1	1	0					1	0	1	1	0	0	0					1	0	1	1	0	0	0	
3	0	1	1	0	0	0	0	0					0	0	0	0	1	1	0	0					0	0	0	1	0	0	1	0					1	1	1	0	0	0	0					1	1	1	0	0	0	0	
4	1	1	1	0	0	0	0	0					0	0	1	0	0	0	0	0					1	0	0	0	0	1	1	1					1	0	1	1	0	0	0					1	0	1	1	0	0	0	
5	1	0	0	0	0	0	0	1					0	0	0	0	0	0	0	0					0	0	0	0	1	0	1	0					1	0	1	0	0	0	0					1	0	1	0	0	0	0	
6	0	0	0	0	0	0	0	0					0	0	0	0	0	1	0	0					0	0	0	0	1	0	1	0					1	0	1	0	0	0	0					0	1	1	0	1	0	0	
7	0	0	0	0	0	0	1	0					0	0	0	0	1	0	0	0					0	0	0	0	1	0	1	0					1	0	1	0	0	0	0					0	0	1	0	1	0	1	
8	0	0	0	0	0	1	0	0					0	0	0	0	1	1	0	0					0	1	1	0	1	0	1	0					1	0	1	0	0	0	0					1	0	1	0	0	0	0	
9	0	0	0	0	0	0	1	0					0	0	0	0	0	1	0	0					0	0	0	0	1	0	1	0					1	0	0	0	0	0	0					1	0	0	0	0	0	0	
0A	0	0	0	0	1	0	1	0					1	1	0	0	1	0	0	1					0	0	0	0	0	0	1	0					1	0	0	0	0	0	1					1	0	0	0	0	1	1	
0B	0	0	0	0	0	1	0	0					0	0	1	0	0	0	0	0					0	1	1	0	0	0	1	0					1	0	1	0	0	0	0					1	0	1	0	0	0	0	
0C	0	0	0	0	0	1	0	0					0	1	0	0	1	1	1	0					0	0	0	1	1	1	1	0					1	0	1	0	0	0	0					1	0	1	0	0	0	0	
0D	0	0	0	0	1	0	1	0					0	1	0	0	0	1	1	0					0	0	0	1	1	1	1	0					1	0	0	0	0	1	1					1	0	0	0	0	1	0	
0E	0	0	0	0	0	1	0	0					1	0	1	0	0	0	0	0					0	0	0	1	1	0	1	0					1	0	1	0	0	0	0					1	0	1	0	0	0	0	
0F	0	0	0	0	0	1	0	0					0	0	0	0	1	1	0	0					0	0	0	0	1	0	1	0					1	1	1	0	0	0	0					1	1	1	0	0	0	0	
10	0	0	0	0	0	1	0	0					0	0	0	0	0	0	0	0					0	0	0	1	1	0	1	0					1	0	1	0	0	0	0					1	0	1	0	0	0	0	
11	1	1	0	0	0	0	0	0					0	0	0	0	0	0	0	0					0	0	0	0	0	1	1	0					1	0	1	1	0	0	0					1	0	1	1	0	0	0	
12	0	0	0	0	0	1	0	0					0	0	0	1	0	0	0	0					0	0	0	1	0	0	1	0					1	0	1	1	0	0	0					1	0	1	1	0	0	0	
13	1	0	0	0	0	1	0	0					0	0	0	1	0	0	0	0					0	0	0	0	1	0	1	0					1	0	1	1	0	0	0					1	0	1	1	0	0	0	
14	0	0	0	0	0	1	0	0					0	0	1	0	0	0	0	0					0	0	0	0	1	0	1	0					1	0	1	1	0	0	0					1	0	1	1	0	0	0	
15	1	0	0	0	0	1	0	0					0	0	0	0	0	0	0	0					0	0	0	0	1	0	1	0					1	0	0	1	0	0	0					1	0	0	1	0	0	0	
16	1	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0					0	0	0	0	1	0	1	0					1	0	1	0	0	0	1					1	0	1	0	0	0	1	
17	0	0	0	0	0	1	0	0					1	1	0	0	0	1	0	0					0	0	0	0	0	0	1	0					1	0	1	0	0	0	0					1	0	1	0	0	0	0	
18	0	0	0	0	0	1	0	0					0	0	0	1	0	0	0	1					0	0	0	1	1	1	1	0					1	0	0	0	0	0	0					1	0	0	0	0	0	0	
19	1	0	0	0	0	0	1	0					0	0	0	0	0	1	1	0					0	0	0	0	1	0	1	0					1	0	0	1	1	1	0					1	0	0	1	1	1	0	
1A	1	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0					0	0	0	0	1	0	1	0					1	0	1	0	0	1	0					1	0	1	0	0	1	0	
1B	0	0	0	0	0	1	0	0					1	0	0	0	1	1	0	0					0	0	0	0	1	0	1	0					1	0	1	0	0	0	0					1	0	1	0	0	0	0	
1C	0	0	0	0	0	1	0	0					0	0	1	0	0	0	0	0					0	0	0	0	1	0	1	0					1	0	1	0	0	0	0					1	0	1	0	0	0	0	
1D	1	0	0	0	0	0	1	0					0	0	0	0	0	1	0	0					0	0	0	0	1	0	1	0					1	0	0	1	0	0	0					1	0	0	1	0	0	0	
1E	1	0	0	0	0	1	0	0					0	0	0	0	0	0	1	1					1	0	0	1	1	0	1	0					1	0	1	0	0	0	0					1	0	1	0	0	0	0	
1F	0	0	0	0	0	1	0	0					0	0	1	0	0	0	0	0					0	1	1	0	1	0	1	0					1	1	1	0	0	0	0					1	1	1	0	0	0	0	

D.28 Sequensor (logic d)

The sequensor logic is driven by the OSCFLO timing signal from the CPU and the control commands. OSCFLO is derived directly from the CPU sequensor and has the same 22.5 MHz frequency, with a duration of 45 nsec. One FPP sequensor cycle is produced for each microinstruction. Each cycle begins with an AP pulse and ends when the next AP pulse starts the next cycle. At each AP time, a new microprogram address is loaded into register RA (logic c) to select the next microprogram. At time BP, the conditions specified by the microinstruction are executed, such as data transfer from one register to another, arithmetic operation, etc. The timing signals T1, T2, T3, T4 are used within the sequensor logic only.

D.29 The sequensor is controlled by the microprogram to run in one of four modes: S135, S180, M090, M135. In either of the two single-cycle modes (S135, S180), the sequensor produces a cycle of a fixed length (either 135 or 180 nsec) and with a single BP pulse. In either of the two multiple-cycle modes (M090, M135), the sequensor produces a variable-length cycle where the BP pulse is repeated (every 90 or 135 nsec) until the repeat condition (REPCOND) is disabled. The Sequensor Cycles and the Repeat Condition code are shown on Figure D-6.

D.30 Instruction Loading (logic d)

An FPP instruction is loaded into the K-register under CPU Bus control during a CPU Fetch microprogram, while the FPP idles in the WAIT microinstruction (/00). The FPP instruction code (Figure D-6) is loaded into the K-register on the trailing edge of BSYCPUAN/BN. The load-instruction timing is shown in Figure D-5.

D.31 Operand Loading

Operands are loaded into the FPP registers M and EM during instruction FFL (to M only), FLD, and the four arithmetic instructions. The operand source is the CPU for the FFL instruction and the memory for the other instructions (Figure D-2). Part or all of the three-word operand (double-word mantissa and one-word exponent) is loaded during microinstructions WAIT (/00), RDM2 (/01), and RDEXP (/04), according to the specific instruction.

D.32 Operand loading is performed under CPU Bus control. The Bus timing is shown on Figure D-4. The CPU control signals BSYCPUAN and TMFN together enable the sequensor T3 count (logic d). Gating the operand word from the BIO lines into the M or EM register is controlled by the microprogram control bits (Figure D-6); in both cases, the data are gated on the leading edge of BP.

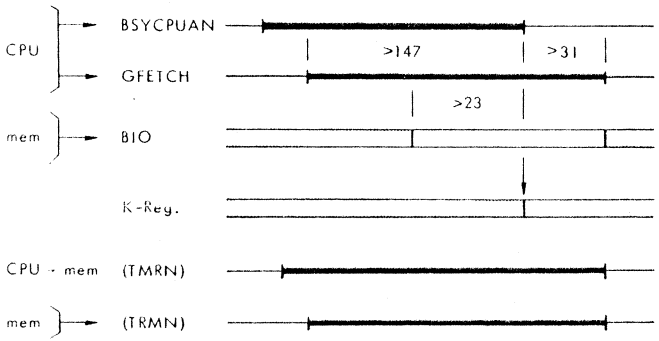
D.33 Store Operand

A Store-Operand operation is performed under CPU Bus control to transfer an integer operand to the CPU (FFX instruction) or to transfer a floating-point operand to the memory (FST instruction). The timing is shown on Figure D-5. The CPU control signals BOFFN and BSYCPUAN/BN (logic d) are used to enable the sequencing of Bus-selection bits BIOS0/S1. The same two control signals gate the selected data onto the BIO lines (logic b).

D.34 FPP Operation Control

The FPP operation processing is controlled by CPU signal FLOACT (Figure D-5; logic d) which is derived from the CPU microprogram bit GFLOT. FLOACT is received at the beginning of an FFX instruction. FLOACT controls the IDLE branch of the microprogram control to start the FFX operation. For the FLD and the four arithmetic instructions, FLOACT is received near the end of the CPU instruction to time the actual processing and to synchronise the end of the operation. FLOACT is a condition for loading the CR in the CPU.

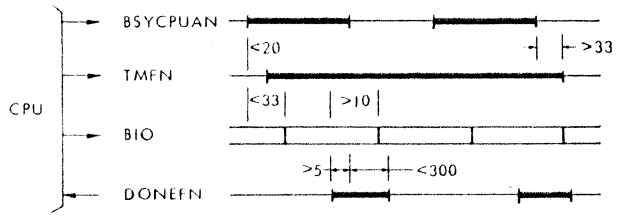
INSTRUCTION LOADING



Note: CPU drops BSYCPUAN at fixed time for Fetch

OPERAND LOADING

FFL Instruction



FLD, FAD, FSU, FMU, FDV Instructions

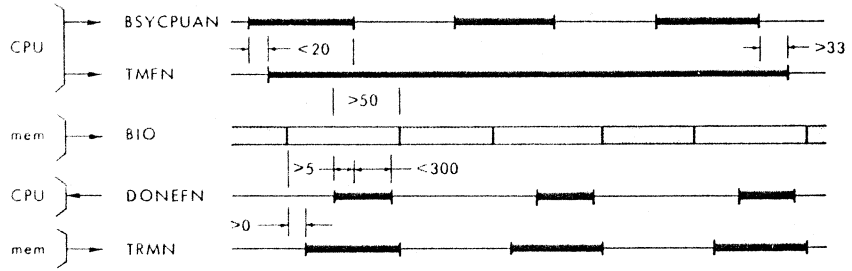
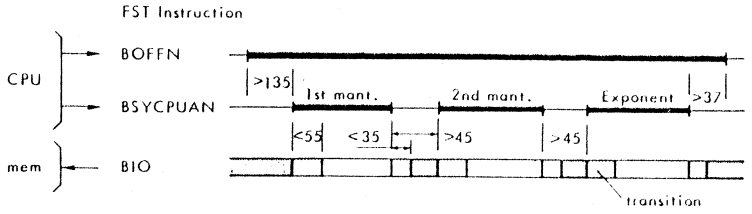


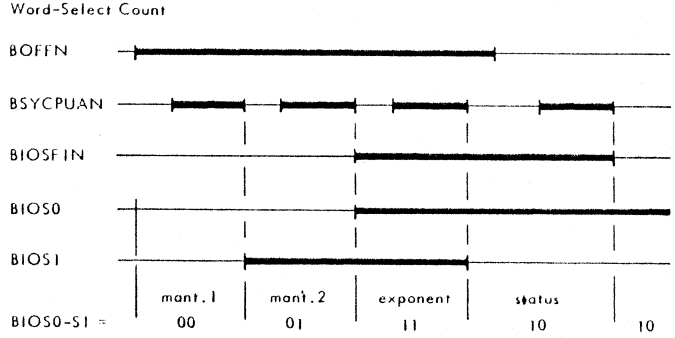
Figure D-5 FPP Timing (A)

STORE OPERAND

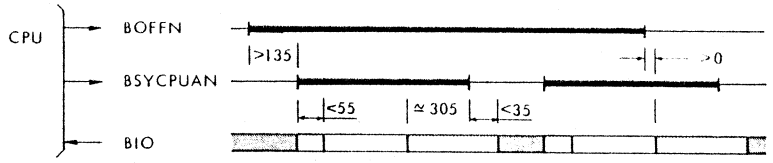
FST Instruction



Word-Select Count



FFX Instruction



FPP OPERATION PROCESSING

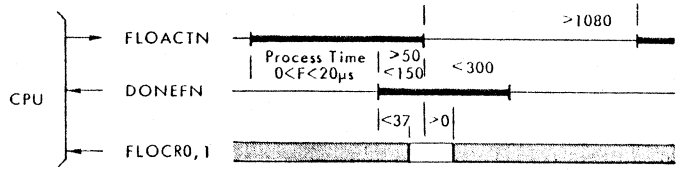
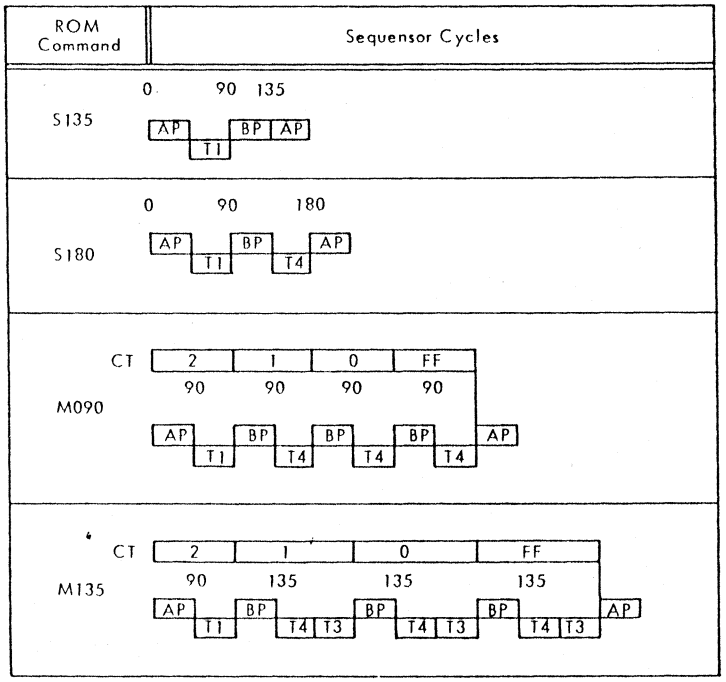


Figure D-5 FPP Timing (B)

Figure D-5 FPP Timing (A)

Figure D-5 FPP Timing (B)



S = Single Cycle
M = Multiple Cycle

REPEAT CONDITION

MUEFON	ALBUF2	ALBUF3	REPCOND
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	$\overline{\text{REPTCN}}$
1	0	1	$\overline{\text{REPTCN}}$
1	1	0	$\overline{\text{REPTCN}}$
1	1	1	$\overline{\text{REPTCN}}$

$\overline{\text{REPTCN}}$ = inactive
Repeat-Terminated Count
(Not CT0-7 = 1 = /FF)

Figure D-6 Logic Control Codes (A)

INSTRUCTION CODES

Inst.	BIO Lines, and K-Reg. Bits								K DECODE						KDVFLD	FLFX
									SST (K05)			INST				
	4	5	6	7	8	9	10	15	0	1	2	0	1	2		
FFL	1	0	0	1	0	0	0	0	0	1	0	0	1	0	0	1
FFX	1	0	0	1	0	0	0	1	0	1	1	0	1	1	0	1
FLD	0	0	0	1	0	-	-	0	0	0	0	0	0	0	1	0
FST	0	0	0	1	0	-	-	1	-	-	-	-	-	-	-	-
FAD	1	1	0	0	0	-	-	-	1	0	0	1	0	0	0	0
FSU	1	1	0	1	0	-	-	-	1	0	1	1	0	1	0	0
FMU	1	1	1	0	0	-	-	-	1	1	0	1	1	0	0	0
FDV	1	1	1	1	0	-	-	-	1	1	1	1	1	1	1	0

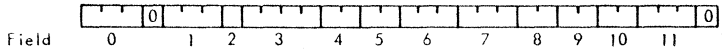
SST	BIO Lines															
	4	5	6	7	8	9	10	11	12	13	14	15				
							Device Address									
	1	-	-	-	1	1	0	0	0	0	0	0				
							KOR2D									

Figure D-6 Logic Control Codes (B)

Figure D-6 Logic Control Codes (A)

Figure D-6 Logic Control Codes (B)

Microinstruction Word Format



Field 0 — Next Address Selection

		μSNA0	μSNA1	μSNA2	(see Table D-1)
0	EXP0	0	0	0	Explicit, address </10
1	IDLE	0	0	1	Idle test
2	TEFL	0	1	0	Read exponent or FFL test
3	REST	0	1	1	Result test
4	EXP1	1	0	0	Explicit, address >/0F
5	IFIX	1	0	1	Fix test
6	ALIGN	1	1	0	Alignment test
7	OPER	1	1	1	Operation test

Field 1 — Sequensor

		μT3	μT4N	μREP	
0	S180	0	0	0	Single BP, AP in 180ns
1	M090	0	0	1	Multiple BP every 90ns
2	S135	0	1	0	Single BP, AP in 135ns
5	M135	1	0	1	Multiple BP every 135ns

Field 2 — End

		μEND	Microinstruction /05 only
1	S	1	End of Process

Field 3 — Mantissa ALU Selection

		μALU0	μALU1	μALU2	μALUCE	
0	ALFA	0	0	0	0	(FA) → ALUM
1	ALUZ	0	0	0	1	Zero → ALUM
2	ALEM	0	0	1	0	(M) → ALUM
4	MULT	0	1	0	0	(FA) + or - (M) → ALUM*MULTI Algorithm
8	ADSU	1	0	0	0	(FA) + or - (M) → ALUM*K07
10	SUAD	1	0	1	0	(M) + or - (FA) → ALUM*K07
12	DIVI	1	1	0	0	(FA) + or - (M) → ALUM*DIVI Algorithm

* = according to

Figure D-6 Logic Control Codes (C)

Figure D-6 Logic Control Codes (C)

Field 4 — Mantissa Accumulator Control

		μFAS1N	μFAS0N	
0	NFA	0	0	FA unchanged
1	RFA	0	1	(ALUM) Right-shifted → FA
2	LFA	1	0	(ALUM) Left-shifted → FA
3	AFA	1	1	(ALUM) → FA

Field 5 — Q-Register Control

		μQS0	μQS1	
0	NCQ	0	0	Q unchanged
1	SLQ	0	1	Shift-Left Q
2	SRQ	1	0	Shift-Right Q
3	LDQ	1	1	Load (FAB) → Q

Field 6 — M-Register Control

		μMSELN	μLDML	μLDMR	
0	MNC	0	0	0	M unchanged
3	MYQ	0	1	1	(Q) → M
4	MSB	1	0	0	BIO selected, but no clock (loading exponent)
5	MBR	1	0	1	BIO → M Right (loading 2nd. mant.)
6	MBL	1	1	0	BIO → M Left (loading 1st. mant.)

Field 7 — Exponent ALU Selection

		μEALU0	μEALU1	μEALU2	
0	EAMB	0	0	0	(EFB) - (EM) → EALU
1	EACB	0	0	1	(EFB) - (EM) - 1 → EALU
2	EQEM	0	1	0	(EM) → EALU
4	EALZ	1	0	0	Zero → EALU
6	EQFB	1	1	0	(EFB) → EALU
7	EAPB	1	1	1	(EFB) + (EM) → EALU

Figure D-6 Logic Control Codes (D)

Figure D-6 Logic Control Codes (D)

Field 8 — EM-Register Control

		μEMS	μEMLD	
1	EM30	0	1	30 → EM
2	EMNC	1	0	EM unchanged
3	EMLB	1	1	BIO → EM

Field 9 — EF-Register Control

		μEFON	μEF1	(EFLOADN)	
0	EFM1	L	L	H	(EF) -1 → EF [decrement]
1	EFP1	L	H	H	(EF) +1 → EF [increment]
2	EFNC	H	L	H	EF unchanged
3	EFLD	H	H	L	(EALU) → EF [load]

Field 10 — CT-Counter Control

		μCTON	μCT1	
0	CTM1	0	0	(CT) -1 → CT [decrement]
1	CTP1	0	1	(CT) +1 → CT [increment]
2	CTNC	1	0	CT unchanged
3	CTLD	1	1	(EALU) → CT [load]

Field 11 — Miscellaneous Commands

		μGP0	μGP1	μGP2	
1	GFUNM	0	0	1	Set unnormalized flag
2	GFXDV	0	1	0	Set DVZO or OVF FIX
3	GFEVF	0	1	1	Set overflow on exponent arithmetic
4	GFEVP	1	0	0	Set overflow on exponent count-up
5	GFEVN	1	0	1	Set underflow on exponent count-down
6	GFFIX	1	1	0	Validation of negative FFX correction

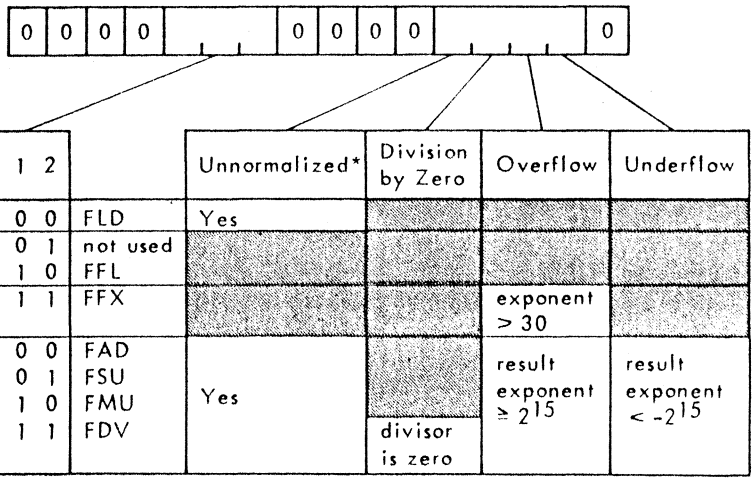
Figure D-6 Logic Control Codes (t)

D.35 STATUS AND INTERRUPTS

The floating-point instruction executions are not interruptable; if a Memory Management Unit detects a Page Fault, the floating-point instruction is stopped like any other instruction. An abnormal condition detected during a floating-point instruction execution sets the corresponding flag in the status register, sets the condition register to 3, and generates a Floating-Point interrupt. The Floating-Point interrupt can be connected only to one of the eight internal interrupts (refer to CPU Section I, Interrupt System).

D.36 At the moment any error condition is detected, the activating status-register bit loads the instruction code from the D-register into the INST register. The INST code is then included with the status word to specify which instruction type caused the error. The system should respond to the FPP interrupt by sending an SST instruction with device address equal to zero. The floating-point status is then transferred to the CPU, as follows:

STATUS



* If Unnormalized is set, the operation is aborted, no other flag is set, and the FPP accumulator remains unchanged.

D.37 The interrupt and the error bit (right byte) of the status word are reset at the end of an SST instruction. If another floating-point instruction is performed before the SST instruction is received, the error bits of the status word indicate the accumulated status (logical Or) of both instructions. The INST code in the left byte, however, retains the code for the first instruction where error status was set.

D.38 SIGNAL LIST

A complete list of all FPP input and output signals is given in Table D-4.

D.39 CARD LAYOUT

The layout of the FPP card is provided in Figure D-8. There are no U-links or other operator controls located on the FPP card.

D.40 PARTS LIST

A list of FPP components is provided in Table D-5.

Table D-4 FPP Signal List

Signal	Conn.	Logic Sheet	Remarks
INPUT TO FPP			
BIO00-15N	3---	D	
BOFFN	5B13	D	BIO contents must be defined by the FPP
BSYCPUAN	5A12	D	CPU is master of the Bus
CLEARN	3A39	D	
FLOACT	5A11	D	CPU activation signal for FFX and CR
GFETCH	5A13	D	CPU fetch cycle
MFAULTN	5B20	D	MMU detects a Page Fault
OSCFLO	5A17	D	CPU clock signal
TMFN	5B12	D	CPU microcommand bit
TMPN	3A31	D	Bus timing from CPU for SST dialogue
TRMN	3A28	D	Bus timing from memory
OUTPUT FROM FPP			
ACN	3A34	D	Accept for SST command
BIO00-15N	3---	B	
DONEFN	5A14	C	FPP operation was done
FLOCR0	5B14	C	FPP condition register bits to CPU
FLOCR1	5A15	C	
FPPABS	5B15	C	Held inactive when FPP board is inserted
INTFPPN	3A16	D	FPP interrupt
TPMN	3A32	D	Bus timing to CPU for SST command

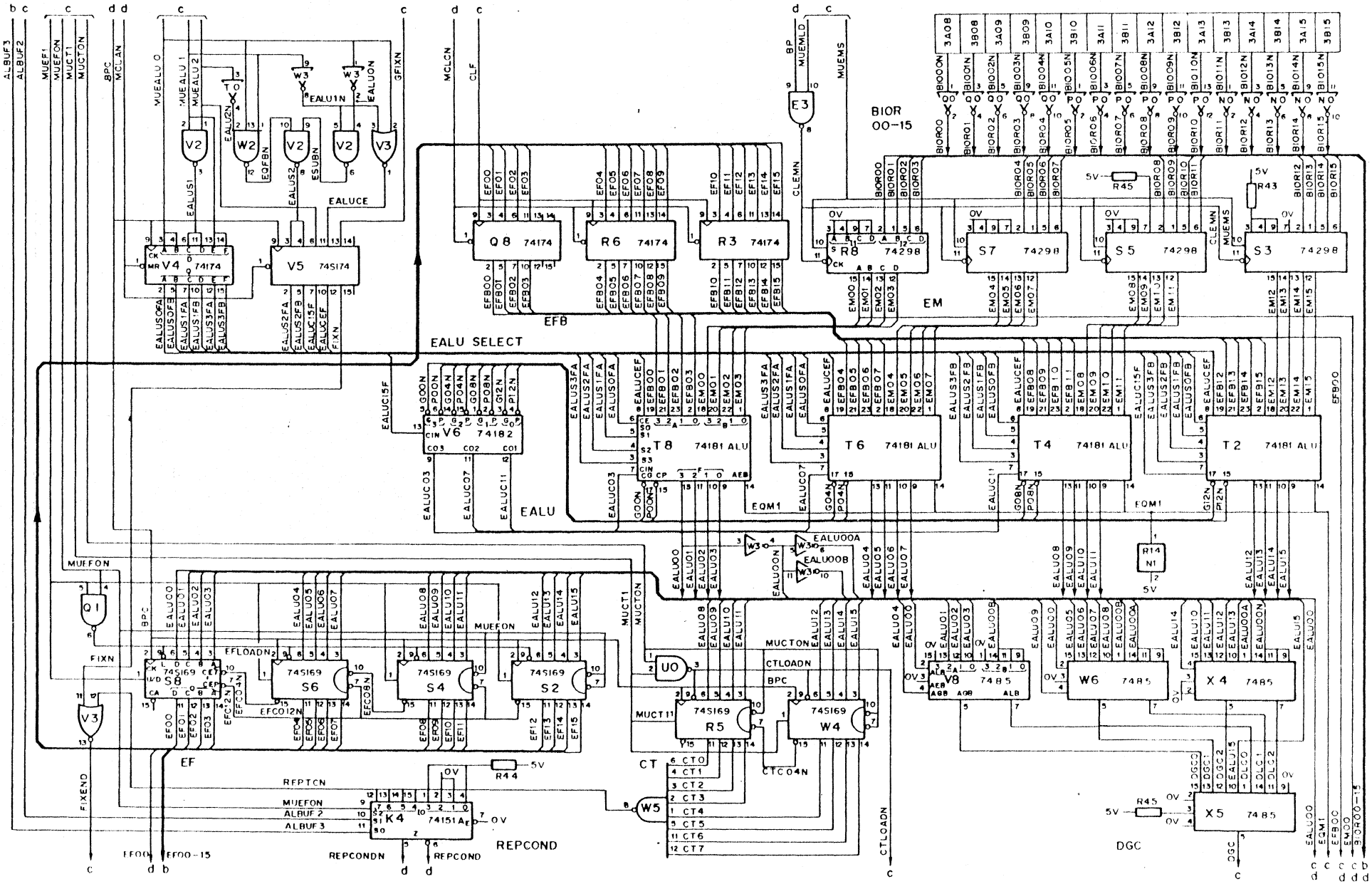


Figure D-7a EPP Logic Diagram

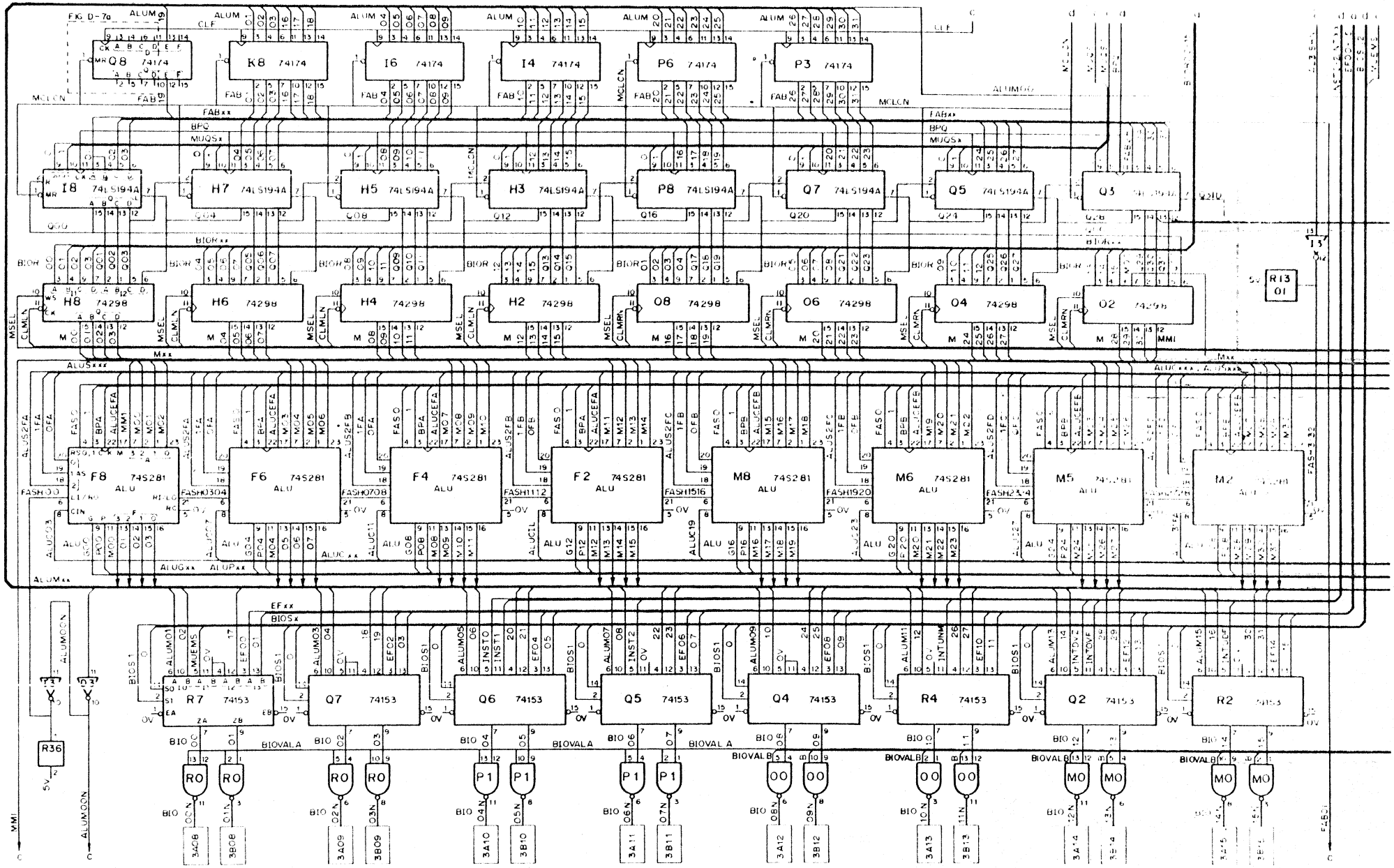


Figure D-7b FPP Logic Diagram

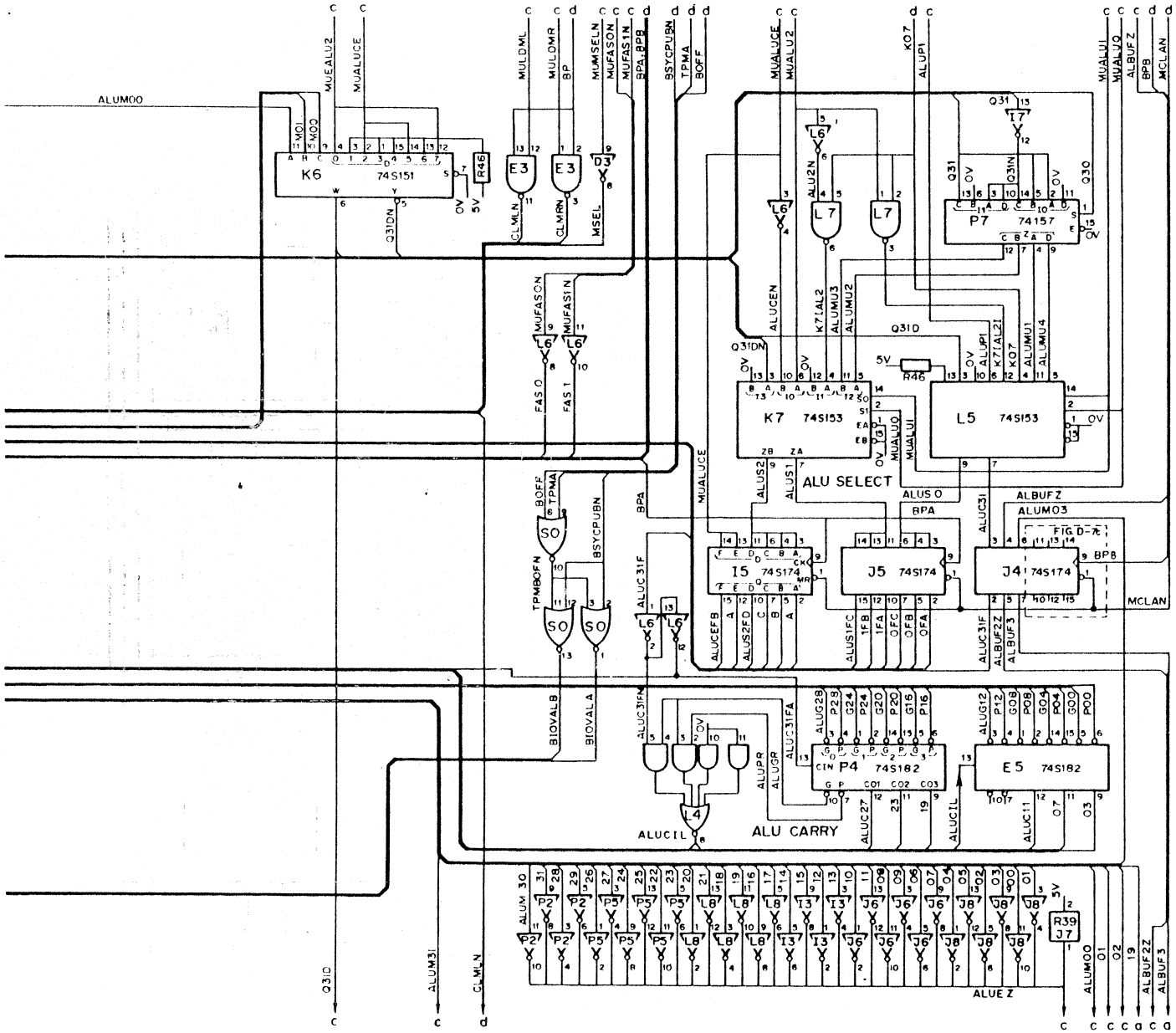


Figure D-7b EPP Logic Diagram

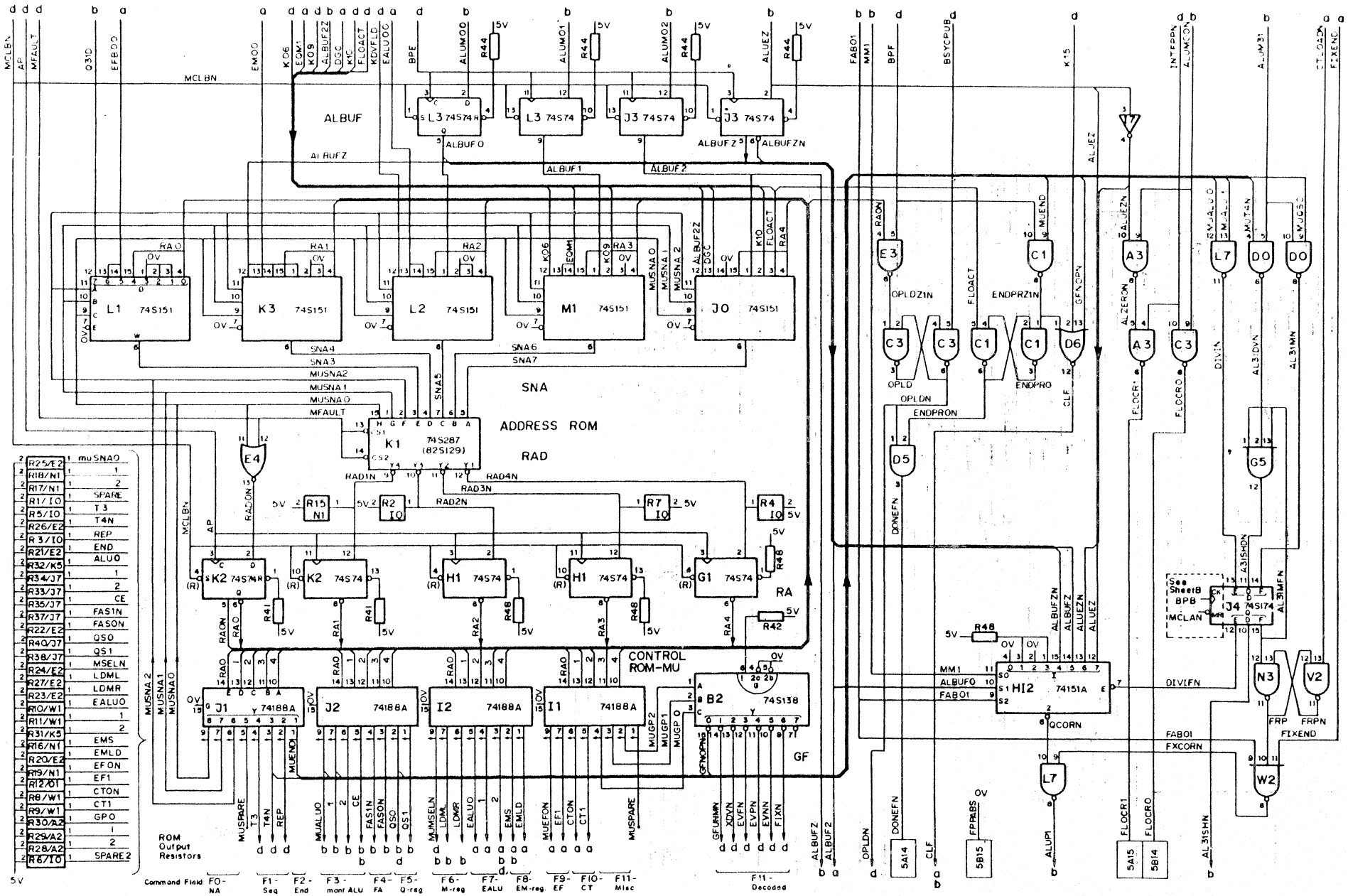


Figure D-7c FPP Logic Diagram

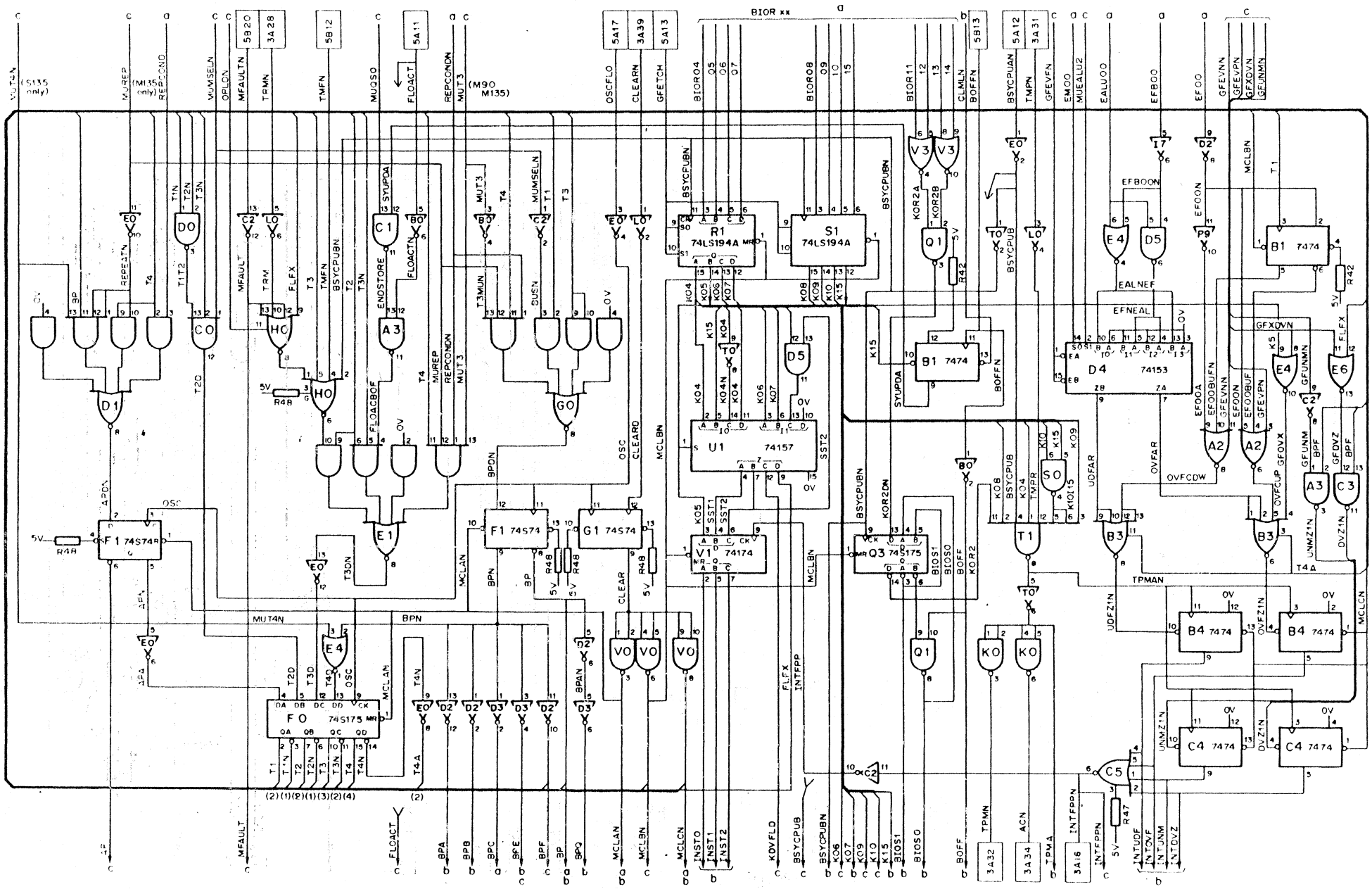


Figure D-7d FPP Logic Diagram

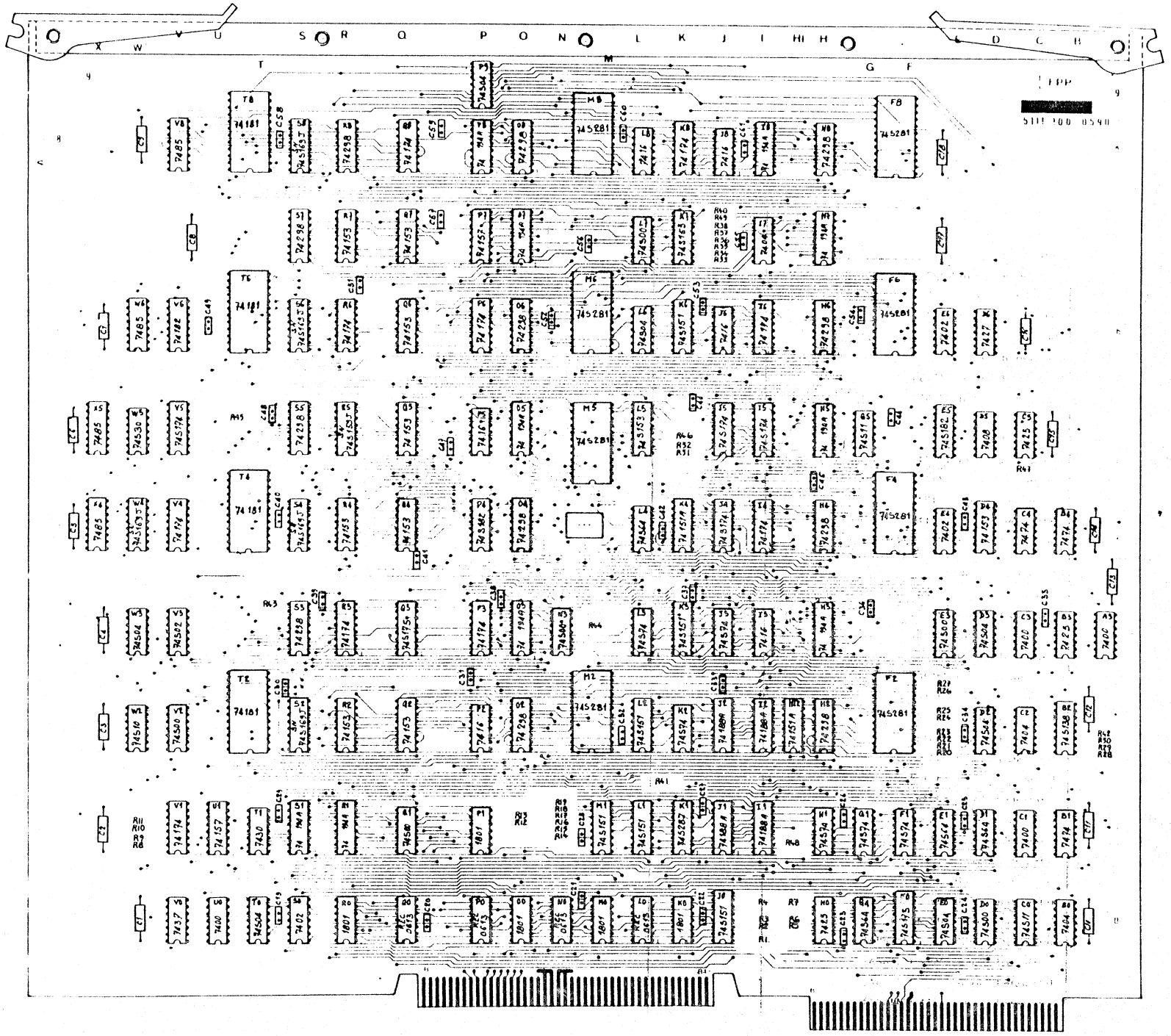


Figure D-8 FPP Card Layout

Table D-5 FPP Parts List

Reference	Description	12NC Code
	Printed circuit	5111 100 05901
A3, C1, C3, U0.	Integrated circuit 7400	
E4, E6, S0.	Integrated circuit 7402	
B0, C2, I7.	Integrated circuit 7404	
D5.	Integrated circuit 7408	
I3, J6, J8, L8, P2, P5.	Integrated circuit 7416	
B3, C5, H0.	Integrated circuit 7425	
D6.	Integrated circuit 7427	
T1.	Integrated circuit 7430	
V0.	Integrated circuit 7437	
B1, B4, C4.	Integrated circuit 7474	
V8, W6, X4, X5.	Integrated circuit 7485	
B2.	Integrated circuit 74S138	
H12, K4.	Integrated circuit 74151A	
D4, Q2, Q4-7, R2, R4, R7.	Integrated circuit 74153	
P7, U1.	Integrated circuit 74157	
I4, I6, K8, P3, P6, Q8, R3, R6, V1, V4.	Integrated circuit 74174	
T2, T4, T6, T8.	Integrated circuit 74181	
V6.	Integrated circuit 74182	
C0, G5.	Integrated circuit 74S11	
F1.	Integrated circuit 2721 (82S129)	
H2, H4, H6, H8, O2, O4, O6, O8, R8, S3, S5, S7.	Integrated circuit 74298	
D0, E3, L7, N3, Q1, V2.	Integrated circuit 74S00	
V3.	Integrated circuit 74S02	
D2, D3, E0, L6, P9, T0, W3.	Integrated circuit 74S04	
W2.	Integrated circuit 74S10	
W5.	Integrated circuit 74S30	
D1, E1, G0, L4.	Integrated circuit 74S64	
F1, G1, H1, J3, K2, L3.	Integrated circuit 74S74	
J0, K3, K6, L1, L2, M1.	Integrated circuit 74S151	
K7, L5.	Integrated circuit 74S153	
I5, J4, J5, V5.	Integrated circuit 74S174	
E5, P4.	Integrated circuit 74S182	
F2, F4, F6, F8, M2, M5, M6, M8.	Integrated circuit 74S281	
H3, H5, H7, I8, O3, O5, O7, P8, R1, S1.	Integrated circuit 74LS194A	
F0, Q3.	Integrated circuit 74S175	
K0, M0, O0, P1, R0.	Integrated circuit 1801	
L0, N0, P0, Q0.	Integrated circuit REC0613	
R5, S2, S4, S6, S8, W4.	Integrated circuit SN74S169J	
J1.	Integrated circuit 2681 (74188A)	
I2.	Integrated circuit 269J (74188A)	
I2.	Integrated circuit 2701 (74188A)	
I1.	Integrated circuit 2711 (74189A)	

Table D-5 FPP Parts List contd.

Reference	Description	12NC Code
C1-18.	Capacitor 10µF, 25V FITCO.	
C19-49, 51-61.	Capacitor 10µF.	
R1, 3, 5, 6, 8, 9, 19-24, 26-30, 32-34, 36, 37, 38, 40-48.	Resistor 1Kn, 1/8W, 5%.	
R39.	Resistor 220n, 1/8W, 5%.	
R2, 4, 7, 14, 15.	Resistor 470n, 1/8W, 5%.	
R10-12, 16-18, 35.	Resistor 4.7Kn, 1/8W, 5%.	
R25, 31.	Resistor 10Kn, 1/8W, 5%.	

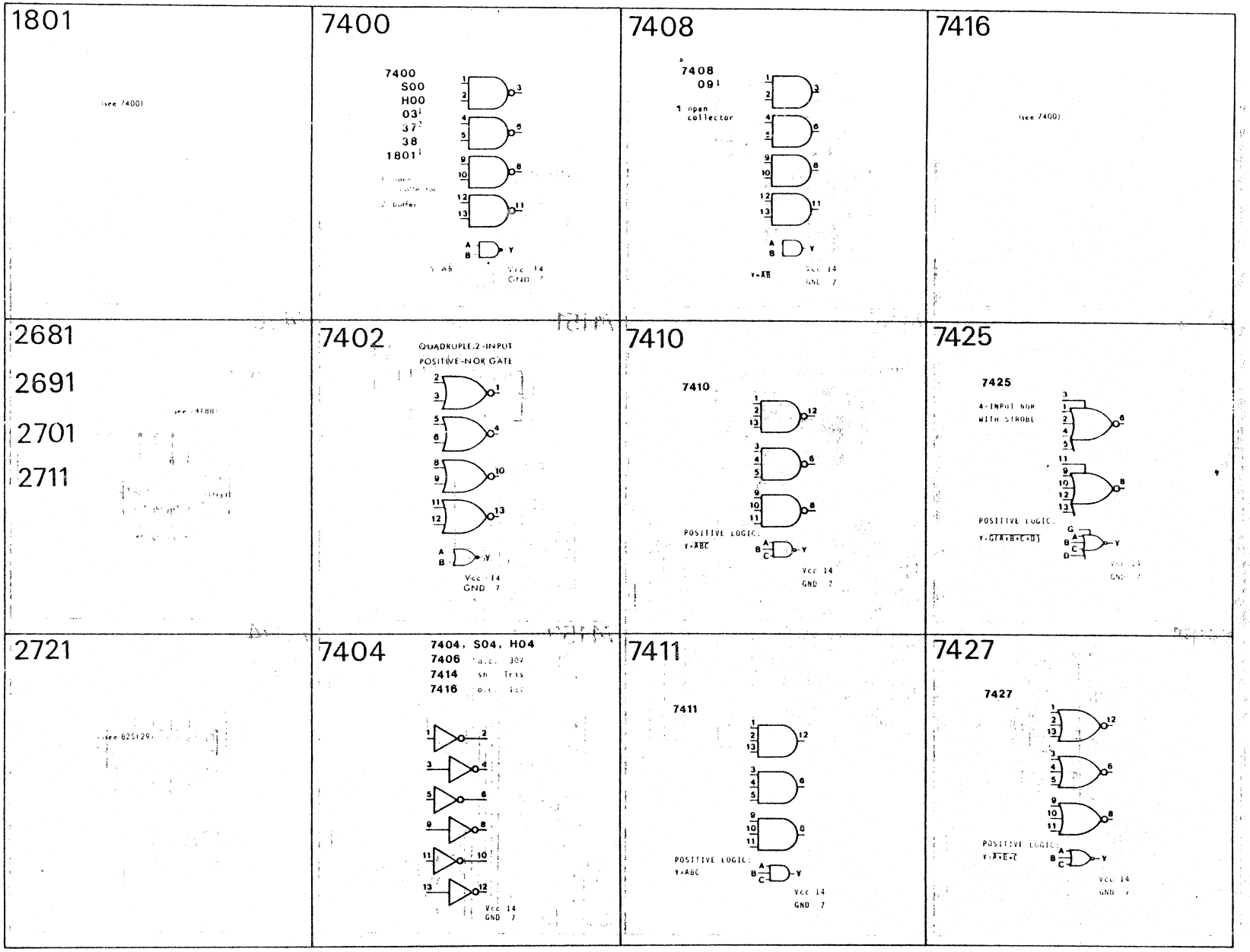
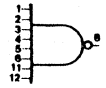


Figure D-9a FPP IC Guide

7430

7430

8-INPUT NAND GATE

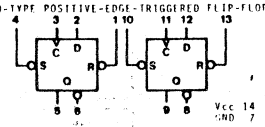


Vcc 14
GND 7

7474

7474, H74, S74

DUAL D-TYPE POSITIVE-EDGE-TRIGGERED FLIP-FLOPS

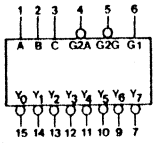


Vcc 14
GND 7

FUNCTION TABLE					
INPUTS				OUTPUTS	
S	R	C	D	Q	$\bar{Q}$
L	H	X	X	H	L
H	L	X	X	L	H
L	L	X	X	H	H
H	H	↑	H	H	L
H	H	↑	L	L	H
H	H	L	X	Q_0	$\bar{Q}_0$

74138

DECODER



Vcc 16
GND 8

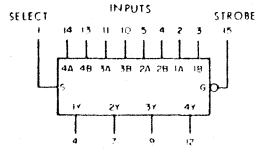
FUNCTION TABLE

INPUTS				OUTPUTS							
ENABLE G1	ENABLE G2	SELECT C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
X	H	X	X	X	H	H	H	H	H	H	H	H
L	X	X	X	X	H	H	H	H	H	H	H	H
H	L	L	L	L	L	H	H	H	H	H	H	H
H	L	L	L	H	H	L	H	H	H	H	H	H
H	L	L	H	L	H	H	L	H	H	H	H	H
H	L	L	H	H	H	H	H	L	H	H	H	H
H	L	H	L	L	H	H	H	H	L	H	H	H
H	L	H	L	H	H	H	H	H	H	L	H	H
H	L	H	H	L	H	H	H	H	H	H	L	H
H	L	H	H	H	H	H	H	H	H	H	H	L

*G2 = G2A+G2B

74157

QUAD 2-to-1-LINE DATA SELECTORS/MULTIPLEXERS



Vcc 16
GND 8

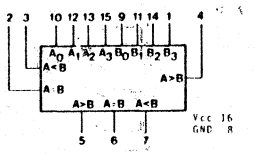
7437

(see 7400)

7485

7485

4-BIT MAGNITUDE COMPARATOR



Vcc 16
GND 8

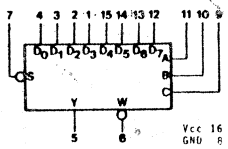
FUNCTION TABLES

COMPARING INPUTS					CASCADING INPUTS			OUTPUTS		
A3,B3	A2,B2	A1,B1	A0,B0		A>B	A<B	A=B	A>B	A<B	A=B
A3>B3	X	X	X		X	X	X	H	L	L
A3<B3	X	X	X		X	X	X	L	H	L
A3=B3	A2>B2	X	X		X	X	X	H	L	L
A3=B3	A2<B2	X	X		X	X	X	L	H	L
A3=B3	A2=B2	A1>B1	X		X	X	X	H	L	L
A3=B3	A2=B2	A1<B1	X		X	X	X	L	H	L
A3=B3	A2=B2	A1=B1	A0>B0		X	X	X	H	L	L
A3=B3	A2=B2	A1=B1	A0=B0		X	X	X	L	H	L
A3=B3	A2=B2	A1=B1	A0<B0		H	L	L	H	L	L
A3=B3	A2=B2	A1=B1	A0=B0		L	H	L	L	H	L
A3=B3	A2=B2	A1=B1	A0=B0		L	L	H	L	L	H
*85, *585										
A3=B3	A2=B2	A1=B1	A0=B0		X	X	H	L	L	H
A3=B3	A2=B2	A1=B1	A0=B0		H	H	L	L	L	L
A3=B3	A2=B2	A1=B1	A0=B0		L	L	L	H	H	L
*L85										
A3=B3	A2=B2	A1=B1	A0=B0		L	H	H	L	H	H
A3=B3	A2=B2	A1=B1	A0=B0		H	L	H	H	L	H
A3=B3	A2=B2	A1=B1	A0=B0		H	H	H	H	H	H
A3=B3	A2=B2	A1=B1	A0=B0		H	H	L	H	H	L
A3=B3	A2=B2	A1=B1	A0=B0		L	L	L	L	L	L

H=high level; L=low level; X=irrelevant

74151

DATA SELECTOR/MULTIPLEXER



Vcc 16
GND 8

FUNCTION TABLE

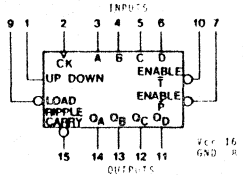
INPUTS				OUTPUTS	
SELECT C	B	A	STROBE	Y	W
X	X	X	H	L	H
L	L	L	L	D0	D0
L	L	H	L	D1	D1
L	H	L	L	D2	D2
L	H	H	L	D3	D3
H	L	L	L	D4	D4
H	L	H	L	D5	D5
H	H	L	L	D6	D6
H	H	H	L	D7	D7

D0,D1...D7 = the level of the D respective input

74169

74169

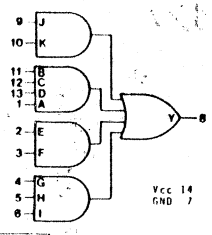
SYNCHRONOUS 4-BIT UP/DOWN COUNTERS



Vcc 16
GND 8

7464

4-2-2-3-INPUT AND-OR-INVERT GATES



Vcc 14
GND 7

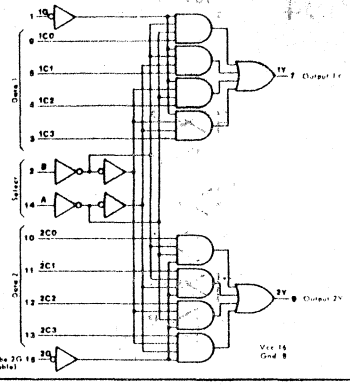
FUNCTION TABLE

INPUTS						OUTPUTS	
PRESET	CLEAR	CLOCK	J	K		Q	$\bar{Q}$
L	H	X	X	X		H	L
H	L	X	X	X		L	H
L	L	X	X	X		H	H
H	H	↓	L	L		Q_0	$\bar{Q}_0$
H	H	↓	H	L		H	L
H	H	↓	L	H		L	H
H	H	↓	H	H		TOGGLE	
H	H	↑	X	X		Q_0	$\bar{Q}_0$

Positive Logic: Y=ABCD+EF+GHI+JK

74153

DUAL 4-LINE-TO-1-LINE DATA SELECTOR/MULTIPLEXER

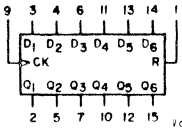


Vcc 16
Gnd 8

74174

74174

HEX D-TYPE FLIP-FLOPS WITH COMMON CLOCK AND RESET



Vcc 16
GND 8

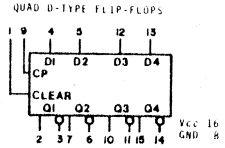
FUNCTION TABLE				
INPUTS			OUTPUTS	
R	C	D	Q	$\bar{Q}$
L	X	X	L	H
H	↑	H	H	L
H	↑	L	L	H
H	L	X	Q_0	$\bar{Q}_0$

Figure D-9b FPP IC Guide

74175

74175

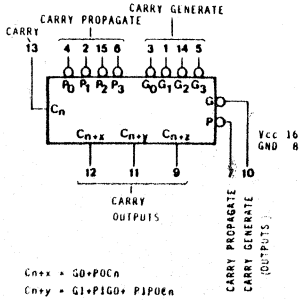
QUAD D-TYPE FLIP-FLOPS



FUNCTION TABLE				
INPUTS			OUTPUTS	
R	L	D	Q	$\bar{Q}$
L	↑	↑	L	H
H	↑	L	L	H
H	↑	H	H	L
H	L	X	Q_0	$\bar{Q}_0$

74182

LOOK-AHEAD CARRY GENERATOR



$C_{n+x} = G0 + P0C_n$

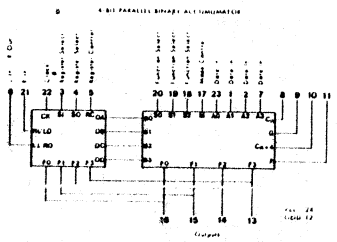
$C_{n+y} = G1 + P1G0 + P1P0C_n$

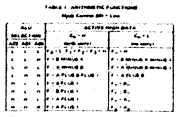
$C_{n+z} = G2 + P2G1 + P2P1G0 + P2P1P0C_n$

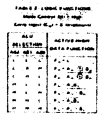
$G = G3(P3+G2)\ (P3+P2+G1)\ (P3+P2+P1+G0)$

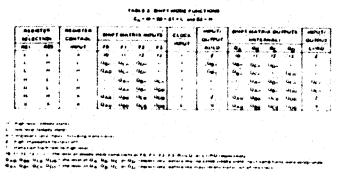
$P = P3P2P1P0$

74281







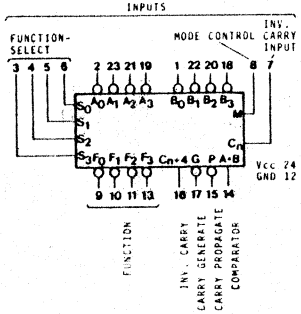


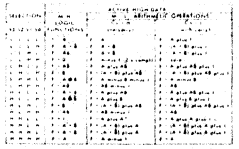
REC 0613

(see 7404)

74181

ARITHMETIC LOGIC UNIT

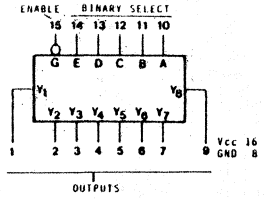




74188

74188A

256-BIT PROGRAMMABLE READ-ONLY MEMORY

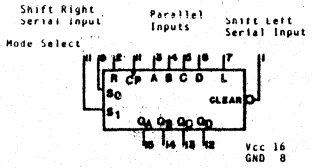


Organized as 32 Words of 8 Bits Each

74194

74194

4-BIT SHIFT REGISTER



82129

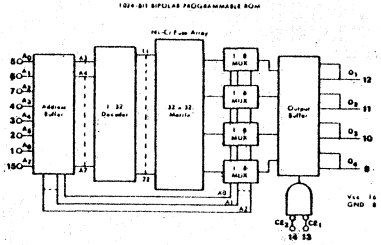


Figure D-9c FPP IC Guide