


DA 4000

CYBER HARDWARE FOR ANALYSTS

Section 2 of 2

**CONTROL DATA
EDUCATION COMPANY**
 a service of
CONTROL DATA CORPORATION

DETAILED PAK DIAGRAM (CPU 3.22)

SHIFT SEQUENCE

The shift sequence controls the operations necessary to perform the following instructions:

20ijk	Left Shift (Xi) by jk
21ijk	Right Shift (Xi) by jk
22ijk	Left Shift (Xk) Nominally (Bj) Places to Xi
23ijk	Right Shift (Xk) Nominally (Bj) Places to Xi
24ijk	Normalize (Xk) to Xi and Bj
25ijk	Round Normalize (Xk) to Xi and Bj
26ijk	Unpack (Xk) to Xi and Bj
27ijk	Pack (Xk) and (Bj) to Xi
43ijk	Form Mask of jk Bits to Xi

SHIFT 20, 21

The 20 instruction reads the selected Xi operand and shifts the 60-bit word left circularly by jk bit positions. The bits which are shifted off the upper end are inserted in the lowest order bit positions.

The 21 instruction reads the selected Xi operand and shifts the 60-bit word right with sign extension by jk bit positions.

NOMINAL SHIFT 22, 23

The 22 instruction reads the selected Xk operand and shifts the 60-bit word either left or right as specified by (Bj). If (Bj) is positive, the data is shifted left circularly by the number of bit positions designated by (Bj). If (Bj) is negative, the data is shifted right with sign extension by the ones complement of the number of bit positions designated by (Bj).

The 23 instruction operates in a manner similar to a 22 instruction except that if (Bj) is positive right shifts are performed, and if (Bj) is negative left shifts are performed.

When shifting right, if the shift count in F is $> 177_8$, gating of the shift network to I5 during SH264 is blocked. A result of zeros is sent to the Xi register.

NORMALIZE 24, 25

The normalize instruction reads the selected Xk operand and performs a normalize operation on this word, delivering the normalized result back to the Xi register and the normalize count to the Bj register.

Normalization involves left shifting the coefficient until bit 47 is different from the coefficient sign bit. The exponent is decreased by the number of bit positions shifted. The normalize count used to shift the coefficient is developed by the normalize network. The normalize count is sent to the SK register during SH164 to enable the desired shift; it is also sent to the F register for subsequent writing into Bj during common time.

At the beginning of the normalize instruction, the Xj exponent is checked for indefinite or infinite operands. An indefinite or infinite operand causes the Xk operand to be returned to Xj unchanged, and gates zeros to Bj.

The normalize instruction also checks for exponent underflow after the normalize count is subtracted. If underflow is detected, the C register is cleared to zeros before initiating common time. The resulting operand sent to the Xi register will contain a zero exponent and coefficient.

The 25 instruction operates in a manner similar to the 24 instruction, except that bit 107 is set in the C register before sending C to the shift network. This round bit has the effect of increasing the magnitude of the coefficient by one half the value of the least significant bit, after the shift is performed.

In addition to checking for underflow, the 24 instruction checks for a coefficient equal to zero. The end case result, when coefficient equal to zero is detected, is the same as underflow. (See table 5-2-16.)

8

7

6

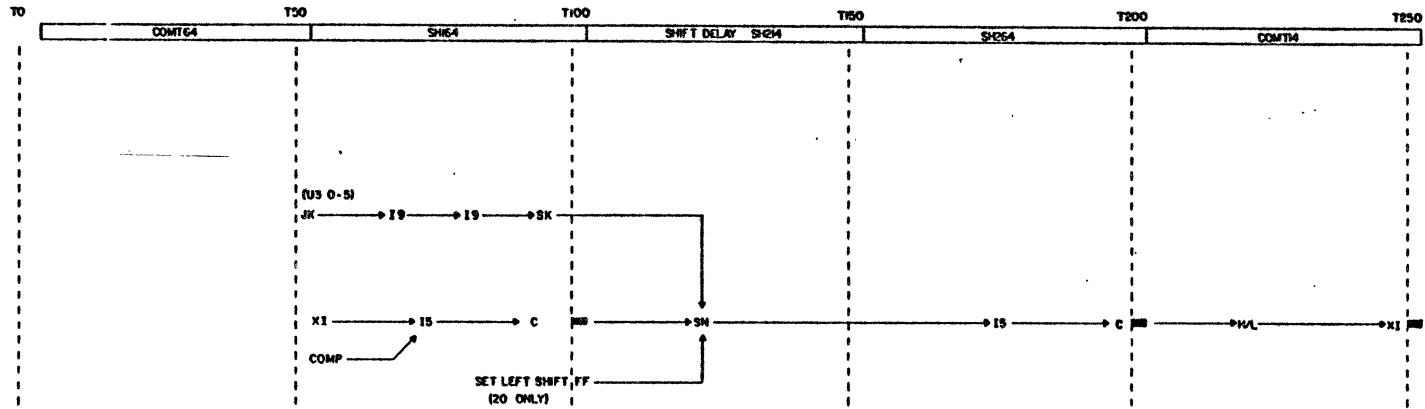
5

4

3

2

1



TERM NAME	COMMAND OR FUNCTION	DPD NO
SH150		22
SHSLHI	SELECT XI RGR	22
F4GK	119	10
SH16A		22
SL190-SL191	119	9
HSH16A		22
SKS1-SKS2	19	9
SH16A		22
S15185		22
S15285	15-107	22
S1516A		22
S15157	(155A-155Z-155I)	22
S15267		22
SH16A	[FURN]	
BL15C	COMP 15	21
SH16A		22
SHENBC	ENABLE C RGR	22
SH16A	[20]	
RS	SET LEFT SHIFT FF	22

TERM NAME	COMMAND OR FUNCTION	DPD NO	TERM NAME	COMMAND OR FUNCTION	DPD NO
SH26A		22	CBXWD		15
S15085		22	SELXI	SELECT XI RGR	2
S15285	15-107-15-107	22	COXWX	ENABLE WRITE STROBE X RGR	2
S1500A		22			
S15057		22	CBXWD		15
S15267	(155A-155Z-155I)	22	HLSL	SELECT HIGHER	10
SH26A		22			
SHENBC	ENABLE C RGR	22			
SH26A		22			
SHNEXT	ENABLE COMP TIME, F	22			



INSTRUCTION FLOW
SEQUENCE: SHFT
INSTRUCTIONS: 20, 21

DATE: 34570
REV: 0
PAGE NO: 19981000
REV: A
FIGURE 3-2-15
PAGE NO: 3-2-46 3

8

7

6

5

4

3

2

1

TABLE 5-2-16. OVERFLOW AND UNDERFLOW CONDITIONS

OVERFLOW		
INSTRUCTIONS	OVERFLOW CONDITION	RESULT
Normalize (24, 25)	None	---
UNDERFLOW		
INSTRUCTIONS	UNDERFLOW CONDITION	RESULT
Normalize (24 only)	Initial coefficient = ± 0	$X_i = 0000\ 0\dots 0_8$, $(B_j) = 60_8$
Normalize (24, 25)	Final Exponent $\leq -2000_8$	$X_i = 0000\ 0\dots 0_8$, (B_j) are correct (See Note below.)
Note: Underflow of Exponent During Normalization: The final (B_j) are the same as if underflow had not occurred. In particular, if the initial coefficient is zero, (B_j) are equal to 60_8 .		

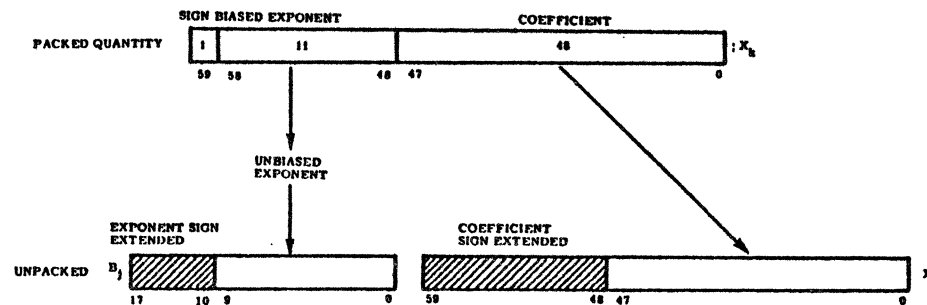
Error Exit Conditions

If X_k contains an infinite quantity ($3777\ x\dots x_8$ or $4000\ x\dots x_8$) or an indefinite quantity ($1777\ x\dots x_8$ or $6000\ x\dots x_8$), an optional exit mode selection is provided. The CPU response is dependent on whether the appropriate exit mode selection was made and the monitor flag /MEJ/CEJ condition.

An exit condition sensed (ECONDS) sets the ERROR EXIT FF (3.17) at the same time as the next RNI sequence is initiated. Error exit clears the U3 instruction register, thus forcing a return jump error exit sequence.

UNPACK, PACK 26, 27

The 26 instruction reads the selected X_k operand, unpacks this word from the floating point format, and delivers the coefficient to the X_i register and the exponent to the B_j register.



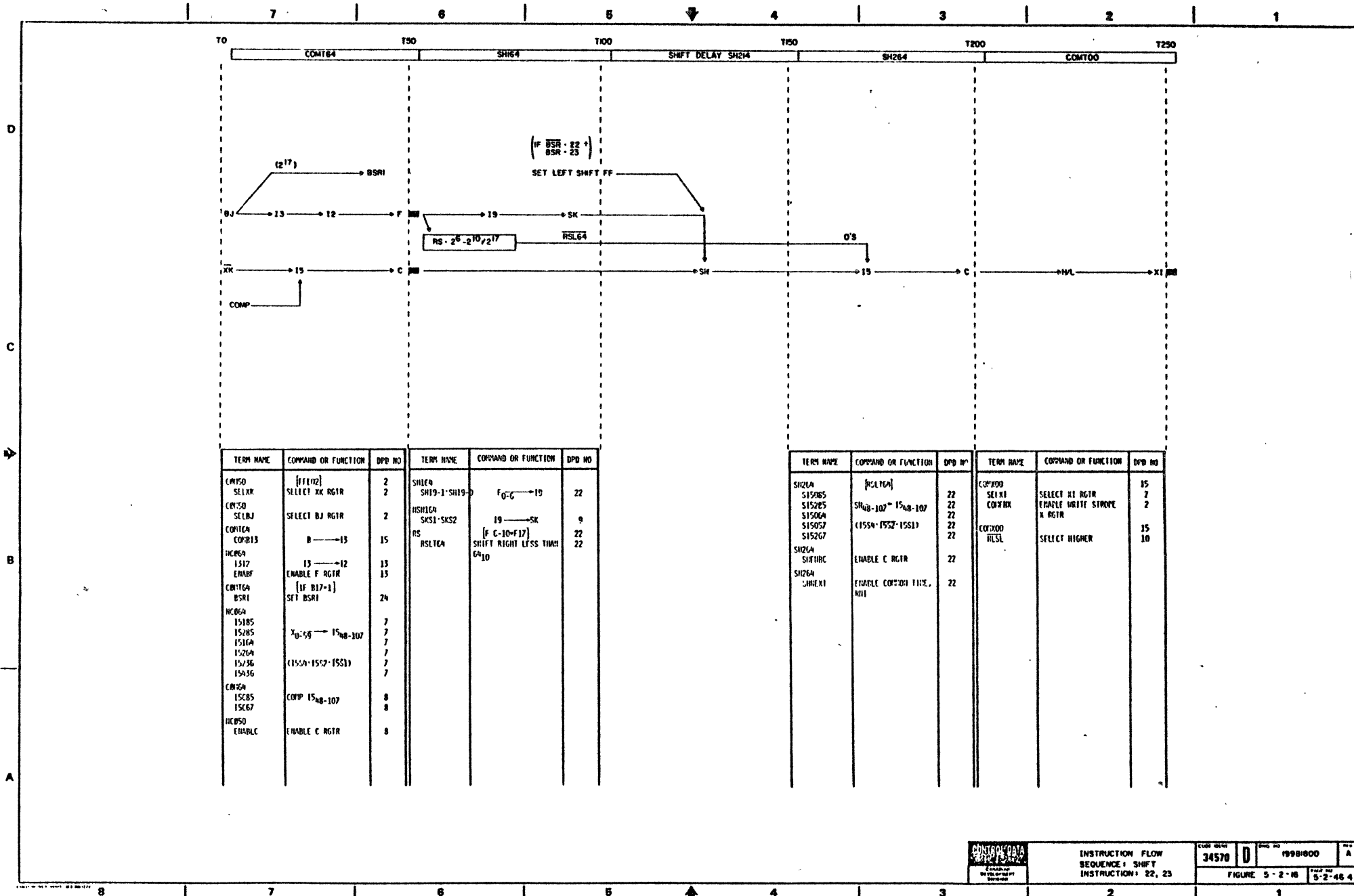
The 60-bit word delivered to the X_i register during common time (COMT00) consists of the lowest 48 bits unaltered from X_k , plus 12 bits equal to the sign bit.

The 18-bit quantity delivered to the B_j register during common time (COMT00) consists of the X_k exponent unbiased and sign extended. Unbiasing the exponent and sign extension is performed through I3 during SH114.

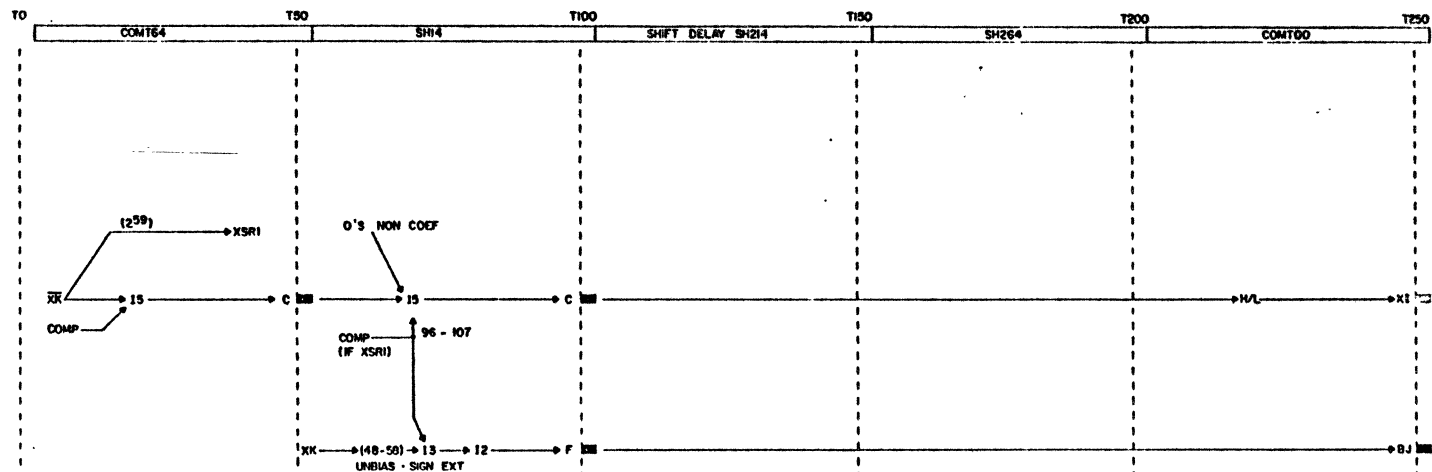
The 27 instruction performs the reciprocal process of the 26 instruction. The unpacked quantities in X_k and B_j are packed in floating point format and delivered to the X_i register.

MASK 43

The 43 instruction generates a masking word using the 6-bit j_k quantity to designate the width of the masking field. The quantity is sent to the SK register. The C register is cleared to zero and sent to the shift network. During the shift period, C is right shifted by the j_k quantity in SK . One-bits are forced to the shift network sign extension scheme, thus replacing each shifted zero bit with a one-bit. The completed masking word sent to the X_i register consists of one-bits in the highest order j_k bit positions, and zero bits in the remainder of the word.



INSTRUCTION FLOW
SEQUENCE: SHIFT
INSTRUCTION: 22, 23



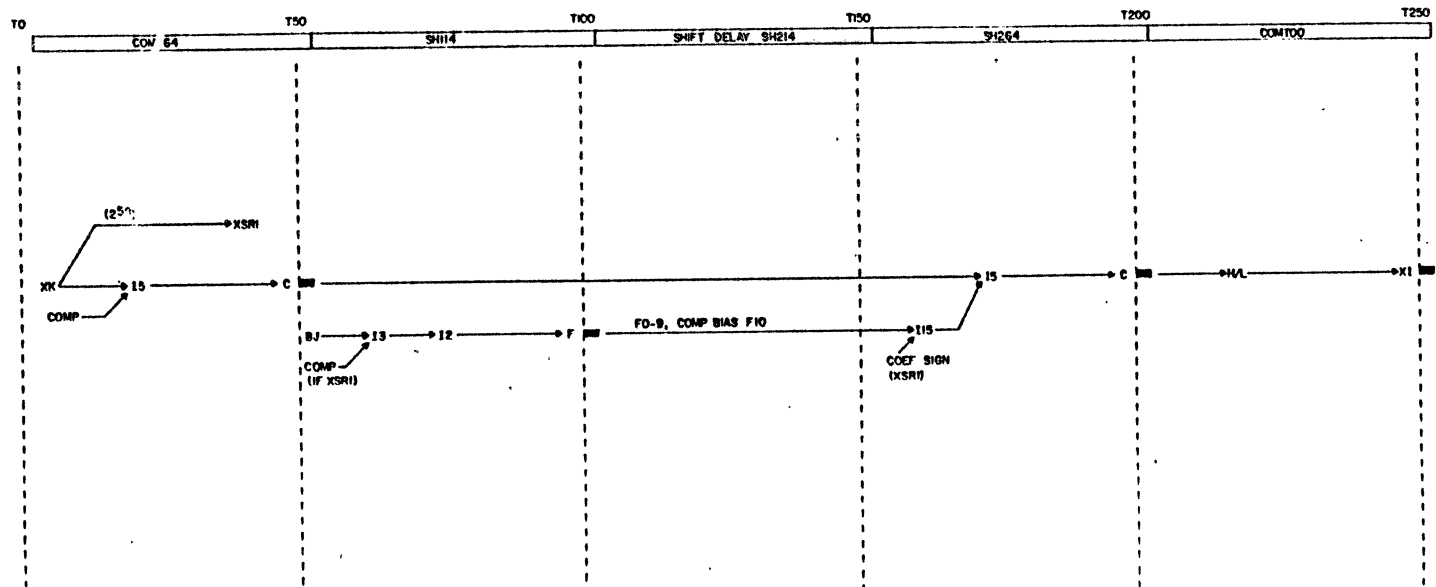
TERM NAME	COMMAND OR FUNCTION	DPD NO	TERM NAME	COMMAND OR FUNCTION	DPD NO
CO1T64	FFC02	2	SH14	O'S NON COEF (15 0-47, 96-107)	8
SELXK	SELECT XK RGTR	2	SH11A	[XSR1]	
NC064	X0-59 → 154B-107	7	SH15C	COMP 15 96-107	22
15185		7	SH100		
15285		7	SHSLXK	SELECT XK RGTR	22
15164		7	SH11A	SHX13	22
15264	(1550A 1552 1551)	7	SHX13	UNBIAS - SIGN EXT	12
15236		7	SHX13	→ 13 0-17	
15436		7	SH11A		
CO1E4			SH1312	13 → 12	22
15C85	COMP 154B-107	8	ENAGF	ENABLE F RGTR	13
15C85		8	SH11A	[XSR1]	
15C67		8	SH13C	COMP 13	22
CO1T64	IF X59-1		SH11A		
XSR1	SET XSR1	24	S15185	C48-95 → 154B-95 (1550A 1552 1551)	22
NC059					
ENABLEC	ENABLE C RGTR	8			

TERM NAME	COMMAND OR FUNCTION	DPD NO	TERM NAME	COMMAND OR FUNCTION	DPD NO
SH264	SHX2T	22	CO1T00		15
	ENABLE COMPON TIME, PH		SELX1	SELECT X1 RGTR	2
			SELBJ	SELECT B J RGTR	2
			CO1E1X	ENABLE WRITE	2
			CO1E1B	STRIDE X, B RGTR'S	2
			CO1T00		15
			HL5L	SELECT HIGHER	10



INSTRUCTION FLOW
SEQUENCE 1 SHFT
INSTRUCTION 26

8 | 7 | 6 | 5 | 4 | 3 | 2 | 1



TERM NAME	COMMAND OR FUNCTION	DPD NO	TERM NAME	COMMAND OR FUNCTION	DPD NO
CR150	[11E07]	2	SH1100	SELECT BJ RGR	22
SELXK	SELECT XK RGR	2	SELBJ		2
HC064			SH114	B → 13	22
15185		7	SH113		
15285	Y0-59 → 1548-107	7	SH114	13 → 12	22
15164		7	SH1312		22
15264		7	ENAF	ENABLE F RGR	13
15236	(1554-1552-1551)	7	SH114	[XSR1]	
15436		7	SH113C	COMP 13	23
COM64					
15C85	COMP 15 48-107	8			
15C67		8			
CO1164	[IF XSR1-1]				
XSR1	SEL XSR1	24			
HC050					
ENABLEC	ENABLE C RGR	8			

TERM NAME	COMMAND OR FUNCTION	DPD NO	TERM NAME	COMMAND OR FUNCTION	DPD NO
SH264			COM100		
S15185	C48-95 → 1548-95 (1554-1552-1551)	22	SELX1	SELECT X1 RGR	15
			COM100	ENABLE WRITE STROBE X RGR	2
SH264	F0-9 → 1548-59 F10- XSR1	8	COM100		
FEX115			HLSC	SELECT FIGHTP	15
SH264					10
S15267	11548-59 → 1548-107 (1554-1552-1551)	22			
SH264	SHENDC	22			
SH264	SH264				
SH264	SH2EXT	22			
	ENABLE COMMAND TIME, PNT				

DETAILED PAK DIAGRAM (CPU 3.23)

BOOLEAN SEQUENCE

The Boolean sequence controls the operations necessary to perform the following instructions:

Transmit

10ijx	Transmit (Xj) to Xi
14ixk	Transmit the Complement of Xk to Xi

Logical

11ijk	Logical Product of (Xj) and (Xk) to Xi
12ijk	Logical Sum of (Xj) and (Xk) to Xi
13ijk	Logical Difference of (Xj) and (Xk) to Xi
15ijk	Logical Product of (Xj) and Complement (Xk) to Xi
16ijk	Logical Sum of (Xj) and Complement (Xk) to Xi
17ijk	Logical Difference of (Xj) and Complement (Xk) to Xi

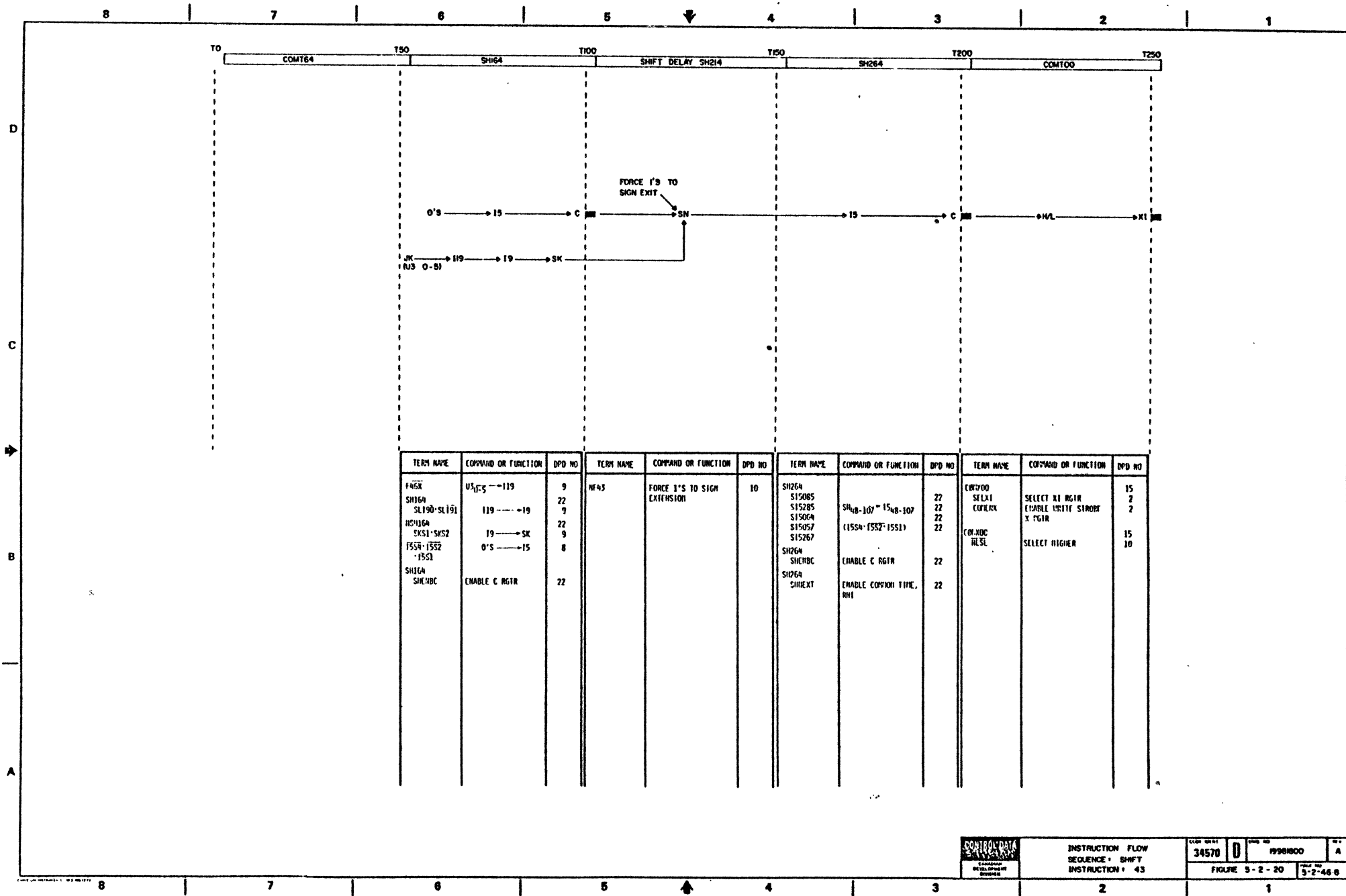
The 10 instruction transfers a 60-bit word from register Xj to register Xi.

The 14 instruction extracts the 60-bit word from operand register Xk, complements it, and transmits the complemented quantity to operand register Xi.

The 11-13 instructions perform the logical product (AND function), logical sum (inclusive OR function), and logical difference (exclusive OR function) of 60-bit words from operand registers Xj and Xk, and place the result in operand register Xi.

The 15-17 instructions perform the logical product (AND function), logical sum (inclusive OR function), and logical difference (exclusive OR function) of the 60-bit quantity from operand register Xj and the complement of the 60-bit word from operand register Xk, and place the result in operand register Xi.

The arithmetic operations for instructions 11-17 are performed by the D adder. The Boolean sequence controls the logical operation codes sent to the D adder which, in turn, directs the D adder ALU to perform the required logical operation.



TERM NAME	COMMAND OR FUNCTION	DPD NO	TERM NAME	COMMAND OR FUNCTION	DPD NO	TERM NAME	COMMAND OR FUNCTION	DPD NO	TERM NAME	COMMAND OR FUNCTION	DPD NO
FGX	U ₁ ES → 119	9	HF45	FORCE I'S TO SIGN EXTENSION	10	SH264		22	COM700	SELECT RI RGTR	15
SH164		22				S15085		22	SELX1	ENABLE 15STTY STROB	2
SL190-SL191	119 → 19	9				S15285	SH ₄₈₋₁₀₇ → 15 ₄₈₋₁₀₇	22	COM4RX	X FGTR	2
HS1164		22				S15004	(155A-155Z-1551)	22	COMJOC		15
SKS1-SKS2	19 → SK	9				S15057		22	HLSL	SELECT HIGHER	10
155A-155Z	0'S → 15	8				S15267		22			
SH164		22				SH264	ENABLE C RGTR	22			
SH164	ENABLE C RGTR	22				SH164	ENABLE COMMON TIME, RH	22			

CONTROL DATA
CORPORATION
MEMPHIS, TENNESSEE

INSTRUCTION FLOW
SEQUENCE - SHIFT
INSTRUCTION - 43

LOG NO. 34570
REV. NO. 0
DRAWING NO. 19981800
FIGURE 5-2-20
REV. NO. 3-2-66 B

8

7

6

5

4

3

2

1

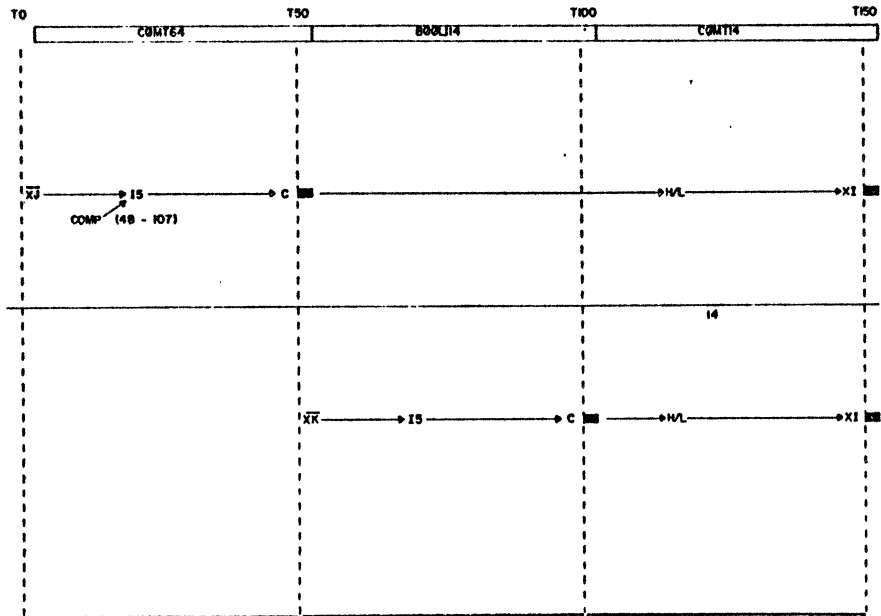
D

C

B

B

A



TERM NAME	COMMAND OR FUNCTION	DPD NO	TERM NAME	COMMAND OR FUNCTION	DPD NO	TERM NAME	COMMAND OR FUNCTION	DPD NO
CR:50		15	HB100		23	CR:50	SELECT XJ RGTR	15
SELXJ	SELECT XJ RGTR	13	SXK		23	SELXI	SELECT XI RGTR	2
CR:6A		15	SELXK	SELECT XK RGTR	5	CR:6X	ENABLE WRITE STROBE X RGTR	2
15C85	COMP 15 48-95	8	HB114		23	HESL	SELECT HIGHER	10
15C67	COMP 15 96-107	8	15185		7			
CR:6A		15	15285	X ₀₋₅₉ → 15 ₄₈₋₁₀₇	7			
15185		7	15164		7			
15285	X ₀₋₅₉ → 15 ₄₈₋₁₀₇	7	15264		7			
15164		7	15236	(1554-1552-1551)	7			
15264		7	15436		7			
15236	(1554-1552-1551)	7	DBL-1114	[14]	23			
15436		7	BL15C	BLOCK COMP 15				
CR:50		15	DFOL-1114		23			
EHABLC	ENABLE C RGTR	8	DFOLC	[14]	23			
			EHABC	ENABLE C RGTR	8			
			DFOL-1114	[10 + 14]	23			
			DFX11	ENABLE RHL. COMP: TIME	23			



INSTRUCTION FLOW
SEQUENCE: BQCLEAN
INSTRUCTION: IQ, 14

34570 **D** 1998400
FIGURE 5-2-21 5-2-48

8

7

6

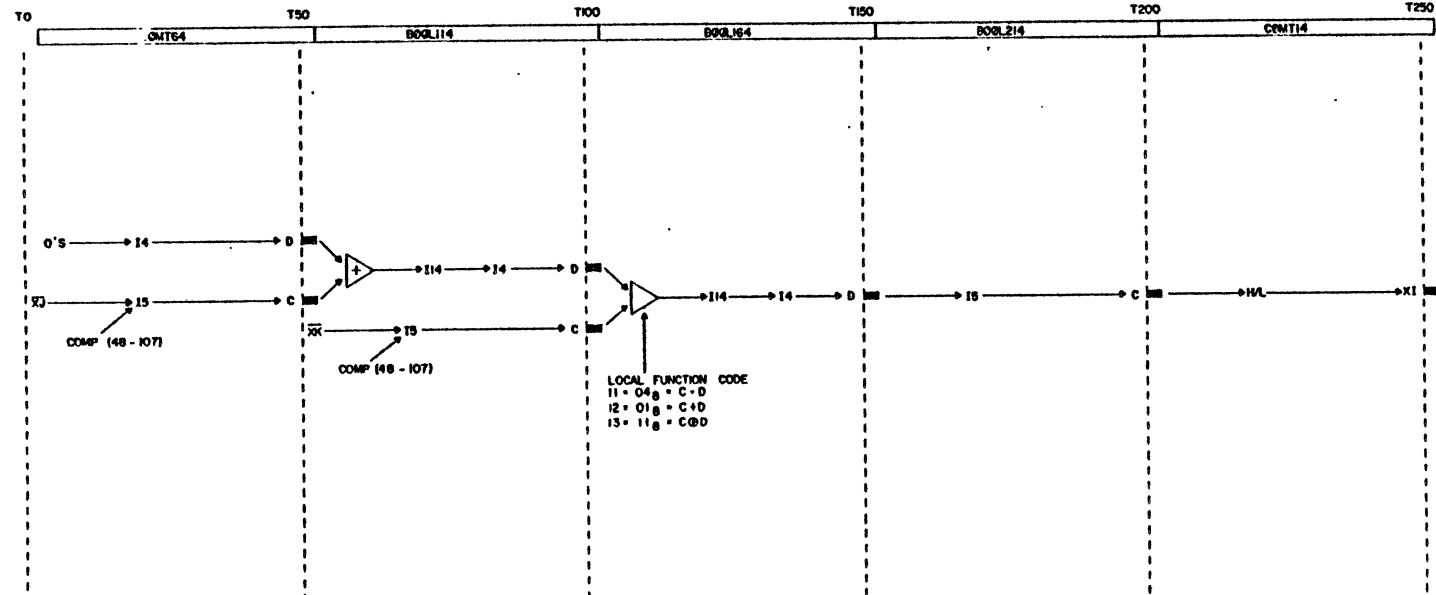
5

4

3

2

1



TERM NAME	COMMAND OR FUNCTION	DPD NO	TERM NAME	COMMAND OR FUNCTION	DPD NO	TERM NAME	COMMAND OR FUNCTION	DPD NO	TERM NAME	COMMAND OR FUNCTION	DPD NO	TERM NAME	COMMAND OR FUNCTION	DPD NO
CM150		15	DPOL-1114		23	DEAL-1164		23	HE214		23	CO1X00		15
SELXJ	SELECT XJ RGR	13	BOOLND	ENABLE D RGR	5	BOEND	ENABLE D RGR	5	15085	04g-107 → 1548-107	7	SELXI	SELECT XI RGR	2
CM164		14	ENABLD	ENABLE D RGR	5	ENABLD	ENABLE D RGR	5	15064	(1554-1552-1551)	7	DOENX	ENABLE WRITE STROBE X RGR	7
15C85	COMP 15 48-95	8	ND100		23				15136	(1554-1552-1551)	7			
15C67	COMP 15 96-107	8	SXK		2				DPOL-1214		23	HLSL	SELECT HIGHEN	10
NC064		15	SELXK	SELECT XK RGR	2				BOENC	ENABLE C RGR	8			
15185		7	ND114		23				ENABC	ENABLE C RGR	8			
15285	0's → 1548-107	7	15185		7				DPOL-1214		23			
15164		7	15285	0's → 1548-107	7				BRK1T	FRAPLE RM1, COMMON TIME	23			
15204	(1554-1552-1551)	7	15164		7									
15236		7	15236	(1554-1552-1551)	7									
15436		7	15436		7									
CM150		15												
ENABC	ENABLE C RGR	8	DPOL-1114	11 + 12 + 13	23									
NC064		15	BLT5E		23									
140-141	0's → 14	5	15C85	COMP 15 48-95	7									
CM150		15	15C67	COMP 15 96-107	7									
ENABLD	ENABLE D RGR	5	DPOL-1114		23									
			BOOLNC		23									
			ENABC	ENABLE C RGR	8									



INSTRUCTION FLOW
 SEQUENCE - BOOLEAN
 INSTRUCTION - 11, 12, 13

DETAILED PAK DIAGRAM (CPU 3, 24, 3, 25, 3, 26)
 FLOATING POINT ADD SEQUENCE (FAD)

The FAD sequence controls the operations necessary to perform the sum or difference of two floating point quantities in Xj and Xk. The packed result is returned to the Xi register.

The floating point instructions controlled by the FAD sequence are as follows:

30ijk	Floating Sum of (Xj) and (Xk) to Xi
31ijk	Floating Difference of (Xj) and (Xk) to Xi
32ijk	Floating Double Precision Sum of (Xj) and (Xk) to Xi
33ijk	Floating Double Precision Difference of (Xj) and (Xk) to Xi
34ijk	Round Floating Sum of (Xj) and (Xk) to Xi
35ijk	Round Floating Difference of (Xj) and (Xk) to Xi

The FAD sequence is initiated by GOFAD from the common time sequence. The operands are obtained from the selected Xj and Xk registers. The exponents are extracted and tested for infinite ($3777_8 + 4000_8$) or indefinite ($1777_8 + 6000_8$) operands. An infinite or indefinite operand causes the FAD sequence to abort and enables the end case exit sequence.

The floating sum or difference operation involves the addition of two floating point coefficients that have equal exponents. Exponent equalization is accomplished by right shifting the coefficient of the smaller exponent a number of places equal to the absolute difference of the two exponents. A right shift decreases the size of the coefficient (moves the binary point left) and the exponent is therefore made larger. Once the exponents are equalized, the sum or difference of the coefficients is computed in the D adder. At the conclusion of the add operation, the binary point is considered to be located between bit positions 47 and 48 of the 108-bit D register.

Single precision instructions (30, 31, 34, 35) use the coefficient result contained in bit positions 48-95 of the D register, and pack the computed exponent. Double precision instructions (32,33) use the lower 48 bits of the D register and subtract 60_8 from the computed exponent before packing. This shifts the binary point to the right of bit 0 which is necessary to express the result as an integer.

Coefficient overflow is checked during FAD364 by examining D register bits 96 and 97. If D register bits $96 \neq 97$, coefficient overflow has occurred. The coefficient is right shifted by one, and the exponent is increased by one.

Exponent underflow is checked during FAD414. Underflow is detected when the exponent is less than -1777_8 after correction during FAD364. Exponent underflow causes the FAD sequence to abort normal exit and enables the end case exit sequence.

The final coefficient and exponent plus bias are packed in 15 during FAD414. D register bit 107 controls complementing the exponent if the resulting coefficient sign is negative. FAD414 enables the common time sequence (COMT00) and the RNI sequence. Common time allows the contents of C to be stored in Xi.

ROUND OPERATION (34, 35 INSTRUCTIONS)

The 34 and 35 instructions operate in the same manner as described, except that the coefficients are rounded before the addition process to produce a rounded sum or difference.

The round bit is attached at the right end of both coefficients (bit 47) during FAD114 and FAD164. During FAD214, the round bit is removed from the coefficient with the smaller exponent when the following conditions are present:

1. $34 . \overline{BON} . XSR1 = XSR2$; or
2. $35 . \overline{BON} . XSR1 \neq XSR2$

The round bit increases the absolute value of the coefficient by one half the value of the least significant bit.

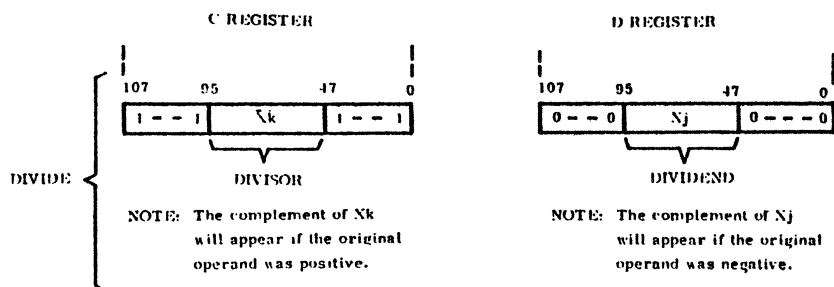
FLOATING POINT MULTIPLY/DIVIDE SEQUENCE (FMD)

The FMD sequence controls the operations necessary to perform multiplication or division of floating point quantities in Xj and Xk. Multiply instructions 40, 41, 42, form the product of multiplier Xj times multiplicand Xk and send the result to Xi. Divide instructions 44 and 45, form the quotient of the dividend Xj divided by the divisor Xk and send the result to Xi.

The FMD sequence also controls the operations necessary to count the number of one-bits in Xk (population count instruction 47) and store the result in Xi.

DIVIDE STEPS

Division is accomplished by repetitive subtractions in the D adder. The D register contains the coefficient of the dividend X_j and the C register contains the complemented coefficient of the divisor X_k . The C and D registers initially appear at the input to the D adder as follows:



Before the first divide iteration, the X_j coefficient (dividend) in the C register is transferred via I14 to I4 where a right shift of one occurs. This reduces the dividend by one half. The dividend now in D is subtracted from the divisor (X_k coefficient) in C. If an end-around-carry occurs as a result of the subtraction, a divide fault is detected, since the coefficient of the dividend must be less than twice that of the divisor. A divide fault aborts the FMD sequence and enables end case exit.

The divide iterations are performed during FMD264 through FMD2664. The SK counter contains the 60_8 iteration count. Each 50 ns clock pulse decrements the counter by one until all iterations have been performed.

Each iteration checks for an end around carry condition from the D adder after the divisor in C has been subtracted from the dividend in D. If end around carry does not occur, the dividend in D is left shifted one place through I4 and returned to D before the next iteration. If end around carry does occur, a quantity one is gated to I14 bit position 0 and the D adder output is left shifted one place through I4 and sent to D. In this way the D register receives an additional quotient bit for each iterative step as the dividend is left shifted through the register. This process continues until the quantity in SK is reduced to zero. After the last iteration, the D register will contain the complete quotient in the lower 48 bits and the remainder in bit positions 48 through 95.

At this time the binary point is considered to be between bit positions 46 and 47 and must be shifted to the right of bit 0 to represent the quotient as an integer. This is accomplished by subtracting 57_8 from the X_j exponent during FMD114.

ROUND OPERATION

Rounding is accomplished by adding a quantity of $1/3$ during the division process. Round bits are added during the divide steps of a 45 instruction each time the SK register contains an even count, except during the first iteration divide. This forces a 1-bit into the D register bit 48 so that successive iterations bring in the $1/3$ round quantity of $25 \dots \dots \dots 25_8$.

EXPONENT AND RESULT FORMATION, DIVIDE

The final exponent for the quotient is formed by subtracting the exponent of the divisor X_k from the exponent of the dividend X_j in the F adder during FMD2764. The exponent of the dividend X_j will already have had a constant of 57_8 subtracted from the exponent value. The result exponent formed in the F adder will thus represent the coefficient as an integer.

During FM2764, the quotient is checked for normalization (D register bit 47 \neq 0). If it is necessary to normalize the quotient, the quantity 1 is subtracted from the result exponent in F while the D register is shifted left by one through I4. If the remainder in D between bit positions 48-95 is \geq the divisor in C, an end around carry from the D adder sets bit 0 of the quotient through I4. A normalized result is thus formed and returned to D.

Exponent overflow or underflow is checked during FM2814 by determining that the absolute value of the exponent is greater than 1777_8 . Exponent overflow or underflow causes the FMD sequence to abort normal exit, and enable the end case exit sequence.

The final quotient and exponent plus bias are packed in I5 during FM2814. I5 is complemented if $XSR1 \neq XSR2$. FM2814 enables the common time sequence (COMT00) and the RNI sequence. Common time allows the quotient from C, plus exponent and signs, to be stored in XI.

END CASE SEQUENCE

The end case sequence checks the formation of infinite, indefinite and zero results when executing floating point instructions controlled by the FAD and FMD sequences.

The floating multiply and divide instructions controlled by the FMD sequence are as follows:

40ijk	Floating Product of (Xj) and (Xk) to Xi
41ijk	Round Floating Product of (Xj) and (Xk) to Xi
42ijk	Floating Double Precision Product of (Xj) and (Xk) to Xi
44ijk	Floating Divide (Xj) by (Xk) to Xi
45ijk	Round Floating Divide (Xj) by (Xk) to Xi

PREPARATION OF OPERANDS

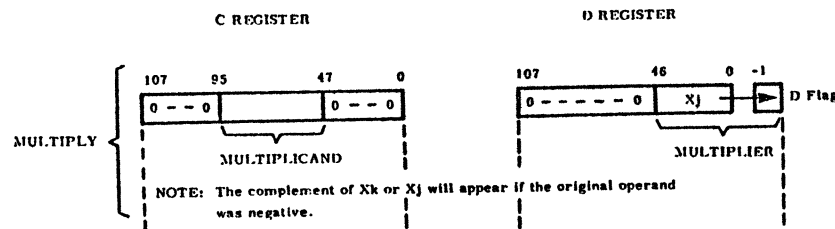
The operands are obtained from the selected Xj and Xk registers. The exponents are extracted and tested for infinite ($3777_8 + 4000_8$), indefinite ($1777_8 + 6000_8$), or zero ($0000_8 + 7777_8$). An infinite, indefinite or zero operand causes the FMD sequence to abort and enables the end case exit sequence. Zero exponents in both Xj and Xk enable integer multiply. Integer multiply blocks end case exit.

The bias for each exponent is removed in I3 and sign extended. The Xj exponent is transferred from I3 through I2 to F. The Xk exponent is transferred from I3 to E. With both exponents at the input to the small adder, a subsequent add during FMD2714 produces the final result exponent before any correction is made.

MULTIPLY STEPS

During common time, the shift and iteration counter is preset with 60_8 to allow the Xj coefficient to be shifted right 48 bits to align with bit 0 of the C register. Since all numbers are considered integers rather than fractions, the binary point is considered as being to the right of bit 0. Right shifting the Xj coefficient (multiplier) 48 bits places it in the proper position for the multiplication process.

The C and D registers initially appear at the input of the D adder as follows:



Just before the multiply iterations, the Xj coefficient is transferred via I14 to I4 where a right shift of one occurs. The shifted bit is sent to the D flag register. The multiply iterations are performed during FMD264 through FMD2664. The SK counter contains the 60_8 iteration count. Each 50 ns clock pulse decrements the counter by one until all iterations have been performed.

The D flag monitors the condition of the lowest order bit of D. Before the first iteration, the multiplier was right shifted one into the D flag. The D flag now determines the first operation. If the D flag is set, the output of the D adder is right shifted one and sent back to the D register. If the D flag is clear, the output of the D register is right shifted one and sent back to the D register. After the first iteration, the D register holds the partial product and the remaining bits of the multiplier. This process continues until the quantity in SK is reduced to zero. After the last iteration, the D register contains the final product with the multiplier shifted end off out of the register.

On the last iteration, bit 46 of C is set while the rest of C is cleared to zeros. C is added to the product in D during FM2714 to form a rounded result. The rounded product is sent to the D register on a 41 instruction only.

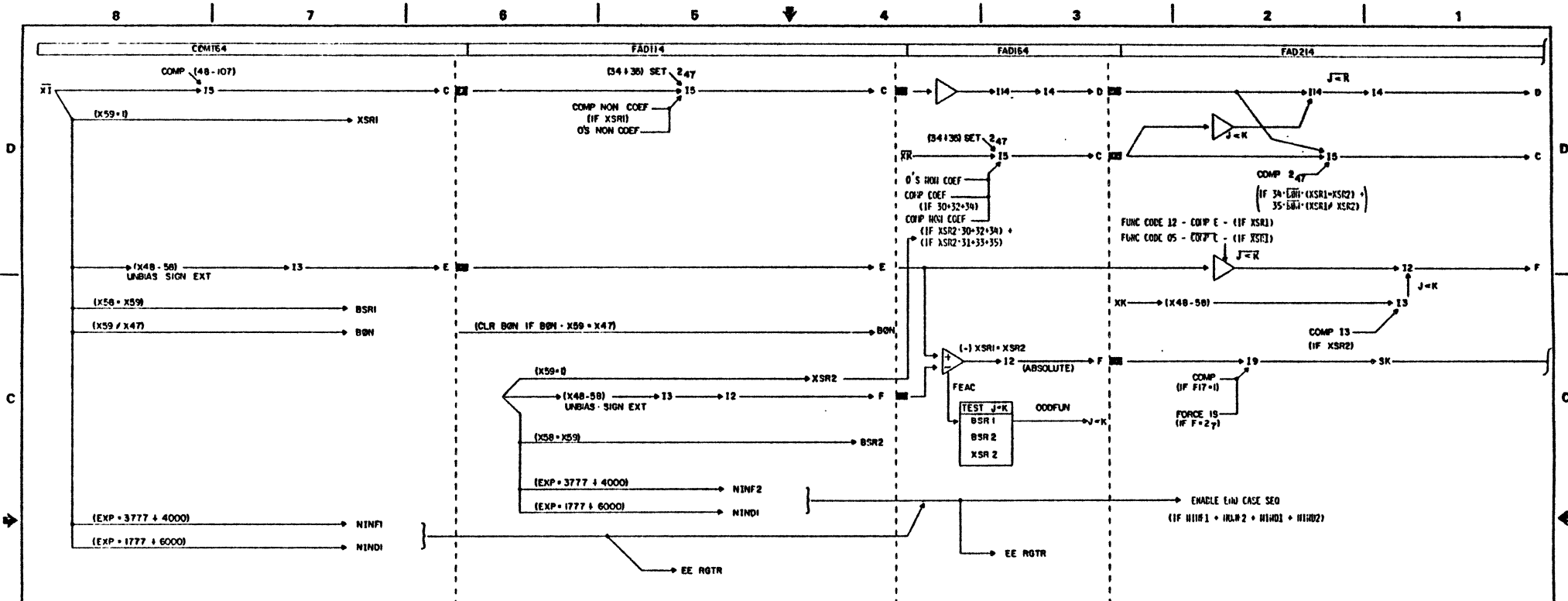
EXPONENT AND RESULT FORMATION, MULTIPLY

The final exponent for the 96-bit product is formed in the F adder during FM2714. For single precision instructions 40, 41, the exponent would already have been adjusted by 60_8 . Adjustment of the exponent for single precision instructions is performed during FMD164, where 60_8 is added to the Xj exponent in F. The exponent is therefore made relative to the upper 48 bits of the product, or zero is added to maintain an exponent relative to the 96-bit product.

If it is necessary to normalize the product during FM2764, the quantity 1 is subtracted from the result exponent in F, while the product in D is left shifted by one through I4 and returned to D.

Exponent overflow or underflow is checked during FM2814 by determining that the absolute value of the exponent is greater than 1777_8 . Exponent overflow or underflow causes the FMD sequence to abort normal exit, and enable the end case exit sequence.

The final product and exponent plus bias are packed in I5 during FM2814. I5 is complemented if $XSR1 \neq XSR2$. FM2814 enables the common time sequence (COMT00) and the RNI sequence. Common time allows the upper or lower product from C, plus exponent and signs, to be stored in Xi.



TERM NAME	COMMAND OR FUNCTION	DPD NO
CR-50	SELECT XJ RGR	15
SELXJ		2
CR-5A		15
FSC85	COMP 15 48-95	8
FSC67	COMP 15 96-107	8
CR-50		15
ENABLE	ENABLE C RGR	8
CR-16A		15
XSR1	SET XSR1 (IF X59=1)	24
CR-17A		15
CEAF13	UNBIAS SIGN EXT 13 0-17	12
SEXP13		12
CR-50		15
ENABLE	ENABLE E RGR	13
CR-5A		15
BSR1	SET BSR1 (IF X58 = X59)	24
CR-5A		24
NINF1	SET NINF1 (IF EXP=3777+4000)	24
NIND1	SET NIND1 (IF EXP=1777+6000)	24
CR-5A		24
BSR	SET BSR FF (IF X55=X47)	24

TERM NAME	COMMAND OR FUNCTION	DPD NO
ISS1-ISS2	0'S NON COEF (15 0-47, 96-107)	8
ISSA		
FD114-XSR1	COMP 15 0-46	25
FISC06	COMP 15 47	25
FISC67	COMP 15 96-107	25
FD114	34+35 15-15 47 (ISS4-ISS2-ISS1)	25
F15057		
F15157		
FAD100		24
FAEXK	SELECT XK RGR	24
FD114	X48-58 (EXP)	24
FEXP13	UNBIAS SIGN EXT 13 0-17	12
SEXP13		12
FD114		24
EDXSR2		26
BSR2	SET BSR2 (IF X58 = X59)	24
NINF2	SET NINF2 (IF EXP=3777+4000)	24
NIND2	SET NIND2 (IF EXP=1777+6000)	24
XSR2	SET XSR2 (IF X59=1)	24
FD114		24
F1312	13 → 12	24
ENAF	ENABLE F RGR	13
FD114		24
FENEC	ENABLE C RGR	25

TERM NAME	COMMAND OR FUNCTION	DPD NO
FAD114	CLR BSR FF (IF BSR=X59=X47)	24
BSR		
FD114		24
FENEE	ENABLE EE RGR	26
FD114		24
F15195	X48-95 → 15 48-95 (ISS4-ISS2-ISS1)	25

TERM NAME	COMMAND OR FUNCTION	DPD NO
FAD150	SELECT XK RGR	24
FASKK		24
FD114	D-ADD → 114	26
FD16A		
FISC06	(XSR2-30+32+34) + (XSR2-31+33+35) COMP 15 0-47,96-107	25
FISC47		25
FISC67		25
FD16A		
FISC85	30+32+34 COMP 15 48-95	25
ISS1-ISS2	0'S NON COEF (15 0-47, 96-107)	8
ISSA		
FD16A		
F15057	(34+35)	25
F15157	(ISS4-ISS2-ISS1)	25
FD16A		
FENBC	ENABLE C RGR	25
FD16A		
FENBD	ENABLE D RGR	26
FD15A		
FENBF	ENABLE F RGR	26
FD16A		
F15185	X48-95 → 15 48-95 (ISS4-ISS2-ISS1)	25
F15285		25
FD16A		
FENBE	ENABLE EE RGR	26

TERM NAME	COMMAND OR FUNCTION	DPD NO
FD214-JLTK	(J=R)	24
ED114	D → 114	26
FD114	C → D-ADD → 114	26
FD214-JLTK	(J=R)	25
F1500M	0-10 → 15 0-107	25
F15057		25
F15085	(ISS4-ISS2-ISS1)	25
F15067		25
FD214-JLTK	(J=R)	25
F1510N	0-10 → 15 0-107	25
F15157		25
F15185	(ISS4-ISS2-ISS1)	25
F15167		25
FD214		
FENBD	ENABLE D RGR	26
FENBC	ENABLE C RGR	25
HF4214-XSR1	E → FADD → 12 (XJ UNBIASED EXP)	12
FA2200	SELECT XK RGR	24
FASKK		24
HF4214	X48-58 (EXP)	24
FEXP13	UNBIAS SIGN EXT 13 0-17	12
SEXP13		12

TERM NAME	COMMAND OR FUNCTION	DPD NO
FD214-JLTK	(J=R)	24
ED114	D → 114	26
FD114	C → D-ADD → 114	26
FD214-JLTK	(J=R)	25
F1500M	0-10 → 15 0-107	25
F15057		25
F15085	(ISS4-ISS2-ISS1)	25
F15067		25
FD214-JLTK	(J=R)	25
F1510N	0-10 → 15 0-107	25
F15157		25
F15185	(ISS4-ISS2-ISS1)	25
F15167		25
FD214		
FENBD	ENABLE D RGR	26
FENBC	ENABLE C RGR	25
HF4214-XSR1	E → FADD → 12 (XJ UNBIASED EXP)	12
FA2200	SELECT XK RGR	24
FASKK		24
HF4214	X48-58 (EXP)	24
FEXP13	UNBIAS SIGN EXT 13 0-17	12
SEXP13		12

TERM NAME	COMMAND OR FUNCTION	DPD NO
FD214-JLTK	(J=R)	24
F1312	13 → 12 → F	26
ENAF	ENABLE F RGR	13
FD214-JLTK	(J=R)	25
FENBF	12 → F	26
FD214		
FENSK	ENABLE PRESENT SK	26
SL191-SL190	F → 19	9
FENR		
SL191-SL190	F → 19	9
TC01		
FL128-2X	1'S → 19	9
FD214		
FALHC	ENABLE END CASE SEQ	24

(Part 1 of 2)

INSTRUCTION FLOW SEQUENCE - FAD INSTRUCTION 30,32,31,33,34,35

FIGURE 5-2-24

The FAD sequence enables the end case sequence at FAD214 time when an infinite or indefinite operand is detected, or at FAD414 time when underflow is detected.

The FMD sequence enables the end case sequence at FMD214 time when an infinite, indefinite or zero operand is detected (except during multiply when both operands are zero); when a divide fault is detected; or, at FMD2814 time, when overflow or underflow is detected.

Overflow and Underflow

Exponents lying outside the range -1777_8 to $+1777_8$ cannot be generated during execution of floating point arithmetic instructions. An attempt to generate an exponent greater than $+1777_8$ yields an infinite result (overflow). An attempt to generate an exponent less than -1777_8 yields a zero result (underflow).

Indefinite

A positive indefinite (1777_8) or negative indefinite (6000_8) operand generates an indefinite indicator plus zero coefficient to the C register. A positive indefinite result indicator is generated whenever a calculation cannot be resolved. The indefinite indicator corresponds to a -0 exponent and a zero coefficient.

The common time and RNI sequences are enabled by the end case sequence after the proper 60-bit result is sent to the C register. Common time allows the end case result in C to be stored in the Xi register.

ERROR EXIT CONDITIONS

If an attempt is made to use an indefinite or infinite operand in floating arithmetic sequences, an optional exit mode selection is provided. The CPU response is dependent on whether the appropriate exit mode selection was made and the monitor flag /MEJ/CEJ condition.

An exit condition sensed (ECONDS) sets the error exit FF (CPU 3.17) at the same time as the next RNI sequence is initiated. Error exit clears the U3 instruction register, thus forcing a return jump error exit sequence.

POPULATION COUNT 47

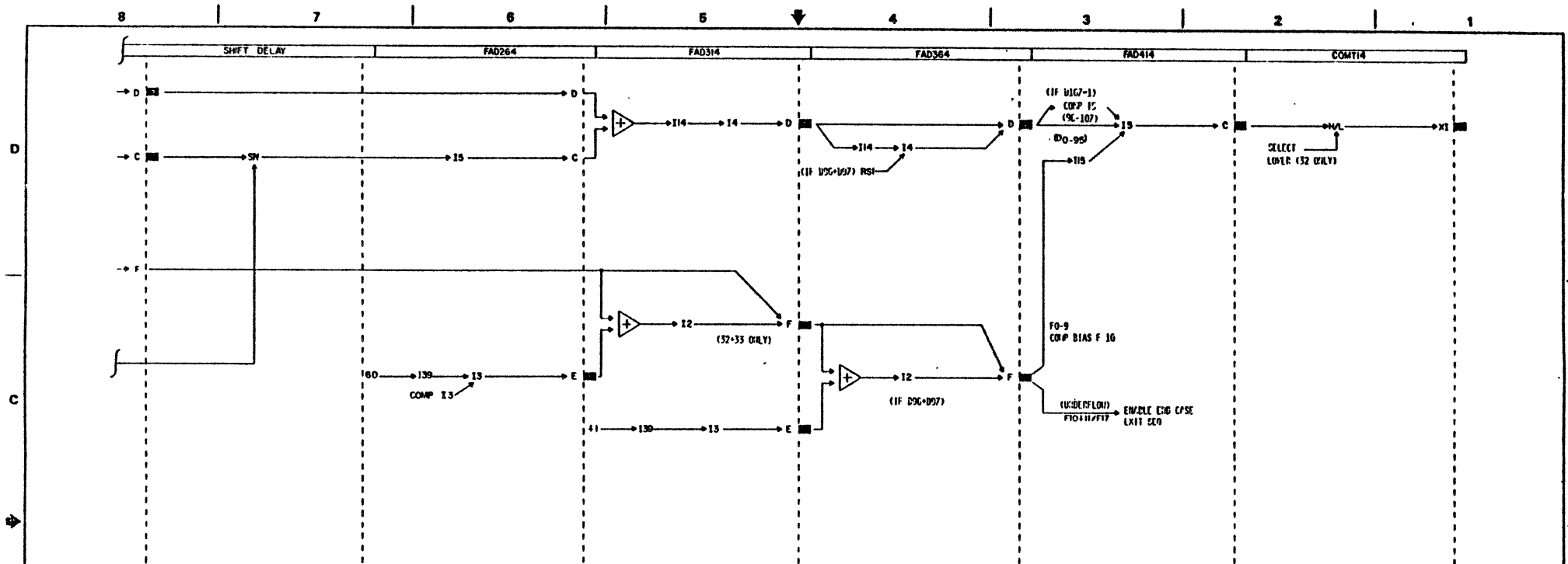
The population count instruction is controlled by the FMD sequence. The instruction counts the number of 1-bits from a selected Xk register and delivers the count value to a selected Xi register.

TABLE 5-2-17. OVERFLOW AND UNDERFLOW CONDITIONS

OVERFLOW		
INSTRUCTIONS	OVERFLOW CONDITION	RESULT
Upper Sum (30, 31, 34, 35)	None (see Note below.)	---
Lower Sum (32, 33)	None	---
Upper Product (40, 41)	$*n_1 + n_2 + 60_8 \geq 2000_8$	$X_i = 3777\ 0\dots 0_8$ or $4000\ 0\dots 0_8$ (True Sign)
Lower Product (42)	$n_1 + n_2 \geq 2000_8$	
Quotient	$n_1 - n_2 - 57_8 \geq 2000_8$	
UNDERFLOW		
INSTRUCTIONS	UNDERFLOW CONDITION	RESULT
Upper Sum (30, 31, 34, 35)	None	---
Lower Sum (32, 33)	Final Exponent $\leq -2000_8$	$X_i = 0000\ 0\dots 0_8$
Upper Product (40, 41)	$n_1 + n_2 + 57_8 \leq -2000_8$	
Lower Product (42)	$n_1 + n_2 - 1 \leq -2000_8$	$X_i = 0000\ 0\dots 0_8$
Quotient (44, 45)	$n_1 - n_2 - 60_8 \leq -2000_8$	
<p>*n_1 and n_2 are the initial exponents.</p> <p>Note: Overflow of Upper Sum: Overflow cannot occur unless one operand is infinite. In this case the result is as indicated. If a one-place Right Shift occurs when the larger operand exponent is equal to $+1776_8$, a correct result with exponent $+1777_8$ is generated.</p>		

The counting process is accomplished by left shifting the Xk operand in the D register one bit at a time into the D flag register. For every 1-bit shifted into the D flag, +1 is gated to the F register. The SK counter contains a count of 74_8 , providing the required iterations to shift each bit into the D flag.

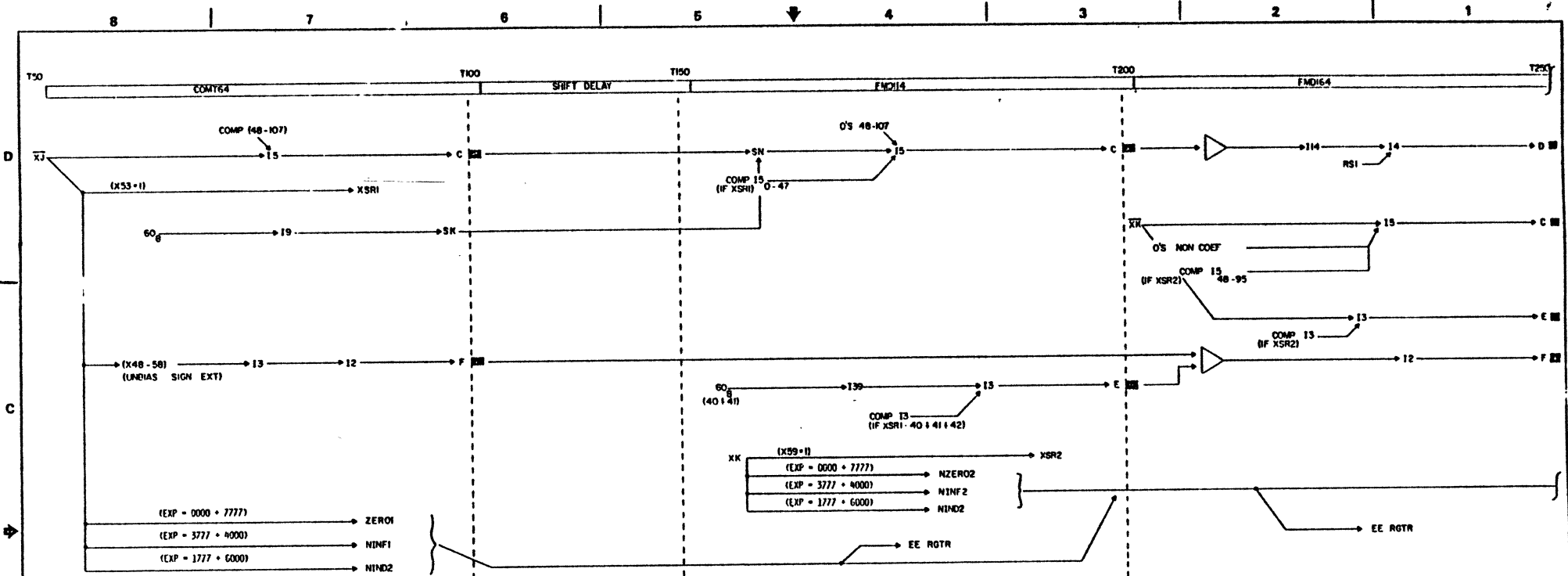
The resulting count value (maximum 74_8) is gated from F to the C register, and during common time to the selected Xi register.



TERM NAME	COMMAND OR FUNCTION	DPD NO	TERM NAME	COMMAND OR FUNCTION	DPD NO	TERM NAME	COMMAND OR FUNCTION	DPD NO	TERM NAME	COMMAND OR FUNCTION	DPD NO	TERM NAME	COMMAND OR FUNCTION	DPD NO	TERM NAME	COMMAND OR FUNCTION	DPD NO
NOTE:	COMMANDS ARE SHOWN FOR FAD214		FAD264			FD114	D-ADD → 114	26	FAD384			FAD414			COM14		
F0214	[34-B0H-XSR1-XSR2]	25	F15004		25	FD314		25	FAD384	D9C-D97	26	F15004	D9-95 → 150-95	25	COM14	SELECT X1 RGR	15
F15C47	[35-B0H-XSR1-XSR2]	25	F15005	SN → 107 → 150-107	25	FENB0	ENABLE D RGR	26	FAD384	14-21 → 14-20	26	F15057		25	COM14	ENABLE WRITE STROBE X RGR	2
	COMP 15 47 (15C1-15C2)	25	F15257	(1554-1552-1551)	25	FD314	CONST0	1	FD114	D → 114	26	F15085	(1554-1552-1551)	25	COM14	[32-33] SELECT LOWER	15
F0214	[XSR2]	26	F15285		25	FD314	+1 → 139	1	FAD384			F15267	F 0-9 → 11508-59	8			10
FAM13C	[XSR2]	26	F15285		25	F1391	139 → 13	26	FENB0	ENABLE D RGR	26	F15267	F10	25			
SC0H3	COMP 13	12	F15067		25	S13913		12	FAD384	ENABLE F RGR	26	FD414-D107	11508-59 → 1506-107	25			
			FD267			FD314			FENB0	ENABLE F RGR	26	F15C67	COMP 15 9C-107	25			
			FENBC	ENABLE C RGR	25	FD314	ENABLE E RGR	26	FD414	ENABLE C RGR	25	FD414	ENABLE C RGR	25			
			FD26A			FD314	[52-53]		FD414	ENABLE E RGR	25	FD414	ENABLE C RGR	25			
			CONST4	60 → 139	1	FENBF	ENABLE F RGR	26	FAD414	ENABLE RHI COMMAND TIME	24						
			CONST5														
			FD26A														
			F13913	139 → 13	26												
			S13913		12												
			FD26A														
			FENBE	ENABLE E RGR	26												
			FD264														
			FAM13C	COMP 13	26												
			SC0H3		12												

(Part 2 of 2)

	INSTRUCTION FLOW SEQUENCE - FAD	FORM 34570	REV D	1990/000	REV A
	INSTRUCTION - 30, 32, 34, 33, 34, 35	FIGURE 5-2-24		PAGE NO 5-2-50.9	



TERM NAME	COMMAND OR FUNCTION	DPD NO
CM50		15
SELXJ	SELECT XJ RGTR	13
COMP64		15
ISCB5	COMP 15 48-95	8
ISCB7	COMP 15 96-107	8
COMP64		15
SL151-SL150	60 → 19	9
NCB64		15
SV1-SK2	PRESET SK CNTR	9
CM50		15
ENABLC	ENABLE C RGTR	8
COMP64		15
XSR1	(IF X59=1)	24
COMP64	48-58 (EXP)	15
CEXP13	UNBIAS SIGN EXT	15
SEXP13	→ 13 0-17	12
NCB64		15
I312	13 → 12	13
ENABF	ENABLE F RGTR	13
CM64		15
NINF1	SET NINF1 (IF EXP=3777+0000)	24
NIND1	SET NIND1 (IF EXP=1777+0000)	24
NZERO1	SET NZERO1 (IF EXP=0000+7000)	24
CM64		15
BB1	SET BDN FF (IF X59 ≠ 47)	24

TERM NAME	COMMAND OR FUNCTION	DPD NO
NCB64		15
15185	70-59 → 15 48-107	7
15285		7
15184		7
15264		7
15236		7
15436		7

TERM NAME	COMMAND OR FUNCTION	DPD NO
ISS1-1552	0'S → NON COEF (15 48-107)	8
ISS4		
FMI114	XSR1	25
F15C0E	COMP 15 0-46	25
F15C97	COMP 15 47	25
FMI114		24
FENBC	ENABLE C RGTR	25
FMD100		24
FMSAK	SELECT XK RGTR	24
FMI114		24
F15SR2	SET XSR2 (IF X59=1)	24
XSR2		24
NINF2	SET NINF2 (IF EXP=3777+0000)	24
NIND2	SET NIND2 (IF EXP=1777+0000)	24
NZERO2	SET NZERO2 (IF EXP=0000+7777)	24
FMI114		24
BB1	CLR BDN FF (IF BDN X59=147)	24
NIMP	(IF 40-42 BDN) (NZERO1-NZERO2) BLOCK END CASE EXIT FOR INTEGER MULTIPLY	25
FMI114		24
COM15		1
COM15	60 → 139	1
FMI114		26
F13913	139 → 13	12

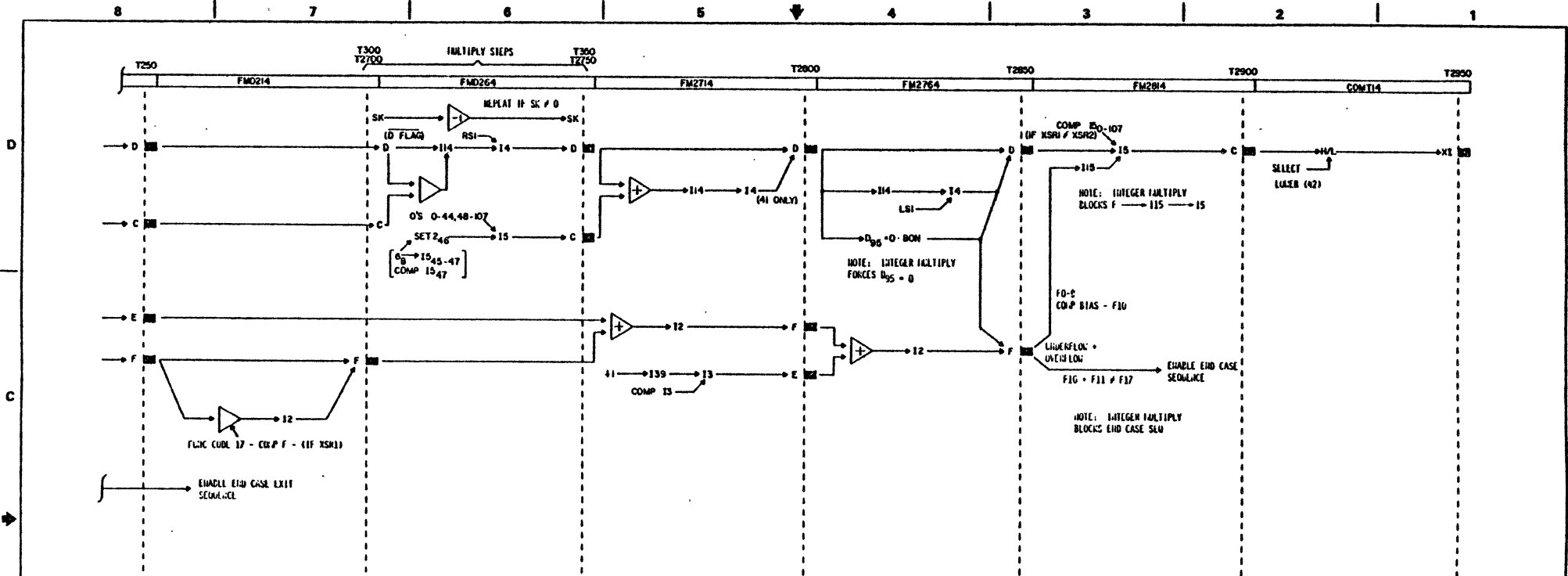
TERM NAME	COMMAND OR FUNCTION	DPD NO
FMI114		24
F1513C		26
SCM13		12
FMI114		26
FENBE	ENABLE E RGTR	26
FMI114		25
F15004		25
F15204		25
F15057		25
F15257		25
FMI114		26
FENBEE	ENABLE EE RGTR	26

TERM NAME	COMMAND OR FUNCTION	DPD NO
FMI114	D ADD → 114	26
FMI164	14-20-14-21	26
	RS1 → 14	
FMD150		24
FMSAK	SELECT XK RGTR	24
ISS4-1552	0'S → NON COEF (15 0-47, 96-107)	8
ISS1		
FMI114	XSR2	24
F15CB5	COMP 15 48-95	25
FMI114	48-58 (EXP)	15
CEXP13	UNBIAS SIGN EXT	15
SEXP13	→ 13 0-17	12
FMI114	XSR2=40 + 41 + 42	26
F1513C		26
SCM13	COMP 13	12
FMI114		26
FENBF	ENABLE F RGTR	26
FENBE	ENABLE E RGTR	26
FMI114		26
FENBD	ENABLE D RGTR	26
FENBC	ENABLE C RGTR	26
FMI164		26
F15185	70-59 → 15 48-95	7
F15285	(ISS4-1552-1551)	7

TERM NAME	COMMAND OR FUNCTION	DPD NO
FMI114		26
F15CB7	40 + 40	26
FMI114	COMP 15 96-107	26
FENBEE	ENABLE EE RGTR	26

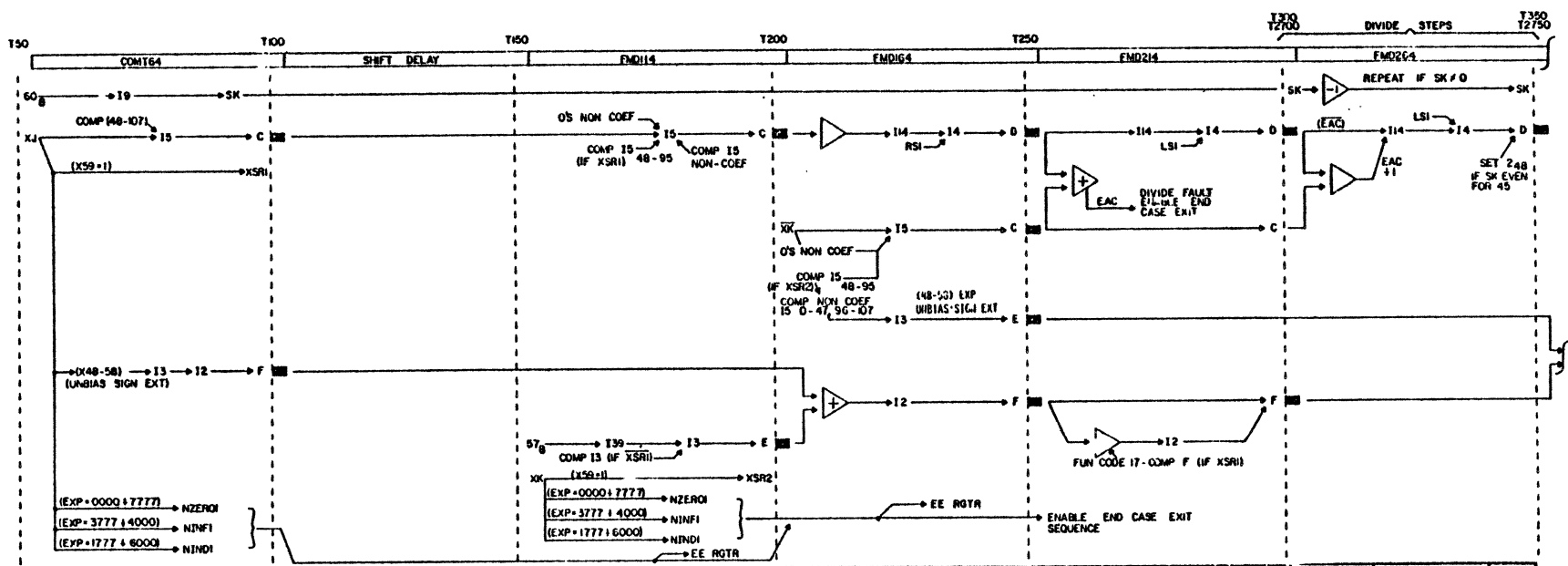
(Part 1 of 2)

INSTRUCTION FLOW SEQUENCE - FMD	INSTRUCTION - 40, 41, 42	34570	19981800
FIGURE 5-2-25		3-2-50 6	



TERM NAME	COMMAND OR FUNCTION	DPD NO	TERM NAME	COMMAND OR FUNCTION	DPD NO	TERM NAME	COMMAND OR FUNCTION	DPD NO	TERM NAME	COMMAND OR FUNCTION	DPD NO	TERM NAME	COMMAND OR FUNCTION	DPD NO	TERM NAME	COMMAND OR FUNCTION	DPD NO	
F11214	[XSR1] ENABLE F RGR	26	F12264	SKS1-SKS2 DECREMENT SK COUNTER	9	F1114	D ADDER → 114	26	F2764	D RGR → 114	26	F2814	CURK00	15				
			F1264	[D FLAG] 40+41+42	26	F2714	41	26	F15004	D ₀₋₉₅ → 15 ₀₋₉₅	25		SELX1	SELECT X1 RGR	2			
			F1114	D RGR → 114	26	F2714	ENABLE D RGR	26	F15057	(TSS4-TSS2-TSS1)	25		COEXK	ENABLE WRITE STORE X RGR	2			
			F1114	D ADD → 114	26	F2714	+1 → 139	1	F15085	(F0-9) F10	25							
			F1264	14-20-14-21	26	F2714	139 → 13	26	F15267	F0-9 → 115 ₄₈₋₅₉	25		COXK00-42		15			
			F11264	SK00	8	F13913	139 → 13	12	F2814	115 ₄₈₋₅₉ → 15 ₄₈₋₁₀₇	25		HLSL	SELECT LOWER	10			
			F1264	FEHDC	25	F2714	COMP 13	26	F2814	FEHBC	25							
			F1504-TSS1	0'S → 15 ₀₋₄₄	8	F2714	COMP 13	12	F2814	FEHBC	25							
			F1504-TSS1	48-107	8	F2714	ENABLE F RGR	26	F2814	FEHBC	25							
			F15057	6 ₈ → 15 ₄₅₋₄₇	25	F2714	ENABLE E RGR	26	F2814	FEHBC	25							
			F15157	COMP 15 ₄₇	25				F2814	FEHBC	25							
			F15257	COMP 15 ₄₇	25				F2814	FEHBC	25							
			F1547	COMP 15 ₄₇	25				F2814	FEHBC	25							
			F1264	FLHDD	26				F2814	FEHBC	25							

(Part 2 of 2)



TERM NAME	COMMAND OR FUNCTION	DPD NO
CB:50	SELECT XJ RGTR	15
SEL1J		13
CM:4A	COMP 15 48-95	8
15C85	COMP 15 96-107	8
15C67		8
CO:10A	CO → 19	15
SCT91-SCT90		9
HC06A	ORESET SK CTR	9
SK1-SK2		9
CU:150	ENABLE C RGTR	15
ENABLC		8
CO:16A	SET XSR1 (IF X59=1)	24
XSR1		24
CO:16A	X48-58 (EXP)	15
EXP13	UNBIAS-SIGN EXT	15
SE:13	→ 13 0-17	12
HC06A	13 → 12	13
1512	ENABLE F RGTR	13
E:17F		13
CO:16A	SET H10F1	24
H10F1	(IF EXP=3777+0000)	24
CO:16A	SET H10C1	24
H10C1	(IF EXP=1777+6000)	24
CO:16A	SET H2ER01	24
H2ER01	(IF EXP=0000+7000)	24
CO:16A	SLT DB1 FF	24
DB1	(IF X59=47)	24
HC06A		7
15185	X0-59 → 15 48-107	7
15285		7
1516A	(155A-1552-1551)	7
1526A		7
15236		7
1543C		7

TERM NAME	COMMAND OR FUNCTION	DPD NO
155A-1552	O'S NON COEF (15 0-47, 96-107)	8
F1114	XSR1	25
F15C85	COMP 15 48-05	25
F1214	ENABLE C RGTR	25
F15100	SELECT XK RGTR	24
F15XK		24
F1114	SET XSR2 (IF X59=1)	24
EXP13	SET H10F2 (IF EXP=3777+0000)	24
H10F2	(IF EXP=1777+6000)	24
H10D2	SET H2ER2 (IF EXP=0000+7777)	24
H2ER2		24
F1214	CLR DB1 FF (IF DB1=X59=X47)	24
DB1	44 + 45	24
F1214	CONST0	1
CONST0	57 → 139	1
CONST1		1
CONST2		1
CONST3		1
CONST5		1
F1114	139 → 13	12
F13913		12
S13913		12
F1114	ENABLE E RGTR	26
FE:16E		26
F1114	ENABLE EE RGTR	26
FE:16E		26

TERM NAME	COMMAND OR FUNCTION	DPD NO
F0114	D ADD → 114	26
F1114	114 → 114	26
F1214	RS1 → 114	26
F1214	SELECT XK RGTR	24
F15A		24
155A-1552	O'S NON COEF (15 0-47, 96-107)	8
1551		8
F110A	X48-58 (EXP)	26
EXP13	UNBIAS-SIGN EXT → 13 0-17	12
SE:13		12
F110A	(XSR2+44+45)	26
F1113C	COMP 13	12
SC:113		12
F110A	ENABLE F RGTR	26
FE:16F		26
F110A	ENABLE D RGTR	26
FE:16D		26
F110A	ENABLE C RGTR	26
FE:16C		26
F1214	X0-47 → 15 48-95	25
F15285	(155A-1552-1551)	25
F110A	[44 + 45]	25
F15C06	COMP 15 0-47, 96-107	25
F15C47		25
F110A	ENABLE EE RGTR	26
FE:16E		26
F110A	(XSR2)	24
F15C65	COMP 15 43-95	25

TERM NAME	COMMAND OR FUNCTION	DPD NO
F1214	[44+45-EN107]	26
FD114	D RGTR → 114	26
H114C	SET FCC	24
F1214	DIVIDE FAULT ENABLE END CASE PLOCK	24
F1214	114 → 14	26
F1214	14-20-16-21	26
F1214	LS1 → 14	26
F1214	(XSR1)	26
F1214	ENABLE F RGTR	26
F1214	ENABLE D RGTR	26

TERM NAME	COMMAND OR FUNCTION	DPD NO
F126A	SKS1-SKS2	9
F126A	DECREMENT SK COUNTER	26
F110A	[44 + 45]	26
EN107	D RGTR → 114	5
1145	[44 + 45]	5
F126A	D ADD → 114	26
F126A	114 → 14	26
F126A	14-20-14-21	26
F126A	LS1 → 14	26
F126A	ENABLE D RGTR	26
F126A	[SK00+45] 1 → D48	5

(Part 1 of 2)



INSTRUCTION FLOW SEQUENCE - FMD INSTRUCTION - 44, 45

CODE 34570 D 19981800 FIGURE 5-2-26

8

7

6

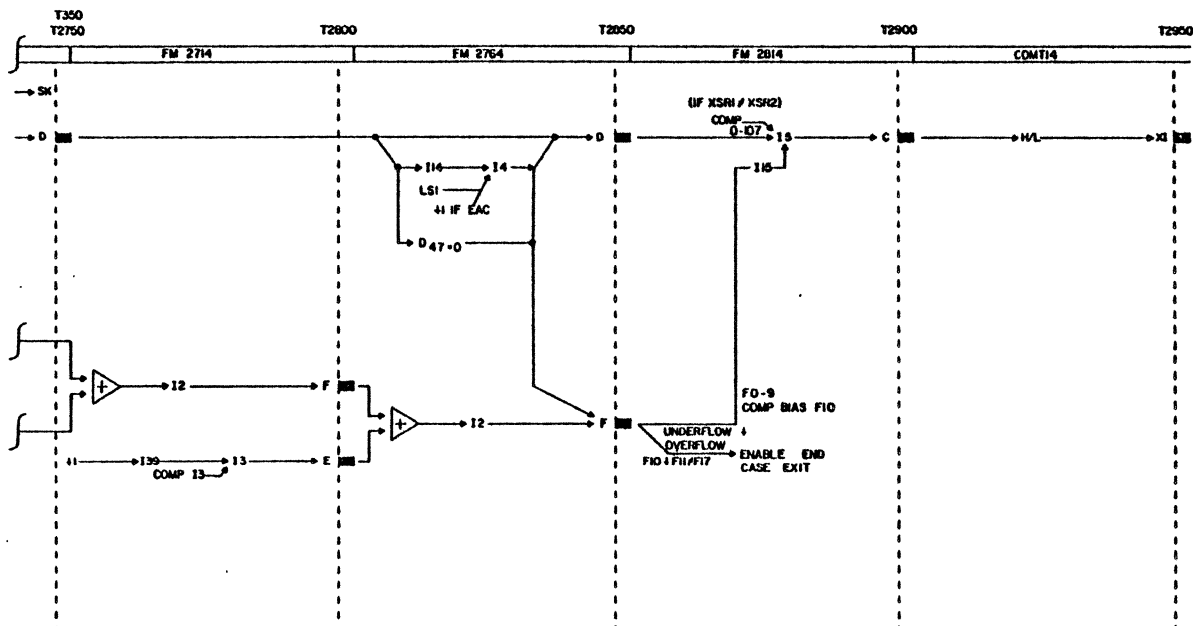
5

4

3

2

1



TERM NAME	COMMAND OR FUNCTION	DPD NO	TERM NAME	COMMAND OR FUNCTION	DPD NO	TERM NAME	COMMAND OR FUNCTION	DPD NO	TERM NAME	COMMAND OR FUNCTION	DPD NO
F2714 CONSTO	+1 → 139	1	F2764 FD114	D RGTR → 114	26	F2814 F15004	D ₀₋₉₅ → 15 ₀₋₉₅	25	COMX00	SELECT XI RGTR	15
F2714			F2764			F15057	(1554-1552-1551)	25	SELX1	ENABLE WRITE STROBE	2
F13913 S13913	139 → 13	26		114 → 14 LS1	26	F15065	F ₀₋₃ → 11 ₀₄₋₅₈	25	COMENX	ENABLE WRITE STROBE X RGTR	2
F2714			F2764	[D ₄₇₋₄₄₊₄₅]		F15267	F ₁₀	8	HLSL	SELECT HIGHER	10
F4K13C SC4M13	COMP 13	26	F2764	{ ENABLE D RGTR ENABLE F RGTR	26	F15267	115 ₄₈₋₅₈ → 15 ₉₆₋₁₀₇	25			
F2714			F2714			F2814	[44 + 45, XSR14 XSR2	25			
F4EBF FEHBC	ENABLE F RGTR ENABLE E RGTR	26				F15C06	COMP 15 ₀₋₁₀₇	25			
						F15C47		25			
						F15C85		25			
						F15C67		25			
						F2814	ENABLE C RGTR	25			
						FEHBC					
						F2814	ENABLE RNI, COMMON TIME	24			
						F2814					
						F2814	F10 + F11 = F17	24			
						F2814	ENABLE END CASE F X11	24			
						FBNF78					
						FEHBC					

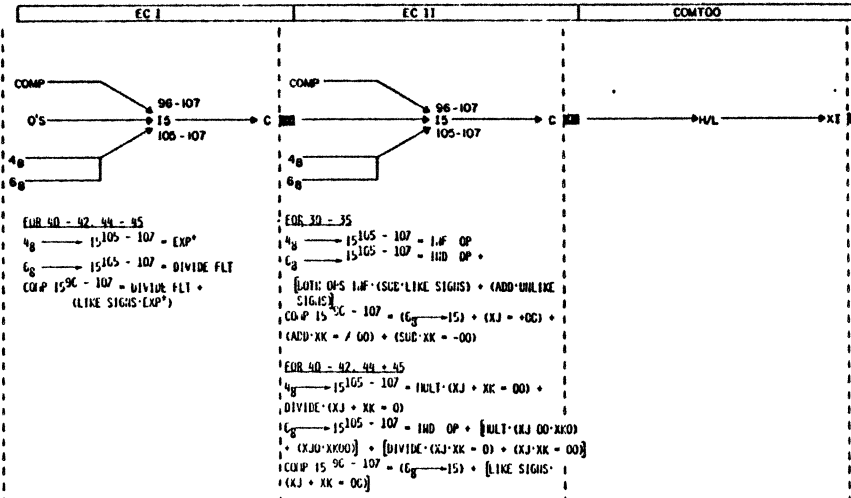
(Part 2 of 2)



INSTRUCTION FLOW
SEQUENCE • F740
INSTRUCTION • 44, 45

Doc ID: 34570
Rev: D
1998800

FIGURE 5-2-26



RESULTS

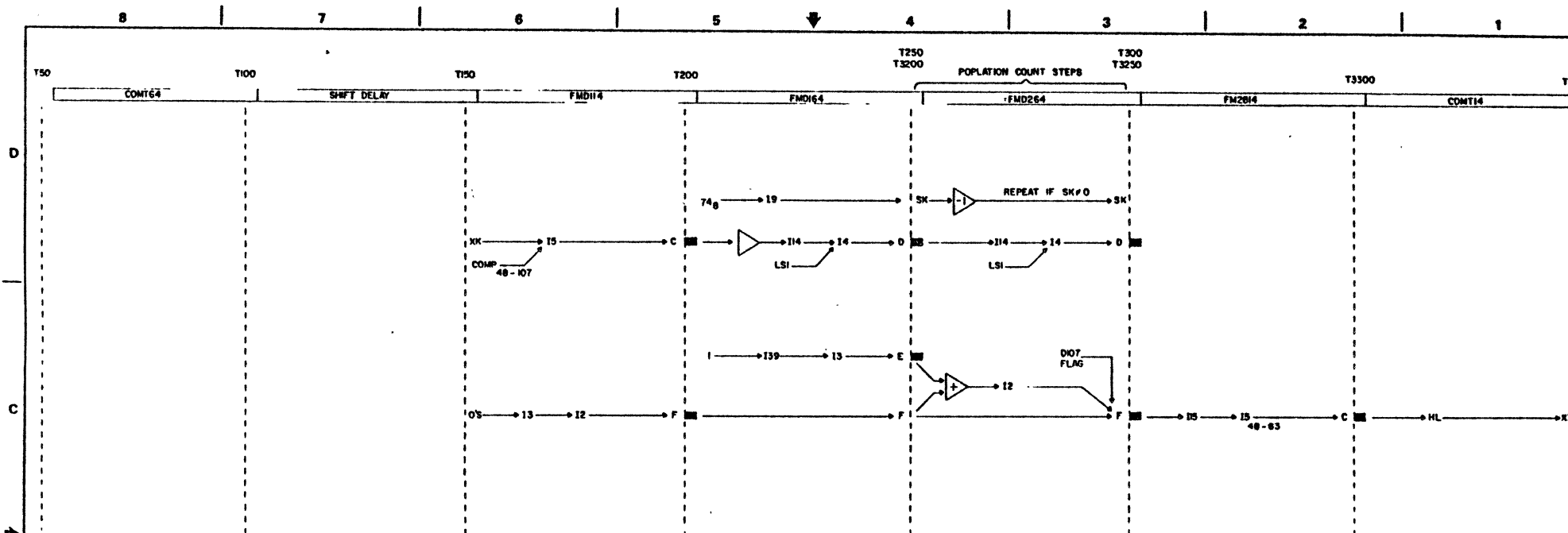
0000	0	0	ZERO
3777	0	0	+OO
4000	0	0	-OO
1777	0	0	IND

TERM NAME	COMMAND OR FUNCTION	DPD NO	TERM NAME	COMMAND OR FUNCTION	DPD NO	TERM NAME	COMMAND OR FUNCTION	DPD NO
FAENDC	[IND + INF + UNDERFLOW]	24	ECII			COMXOO		15
FVENDC	[IND + INF + UNDERFLOW + ZERO OPERAND + DIVIDE FAULT]	24	E1510	4b → 15 ¹⁰⁵⁻¹⁰⁷	25	SELXI	SELECT XI RGTR	2
ENDC	ENABLE END CASE EXIT (ECI)	25	E15257	8b → 15 ¹⁰⁵⁻¹⁰⁷	25	COFENK	ENABLE WRITE STROBE X RGTR	2
1554-1552-1551	0'S → 15	8	EC1567	COMP 15 ⁹⁶⁻¹⁰⁷	25			
ECI			ECII	IND + INF + ZERO				
E1510	4b → 15 ¹⁰⁵⁻¹⁰⁷	25	ECENDC	ENABLE C RGTR	25			
E15257	8b → 15 ¹⁰⁵⁻¹⁰⁷	25	ECII					
EC1567	COMP 15 ⁹⁶⁻¹⁰⁷	25	ECENIT	ENABLE RNT, COMMON TIME	25			
ECI								
ELENDC	ENABLE C RGTR	25						



END CASE EXIT SEQ

DOC# 34570	FORM NO 19981800	REV A
FIGURE 5-2-27		PAGE NO 5-2-5010



TERM NAME	COMMAND OR FUNCTION	DPD NO	TERM NAME	COMMAND OR FUNCTION	DPD NO	TERM NAME	COMMAND OR FUNCTION	DPD NO	TERM NAME	COMMAND OR FUNCTION	DPD NO	TERM NAME	COMMAND OR FUNCTION	DPD NO
F1114	(F1114)	25	SET90-SEL191	74 _b → 19	9	F1264	SKS1-SKS2	9	F1264	DECREMENT SK	9	F1264	F0-17 → 1150-17	8
F15185	(F15185)	25	COEX-1			F1264	COUINTER		F2814	SELECT X1 RGR	15	COJ00	SELECT X1 RGR	15
F15285	X0-59 → 1548-107	25	F1114	D ADD → 114	26	F1264	D ADD → 114	26	F15285	115 → 1548-63	25	COEX-1	ENABLE WRITE STROBE X RGR	2
F15167	(155A-1552-1551)	25	F116A	114 → 114	26	F1264	[47]		F2814	ENABLE C RGR	25			
F1114	(F1114)	25	F116A	114 → 114	26	F1264	14-20 (14-21)	26	F2814	ENABLE D RGR	26			
F11715	COMP 15 48-107	24	F116A	LSI → 114	26	F1264	LSI → 114	26	F2814	ENABLE RGR, COMMON TIME	24			
F15085		25	F116A	ENABLE D RGR	26	F1264	ENABLE D RGR	26	F1264					
F1114	(F1114)	25	F116A	(F1114)		F1264	ENABLE D RGR	26						
F1114	ENABLE C RGR	25	F116A	COJ00	1	F1264	[47-D FLAG]	26						
DEFAULT	O'S → 13	12	F1114	(F1114)		F1264	ENABLE F RGR	26						
F1114	F1114	26	F13913	139 → 13	26									
F1312	13 → 12	26	F116A	ENABLE E RGR	26									
F114F	ENABLE F RGR	13												

DETAILED PAK DIAGRAM (CPU 3.27)

ECS SUBSYSTEM SEQUENCE

Detection of an ECS instruction (011JK or 012JK) in parcels 0 and 1 causes an ECS request to be sent to the ECS coupler. The continuation of the ECS sequence is suspended until the accept (ECSACP) is returned. The ECS sequence is responsible for making address range tests for both the ECS address and the CM address. This is done in each case by comparing the last word address against the field length (FLE or FL). A test for negative word count is also performed. Violation of any of these conditions causes an address range error (AOR) and aborts the ECS sequence. An attempt to execute an ECS instruction from the wrong parcel, or when no ECS coupler is present, forces an illegal instruction fault.

The ECS sequence sends the word count ($Bj + K$) and the ECS starting address ($X0\ 0-23 + RAE$) to the ECS coupler, and the starting address ($A0 + RA$) to CMC. If none of the abort conditions are present, a start transfer is sent to the coupler to initiate movement of data. The CPU remains idle during the transfer.

An error exit (ERRABT) or normal exit (ENDTRC) will be sent to the CPU at the completion of the data transfer. The error exit causes the processor to execute the instruction (usually a branch) that is in parcels 2 and 3 of the ECS instruction word. A normal exit bypasses this instruction and does an initial start RNI to the next word.

FLAG REGISTER

The ECS subsystem also contains a flag register primarily used for communication between processors attached to ECS. Access to this register is through an ECS instruction identified by both bit 23 of X0 and bit 23 of FLE being set. The ECS sequence must be modified when a flag operation is detected. The contents of X0 0-23 are sent to the coupler without the addition of RAE. No data transfer is made; however, the coupler will respond with either error or normal exit depending on the flag function bits in X0 and the condition of the flag register.

EXCHANGE BREAKIN

If an exchange request arrives during the execution of an ECS transfer, the ECS instruction is terminated. No provision is made for maintaining addresses or number of words transferred. Consequently, the P register value stored in the exchange package will point to the ECS instruction. It will be reinitiated at the next execution interval of the program as if no ECS data had been processed.

8

7

6

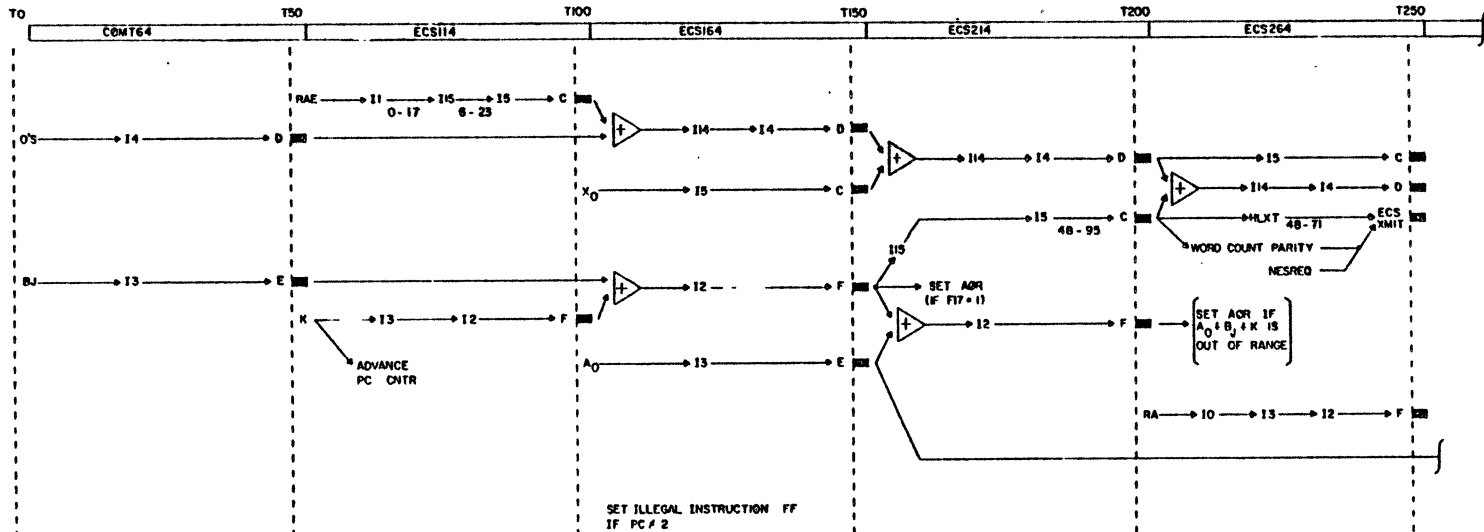
5

4

3

2

1



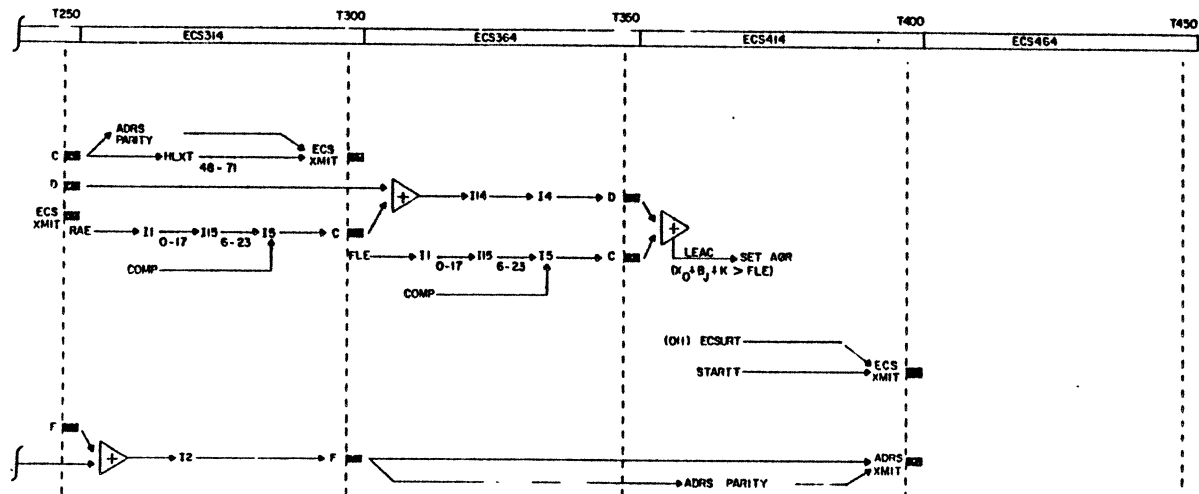
TERM NAME	COMMAND OR FUNCTION	DPD NO	TERM NAME	COMMAND OR FUNCTION	DPD NO	TERM NAME	COMMAND OR FUNCTION	DPD NO	TERM NAME	COMMAND OR FUNCTION	DPD NO	TERM NAME	COMMAND OR FUNCTION	DPD NO		
CB.50	SELECT BJ RGTR	15	ECS114	R1X → 11	27	ECS164	ENABLE D RGTR	27	ECS214	ENABLE D RGTR	27	HE26A	D → 15 ₄₈₋₁₀₇	27		
SEL13		2	SPACE5	11 ₀₋₁₇ → 11 ₆₋₂₃	8	ECS114	ENABLE F RGTR	27	ECSE1D	ENABLE F RGTR	27	15085		7		
CB.114	B → 13	12	ECS115	115 ₀₋₅₉ → 115 ₄₈₋₉₅	27	HE16A		27	ECSE1F	ENABLE F RGTR	27	1506A	(1554-1552-1551)	7		
HEB50	ENABLE E RGTR	13	15285	(1554-1552-1551)	7	HE16A	Y ₀₋₅₉ → 15 ₄₈₋₁₀₇	7	FRG-ESTES	F ₀₋₁₇ → 11 ₀₋₁₇	8	15135		7		
HEB50	ENABLE E RGTR	13	HE114	K → 13	27	ECS214	ES1115	115 ₀₋₅₉ → 15 ₄₈₋₉₅	27	ECS214	ES1115	115 ₀₋₅₉ → 15 ₄₈₋₉₅	27	HE26A	COUPLER PLO →	27
HEB50	ENABLE D RGTR	5	SELX13	ADVPC2	12	ECS164	ENABLE C RGTR	27	15285	(1554-1552-1551)	7	NESREQ	ECS XMIT	27		
HEB50	ENABLE D RGTR	5	ECS114	ES1312	27	HE16A	SELX13	A → 13	27	ECS214	ECSE1C	ENABLE C RGTR	27	ECS264	ENABLE D RGTR	27
			ENAF	ENABLE F RGTR	13	HE16A	ENAF	ENABLE E RGTR	13	ES214	ADR	ENABLE C RGTR [F17=1] SEL ADR	27	EL26A	ENABLE TEST ADR	17
												ST10-S101	RA → 10	4		
												S102				
												HE26A	SL1013	10 → 13	27	
												ECS264	ES1312	13 → 12	27	
												ENAF	ENABLE F RGTR	13	27	

(Part 1 of 2)



INSTRUCTION FLOW
SEQUENCE - ECS
INSTRUCTION - 01, 02

Doc ID: 34970	Rev: D	Part No: 19981800
FIGURE 5-2-29		PAGE 5-2-56 I



TERM NAME	COMMAND OR FUNCTION	DPD NO	TERM NAME	COMMAND OR FUNCTION	DPD NO	TERM NAME	COMMAND OR FUNCTION	DPD NO	TERM NAME	COMMAND OR FUNCTION	DPD NO
ECS314 DRACCS	RAE → 11	27	ECS364 SRLECS	FLE → 11	27	ECS414 SETADR	[FLEAC] SET ADR	27	ECS464 RESETR	[ECSRRT] ENABLE RUI	27
ECS314 FRAC-ES115	11 ₀₋₁₇ → 115 ₆₋₂₃	8	ECS364 FRAC-ES115	11 ₀₋₁₇ → 115 ₆₋₂₃	8				ECS464 ESERPK	[ECSRRT] ENABLE SER EXIT	27
ECS314 ES115 15205	115 ₀₋₅₉ → 15 ₄₈₋₉₅	27	ECS364 ES115 15205	115 ₀₋₅₉ → 15 ₄₈₋₉₅	27						
ECS314 ECS15C	COMP 15	27	ECS364 ECS15C	COMP 15	27						
ECS314 ECS15C	ENABLE F RGTR	27	ECS364 ECS15C	ENABLE D RGTR	27						
ECS314 ECS15C	ENABLE C RGTR	27	ECS364 ECS15C	ENABLE C RGTR	27						

(Part 2 of 2)



INSTRUCTION FLOW
SEQUENCE 1
ECS
INSTRUCTION: 01, 012

CODE SHEET	REV. NO.	DATE
34570	D	1998.800
FIGURE 5-2-29		5-2-56.2

DETAILED PAK DIAGRAM (CPU3.28)
COMPARE/MOVE DATA SECTION (Part One)

The compare move data section consists of four data registers. Three are shown on diagram 3.28; they are the 54-bit R register, the 60-bit Q register, and the 60-bit S register. The remaining register is shown on diagram 3.29; it is the 48-bit T register.

Q REGISTER

The 60-bit Q register is located on the JC module. It is the primary word formation register. Input to the Q register is through selector I30 which allows selection of either the C register bits 48-107 or R register bits 0-47, 108-113.

R REGISTER

The 54-bit R register is also located on the JC module. It is used as a residue register for data right shifted in the C register prior to storing in the Q register. The input to R comes directly from the C register bits 0-47, 108-113.

S REGISTER

The 60-bit S register is located on the JB module. It acts as a buffer register for data stored in the Q register.

During move instructions (464, 465), data words that have been properly formatted in the Q register are transferred to the S register. The output of S gates directly to the HR register and the output transmitters.

During a compare instruction (466, 467), the S register serves a more useful purpose. Data words that have been properly formatted in the Q register are transferred to the S register awaiting subsequent comparison with corresponding words stored in the Q register.

COMPARISON CIRCUITS

Data comparison is performed on a word basis by the S = Q compare circuit located on the JB module. Each JB module is capable of one character comparison.

The output of the S = Q compare circuit generates a compare character equal signal for each character (COME 0-9). Compare character equal will be at one level when the respective characters in S and Q are equal. The compare character equal signals can also be forced to indicate equality by the force equivalence circuits. These circuits are used by the compare collate instruction exclusively.

If all ten characters in the S register and Q register compare equal, the compare word equal signal (CWEQ) will be generated from the JF module. Compare word equal allows comparison of the next pair of words.

If an inequality exists between the characters in S and Q, the respective compare character equal signals for the unequal characters will be at a zero level. These zero level compare character equal signals are monitored by an unequal character position priority encode circuit, located on the JF module. The output of the encoder provides a 4-bit binary code pointing to the first unequal character. This binary code is stored in the character position (CP) register.

FORCE EQUIVALENCE CIRCUITS

The output of the character position register (CP) feeds a +1 incrementer circuit also located on the JF module. The incrementer is used to force equivalence for a collate instruction.

For example, assume that during the execution of a compare collate instruction a pair of characters are found unequal. The character position code for the unequal characters is stored in the character position register. The collating characters corresponding to the unequal characters are read from the collate table and compared. Should they be equal, the instruction continues. The code in the CP register is incremented by one, pointing to the next character to be compared. The incremented value is fed to the force equivalence decoder which generates 10 force equivalence bits (DEC 0-9). The force equivalence bits cause an equal comparison to occur on all characters preceding the unequal character and including the unequal character. Comparison of the remaining characters occurs as described previously.

DETAILED PAK DIAGRAM (CPU 3.29)
COMPARE/MOVE DATA SECTION (Part Two)

The circuitry shown on diagram 3.29 is used by the compare collate (466) and compare uncollated (467) instructions.

COMPARE UNCOLLATED (467)

For a compare uncollated instruction, the circuitry on this diagram determines whether the unequal character in Q was greater than, or less than, the unequal character in S. Selectors I32 and I33 on the JD module gate the unequal character from S via I37 to the TS register, and the unequal character from Q via I37 to the TQ register. The unequal character is selected using the code stored in the character position register (CP).

Each JD module is capable of performing a single bit comparison between TS and TQ. The output of the TS = TQ compare circuit generates a compare bit equal signal for each bit (CMTC 0-5). The compare bit equal signals will be at a one level when the respective bits in TS and TQ are equal.

The compare bit equal signals (CMTC 0-5) are fed to a priority encode circuit on the JE module. The priority encode circuit, scanning from left to right, produces a code pointing to the first unequal bit in the TS and TQ registers. The priority code is then fed to a multiplexer circuit. The multiplexer circuit monitors the TQ register bits 0-5. By using the binary code from the priority encoder, the multiplexer circuit will select the appropriate TQ register bit that compared unequal. If this bit equalled one, TQ would be greater than TS and the QGS signal will be generated.

QGS is used during the exit sequence to condition the X0 register.

COMPARE COLLATE (466)

For a compare collate instruction the circuitry on this diagram performs the collate operation.

I32 and I33 will select the unequal character from S and Q using the code stored in the CP register. These characters are then stored in the TS and TQ registers via the I37 selector.

Assuming that this is the first time a collate operation is being performed during the instruction execution, the T register will not contain a valid collate table word. Consequently, the word position register located on the JE module will not contain a valid word position code.

The word position register (WP) feeds two test circuits located on the JE module. These circuits determine whether the word position code is equal to the upper 3 bits of the unequal character in the TS and TQ registers. However, the output of these two test circuits are blocked by the first collate signal NICOL in its active state.

The collate sequence uses the output from the WP = TQ and the WP = TS test circuits to condition two equality detection FFs. The third equality detection FF is located on the JE module. A TQ = TS test circuit feeds the TQ = TS equality detection FF.

The contents of these three detection FFs determine the sequence of events that occur during a collate operation.

With the WP = TQ and WP = TS test circuits blocked by first collate active, only two equality detection possibilities can occur:

	<u>TQ = TS</u>	<u>TQ = WP</u>	<u>TS = WP</u>
1.	1	0	0
2.	0	0	0

The TQ = TS detection FF indicates that both collate characters are contained in the same collate table word. If $\overline{TQ = TS}$ the collate characters are located in different words of the collate table.

COLLATE TABLE LOOK-UP - $\overline{TQ = TS}$

Assuming TQ = TS, two central memory references are required for collate table look-up. An address pointing to the first word of the collate table is stored in the A0 register. The contents of A0 are added to TS register bits 3-5 to provide the address for the first table look-up. TS register bits 3-5 are also gated to the WP register via I35.

After the word read from the collate table is received at the CR9 register, it is gated to the table register (T) located on the JG module. The table register feeds a priority multiplex circuit capable of selecting one of the eight collate characters from table register. Selection is determined by the binary code selected via I38 located on the JD module. At this time I38 would select the lower 3 bits of the TS register (0-2) to I38, so that the appropriate character is selected in I34. The selected collate character is gated to I37 located on the JD module. I37 will now select the collate character from I34 to be stored in the TS register.

Another memory reference will obtain the second word from the collate table. The procedure is similar to that already described except that the TQ register is used.

After the second word read from the collate table is received at the CR9 register, it is gated to the table register, destroying the previous contents. The lower 3 bits (0-2) of the TQ register are gated to I38, allowing selection of the second collate character from I34 to I37. I37 will select the collate character to be stored in the TQ register.

With both collate characters stored in the TS and TQ registers, the contents of TS and TQ are compared for equality using the comparison circuit located on the JD module. If both collate characters are equal, the collate character equal signal (CCEQ) from the JE module will allow normal continuance of instruction execution.

COLLATE CHARACTER COMPARISON - EQUAL

The remaining circuits on the JE module serve a useful purpose if the result of a collate character comparison is equal. The word position register (WP) will contain a code pointing to the collate table word referenced on the last memory request. This is the word contained in the table register.

If during the same collate instruction execution another collate operation is required, the collate sequence control logic will check the condition of the three equality detection FFs to determine the sequence of events.

Five possible combinations can occur:

	<u>TQ = TS</u>	<u>TQ = WP</u>	<u>TS = WP</u>	<u>MEMORY REQUESTS</u>
1.	-	1	1	NONE
2.	0	0	0	2
3.	1	0	0	1
4.	-	0	1	1
5.	-	1	0	1

If both TQ = WP and TS = WP, both collate characters are stored in the table register; a memory reference is not required.

If TQ = TS and TQ = WP and TS = WP, neither collate character is stored in the table register. Two memory requests are required to obtain the collate characters from the collate table.

Finally, if either TQ = WP or TS = WP, both collate characters are located in the same table word. Only one memory request is required to obtain both characters from the table.

DETAILED PAK DIAGRAM (CPU 3, 30)
COMPARE/MOVE CONTROL SECTION (Part One)

C1, C2 OFFSET REGISTERS

The C1 and C2 offset registers are located on the JX module. C1 provides a 4-bit offset value for the first word of the K1 field. C2 provides a 4-bit offset value for the first word of the K2 field.

I41, I42 - C-ADDER

Selector circuits I41 and I42 are located on the JX module. The outputs of I41 and I42 provide the A and B inputs to the C adder on the JY module. Depending upon the gating term selection of I41 and I42 (both operate in parallel), the C adder may perform three functions:

1. Subtract C2 from C1 (by complement addition)
2. Subtract C1 from C2 (by complement addition)
3. Add SCR + (+12₈)

The C adder output is gated to the shift count register (SCR). The shift count value is used to shift characters in the C register to the appropriate position (depending on the C1, C2 offset value) before loading in the Q register. The shift count value stored in SCR represents the number of characters that must be right shifted. This value is gated to a times-six translator circuit that converts the character shift count to a bit shift count. The translator output (SCRX 1-5) is gated to the shift count register via I19 and I9 (CPU 2.8).

I40, I40 DECODE, I44

Selector I40 located on the JX module provides a 4-bit input path to selector I44 and the I40 decode circuit located on the JJ module. Depending on gating term selection of I40, the contents of any one of four registers may be gated to the character select register (CSR) located on the JZ module.

C1 → I40 → I40 DECODE → CSR
C2 → I40 → I40 DECODE → CSR
LA → I40 → I40 DECODE → CSR
LC → I40 → I40 DECODE → CSR

In the same manner, the gating term selection of I40 allows the contents of any one of three registers to be gated to the partial write register (PW) also located on the JZ module.

SCR → I40 → I40 DECODE → PW
LA → I40 → I40 DECODE → PW
LC → I40 → I40 DECODE → PW

The purpose of the I40 decoder is to transform the 4-bit code from one of the registers listed above to a 10-bit code representing the ten character positions in a word.

Selector I44 located on the JX module provides a 16-bit input path to the LE register located on the JM module (CPU 3.31). The gating term selection of I44 allows the contents of any one of three registers, or a generated constant value to be gated to the LE register.

C1 → I40 → I44 → LE
C2 → I40 → I44 → LE
CP → I44 → LE
-12₈ → I44 → LE

CHARACTER SELECT/PARTIAL WRITE REGISTERS (CSR, PW)

The character select and partial write registers contain a 10-bit code, each bit representing one of ten character positions in the Q register. CSR, CSR complement, PW complement and CSR ≠ PW are gated to the I47 selector. Depending on gating term selection of I47, the appropriate CSR/PW bits provide an enable or disable on the Q register input load circuit (CPU 3.28).

DETAILED PAK DIAGRAM (CPU 3.31)
COMPARE/MOVE CONTROL SECTION (Part Two)

LA, LC REGISTERS

The LA and LC registers are located on the JN module. At the beginning of a compare/move instruction, the LA and LC registers will contain an octal representation of the character field length. The length value is gated from CR9 bits 26-29, 48-50 (465, 466, 467) or CR9 bits 26-29, 48-56 (464), via the I46 selector located on the JM module.

LAC1, LAC2 REGISTERS

The LAC1 and LAC2 registers receive the current LA or LC length value via selector I45. The length value in LAC2 is used during compare unequal to determine the character count value to be stored in the X0 register upon instruction conclusion.

LE, LF REGISTERS

The LE and LF registers provide the A and B input path to the L adder located on the JM module. The LE register receives its input from selector I44 located on the JX module (CPU 3.30). Depending on the selections made at I40 and I44 (described previously), the LE register will receive either the contents of C1 or C2, the complemented contents of the character position (CP) register, or a generated constant value of -12_8 .

The LF register receives its input from selector I45. I45 allows the contents of LA, LC, LAC2 or the L adder output via I46 to be gated to the LF register.

L ADDER

The L adder performs four functions required for character length calculations. Initially, the contents of LA and LC are added to the contents of C1 and C2, respectively. The results are returned to the LA and LC registers. During K1 and K2 address sequences, LA and LC are decremented by -12_8 . The address sequence monitors the decremented values of LA and LC to determine a K1 or K2 exhaust condition. In addition, during K2 address sequences for a move, the decremented LC value from the L adder is gated via I45 to the LF register to perform a double subtraction. The double subtraction provides a look-ahead function that detects an exhaust condition on the next K2 address sequence. Finally, the L adder allows the contents of the character position register (CP) to be subtracted from the decremented LA or LC value in the LAC2 register on compare unequal. The resultant value will be stored in the X0 register upon instruction termination.

DETAILED PAK DIAGRAM (CPU 3.32)
INSTRUCTION DECODE SEQUENCE, START SEQUENCE

The instruction decode sequence is initiated from the common time sequence by the GOCMU signal. The sequence decodes a 460 (pass), 464 (move indirect), 465 (move direct), 466 (compare collate), 467 (compare uncollate).

A decode of 460-463 for a pass instruction will generate the NOP signal. NOP enables the RNI sequence.

A decode of 464 for move indirect generates the enable increment sequence signal (EINCS). The increment sequence will use bit positions 30-50 of the instruction word to address (Bj) + K, which will be the address of a 60-bit descriptor word. On receipt of the descriptor word from memory, the accept sequence will generate a GO464 signal to initiate the instruction decode sequence once again.

GOCMU for a 465, 466 or 467, or GO464, and the character length value not equal zero will generate the enable start sequence signal (ESTAHL).

MOVE INSTRUCTION (464, 465 - Refer to timing diagram, figure 5-2-31)

During the instruction decode and start sequence for a move, the following will occur:

1. C2 offset plus character length value in LC are added in L adder. Result returned to LC.
2. C1 is subtracted from C2 (by complement addition); the result is stored in the shift count register (SCR). The output of the C adder is monitored by the $C2 \geq C1$ FF located on the JL module. If $C2 \geq C1$, a carry signal (CADDCC) will enable setting the $C2 \geq C1$ FF.
3. C1 and C2 are tested for out of range condition by the I40 decode circuit. The C12AOR signal will be generated if $C1 \geq 10_{10}$ or $C2 \geq 10_{10}$. C12AOR will set the C1/C2 AOR FF located on the JL module.

4. If $C1 > C2$ (that is $C2 \geq C1$ FF reset), $+12_8$ is added to the shift count value in the SCR register.
5. The shift count value in SCR is gated to a times-six decoder circuit, then to I19, I9 and the SK register.
6. C1 offset plus character length in LA are added in L adder. Result returned to LA.
7. SCR register shift count value is gated to the I40 decoder and stored in the partial write register for use by the data sequences.
8. -12_8 generated at I44 is stored in the LE register for subtraction during the address sequences.

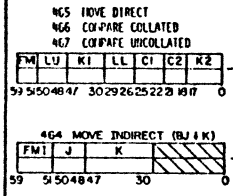
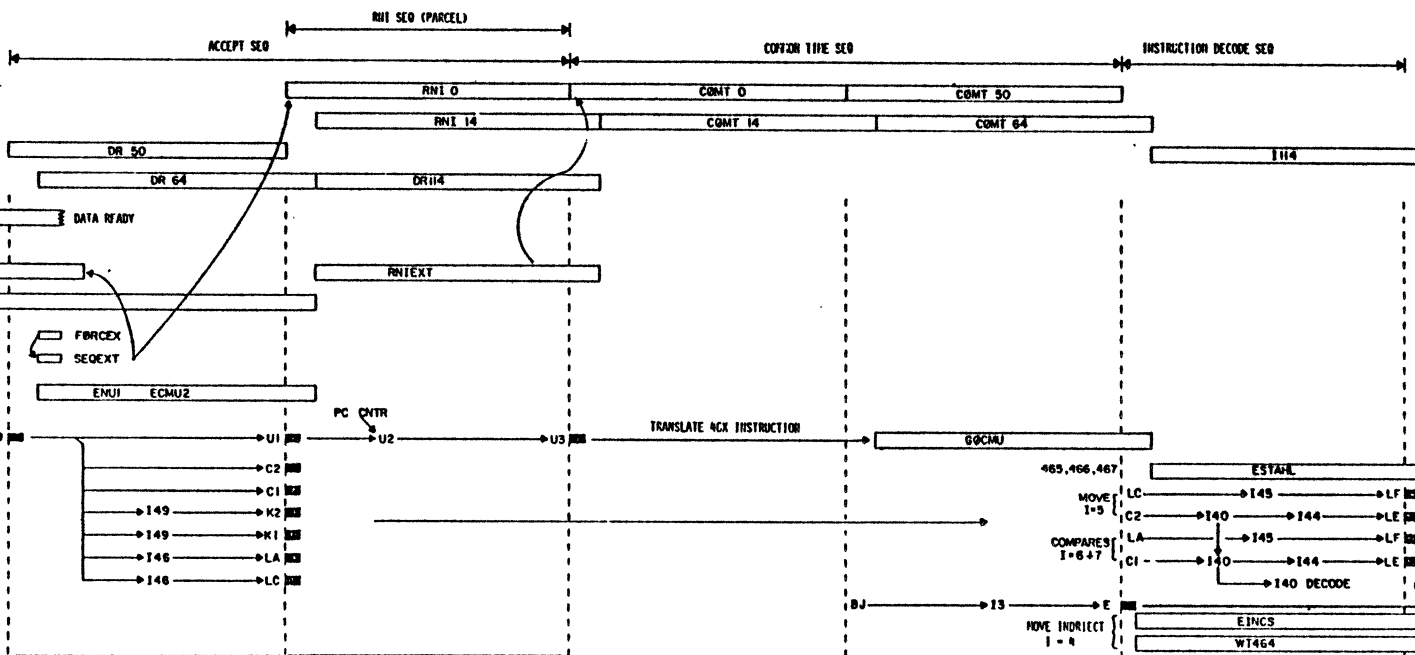
COMPARE INSTRUCTION (466, 467 - Refer to timing diagram, figure 5-2-31)

The instruction decode and start sequence is similar to that of a move instruction except for the following:

1. The addition of C1 + LA occurs before the addition of C2 + LC. This is because the address sequence addresses the K2 word first for a move and K1 word first for a compare.
2. If $C1 > C2$, as determined by step 2 above, the sequence will not add $+12_8$ to the SCR register (as in step 4); it will, however, subtract C2 from C1 to obtain a new shift count value in SCR.

The start sequence initiates the address sequence by generating the clear block K address FF signal (CBKOCF). The address sequence operates in parallel with the start sequence starting at S164 time.

D
C
B
A



TERM NAME	COMMAND OR FUNCTION	DPD NO	TERM NAME	COMMAND OR FUNCTION	DPD NO
DR50	(INITST:RNI11)	16	RNI TO		14
FORCEX	FORCE EXIT	14	U23	U2 → U3	1
SEQEXT	SEQUENCE EXIT		RNI T14	(RNI FF SET BY PREVIOUS RNI SEQ)	14
DR64	(NRVT11)	16	RNIEXT	ENABLES COMMON TIME SEQ	17
ENUI	CR9 → U1	1			
ECHUR		16			
ECHIC2	CR9 18-25 → C1, C2	30			
G400		32			
G46Z	CR9 26-29 → 146	31			
	48-50				
G46Z	CR9 30-47 → 149	3			
	0-17				
ECHUR		16			
ELAOCF		16			
ELKOCF		16			
ELA	146 0-15 → LA, LC	31			
ELC		31			
ECHUR		16			
EK1OCF		16			
EK2OCF		16			
EK1	149 30-47 → K1	3			
EK2	149 0-17 → K2	3			

TERM NAME	COMMAND OR FUNCTION	DPD NO
COM50		15
SELBJ	ADRS BJ RGR	2
COM64	46X	15
COM13		15
SEL13	BJ → 13	12
NC050		15
ENABE	13 → E	13

TERM NAME	COMMAND OR FUNCTION	DPD NO
1114		32
LASHV	(1-6-7-COMPARE)	37
G450		31
G450-G451	LA → 145	31
G450-G451	LC → 145	31
1114		32
ELFIAP		31
ELF	145 → LF	31
C140HL	(1-6-7-COMPARE)	32
G401		32
C240HL	(1-5-MOVE)	32
G400		30
G400-G401	C2 → 140	30
G400-G401	C1 → 140	30
1114		32
4044HF		30
G441		30
G440-G441	140 → 144	30
1114		32
ELEIAP		31
ELE	144 → LE	31
ESTABL	EN START SEQ (PC000-LP0-465+466+467)	32

FOR 465 466 467 INSTRUCTIONS SEQUENCE WILL CONTINUE WITH START SEQ

(FIELD LENGTH IN CHARACTERS)

(CHARACTER POSITION OFFSET FOR K2)

(FIELD LENGTH IN CHARACTERS)

(CHARACTER POSITION OFFSET FOR K1)

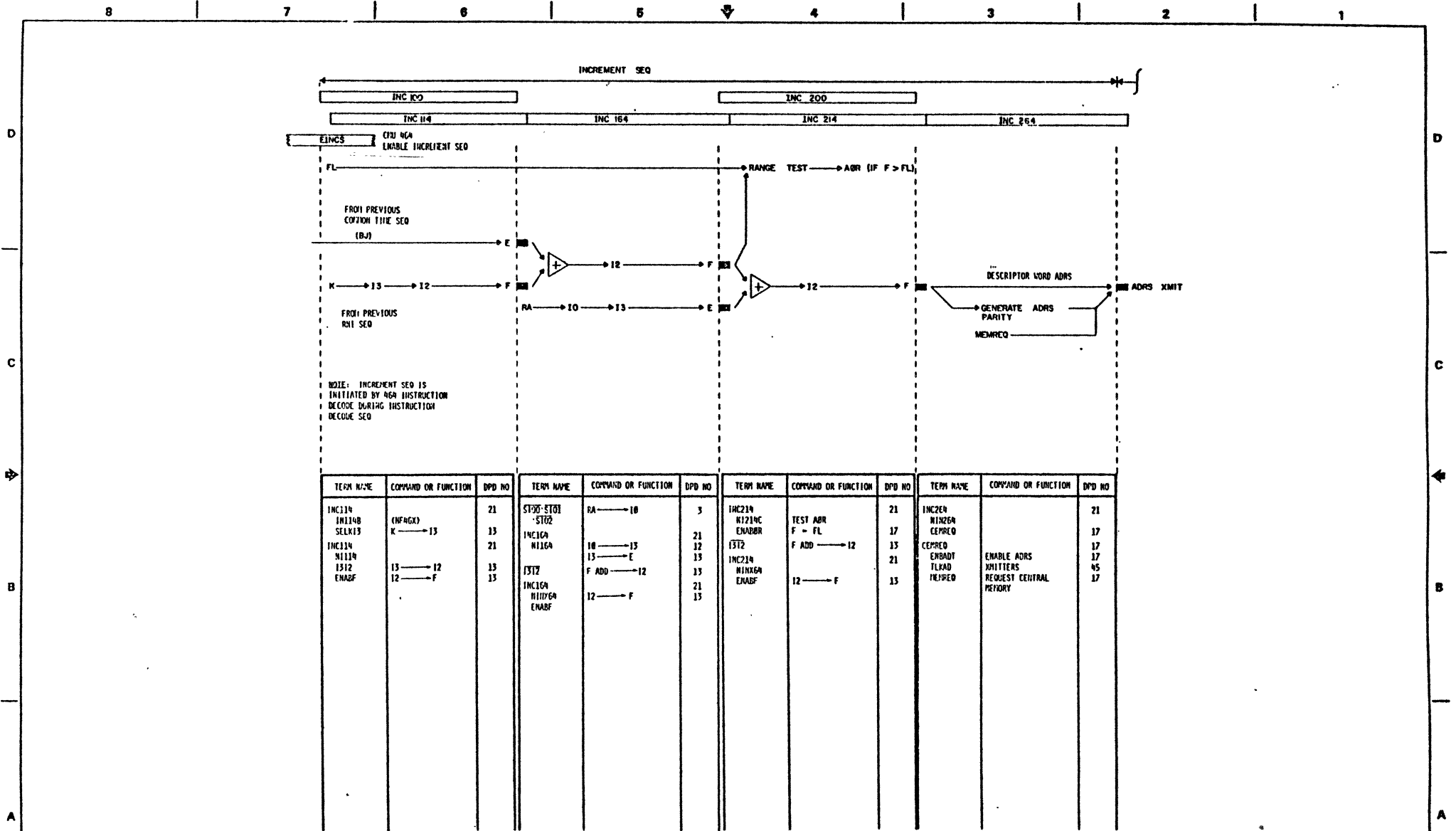
(IF C1 OR C2 ≥ 10)

FOR 464 INSTRUCTION SEQUENCE WILL CONTINUE WITH INCREMENT SEQ

0JJ	C12ABR	TEST 140Z10	30
		ENABLE ABR INHIBIT START SEQ T16A	32
EINC5	WT464	(GOCMU-1-4) SET WT66A FF	32

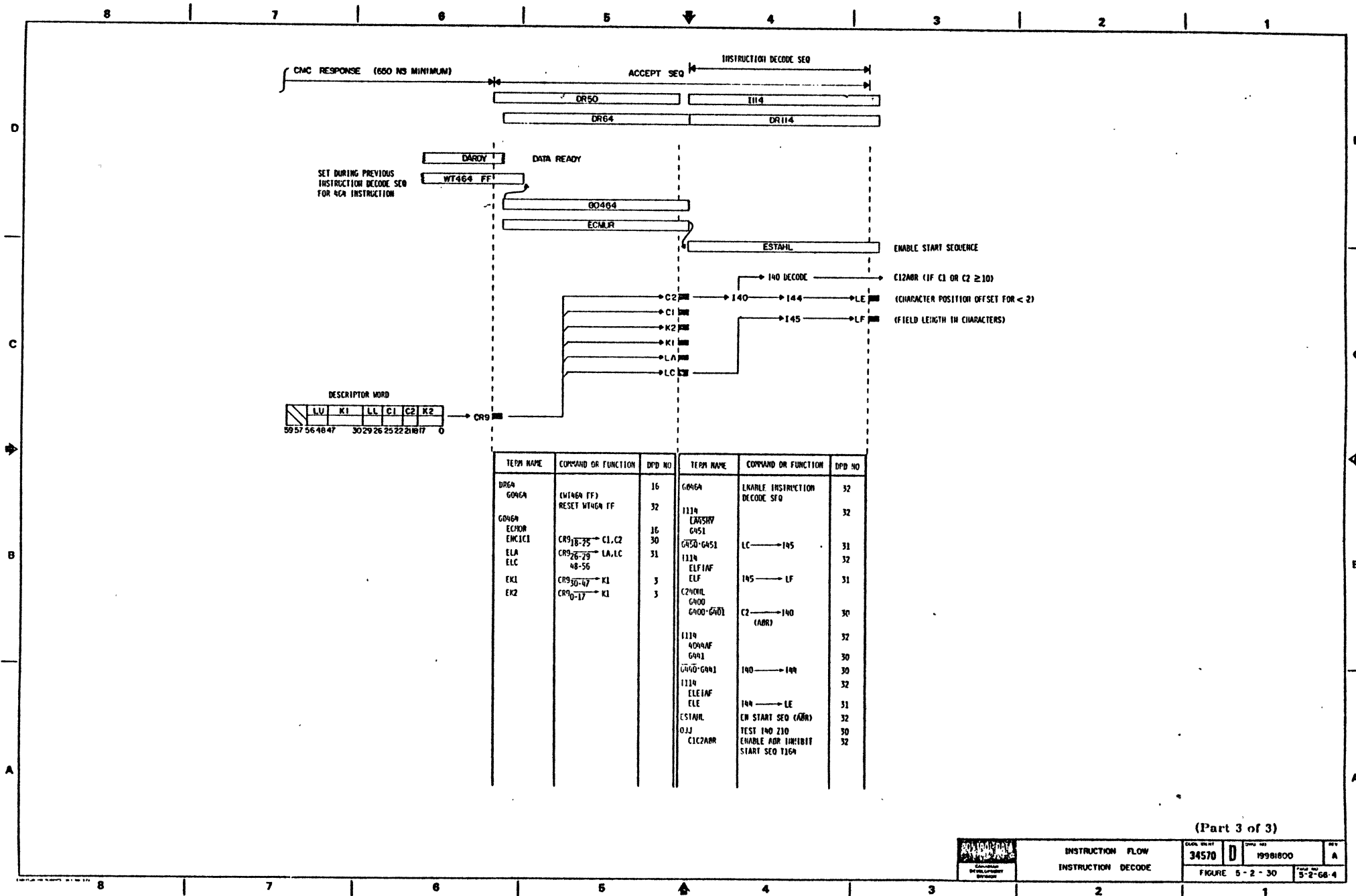
CONTINUED

(Part 1 of 3)



TERM NAME	COMMAND OR FUNCTION	DPD NO	TERM NAME	COMMAND OR FUNCTION	DPD NO	TERM NAME	COMMAND OR FUNCTION	DPD NO	TERM NAME	COMMAND OR FUNCTION	DPD NO
INC114	(NFRGX)	21	ST00-ST01	RA → 10	3	INC214	TEST ADR	21	INC264		21
INI14B			ST02			R1214C	F = FL	17	INX264		17
SELK13	K → 13	13	INC114	10 → 13	21	ENABR			CEPREQ		17
INC114		21	NI164	13 → E	13	ST2	F ADD → 12	13	CEPREQ	ENABLE ADRS	17
NI114						INC214		21	ENBAD	XMITTERS	45
IS12	13 → 12	13	ST2	F ADD → 12	13	NINX64			TLKAD	REQUEST CENTRAL	17
ENABF	12 → F	13	INC164		21	ENABF	12 → F	13	TRJREQ	MEMORY	17
			NI1064	12 → F	13						

(Part 2 of 3)



(Part 3 of 3)

DETAILED PAK DIAGRAM (CPU 3.33, 3.34, 3.35, 3.36, 3.37)

ADDRESS SEQUENCE

Initially, the address sequence is enabled by the start sequence clearing the block K address FF located on the JS module (CPU 2.34). The K1 address FF located on the JR module (CPU 3.34) is set during the start sequence by SK1ALR for a compare instruction, or reset during the start sequence by CK1ALR for a move instruction. Also, the 1st address FF located on the JS module will be set by CMU master clear (CMUCC) which is activated at the beginning and end of every CMU instruction.

MOVE INSTRUCTION (464, 465 - Refer to timing diagram, figure 5-2-32)

1st ADDRESS

1. The contents of the K2 register are loaded in the F register of the small adder. RA is loaded into E. The resultant relative address from the small adder is stored in the F register and transmitted to central memory control.

The transmission of a K address and memory request to central memory control is enabled by address sequence K264. The K264 timing chain is enabled only by specific conditions to ensure a valid address is being transmitted. 1st address FF enables K264 and, assuming the address transmitter register is not full (indicated by ADRS XMIT FF reset), the enable address transmit signal (EATOR) will be generated.

2. During 1st address, the length in LC will be decremented by -12_8 in the L adder. This is performed to test for a K2 exhaust on the first word. If K2 has exhausted (that is, $LC \leq 12$), the K2 exhaust FF located on the JQ module (CPU 3.36) will be set, and the 1st & last FF will be set. The 1st & last FF will enable the short data sequence. On the 1st address, the decremented length count will not be transferred to the LC register. LC will remain at its original value.
3. The data counter located on the HT module (CPU 3.39) is incremented by one. The increment is enabled by the update data counter signal (UPDKPJ) from the JP module (CPU 3.33). The counter contains a count representing the number of words requested from memory. As each word is received, the count is decremented by one.

4. Clear 1st address FF, set 2nd address FF, clear K1 address FF (CPU 3.35).

2nd ADDRESS

1. With the K1 address FF set, K1 will be addressed in a manner similar to step 1 above.

The K264 address sequence will not be enabled until central memory control has generated an accept for the previous memory request. The accept will reset the ADRS XMIT FULL FF (ATFNSR) located on the JS module (CPU 3.35).

2. The length in LA will be decremented by -12_8 in the L adder. The group carry bit (LADDG) from the L adder carry look-ahead network is monitored on the JQ module (CPU 3.36). Absence of a carry indicates an exhaust condition and will set the K1 exhaust FF. If $LA > 12_8$, the enable LA signal ELA0QF will allow the decremented count to be stored back in the LA register.
3. Increment data counter - described in step 3 above.
4. The buffer counter located on the JV module (CPU 3.37) is incremented by one. The increment is enabled by the update buffer counter signal (UPBKPV) from the JP module (CPU 3.36). The counter contains a count representing the number of K1 words requested from memory. For every word written into K2, the buffer counter will be decremented by one.
5. Clear 2nd address FF, set 1st write FF located on JS module (CPU 3.35).

The K1 address FF will determine whether the address sequence is to perform additional K1 read requests, or formulate the first K2 write address. The K1 address FF, located on the JR module (CPU 3.34), will remain set until buffer counter reaches a count of 5, or the K1 address has been exhausted. At that point, the K1 address FF is reset and, with the 1st write FF set, the address sequence will prepare the K2 address.

1st WRITE. K1 ADRS. 1st ADRS. 2nd ADRS

1. With the K1 address FF set, the contents of the K1 register are gated to the E register, +1 is forced to the F register by the ITOFN signal generated from the JP module (CPU 3.33). The result K1 + 1 is returned to the K1 register by the enable K1 signal EK1NRC. RA is added to the contents of F and the resultant K1 address is transmitted to central memory control.

The remainder of the sequence will be the same as steps 2, 3 and 4 of 2nd address described above.

1st WRITE. K1 ADRS. 1st ADRS. 2nd ADRS

1. With the K1 address FF reset, K2 is gated to F, RA is gated to E. The result from the small adder produces a relative memory address for the first K2 word.

The transmission of the K2 address and a memory write request will not occur until the HR register is loaded with a data word to be written into memory. Loading of the HR register is controlled by the data sequence. With the HR register loaded, the HR full FF is set to enable K264 of the address sequence. K264, in turn, enables address transmit (EATOR) and data transmit by generating EDTOS from the JS module (CPU 3.35).

2. The length in LC will be decremented by -12_8 in the L adder. This is performed to test for K2 exhaust. If $LC \geq 12_8$, the enable LC signal ELCOQF generated from the JQ module will allow the decremented count to be stored back in the LC register.
3. The sequence then performs a double subtraction that monitors whether the next K2 sequence will exhaust the length in LC. The look-ahead function allows the address sequence to read the last K2 word before performing the last K2 write. The decremented contents of LC from the L adder are enabled to the LF register via I45, and thus the second subtraction is performed. The absence of a group carry or pass (\overline{LADDG} , \overline{LADDP}) from the L adder indicates $L < 12$; if K1 has already been exhausted, the LAC < 12 FF will be set.

Setting of the LAC < 12 FF enables setting the K1 address FF. The address sequence will thus perform a K1 read sequence on the next cycle to obtain the last word of K2. Since K1 must have exhausted in order to set LAC < 12, the current K2 address used to produce the memory address in step 1 is stored in K1 by the EK1NRC signal on the JR module. Thus both K1 and K2 will contain the same K2 address.

4. Reset 1st write FF

If the K1 exhaust FF is not set, the address sequence will toggle between K1 read and K2 write until K1 has exhausted. The address sequences for K1 and K2 are identical to those already described but with two exceptions, (a) and (b), listed below.

1st ADRS. 2nd ADRS. K1 ADRS

- (a) K1 is gated to E, +1 is forced to F, result K1 + 1 returned to K1.
- (b) Data counter and buffer counter are incremented by one.

Once K1 has exhausted, the address sequence will perform K2 write until the LAC < 12 FF is set, at which time the last K2 read is performed.

LAC < 12 FF. K1 ADRS (Last K2 READ)

1. With the K1 address FF set, the contents of the K1 register (K1 will contain the previous K2 address) are gated to the E register, +1 is forced to the F register. The result from the small adder is added to RA to produce a relative memory address for the last K2 word. This address is transmitted to central memory control.
2. The data counter is incremented by one. The increment is enabled by the update data counter signal (UPDKQJ) from the JQ module (CPU 3.36).
3. Reset K1 address FF.

With the last K2 read performed, the address sequence will generate the K2 address for the last K2 write.

LAC < 12. K1 ADRS (Last K2 WRITE)

1. With the K1 address FF reset, the contents of the K2 register are gated to the E register, +1 is forced to the F register. The result from the small adder is added to RA to produce a relative memory address for the last K2 write.

The transmission of the last K2 address and a memory write request will not occur until the HR full FF is set by the data sequence.

COMPARE INSTRUCTION (465, 467 - Refer to timing diagram, figure 5-2-32)

The address sequencing for a compare instruction is similar to that of a move with the following exceptions:

1st ADDRESS

1. K1 is addressed rather than K2, as in a move.
2. LA is decremented by -12_8 . K1 exhaust test is performed. The 1st & last FF cannot be set during 1st address if K1 has exhausted. If $C1 > C2$, the LA value before it is decremented is stored in LAC1 and the previous contents of LAC1 are stored in LAC2.
3. The data counter and buffer counter are incremented by one.

2nd ADDRESS

1. K2 is addressed rather than K1, as in a move.
2. LC is decremented by -12_8 . K2 exhaust test is performed. If K2 has exhausted and K1 exhausted on the previous sequence, the 1st & last FF is set to enable the short data sequence. If $C2 \geq C1$, the LC value before it is decremented is stored in LAC1, and the previous contents of LAC1 are stored in LAC2.
3. The data counter and buffer counter are incremented by one.

With the buffer counter equal to 4, the block K ADRS FF (CPU 3.35) is set. The block K ADRS FF will prevent further address sequences from occurring until the compare sequence has compared the first pair of words. The compare sequence decrements the buffer counter by two and resets the block K ADRS FF.

Address sequencing will continue until K1 and K2 exhaust, or a compare unequal occurs.

COMPARE COLLATE - COLLATE TABLE LOOK-UP - A0 ADRS

The collate sequence will set the A0 ADRS FF and clear the block K ADRS FF, allowing the address sequence to generate the correct table word address.

1. The collate sequence will load the upper bits (3-5) of the TQ or TS register in the E register. The address sequence will load the contents of the A0 address register in the F register. The result from the small adder is added to RA to produce the relative memory address for the desired table word.
2. The address sequence will clear the A0 ADRS FF and set the block K ADRS FF.

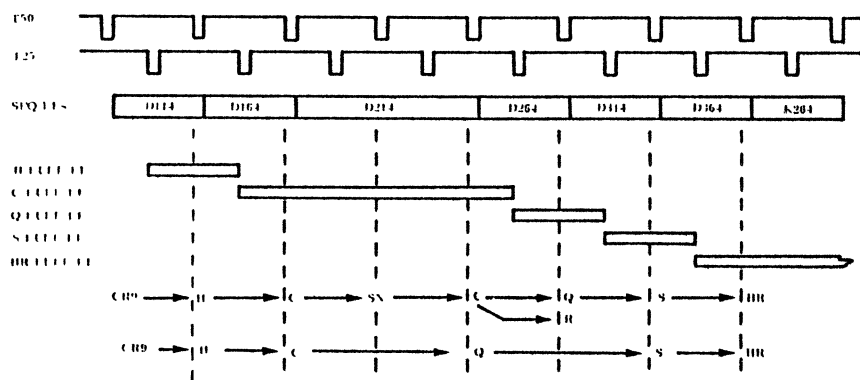
Further address sequencing will be blocked, unless the collate sequence sets the A0 ADRS FF and clears block K ADRS once again.

DETAILED PAK DIAGRAM (CPU 3.38, 3.39, 3.40)

DATA SEQUENCE

Central memory control generates the data ready signal (DARDY), 50 ns before the transmission of requested data. The accept sequence located on the GM module (CPU 3.16) is conditioned by data ready. Data ready starts the accept sequence timing chain: DR50, DR64. The leading edge of DR50 generates central memory data ready (CMDRM) to the data sequence IIT module (CPU 3.39). At the next clock, CMDRJX starts the data sequence timing chain: D164, D214, D264, D314, and D364.

The basic data path flow through the data sequence is shown on the illustration below.



The time interval between the leading edge of CMDRJX and the beginning of D164 is considered as D114.

During time intervals D114 through D364, data in CR9 can be propagated in succession through five registers (H, C, Q, S, and HR) with the loading of each controlled independently by the data sequence. The full conditions of these registers are

monitored by five control FFs: H full, C full, Q full, S full, and HR full, located on the IIT and JW modules (CPU 3.39; 3.38). Each control FF is enabled by a special 25 ns clock that occurs half way between the regular clock. A full condition alerts the next sequence interval to enable continuance of data propagation.

Normally, D214 allows for the realignment of character positions in C. Realignment is performed by right shifting the desired number of characters through the shift network and returning the shifted results to the C register. The C full FF remains set while the shift is taking place. When a realignment of data in C is not required, D214 stores the unshifted data from C directly into Q. Depending on sequence conditions, Q full may be set at this point.

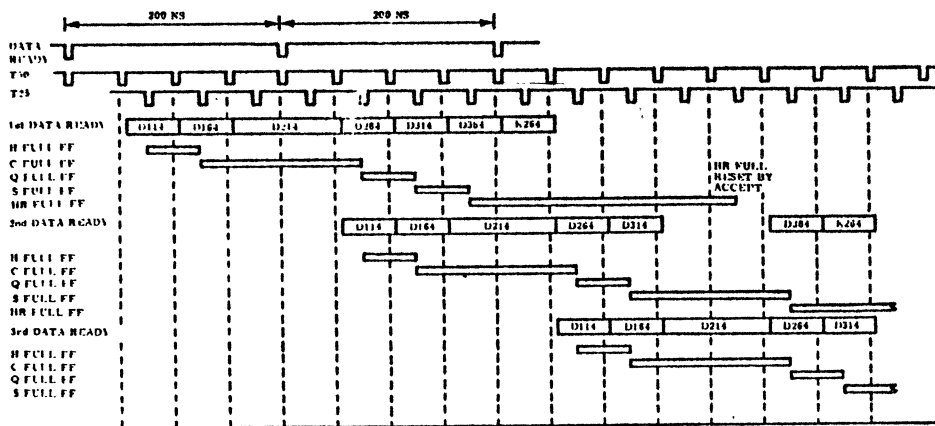
Normally, D264 allows for the storage of shifted data from C (bits 48-107) into Q. Any shift residue characters, (those shifted past the tenth character position, bit 48), are transferred into R for temporary storage.

Similarly, D314 allows for the storage of data from Q into S; D364 allows for the storage of data from S into HR. HR full alerts address sequence K264 to initiate a memory write request. Prior to the generation of HR full, the address sequence will already have placed a K2 address into F.

To prevent writing, a compare instruction blocks D 364 altogether.

Consecutive data transmission to central memory control is dependent on the receipt of a signal that acknowledges memory acceptance of the previous write data. Acceptance of a write request which is unduly delayed causes stacking of data in the S, Q, C and H registers. It is specifically because of this stacking capability that the address sequence monitors the buffer counter to ensure that only five K1 address requests ($C2 \geq C1$), or six K1 address requests ($C1 > C2$) can be issued prior to a write.

An example of data stacking is illustrated below.



Three data ready responses are received at their maximum rate of 200 ns. Each data ready initiates a data sequence.

The first data sequence proceeds through D364, where HR full is set. HR full enables address sequence K264, and it is during this time that first write request occurs. The second data sequence starts 200 ns after the first; it proceeds through D314 only. Since HR is full, D364 cannot be enabled. The second data word remains stacked in S with the S full FF set. The third data sequence starts 200 ns after the second; it proceeds through D214. By D214 time, HR full is reset by an accept for the first write request. At the next clock, D364 is enabled for the second word while D264 is enabled for the third. The second and third words are simultaneously transferred from S and C into HR and Q, respectively. HR full and Q full enable K264 and D314 at the next clock. K264 initiates a memory write request for the second word while D314 allows for the transfer of data from Q into S. The third data word will remain stacked in S until the second write accept is received.

MOVE INSTRUCTION (464, 465 - refer to Figure 5-2-33)

Three data detection FFs are utilized by the data sequence to determine path selection. The three FFs, 1st data, 2nd data and 3rd data are located on the HW module (CPU 3.40).

1st data sets at the beginning of a CMU instruction; it enables sequence path selection for receipt of the first word. 2nd data is set at the end of 1st data; it enables sequence path selection for receipt of the second word. Two paths are provided for 2nd data; the path chosen is dependent on the $C2 \geq C1$ FF. 3rd data sets at the end of 2nd data when $C1 > C2$; it enables sequence path selection for receipt of the third word.

1st DATA

Receipt of the first data ready response initiates the first data sequence. The first word received will be the first word of the K2 destination field.

1. Data ready allows the data counter (CPU 3.39) to be decremented by one.
2. The first K2 word is transferred into Q, where it will remain until second data. The entire Q register is enabled because CSR contained zero from the start sequence.
3. Clear 1st data, set 2nd data.

The data sequence is now conditioned for receipt of the second data word. The next data ready will initiate the second data sequence.

2nd DATA

The second word received will be the first word of the K1 source field. The path chosen for 2nd data is dependent on the $C2 \geq C1$ FF. Each path is described separately.

$C2 \geq C1$

With the $C2$ offset greater than, or equal to the $C1$ offset, the first K1 word is shifted to align the first actual character position of K1 with K2. The shifted K1 data from C is transferred into Q, where it combines with the K2 offset stored in Q from 1st data. The shift residue from C is transferred into R for temporary storage until the next sequence. The complete word in Q becomes the first K2 write data; it is transferred into S and HR, at which time the first write request is performed.

1. Data ready allows the data counter to be decremented by one (CPU 3.39).
2. The $C2$ offset in CSR controls loading Q, so that the K2 offset is protected while the shifted K1 occupies the remaining character positions of Q.
3. Clear 2nd data.

All subsequent data responses use the normal path (1st DATA, 2nd DATA, 3rd DATA) until last word is detected.

C1 > C2

With the C1 offset greater than the C2 offset, the first K1 word is shifted to align the first actual character position of K1 with K2. Since a left shift cannot be performed, the shift will cause all characters to reside in the residue portion of C. The residue from C is transferred into R, where it will remain until the next sequence.

1. Data ready allows the data counter to be decremented by one (CPU 3, 39).
2. Clear 2nd data, set 3rd data.

The data sequence is now conditioned for receipt of the third data word. The first K1 word that was aligned with K2 remains in the R register until the next K1 word is received.

3rd DATA

While the second K1 word is being shifted to align K1 with K2, the first residue is transferred from R into Q, where it combines with the K2 offset stored in Q from 1st data. The shifted second K1 word in C is then transferred into Q to occupy its remaining character positions. The shift residue from C is transferred into R for storage until the next sequence. The complete word in Q becomes the first K2 write data, and is transferred into S and IIR, at which time the first write request is performed.

1. Data ready allows the data counter to be decremented by one (CPU 3, 39).
2. The C2 offset in CSR controls the loading of Q, so that the K2 offset is protected while the first K1 residue from R is transferred into Q.
3. The shift count in PW controls the loading of Q, so that the K2 offset and K1 residue previously stored in Q are protected while the second K1 word occupies the remainder of Q.
4. Clear 3rd data.

All subsequent data ready responses use the normal path (1st DATA . 2nd DATA . 3rd DATA), until last word is detected.

1st DATA . 2nd DATA . 3rd DATA (Normal Path)

The normal path operation is similar to 3rd data with the exception that the CSR register will contain zero instead of an offset value.

The data counter is decremented by one for each data ready. When the count equals zero and the LAC < 12 FF is set, the data word in CR9 is the last K2 word. The data path chosen is dependent on the remaining buffer count value.

DT=0 . BF=1 . LAC < 12FF

A data count of zero and a buffer count of one indicate that the block HR FF must have been set on the previous sequence to prevent writing the last K2 word until the partial write characters from K2 are read. (Examples of this condition are illustrated in figures 5-2-34 and 5-2-36.)

The remaining length value in LC determines how many partial write characters from K2 must be returned to K2.

1. The remaining length value from LC is transferred into PW where it controls loading the partial write characters into Q.

DT=0 . BF=0 . LAC < 12 FF

A data count of zero and a buffer count of one indicate that the block HR FF was not previously set. Residue characters from the previous sequence stored in R are transferred into Q. The partial write characters from K2 are transferred into Q. (An example of this condition is illustrated in figure 5-2-35.)

1. The remaining length value from LC is transferred into PW and is used to control the loading of the partial write characters into Q.

COMPARE INSTRUCTION (466, 467 - Refer to figure 5-2-33)

A compare instruction sets the block HR FF (CPU 3, 38); it remains set throughout the instruction to block D364.

The toggle FF located on the HW module (CPU 3, 40) is used by compare to select the appropriate data path for K1 and K2. In its reset state, the toggle FF selects the K1 data path. K1 is always stored in S. When set, the toggle FF selects the K2 data path. K2 is always stored in Q.

The 1st data flip-flop is the only data detection FF utilized by compare. Path selections are determined by the condition of 1st data, $C2 \geq C1$, toggle and last compare.

The first word received (of a pair) will always be a K1 word. The 1st data FF is set at the beginning of a CMU instruction; it enables sequence path selection for the first word received. Four paths are provided for 1st data: ($C2 \geq C1$. TOGGLE), ($C2 \geq C1$. TOGGLE), ($C1 > C2$. TOGGLE), and ($C1 > C2$. TOGGLE).

1st DATA . C2 ≥ C1 . TOGGLE

The first K1 word received is shifted to align K1 with K2. The shift residue is transferred into the R register for storage until the next sequence. The shifted K1 data from C is transferred into Q and S.

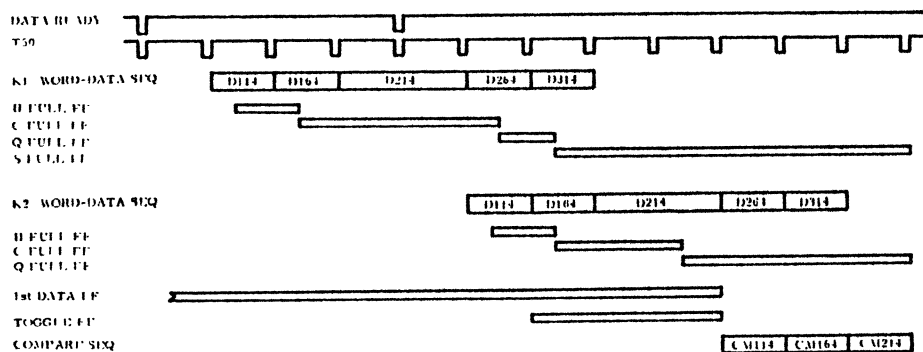
1. Data ready allows the data counter to be decremented by one (CPU 3.39).
2. The C2 offset in the CSR register prevents loading the C2 offset positions of Q.
3. The toggle FF is set; the 1st data FF remains set until receipt of the first K2 word.

1st DATA . C2 ≥ C1 . TOGGLE

The first K2 word received is transferred directly into Q without shifting, since K1 was already shifted to align with K2.

1. Data ready allows the data counter to be decremented by one (CPU 3.39).
2. The C2 offset in the CSR register prevents loading the C2 offset positions of Q.
3. Reset 1st data FF and toggle FF.
4. Enable compare sequence.

A representative timing diagram for 1st data is shown below.



All subsequent data ready responses will use the normal compare data path, (1st DATA . LAST COMPARE), until last compare is detected.

1st DATA . C1 > C2

With $C1 > C2$, the paths for 1st data are similar to those previously described for $C2 \geq C1$, with the following exceptions:

1. Rather than K1 being shifted to align with K2, the opposite is performed; K2 is shifted to align with K1.
2. Because of this change, the loading of Q is controlled by C1 instead of C2.

All subsequent data ready responses will use the normal data path (1st DATA . LAST COMPARE), until last compare is detected.

1st DATA . LAST COMPARE (Normal Path)

The normal paths for compare are similar to the 1st data paths previously described. However, the difference between 1st data and the normal path is that the residue in R from the previous sequence is transferred into Q while the shift is being performed for K1 ($C2 \geq C1$) or K2 ($C1 > C2$). The shifted characters of K1 or K2 are then transferred into Q to combine with the residue, forming a complete word. The current shift residue is transferred into R for storage until the next sequence.

1. The CSR register will always contain zero so that the residue from R can be loaded into the entire Q register.
2. The PW register, which contains the shift count, ensures that only the shifted contents of K1 ($C2 \geq C1$) or K2 ($C1 > C2$) are transferred into Q while the previous residue, (step 1), is protected.

The normal path is used for receipt of data until the last compare FF is set. Last compare is set during the compare sequence when K1 and K2 have been exhausted and the compare sequence determines that the second last pair of words are equal. Data path selection for last compare is determined by the condition of the $C2 \geq C1$ FF, toggle FF and the remaining buffer count.

LAST COMPARE . BF=2

The buffer count of two with the last compare FF set, indicates that one remaining pair of words must be received and compared before the instruction is completed. (An example of this condition is illustrated in figure 5-2-37.)

The data sequence is similar to a normal path except that:

1. CSR contains the remaining length from LA ($C1 > C2$) or LC ($C2 \geq C1$); CSR prevents loading Q with characters that are not part of the K1 or K2 last word field.
2. CSR \neq PW ensures that only the shifted K1 ($C2 \geq C1$) or K2 ($C1 > C2$) characters are transferred into Q while the previous residue is protected, and characters not part of the K1 or K2 field are blocked.

LAST COMPARE , BF=1

A buffer count of one with the last compare FF set, indicates that only one remaining word must be received. (An example of this condition is illustrated in figure 5-2-38.)

With $C2 \geq C1$, the remaining word will be from K2; whereas with $C1 > C2$, the remaining word will be from K1. CSR will contain the remaining length from LA ($C1 > C2$), or LC ($C2 \geq C1$); CSR prevents loading Q with characters that are not part of the K1 or K2 last word field.

COLLATE TABLE LOOK-UP

The data sequence is also used during the compare collate operation to store the collate table word into the T register. The appropriate collate character is selected via I34 and transferred to either the TS or TQ register.

CENTRAL MEMORY CONTROL - ACCEPT RESPONSE

Central memory control responds to a read or write request by generating an accept signal (CMACM) when the request is honored. CMACM sets the CMC accept FF (CPU 3.35) and clears the ADRS XMIT full FF (at full). The reset condition of at full allows initiation of another address sequence, unless the block K ADRS FF is set.

Buffer Counter

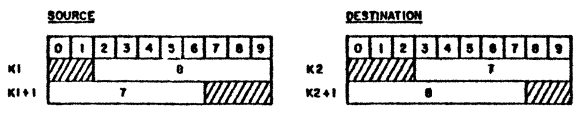
The CMC accept FF for a write operation (K1 ADRS FF) decrements the buffer counter by one. K1 ADRS XMIT FF, located on the JR module (CPU 3.34), prevents decrementing the buffer counter on a read accept. The buffer counter and decrement controls are located on the JV module (CPU 3.37).

Block IIR Controls

When the address sequence detects that the next K2 field length will exhaust (indicated by $LAC < 12$ FF) with a count of two in the buffer counter, the block IIR FF will be set. $LAC < 12$ and a buffer count of two indicate that the last K1 word and last K2 word must be received before the last K2 write is performed.

The block IIR FF is always set at the beginning of a compare instruction, and remains set throughout its execution.

When set, the block HR FF blocks data sequence D364.



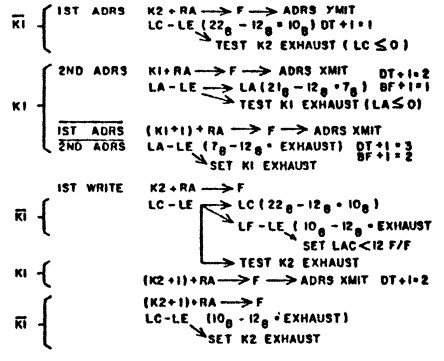
INSTRUCTION DECODE SEQ

LC → LF
C2 → LE

START SEQ

LE + LF → LC + 22₀ (L + C2)
C1 - C2 → SCR + 1 SCR → SK
0 → CSR
SCR → PW + 1
LA - C1 → LA + 21₀ (L + C1)
-12₀ → LE

ADDRESS SEQ



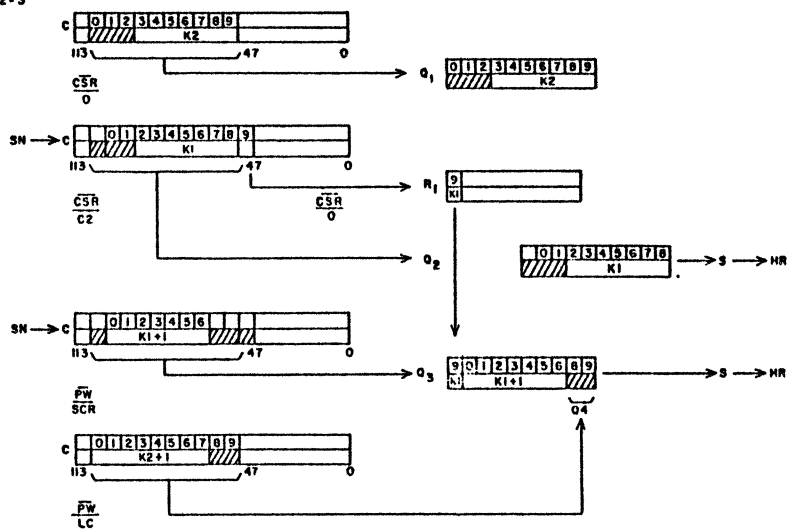
SEQ WILL WAIT UNTIL HR FULL BEFORE ADRS XMIT

SEQ WILL WAIT UNTIL HR FULL BEFORE ADRS XMIT

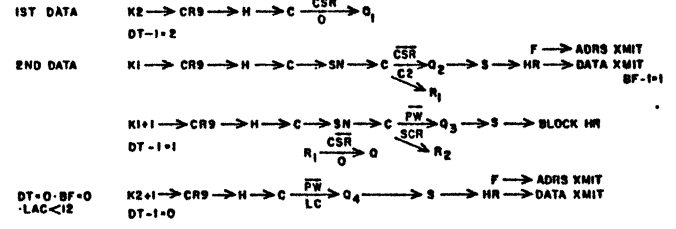
NOTES · DT · DATA COUNTER
BF · BUFFER COUNTER

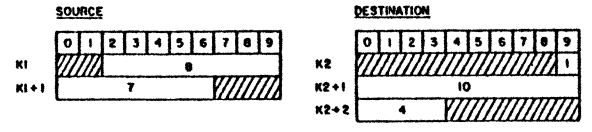
MOVE EXAMPLE NO.1 LONG MOVE C2 > C1

L = 17₀ = 15₁₀
C1 = 2
C2 = 3



DATA SEQ





INSTRUCTION DECODE SEQ
 LC → LF
 C2 → LE

START SEQ
 LE + LF → LC + 30₀ (L + C2)
 C1 - C2 → SCR + 7 SCR → SN
 0 → CSR
 SCR → PW + 7
 LA + C1 → LA + 21₀ (L + C1)
 -12₀ → LE

ADDRESS SEQ

KI

1ST ADRS K2 + RA → F → ADRS XMIT
 LC - LE (30₀ - 12₀ - 16₀) DT + 1 = 1
 TEST K2 EXHAUST (LC ≤ 0)

2ND ADRS K1 + RA → F → ADRS XMIT DT + 1 = 2
 LA - LE → LA (21₀ - 12₀ - 7₀) BF + 1 = 1
 TEST K1 EXHAUST (LA ≤ 0)

1ST ADRS (KI + 1) + RA → F → ADRS XMIT
 2ND ADRS LA - LE (7₀ - 12₀ - EXHAUST) DT + 1 = 3
 SET K1 EXHAUST BF + 1 = 2

KI

1ST WRITE K2 + RA → F
 LC - LE → LC (30₀ - 12₀ - 16₀)
 LF - LE (16₀ - 12₀ - 4)
 TEST ≤ 12 (LF < 12)
 TEST K2 EXHAUST

(K2 + 1) + RA → F
 LC - LE → LC (16₀ - 12₀ - 4)
 LF - LE (4 - 12₀ - EXHAUST)
 SET LAC < 12 F/F
 TEST K2 EXHAUST

KI

(K2 + 2) + RA → F → ADRS XMIT DT + 1 = 2
 (K2 + 2) + RA → F
 LC - LE (4 - 12₀ - EXHAUST)
 SET K2 EXHAUST

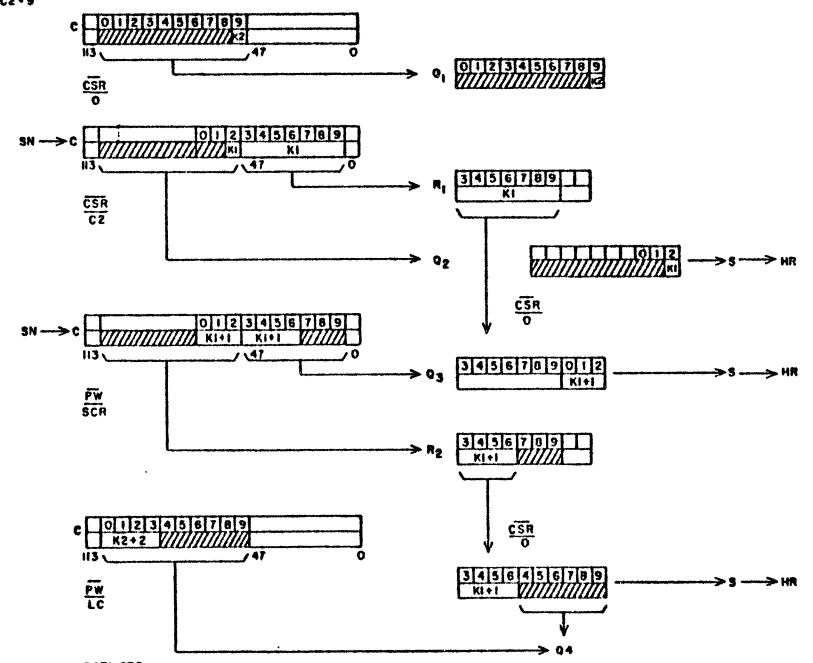
SEQ WILL WAIT UNTIL
HR FULL BEFORE
ADRS XMIT

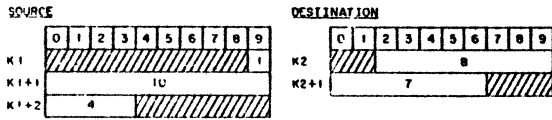
SEQ WILL WAIT UNTIL
HR FULL BEFORE
ADRS XMIT

SEQ WILL WAIT UNTIL
HR FULL BEFORE
ADRS XMIT

MOVE EXAMPLE NO 2 LONG MOVE C2 > C1

L = 17₀ - 15₁₀
 C1 = 2
 C2 = 9

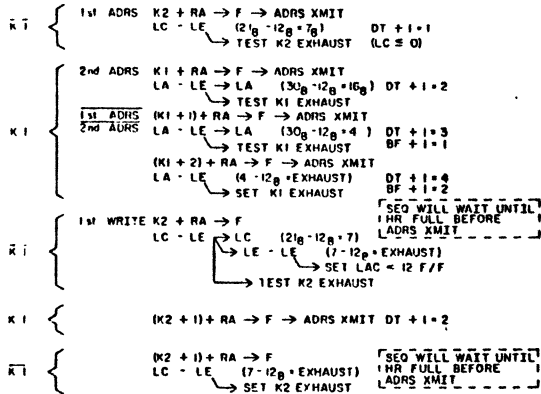




INSTRUCTION DECODE SEQ
 LC → LF
 C2 → LE

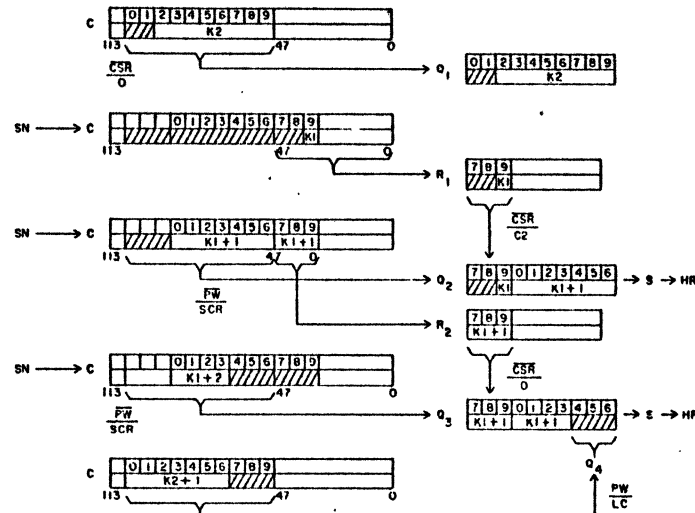
START_SEQ
 LE + LF → LC + 2₁₀ (L + C2)
 (C2 - C1) + 12₈ → SCR = 3₈ SCR → SK
 D → CSR
 SCR → PW = 3₈
 LA + C1 → LA + 30₈
 -12₈ → LE

ADDRESS_SEQ

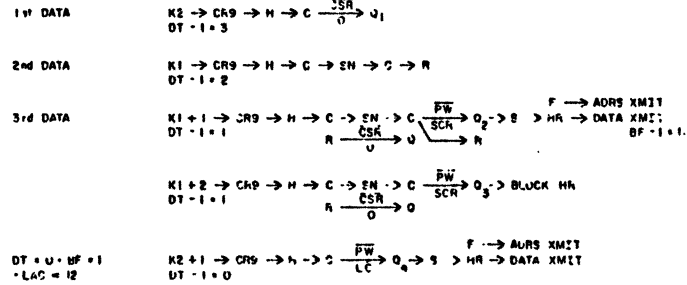


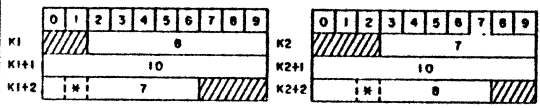
MOVE
 L = 15₁₀
 C1 = 9
 C2 = 2

EXAMPLE #3 LONG MOVE C1 - C2



DATA_SEQ





* NOTE: THIS EXAMPLE ASSUMES INDICATED CHARACTER OF K1+2 AND K2+2 TO BE UNEQUAL

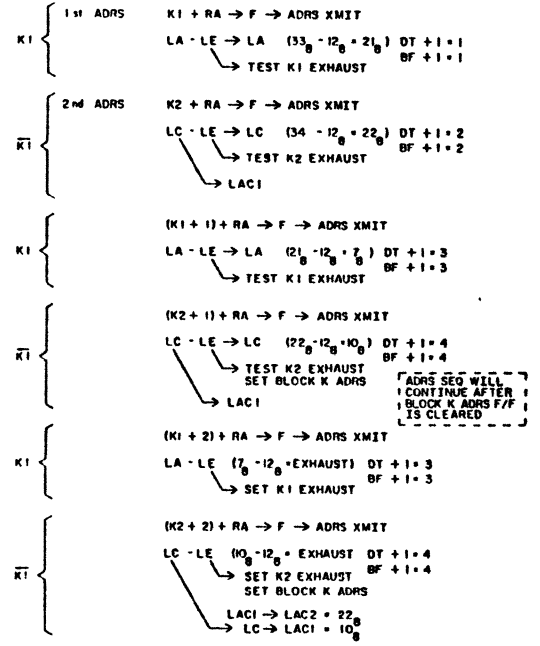
INSTRUCTION DECODE SEQ

LA → LF
C1 → LE

START SEQ

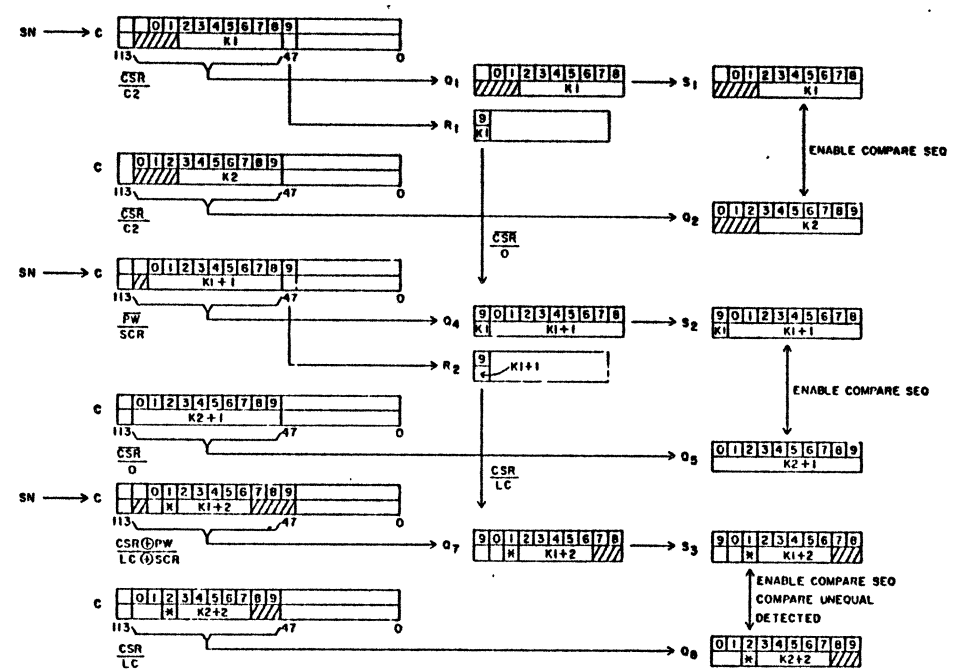
LE + LF → LA + 3₀ (L + C1)
C2 - C1 → SCR + 1 SCR → SK
0 → CSR
SCR → PW + 1
LC + C2 → LC + 3₀ (L + C2)
- 12₀ → LE

ADDRESS SEQ

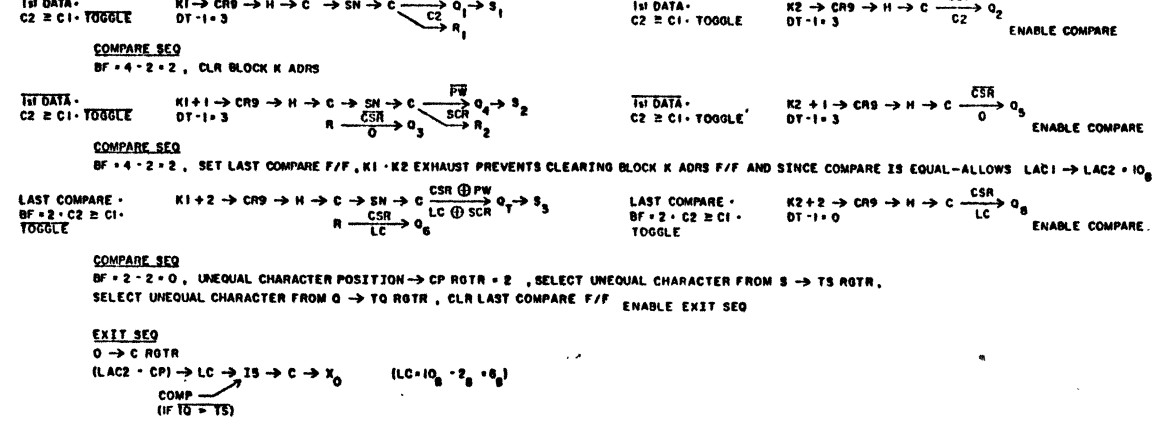


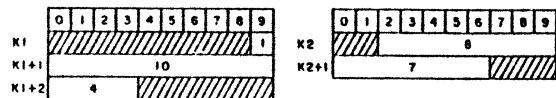
COMPARE
L = 3₀ = 20₀
C1 = 2
C2 = 3

EXAMPLE # 1 C2 ≥ C1



DATA SEQ





THIS EXAMPLE ASSUMES ALL CHARACTERS EQUAL

INSTRUCTION DECODE SEQ

LA → LF
C1 → LE

START SEQ

LE + LF → LA + 30₈ (L + C1)
C1 - C2 → SCR = 7₈ SCR → SK

0 CSR
SCR → PW = 7₈

LC + C1 → LC + 21₈ (L + C2)
-12₈ → LE

ADDRESS SEQ

1st ADRS K1 + RA → F → ADRS XMIT
LA - LE → LA (30₈ - 12₈ + 16₈) DT + 1 + 1
LACI → TEST K1 EXHAUST BF + 1 + 1

2nd ADRS K2 + RA → F → ADRS XMIT
LC - LE → LC (21₈ - 12₈ + 7₈) DT + 1 + 2
TEST K2 EXHAUST BF + 1 + 2

(K1 + 1) + RA → F → ADRS XMIT
LA - LE → LA (16₈ - 12₈ + 4₈) DT + 1 + 3
LACI → TEST K1 EXHAUST BF + 1 + 3

(K2 + 1) + RA → F → ADRS XMIT
LC - LE (7₈ - 12₈ = EXHAUST) DT + 1 + 4
SET K2 EXHAUST BF + 1 + 4

ADRS SEQ WILL CONTINUE AFTER BLOCK K ADRS F/F IS CLEARED

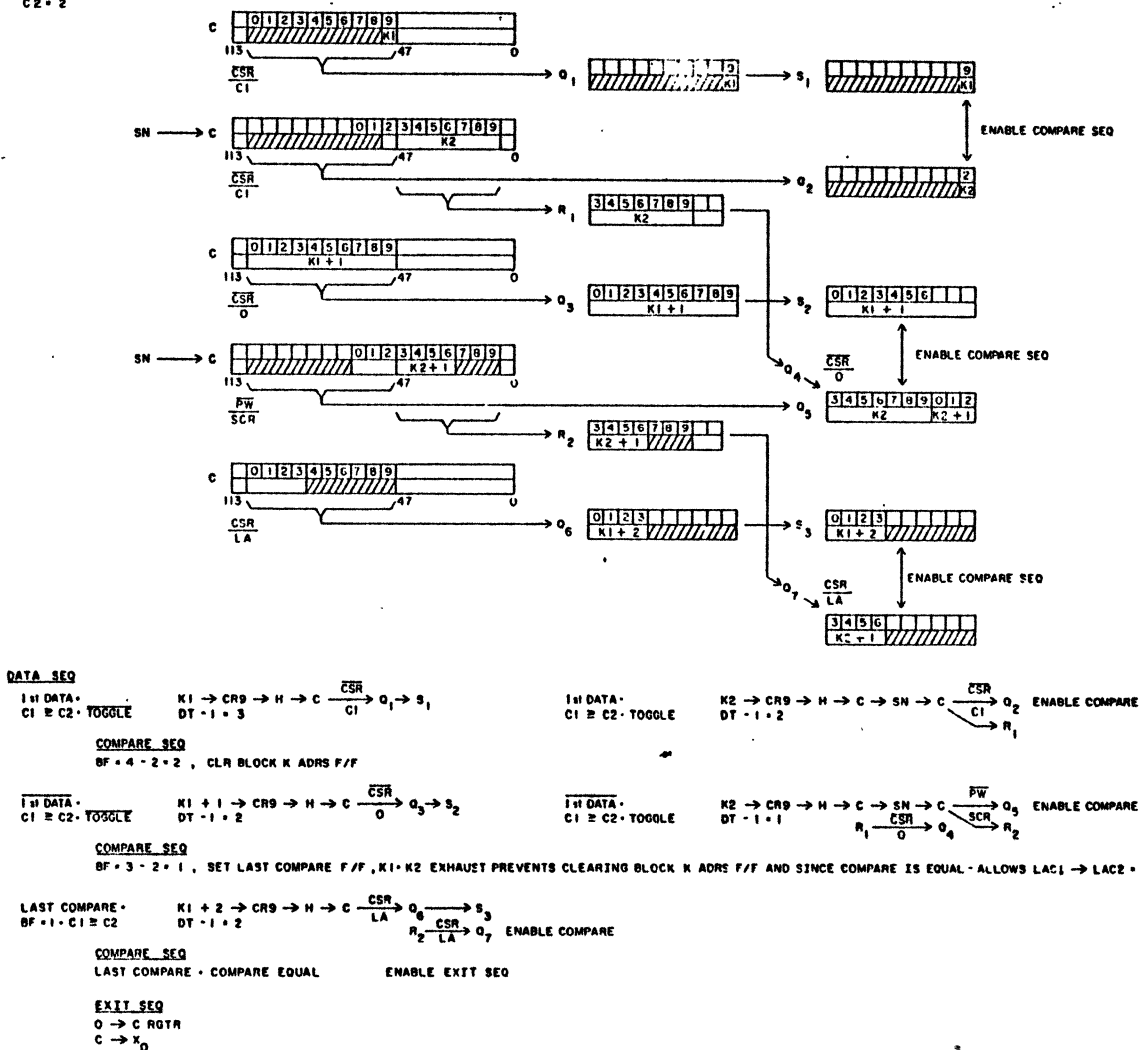
(K1 + 2) + RA → F → ADRS XMIT
LA - LE (4₈ - 12₈ = EXHAUST) DT + 1 + 3
SET K1 EXHAUST SET BLOCK K ADRS F/F BF + 1 + 3
LACI → LAC2 = 16₈
LACI + 4₈

COMPARE

L = 17₈ + 15₈
C1 = 9
C2 = 2

EXAMPLE # 2

C1 = C2



DETAILED PAK DIAGRAM (CPU 3, 41)

SHORT DATA SEQUENCE

The short data sequence is similar in operation to the normal data sequence; however, it is only enabled when the 1st & last FF is set.

1st & last FF is set for a move when K2 exhausts during 1st address, and for a compare when K1 and K2 exhaust during 2nd address.

MOVE INSTRUCTION (464, 465 - Refer to figure 5-2-39.)

1st DATA

1. Decrement data counter by one.
2. The first K2 word is transferred to Q.
3. Clear 1st data, set 2nd data.

2nd DATA

Path selection for 2nd data is determined by $C2 \geq C1$ and the buffer count.

$C2 \geq C1$

With $C2 \geq C1$, the last move equals-zero signal (LMS0TV) selects the appropriate data path.

1. Decrement data counter by one.
2. The shifted K1 word is transferred to Q. The C2 offset in CSR and the shift count in PW control the loading of Q. $CSR \neq PW$ ensures that only the shifted characters from K1 are stored in Q, while the K2 offset, and characters not part of the K2 field, are protected. (An example of this condition is illustrated in figure 5-2-41.)
3. The complete word in Q is transferred to S and HR in preparation for the 1st write request.

$C1 > C2$

With $C1 > C2$, the buffer count is checked. If the count equals zero, the last move equals-zero signal (LMS0TV) is generated. The sequence follows the same path described for $C2 \geq C1$.

If the buffer count equals one, the last move equals-one signal (LMS1TU) is generated. Last move equals-one indicates that two K1 words must be received. (An example of this condition is illustrated in figure 5-2-40.)

1. Decrement data counter by one.
2. After K1 has been shifted, all K1 characters will reside in the residue portion of C. Therefore, the residue must first be transferred to R and then to Q. The C2 offset in CSR and the shift count in PW control the loading of Q.
3. Clear 2nd data, set 3rd data.

3rd DATA

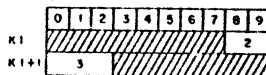
1. Decrement data counter by one.
2. The shifted K1 data is transferred to Q. The loading of Q is controlled by the remaining length value in LC and the shift count in PW.
3. The complete word in Q is transferred to S and HR in preparation for the 1st write.

COMPARE INSTRUCTION (466, 467 - Refer to figure 5-2-39)

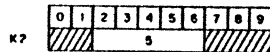
Short data for a compare monitors $C2 \geq C1$ and toggle to determine path selection.

The data path is the same as the one used for a normal data sequence with 1st data, except that for a short compare both CSR and PW are used. CSR will contain the C2 offset value ($C2 \geq C1$) or the C1 offset value, while PW will contain the remaining LC value ($C2 \geq C1$) or LA value ($C1 > C2$).

SOURCE



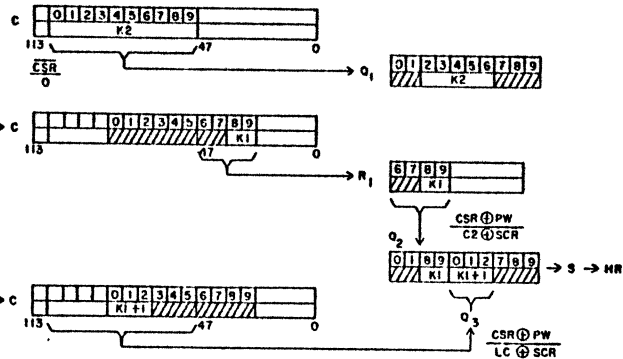
DESTINATION



MOVE EXAMPLE #4

L = 5
C1 = 8
C2 = 2

SHORT MOVE C1 > C2



DATA SEQ - ONE WORD

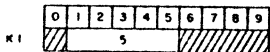
1st DATA K2 -> CR9 -> H -> C -> CSR -> O1
DT - 1 - 2

2nd DATA K1 -> CR9 -> H -> C -> SN -> R1
DT - 1 - 1

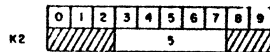
3rd DATA K1 + 1 -> CR9 -> H -> C -> SN -> C -> CSR -> PW -> LC -> SCR -> S -> HR -> DATA XMIT
DT - 1 - 0

5-2-40

SOURCE



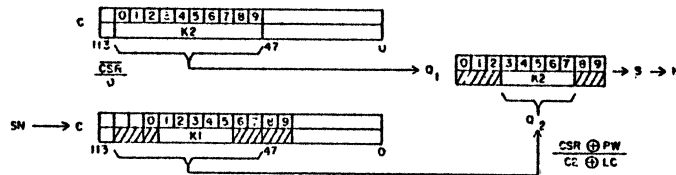
DESTINATION



MOVE EXAMPLE #5

L = 5
C1 = 1
C2 = 3

SHORT MOVE C2 > C1



DATA SEQ - ONE WORD

1st DATA K2 -> CR9 -> H -> C -> CSR -> O1
DT - 1 - 1

2nd DATA K1 -> CR9 -> H -> C -> SN -> C -> CSR -> PW -> C2 -> LC -> O -> S -> HR -> DATA XMIT
DT - 1 - 0

5-2-41

INSTR FLOW EX 1, EX 2		19981800	A
SHORT MOVE C1 > C2, C2 > C1		FIGURES 5-2-40, 41	5-2-84-2

DETAILED PAK DIAGRAM (CPU 3.42)

COMPARE SEQUENCE

The compare sequence is enabled from either the data sequence or the collate sequence.

FROM DATA SEQUENCE

The compare sequence monitors the compare word equal signal (CWEQ) to determine the action to be performed.

Comparison Equal (CWEQ)

1. The buffer counter is decremented by two, which will allow the address sequence to initiate read requests for another pair of words.
2. The block K address FF is reset, enabling the address sequence timing chain at the next clock.
3. S full and Q full are cleared, enabling the data sequence to resume, and another compare to be initiated.

Last compare is detected by the condition (K1 exhaust and K2 exhaust) or (1st & last FF). When both K1 and K2 are exhausted for a normal compare, or 1st & last is set for a short compare, the next compare will be the last. If the result of the last comparison is an equal condition, the exit sequence will be enabled.

Comparison Unequal (CWEQ)

1. The character position register (CP) is enabled, so that a code pointing to the first unequal character (from left to right) can be stored.
2. Using the CP code, the unequal character from both S and Q is stored in TS and TQ, respectively.
3. The compare sequence will enable the exit or collate sequence. The sequence enabled will depend on the instruction type being executed.

DETAILED PAK DIAGRAM (CPU 3.43)

COLLATE SEQUENCE

The collate sequence is enabled from the compare sequence if a compare word unequal is detected during a 466 instruction.

The collate sequence can be divided into two sections, where timing chain sequences CS114, CS164 and CS214 form collate I, and CS264 forms collate II. (Refer to figure 5-2-42.)

COLLATE I

Timing chain sequences CS114 and CS164 are enabled by compare word unequal (CWEQ) from the compare sequence, CM214. The remaining timing chain FF, CS214 is set by require address FF.

Three control FFs are conditioned by CS114 and CS164. They are: address 2, data 2 and require address.

The require address FF allows the address sequence to be enabled by enabling CS214. During CS214, the block K address FF is reset.

The address 2 FF indicates that two passes through the address sequence must be performed, since both collate characters are located in different words.

The data 2 FF indicates that two passes are required through data sequence D164. With address 2 set, each pass occurs after data ready; with address 2 reset, both passes occur consecutively after the first data ready.

The table at right shows the possible equality detection combinations; the resultant settings of the three control FFs, the number of address requests, the passes through D164, and the final code stored in WP.

CS214

CS214 is enabled by the require address FF set during CS164. At CS214, the block K address FF is cleared, and the A0 address FF is set. A0 address conditions the address sequence to transmit a collate table address. The upper three bits (3-5) of TS or TQ, which selects one of the eight possible collate table words, are stored in E and WP. During the address sequence, the collate table address from the A0 register is added to the select code in E to formulate the table word address.

COLLATE II CS264

Collate II is enabled from the data sequence when both collate characters have been loaded into TS and TQ.

If both collate characters are equal, the collate character equal signal (CCEQ) is generated to enable the compare sequence; otherwise the exit sequence is enabled.

Equality Detection FFs			Collate I Control FFs			Final WP	Pass in D164	No. Adrs Requests
TQ=TS	TQ=WP	TS=WP	ADRS2	DATA2	REQ ADRS			
-	1	1	0	0	0	No change	None	None
0	0	0	1	1	1	TQ	2 passes per Data Ready	2
1	0	0	0	1	1	TQ	2 passes for Data Ready	1
-	0	1	0	0	1	TQ	1 pass for Data Ready	1
-	1	0	0	0	1	TS	1 pass for Data Ready	1



* NOTE: THIS EXAMPLE ASSUMES INDICATED CHARACTER OF K1 AND K2 TO BE UNEQUAL
 UNEQUAL CHARACTER OF K1 = 63₈
 UNEQUAL CHARACTER OF K2 = 31₈

COLLATE TABLE

	0	1	2	3	4	5	6	7
A0								
A3		37						
A5				37				

INSTRUCTION DECODE SEQ

LA → LF
 C1 → LE

START SEQ

LE + LF → LA + 10₈ (L + C1)
 C1 - C2 → SCR + 1 SCR → SK 1
 0 CSR
 SCR → PW + 1
 LC + C2 → LC + 7₈

ADDRESS SEQ

K1
 1st ADRS K1 + RA → F → ADRS XMIT DT + 1 + 1
 LA - LE → (10₈ - 12₈ * EXHAUST) BF + 1 + 1
 SET K1 EXHAUST
 LAC1 → 10₈

K1
 2nd ADRS K2 + RA → F → ADRS XMIT DT + 1 + 2
 LC - LE → (7₈ - 12₈ * EXHAUST) BF + 1 + 2
 SET K2 EXHAUST, SET BLOCK K ADRS
 SET 1st & LAST F/F
 LAC1 → LAC2 + 10₈

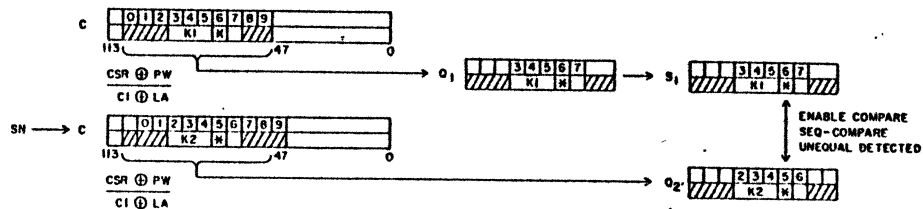
A0 ADRS (A0 → F + E) → F + RA → ADRS XMIT
 CLR A0 ADRS F/F
 SET BLOCK K ADRS
 CLR A2 ADRS F/F

COMPARE - COLLATE

L = 5₈
 C1 = 3
 C2 = 2

EXAMPLE #1

C1 ≥ C2
 SHORT COMPARE



DATA SEQ - ONE WORD

C1 ≥ C2 - TOGGLE
 DT - 1 + 1

CSR ⊕ PW
 C1 ⊕ LA

C1 ≥ C2 - TOGGLE
 DT - 1 + 0

COMPARE SEQ

SET COLLATE IN PROGRESS
 UNEQUAL CHARACTER POSITION → CP RGTR + 6₈
 SELECT UNEQUAL CHARACTER FROM S → TS RGTR + 63₈
 SELECT UNEQUAL CHARACTER FROM Q → TO RGTR + 31₈

COLLATE TABLE POSITION CHARACTER POSITION

5	4	3	2	1	0
6	3				
3	1				

COLLATE SEQ CS114, CS164, CS214

TS + WP, TO + WP, TO + TS
 SET DATA₂ F/F (TO + WP + TS + WP)
 SET ADRS₂ F/F (TO + WP + TS + WP + TO + TS)
 SET REQUIRE ADRS F/F (TO + WP + TS + WP)
 CLR 1st COMPARE F/F
 CLR BLOCK K ADRS F/F
 SET A0 ADRS F/F
 RTS 3-5 → WP
 E RGTR (6₈)

ADRS SEQ WILL START WITH TS CONTENT IN E RGTR

DATA SEQ

COLLATE IN PROGRESS - DT = 0

CR9 → T → I34 → I37 → TS (37₈)
 TS 0-2
 CLR DATA₂ F/F
 CLR T FULL F/F (TO + TS)

COLLATE SEQ CS214 (REQUIRE ADRS F/F - BLOCK K ADRS)

CLR BLOCK K ADRS F/F
 SET A0 ADRS F/F
 CLR REQUIRE ADRS F/F (ADRS₂ F/F)
 RTQ 3-5 → WP
 E RGTR (3₈)

ADRS SEQ WILL START WITH TO CONTENT IN E RGTR

DATA SEQ

COLLATE IN PROGRESS - DT = 0

CR9 → T → I34 → I37 → TO (37₈)
 TO 0-2
 CLR T FULL F/F
 ENABLE COLLATE CS264

COLLATE SEQ CS264

TS + TO ENABLE COMPARE SEQ (IF TS = TO, ENABLE EXIT SEQ)

COMPARE SEQ

CP RGTR + 1 → CP RGTR + 7₈ (ALLOW COMPARE S AND Q CHARACTERS 7, 8, 9)
 ASSUMING REMAINING CHARACTERS COMPARE EQUAL → ENABLE EXIT SEQ
 SET LAST COMPARE F/F

EXIT SEQ

0 → C RGTR
 C → X0 RGTR

DETAILED PAK DIAGRAM (CPU 3.44)

EXIT SEQUENCE

The exit sequence is enabled at the conclusion of a move or compare instruction.

MOVE INSTRUCTION

The enable exit signal (EEXNVF) is generated when: K1 and K2 are exhausted, the enable exit FF is set, and a write accept is received for the last K2 word. EEXNVF enables exit sequence E114. For a move, E114 will clear the C register only. Timing chain sequences E164 and E214 are skipped; E264 is enabled next. (Refer to figure 5-2-44.)

During E264, the contents of C, containing zero, are transferred into X0. The CMU exit signal (EMCEXII) is generated to enable the RNI sequence.

COMPARE INSTRUCTION

The enable exit signal (EEXNYF) is generated by the compare or collate sequences. A compare instruction (467) will generate enable exit if an unequal compare occurs before the last compare. It will also generate enable exit if the last compare is equal. However, the last compare FF will be set, indicating equality on the last compare.

A compare collate instruction (466) will generate enable exit if an unequal compare occurs after the appropriate collate characters are read and compared.

EXIT - COMPARE EQUAL

The normal exit for a compare equal is identical to the exit performed for a move.

EXIT - COMPARE UNEQUAL

The exit sequence for compare unequal is used to calculate the number of characters that were not compared as the result of the unequal condition, and whether K1 is greater or less than K2. The remaining count is contained in the LAC2 register. The character position code in the CP register is subtracted from the remaining count to produce a count equal to the number of characters that have not been compared +1. The count is transferred from LC via I5 into the C register.

If $Q < S$, the C register is complemented via the I5 complement control logic. The complement of C indicates that $K1 > K2$.

If $Q > S$, the C register is not complemented. This indicates $K2 > K1$.

At E264, the count value stored in the C register is transferred into X0. The CMU exit signal (EMCEXII) is generated to enable the RNI sequence.

DETAILED PAK DIAGRAM (CPU 3.45)

INVERTER 17 PARITY GENERATOR OUTPUT XMITTERS

There are five sets of transmitters that allow the CPU to communicate with other units in the system.

P TRANSMITTERS - 2 SETS

The current contents of the CPU P register are continually transmitted to the two PPS chassis. This output also includes a parity bit and the condition of the run FF. The PPS can use these signals to determine abnormal CPU operation.

ECS TRANSMITTERS

The ECS coupler receives the starting address and word count from the CPU during execution of an ECS instruction. An odd parity bit (COXP) accompanies the transmissions. The request (COREQ) is sent to establish the start of an ECS sequence. The write signal (COWRT) will be sent with the address if a data transfer from CM to ECS is to occur. Start transfer (COSTXF) will be sent if no AOR conditions inhibit the data transfer.

CM ADDRESS TRANSMITTERS

The sequences accessing memory develop the gating for loading F into the address transmitters. The request (MEMREQ) will accompany the address if no range error exists.

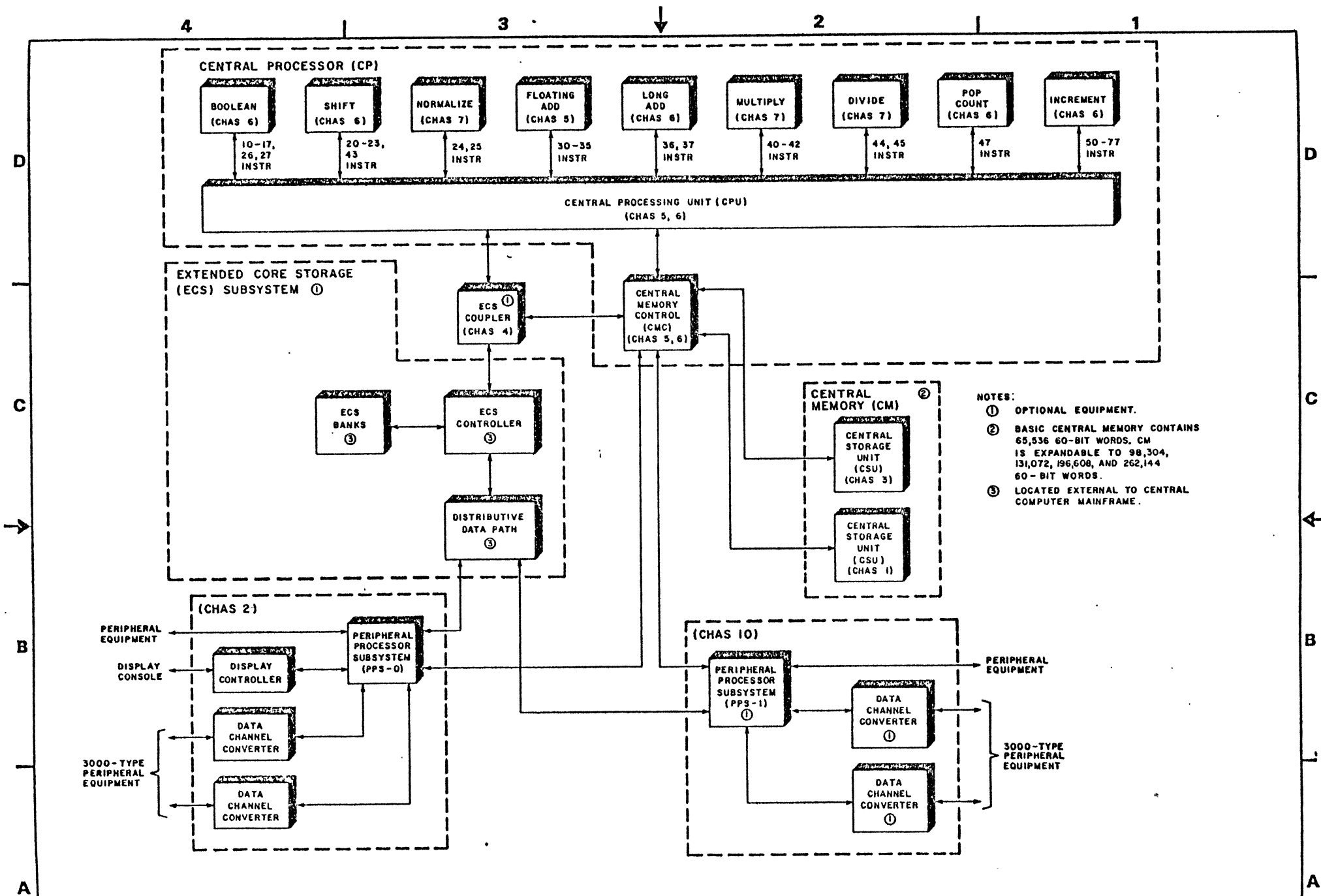
Parity for the address is developed as ADDPAR; however, this signal can be forced to zero by an input from the status and control register. An RNI tag accompanies requests for instructions. This is used in the CMC breakpoint test. Two control signals related to exchange jump are transmitted independently. The request exchange (CPOXRQ) signals CMC when the CPU executes a 013 instruction or an error exit exchange jump is to be made. OK exchange (CPOKX) is a response to an exchange request sent to the CPU by CMC.

CM DATA TRANSMITTERS

The contents of the hold register (HR) are clocked to the data transmitters continually. A single parity bit (ODTPAR) is developed to accompany data transmission. This parity can be forced to zero by the status and control register. A write signal (DWRITE) will be developed by the sequences when the data transmitters contain useful data. This signal will be transmitted to CMC as an indicator of a CM write operation.

I7 AND HOLD REGISTER

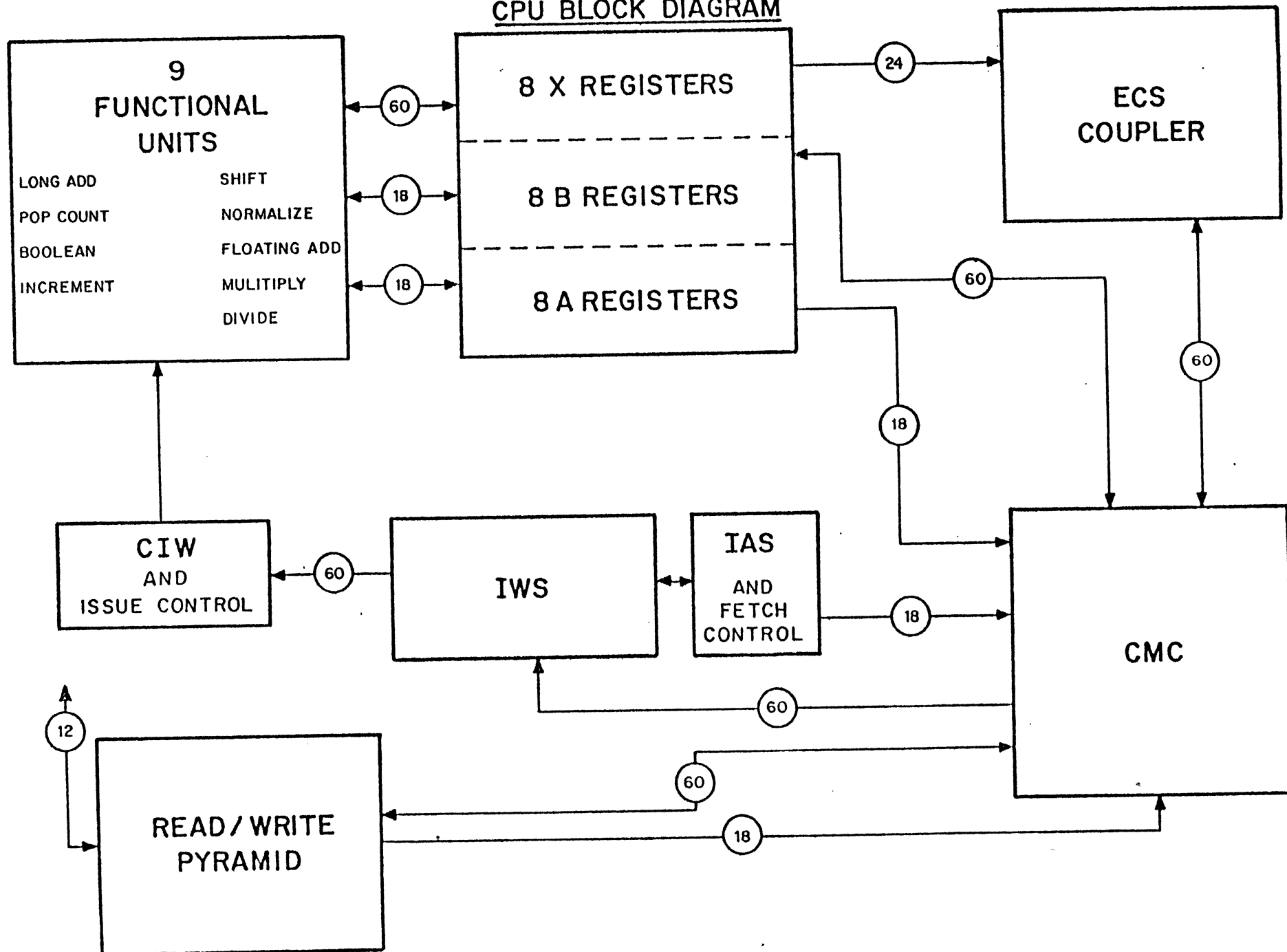
All data to be transmitted to memory is formatted in I7 and placed in the hold register. Clocking of the IIR is conditioned by the sequences which store data. These sequences develop the enable IIR (ENBHR) signal and the various I7 input paths. The following table illustrates the contents of HR for each sequence.



NOTES:

- ① OPTIONAL EQUIPMENT.
- ② BASIC CENTRAL MEMORY CONTAINS 65,536 60-BIT WORDS. CM IS EXPANDABLE TO 98,304, 131,072, 196,608, AND 262,144 60-BIT WORDS.
- ③ LOCATED EXTERNAL TO CENTRAL COMPUTER MAINFRAME.

CYBER 175 BASIC CPU BLOCK DIAGRAM



1 / 5

CENTRAL PROCESSING UNIT

The CPU, together with the functional units, executes programs stored in central memory (CM). The CPU consists primarily of 24 operating registers and a 12-word instruction stack. Data moves to and from the CPU through the operating registers.

INSTRUCTION CONTROL

The principal components of instruction control are the instruction word stack (IWS) and the instruction address stack (IAS). A coincidence test is made between the content of P and the content of each IAS rank during each clock period. Whenever a coincidence occurs, the corresponding 60-bit instruction word in the IWS is sent to the current instruction word (CIW) register for execution.

INSTRUCTION WORD STACK (IWS)

The IWS is a group of 12 registers, individually identified by rank, that holds program instruction words for execution. Data moves through the IWS from rank 12 to rank 1. Rank 12 is filled with CM data, and all other ranks are filled with data from the next-highest-order rank. The data shifted from rank 1 is discarded. Data movement through the IWS takes place only during a shift stack condition.

A shift stack condition occurs when the read address for an instruction word enters central memory control (CMC) and no CM bank conflicts occur. The condition is delayed to arrive coincident with the read data arriving at IWS.

INSTRUCTION ADDRESS STACK (IAS)

The IAS is a group of 12 registers that holds CM program addresses on a one-to-one basis with the program words in the IWS. Rank 1 in the IAS contains the oldest address in the stack, and this address corresponds to the relative CM location from which the word in rank 1 of the IWS was read.

All ranks of the IAS receive information from the next-highest-order rank during a shift stack condition in a manner analogous to that for the IWS. An address is read into rank 12 of the IAS from the next stack address (NSA) register. This address corresponds to the relative CM address for the word arriving at rank 12 of the IWS. When the stack is shifted, the address in rank 1 of the IAS is discarded.

The outputs of all IAS ranks are compared with the content of P each clock period. Rank X coincidence gates the IWS rank X content to the CIW register.

NEXT STACK ADDRESS (NSA)

The NSA register, adder, and advance stack flag (ASF) generate the NSA, which is an 18-bit CM relative address for the next sequential word required by the IWS. The inputs to the NSA register consist of a branch address from the P register or the content of the NSA register plus one or zero from the NSA adder. One is added to the NSA when ASF is set.

P REGISTER

The P register contains the current program execution address, which is an 18-bit relative CM address.

The P register has four possible sources of data. Read data bits 36 through 53 enter the P register from CMC as part of the exchange package. K enters the P register during conditional branch and return jump sequences. K+Bi enters during the unconditional jump instruction (02). P+1 enters during straight-line coding.

Under branch conditions, the P bits enter the instruction fetch address (IFA) register in CMC where they are added to the reference address for CM (RAC) to form the fetch address of an instruction word. The P bits enter the register selection network to become bits 36 through 53 in the first word of the exchange package. The content of the P register enters the return jump exit (RJX) address register during the return jump and error exit sequences.

CURRENT INSTRUCTION WORD (CIW) REGISTER

The CIW register is divided into four 15-bit parcels. All four parcels are loaded in one clock period when an instruction word is read from the IWS. The highest-order parcel in the instruction word is issued first. An instruction issues from the CIW register when the conditions in the functional units and operating registers are such that the functions required in the instruction execution may be completed without conflicting with a previously issued instruction. The other parcels are then left-shifted in the CIW register by either 15 or 30 bits, depending upon the instruction format for the instruction issued.

The exchange sequence register and counter provides three bits during the exchange sequence which become the i, j, and k designators that gate Xi, Aj, and Bk into the exchange package in CM. The counter outputs also gate the P and support registers into CM during the exchange sequence.

X REGISTERS

The eight X registers are the principal operating registers for the CPU. These registers each contain 60 bits and serve as the source and destination for operands in execution of the arithmetic instructions. The X registers receive data from all nine of the functional units and from CMC.

INPUTS

Translator

The translator merges the special case and sign data from the functional units into complement control signals. The signals are used by complement control to generate special case floating-point numbers or to change the sign of a result from a functional unit.

Input to X Network

Every functional unit has at least one result data path, and several of the floating-point units have multiple 60-bit result data paths to the X registers. Instruction control ensures that only one of the nine functional units sends a result to the X registers during any given clock period. The data flow from the functional units is treated in two groups. One group, consisting of the multiply, divide, shift, and normalize units, is treated in a static merge network. Data from these units flows through the complement control network before entering the X registers. The second group, consisting of the remaining functional units and CMC, merges with the data from the complement control network in a second static merge network. The 60 bits from this last network are delivered to all eight X registers. The data is entered into a specific register selected by the X register selection network.

X Register Access Control and Selection

The X register access control and selection functions determine the timing and the specific register selection for each word entered into an X register. The X register selection network selects the destination X register for all instructions. The i designator from the CIW register enters the X register selection network in one of three ways: directly, through a 1-clock-period delay, or through the X register access control. The direct path is used during an exchange sequence. The delayed path is used for all two-cycle (2-clock-period) instructions. The X register access control path is used for all instructions requiring more than 2 clock periods to execute. A four-register delay chain in the X register access control provides the delay required.

For example, register 1 supplies an extra clock period of delay for the 3-clock period normalize instruction's i designator value.

OUTPUTS

The contents of all eight X registers enter a static distribution network along with the i, j, and k designators from the CIW register. The distribution network decodes the i, j, and k designators and gates the contents of the selected Xi, Xj, and Xk registers to the nine functional units and CMC.

The Xi outputs to CMC stores X0 through X7 in the exchange package. The Xj output to jump test is used during the 03 branch instructions. The X0 register is connected directly to ECS instruction control.

A REGISTERS

The eight A registers are used to address CM for operands and store results.

Registers A1 through A5 are used to address CM when reading data from CM to an X register. A read CM reference is initiated whenever one of these A registers is the destination during execution of an increment instruction. The data from the relative CM address is delivered to the corresponding X register. For example, an increment instruction with an A2 register destination causes a CM reference with the CM data sent to the X2 register.

Registers A6 and A7 are used to address CM when writing data into CM from an X register. A write CM reference is initiated whenever one of these A registers is the destination during execution of an increment instruction. The data from the corresponding X register is written into the relative CM address specified in the A register.

INPUTS

Only one data source may transmit data to an A register during a given clock period. The input from the increment unit is gated into the selected A register by the go 2 cycle to A signal during 50 through 57 instruction execution. The CMC input fills A0 through A7 during an exchange sequence.

OUTPUTS

The data in the A register specified by the j designator is delivered to the increment unit and CMC each clock period. The data in the A0 register, the starting relative CM address for ECS instructions, is delivered to CMC and ECS instruction control each clock period. The CMC data path, from the distribution network, is used for storing the A register data in the exchange package in CM during an exchange sequence. The increment unit data path provides data from Aj for 50, 54, 55, 60, 64, 65, 70, 74, and 75 instruction execution.

B REGISTERS

The B registers are intended primarily for indexing functions in program execution. The B0 register does not physically exist in the hardware. In the instruction execution, this register appears to contain all zero bits.

INPUTS

Only one data source may transmit data to a B register during a given clock period. It is gated to the B register by access control. The increment unit input is present during 60 through 67 instruction execution and is gated by the go 2 cycle to Bi signal. The normalize inputs are gated by go normalize during 24 and 25 instruction execution. The CMC inputs are present during an exchange sequence and are gated by signals from exchange destination control. The boolean unit data is gated by the go 2 cycle to Bj signal during 26 instruction execution.

OUTPUTS

Bi and Bj are used by the jump test circuit for 04 through 07 instructions. Bi is sent to the K+Bi adder to form the word count during jump instruction (02) execution. Bj is used by the K+Bj adder for ECS instructions (011 and 012) and the central exchange instruction (013).

SUPPORT REGISTERS

The support registers assist the operating registers during the execution of programs. The registers are entered with CM read data during an exchange sequence. The data in these registers is returned to CM by a second exchange sequence which terminates the execution interval. The P register is also entered with CM read during an exchange sequence.

REFERENCE ADDRESS FOR CM (RAC)

The 18-bit RAC register is added to relative CM addresses to form absolute CM addresses. The P register content is added to RAC to form the absolute program address in CM.

FIELD LENGTH FOR CM (FLC)

The 18-bit FLC register defines the size of the CM field available for program execution.

EXIT MODE (EM)

The 6-bit EM register holds the exit mode selections for a program. The exit mode bits control CPU error processing. Any or all of the six bits can be selected at one time. The exit mode selections are listed below in octal format as they appear in bit positions 48 through 50 and 57 through 59 of the exchange package.

<u>Mode Selection</u>	<u>Condition Sensed</u>
0XX1	Address out of range
0XX2	Infinite operand
0XX4	Indefinite operand
1XX0	ECS flag register parity error
2XX0	CMC input error
4XX0	CM data error

REFERENCE ADDRESS FOR ECS (RAE)

The 21-bit RAE register content is added to the relative ECS address specified by the instruction to form the absolute starting ECS address for block transfers. The lower two octal digits of RAC (bits 36 through 41 of the exchange word) are always zero.

FIELD LENGTH FOR ECS (FLE)

The 24-bit FLE register defines the size of the ECS field available for block transfers. The lower two octal digits of FLE (bits 36 through 41 of the exchange word) are always zero.

MONITOR ADDRESS (MA)

The 18-bit MA register holds an absolute address that specifies the starting address of an exchange package.

EXCHANGE JUMP ADDRESS

A CPU (013) or PP (2600, 2610, or 2620) exchange jump instruction starts or interrupts the CPU and provides CMC with the first address of a 16-word exchange package in CM. This initial address is K+Bj, MA, or A (PPS XJA) in PPS-0 or PPS-1. The CPU holds this address in the XJA register. The exchange package shown in Figure 5-3 provides the following information on a program to be executed.

Program address (P) - 18 bits
Reference address for CM (RAC) - 18 bits
Field length of program for CM (FLC) - 18 bits
Exit mode (EM) - 6 bits
Reference address for ECS (RAE) - 21 bits
Field length of block transfer for ECS (FLE) - 24 bits
Monitor address (MA) - 18 bits

Initial contents of eight A registers - 18 bits

Initial contents of eight X registers - 60 bits

Initial contents of B1 through B7 (B0 constant zero) registers - 18 bits

The period of time that a particular exchange package resides in the CPU registers is called the execution interval. The execution interval begins with an exchange jump that reads the exchange package from CM and enters these parameters into the CPU registers. It ends with another exchange jump that stores the exchange package back into CM. An exchange jump timing diagram is provided behind the CPU logic diagrams. A timing chart is provided in part 15 of this section.

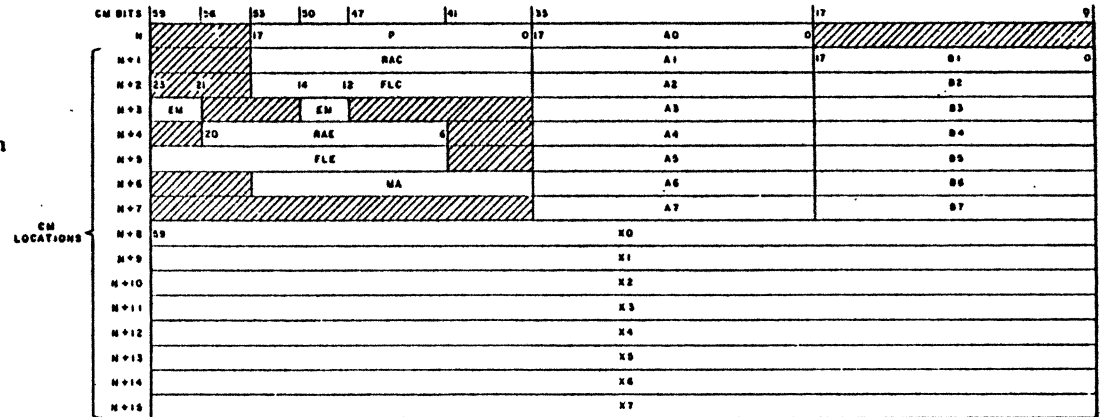
ECS INSTRUCTION CONTROL

The ECS instruction control interfaces the CPU and the ECS coupler. Block transfer starting address information (RAC, RAE, A0, and X0), field length (FLC and FLE), and word count [K+(Bj)] are sent from the CPU to the ECS coupler.

ERROR EXIT CONTROL

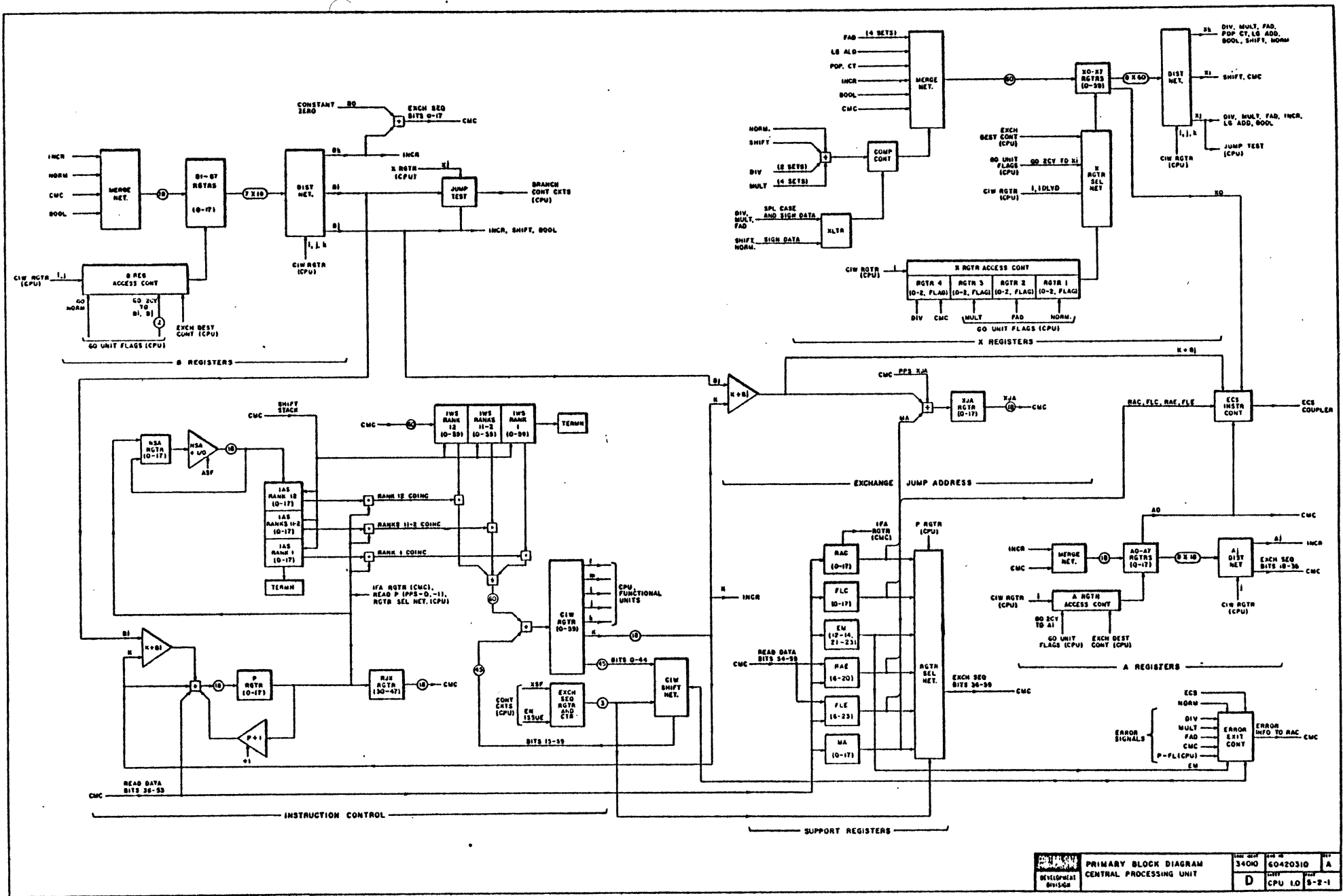
The error exit control monitors CP error conditions. Depending upon the type of error and the exit mode bits, the program in execution may be interrupted. If the error is an illegal instruction, breakpoint, or address range error on RNI or branch, the program interruption is unconditional. For other types of errors, the corresponding exit mode bits must be set before the program can be interrupted. The exit mode bits are:

<u>Bit</u>	<u>Exit Mode</u>
48	Address range error
49	Infinite mode
50	Indefinite mode
57	Parity error on ECS flag register operation
58	CMC input error
59	CM data error



NO HARDWARE REGISTERS EXIST

Figure 5-3. Exchange Package



	PRIMARY BLOCK DIAGRAM	3400	60420310	A
	CENTRAL PROCESSING UNIT	D	CPU I.O.	5-2-1

PART OF INSTRUCTION STACK; PART OF BRANCH

INSTRUCTION STACK

The instruction stack consists of the instruction word and address paths shown on CPU 3.0 and the control circuits shown in CPU 3.3. The principal components of the instruction stack CPU 3.0 are the IWS and the IAS. A coincidence test is made between the content of P and the content of each IAS rank during each clock period. Whenever a coincidence occurs, the corresponding 60-bit instruction word in the IWS is sent to the CIW register. If more than one coincidence occurs, all are sent simultaneously to the CIW register, and the results are merged as a logical sum (A+B).

INSTRUCTION WORD STACK (IWS)

The IWS is a group of 12 60-bit registers, individually identified by rank, that holds program instruction words for execution. Data moves through the IWS from rank 12 to rank 1. Rank 12 is filled with CM data, and all other ranks are filled with data from the next-highest-order rank. The data shifted from rank 1 is discarded. Data movement through the IWS takes place only during a shift stack condition.

A shift stack condition is generated when the fetch address enters SAS and no CM bank conflicts occur. The condition is delayed to arrive coincident with the read data arriving at IWS.

INSTRUCTION ADDRESS STACK (IAS)

The IAS is a group of 12 18-bit registers that holds relative CM program addresses on a one-to-one basis with the program words in the IWS. IAS rank 1 contains the oldest address in the stack, and this address corresponds to the CM location from which the word in IWS rank 1 was read. All IAS ranks are cleared/entered with information from the next-highest-order rank during a shift stack clock period in a manner analogous to that for the IWS.

An address is read into IAS rank 12 from the NSA register. This address corresponds to the CM address for the word arriving at IWS rank 12. The address shifted from IAS rank 1 is discarded.

Each IAS rank has three enable coincidence bits which indicate if the address in that rank is usable. These bits enter rank 12 along with the address from the NSA register. They are shifted through the ranks along with the address. The enable coincidence bits in each rank are cleared when the clear IAS signal voids the stack. Once cleared, the enable coincidence bits for a particular rank remain clear until an address is shifted into the rank.

The IAS is voided when any of the following instructions is executed.

1. Exchange jump (013, 2600, 2610, or 2620)
2. Return jump (010)
3. Jump (02)
4. Jump to address not in IAS (03 through 07)

The output of each IAS rank is compared with the content of P each clock period. Six-bit groups are compared on each 3HA7 module. The enable coincidence bit must be present to achieve group coincidence. The group results enter the 4HB7 modules where they are checked for word coincidence. Word X coincidence gates IWS rank X content to the CIW register.

A maintenance switch located on chassis 5 is used to disable either ranks 1 through 10 or ranks 1 through 4 of the IAS. This allows operation to continue in a degraded mode if a failure occurs in these ranks. The ranks are disabled by preventing one of the three enable coincidence bits from setting.

Operation

The next stack address (NSA) register, adder, advance stack flag (ASF), and associated input circuits generate the NSA. The ASF is set when shift stack is present. This flag advances the count of NSA in the clock period following the shift stack condition. The NSA is an 18-bit CM relative address for the next sequential word required by the IWS.

The two sources of data for the NSA register are the P register and the NSA loop. The content of the P register enters the NSA register when a P to NSA condition exists. The content of the NSA register plus one enters via the NSA loop when ASF is set. When both conditions exist, the P to NSA condition takes priority. When neither condition exists, the same value constantly reenters the NSA register via the NSA loop.

P to NSA signals the arrival of the last fetch word requested prior to a program branch. Thus, all IWS words have arrived from CMC, their corresponding addresses have entered into the IAS, and the branch address can now enter the NSA register.

Address

When the shift stack signal is present and ASF is not set, the address entered into IAS rank 12 is a copy of the content of the NSA register. When shift stack is present and ASF is set, the address entered into IAS rank 12 is NSA plus one. The address that enters IAS also loops back to enter the NSA register.

BRANCH

Branch instructions are executed in a portion of the CPU which is not well defined as a functional unit. This portion contains the program address register (P) and those control flags required to sequence the entry of a new address into the P register as well as the return jump exit (RJX) register and control flags required to execute the return jump instruction.

The P and RJX registers are shown on CPU 3.0, but the associated control flags are shown on several other diagrams. A general discussion of the branch functions, both normal and return jump, and detailed discussions of the P and RJX registers follow. For detailed information on the control flags, refer to the text for the diagram on which the flag is shown.

NORMAL BRANCH EXECUTION

A normal branch instruction, as distinguished from a return jump instruction, has three cases to consider in execution timing. One case occurs when the jump condition is not met in a conditional branch instruction, referred to as the fall-through case. A second case, called branch in stack, occurs when the jump is taken and the destination address is currently in the IAS. A third case, called branch out of stack, occurs when the jump is taken and the destination address is not in the IAS. Each of these cases has a separate timing sequence.

All normal branch instructions begin execution with the setting of the GJF (CPU 3.2, 3.7). This occurs when the registers required by the instruction are free.

The conditions for setting GJF are similar to the conditions for issue of a computation instruction. GJF allows the branch instruction to issue from the CIW register only in the fall-through case. The other two cases involve further sequence control flags.

A branch instruction in which the jump condition is met enters the starting address for the new program into the P register. The P register is entered with a new address when GJF sets. JCF (CPU 3.7) sets at the same time to indicate that the jump has been completed. The branch instruction then issues from the CIW register in the following clock period. If the new program address is in the IAS, the next instruction word is read directly into the CIW register. If the new program address is not in the IAS, a CM reference is initiated to read the new program instruction word and the IAS is voided.

RETURN JUMP EXECUTION

Execution of a return jump instruction begins with the setting of GJF in a manner similar to a normal branch instruction. GJF causes the content of the P register to be entered in the RJX register and the K portion of the return jump instruction to be entered in the P register. RJF (CPU 3.2, 3.3) sets to continue the sequence.

RJF is a delay mechanism to allow the completion of any previous CM instruction word references. RJF clears all IAS ranks for as long as it is set. When all instruction words previously requested have arrived at the IWS, RJF clears and SXF (CPU 3.3) sets to continue the sequence. The content of the P register is transmitted to the IFA register during the last clock period in which RJF is set.

SXF causes a CM reference to write an 0400 instruction into the address specified by the content of the RJX register. The address used to write the branch instruction is in the P and IFA registers. SXF remains set until the address for the CM reference has entered the SAS. SXF then clears and JCF sets to complete the sequence. The content of P is advanced one count during the last clock period in which SXF is set.

JCF allows the return jump instruction to issue from the CIW register. This flag is set for only 1 clock period. During this clock period, the P register contains address K+1. The IFA register contains this same address. No coincidence is possible between the content of P and the IAS because of the previous voiding of the IAS. The instruction stack fetch control therefore initiates the CM references to read the beginning of the called subroutine.

P REGISTER

The P register is an 18-bit register which contains the current program execution address. The current content of the P register is cleared and new data is entered when the enter P signal is present.

The P register enters into a large number of functions so it is physically reproduced six times (one-third of a P register on each 3HA7 module) to provide a fanout of information with a minimal delay. All hardware copies of the P register are entered with the same data in the same clock period and under the same conditions.

Inputs

The P register has four possible inputs. The P register input is made in the 4LA7 modules on the basis of three control conditions. Each condition specifies one of the data paths while the absence of all three conditions selects the fourth data path. Read data bits 36 through 53 enter the P register from CM during the exchange sequence. K bits 0 through 17 enter the P register during conditional branch and return jump sequences. K+Bi enters during the unconditional jump instruction (02). P+1 enters during straight-line coding and increases the content of P by one count in the clock period following the entry of a new word into the CIW register.

Outputs

Refer to the IAS paragraphs previously discussed for two of the P register outputs.

The content of P enters the IFA register (3HI7 modules) in CMC where it is added to RAC to form the fetch address of an instruction word. The content of P enters the RJX register during the return jump sequence and the error exit sequence. The content of P is subtracted from FLC in the 4LH7 module. When P is equal to or greater than FLC, a bit is set in the exit condition register. The content of P enters the 4LT7 modules to become bits 36 through 53 in the first word written into CM during an exchange sequence.

Two buffer registers (4HM7 and 3SW7 modules) delay transmission of the content of P to the PPS for use in a read P (27) instruction. The buffer registers are enabled by the sample P signal which is present when APF is set or 1 clock period after an error exit enter P or exchange jump enter P⁵ condition occurs. The PPS receives the relative address of the instruction currently in execution.

RETURN JUMP EXIT (RJX) REGISTER

The current content of the P register (current execution address plus one) enters the RJX register (4HM7 module) when the enter RJX signal is present. The information is transmitted to the SWS in CMC to construct and write into CM the return jump or error exit word.

The return jump word is 60 bits long and is formed in the SWB; bits 0 through 29 are zeros, bits 30 through 47 are the RJX bits, and bits 48 through 59 form the 0400 instruction code. This word is written into the CM location determined by K of the return jump instruction plus RAC.

The error exit word is 60 bits long and is formed in the SWB; bits 0 through 29 are zeros, bits 30 through 47 are the RJX bits, bits 48 through 53 are exit condition bits, and bits 54 through 59 are zeros. The word is written into CM at location RAC.

PART OF INSTRUCTION ISSUE

The instruction issue function is performed by the CIW register, associated circuits on CPU 3.1, and additional control circuits on CPU 3.2.

Program instruction words are read one at a time from the IWS into the CIW register for execution. Each instruction word is divided into four 15-bit parcels. As many as four instructions may be in the CIW register at one time. These instructions must be executed in sequence, and the proper allowance must be made for the mixture of one-parcel and two-parcel instruction formats.

An instruction issues from the CIW register when the conditions in the functional units and operating registers are such that the functions required in the instruction execution may be completed without conflicting with a previously issued instruction. This is indicated by the presence of the go issue signal which is comprised of the enable issue and registers free signals. The issue process requires 1 clock period. During this clock period, the f, m, i, j, and k designators in the instruction are sent to all appropriate parts of the CPU. The proper control flags are set to execute the functions required in the instruction. Also, the data in the CIW register is shifted to position the next instruction for execution. Except for increment instructions (5X), there is no further record of the instruction once it issues from the CIW register. Therefore, instructions must be executed to completion in a fixed-time framework. No delays are allowable from issue to delivery of data to the destination operating registers. For increment instructions (5X), CM conflicts may cause execution delays. Increment data is held in CMC until the conflicts are resolved.

CURRENT INSTRUCTION WORD (CIW) REGISTER

OPERATION

The CIW register is physically located on the 4HD7, 4HE7, and 4HF7 modules. Because of fanout considerations, the upper 15 bits are distributed over a total of 11 modules, 9 4HD7 and 2 4HE7 modules, while the lower 45 bits are located on 5 4HF7 modules.

The CIW register is divided into four 15-bit parcels. All four parcels are loaded in 1 clock period when an instruction word is read from the IWS. The highest-order parcel in the instruction word is issued first. The other parcels are then left-shifted in the CIW register by either 15 bits or 30 bits, depending upon the instruction format for the instruction issued. The lower-order parcels are replaced with zeros as the data is left-shifted in the register.

The CIW register is cleared and new data enters when the registers free and enable issue signals are present. The presence of either sample IWS or gate 15/30 path signals determines whether the input data is from the IWS or from the lower parcels of the CIW register. The absence of both signals during a go issue condition enters a 60-bit word of zeros in the CIW register.

IWS PATH

Data from the IWS path is selected whenever sample IWS is present. If coincidence occurs between P and IAS, an instruction word enters the CIW register; otherwise, a word of all zeros enters the register. Sample IWS exists whenever the read stack signal is present except during the part of the exchange sequence when XIF is present. XIF inhibits the IWS path when the 15/30 path is entering the i, j, and k designators into the CIW register that writes the A, B, and X registers into the exchange package in CM.

15/30 PATH

Data enters from the 15/30 path whenever the gate 15/30 path signal is present. Circuits on the 4LG7 modules determine which data is entered. The data is either the CIW left-shifted 15 or 30 bits, with vacated lower parcels filled with zeros, or the XSK bits. The translator in the 4LG7 modules uses the f, m, and i designators to select a shift of 15 or 30, depending upon whether the issuing instruction contains one or two parcels. XSK+1 bits 0, 1, and 2 are gated into bit positions 45 through 53 by XSF in the 4LG7 modules.

PARCEL COUNT

The parcel count reflects the number of the parcel in the CIW that is being issued. The parcel-counting function is performed by the parcel counter register on the 3IH7 module and a translator on the 3LK7 module. The count changes each time an instruction issues. When the last parcel in the CIW issues, the counter is cleared.

INSTRUCTION ISSUE CONTROL

The instruction issue function is performed by the control circuits on CPU 3.2 and by the CIW register and associated circuits on CPU 3.1. Refer to the CPU 3.1 description for the definition of instruction issue.

A, B, AND X REGISTER RESERVATIONS

Eight register busy flags are associated with each set of registers, A, B, and X. These busy flags are bit registers with separate set and clear inputs which are forced clear by a master clear. A busy flag is set to reserve a specific register when an instruction issues from the CIW register that delivers a result to that register. The flag remains set until the result is entered in the reserved register. This flag prevents subsequent instructions from reading the content of this register until the new data has been entered.

The register busy flags are located on the 4HC7 modules. The flag(s) that is set when an instruction issues is determined by a translation of the instruction's i, and in the case of the B register, j designators and five outputs of the go unit select translator (4LC7 module).

Before the next instruction in the CIW register can issue, its operand and result register needs are checked against the register busy flags. Coincidence between the outputs of the i, j, and k sense translators and a set register busy flag generates a register busy signal that is used in the registers free translator (4LC7 module). The instruction does not issue until the results of the previous instruction are delivered to the reserved/requested register; at this time, the appropriate selection bits and clear reservation signal clear the selected register busy flag.

The registers quiet signal indicates that no register busy flags are set; no registers are reserved.

REGISTERS FREE

This condition (4LC7 modules) indicates that the operating registers involved in the instruction waiting to issue are now free. That is, no register busy flags are set for an A, B, or X register designated as either an operand or a destination register for the upper instruction in the CIW register.

GO UNIT CIRCUIT

The go unit circuit (4LD7 and 4HG7 modules) consists of the go unit select translator, go unit flags, go unit fanout, multiply busy fanout, and divide busy fanout.

GO UNIT SELECT TRANSLATOR

The go unit select translator (4LD7 module) translates the f, m, i, and j designators of the instruction in the upper parcel of the CIW register into the outputs listed on the diagram. The outputs are generated only when the input designators satisfy the conditions in the first three columns of the translator table. For example, go to Xi is not enabled by 5X instructions with i designator values of zero, six, or seven but is enabled by 5X instructions with i designator values of one through five.

The outputs enable the setting of go unit flags or operating register busy flags when the instruction issues.

GO UNIT FLAGS

The primary function of the go unit flags (4HG7 module) is to control the execution of all instructions involving functional units. With the exception of the multiply and divide units, data flows continuously through the functional units. The outputs of the functional units are discarded, however, unless the applicable go unit flag is set. The go unit flag gates the output of the functional unit to the destination operating register(s). Also, all instructions that execute in 2 clock periods send a go 2 cycle signal to the operating register's access control which selects the correct result register in the following clock period.

The multiply and divide busy flags control data movement through the respective functional units and block further inputs for 1 and 17 clock periods, respectively, following instruction issue.

DIVIDE BUSY FANOUT

The divide busy flag becomes the divide busy fanout outputs shown. Enter Xk, enter exponent, and divide busy are complements of the divide busy flag; release rank 1, 2 borrows and begin sequence signals are not inverted.

ENABLE ISSUE TRANSLATOR

Enable issue indicates that the required functional units, register destination paths, and storage access path are available for the current instruction in the CIW register. This condition does not consider the possible register reservation problems discussed previously.

The enable issue translation primarily prevents the conflict that would occur if two or more instructions attempted to deliver results to the X registers during the same clock period. Thus, an instruction is not allowed to issue if the data to the destination X register would conflict with data from another functional unit already processing data. For example, the normalize instructions (24, 25) are 3-clock-period instructions so they are prevented from issuing when an instruction has previously issued which has 3 clock periods remaining before delivering its results to an X register. Block 5 cycle prevents the 5-clock-period multiply instructions (40 through 42) from issuing when the divide unit results or a CM word is within 5 clock periods of entering an X register.

INSTRUCTION STACK - FETCH CONTROL

The instruction stack consists of the instruction word and address paths on CPU 3.0 and the fetch control circuits on CPU 3.3. Words are requested for the IWS two words ahead of the word currently being executed. Since at least two words (ranks 11 and 12) still remain to be executed in the IWS when a word from IWS ranks 1 through 10 is being executed in the CIW register, no additional words are requested. However, when the rank 11 word is gated to the CIW register, one word is requested from CM in order to remain two words ahead and similarly, when the rank 12 word is gated to the CIW register. Under branch conditions, two words are also requested.

The fetch control circuits monitor the CIW register and instruction stack and fetch one or two words for the IWS, depending upon detected conditions. One word is requested from CM when coincidence is detected between IAS word 11 and the content of P. Two words are fetched when coincidence is detected between IAS word 12 or when there is no coincidence between the content of P and any of the IAS ranks.

The following paragraphs describe conditions in the instruction stack under straight-line code and branch situations.

STRAIGHT-LINE CODE

Straight-line code exists when the program address is advanced sequentially. Since the instruction words are requested for the stack two words ahead of the word currently being executed, coincidence between P and IAS occurs only on ranks 11 and 12. When a word is read from IWS rank 11, F1F sets to request another word. If the requested word arrives before program execution advances to the next word, the stack has shifted data one rank and the next word is again in rank 11. If the requested word does not arrive before program execution advances to the next word, a word is read from IWS rank 12. When this occurs, a second word is requested from CM to accelerate the stack-filling process.

Program execution can proceed so rapidly that the instruction word read from IWS rank 12 has been executed before the first of the two requested words has arrived. When this happens, OSF sets and program execution must wait for the arrival of the first instruction word.

BRANCH IN STACK

Program execution may reach a branch instruction whose destination address is already in the IAS. When this occurs, the content of the P register is altered to the new program address. A coincidence occurs in the IAS during the following clock period, and the corresponding word is read from the IWS to the CIW register. The jump is then completed without a CM reference for a new instruction word.

BRANCH OUT OF STACK

Program execution may reach a branch instruction whose destination address is not in the IAS. When this occurs, the content of the P register is altered to the new program address. No coincidence occurs in the IAS during the following clock period, and the OSF and JOF are both set. A new address is entered in the IFA register, and two words are requested from CM to begin the new program sequence.

It is possible for a branch out of stack to occur whose destination address corresponds to a program word which has already been requested from CM as a result of sequential two-word read-ahead. If the word has not actually arrived at the IWS at the time of the branch test, the jump out of stack occurs, and a duplicate of the first word in the new sequence is read from CM. Execution of the new sequence can begin as soon as the earlier word arrives at the IWS.

WORD COINCIDENCE TEST

The 4LO7 module performs the word coincidence test. When all three bit groups for the same word are coincident with the similar bits of the P register, the three inputs are ones and word X coincidence is generated. Word 11 coincidence exists when the content of IAS rank 11 is the same as P and implies that the CIW register is about to read the next to the last word in the IWS. When this word is read to the CIW register, F1F sets to request an instruction word from CM. Word 12 coincidence exists when the content of IAS rank 12 is the same as P and implies that the CIW register is about to read the last word in the IWS. When this word is read to the CIW register, both fetch flags (F1F and F2F) set to request a total of two instruction words from CM. No coincidence is generated when no IAS ranks are coincident with P and denotes that a two-instruction word fetch is required.

ACCESS CONTROL FLAG TRANSLATOR

The access control flag translator (6HK7 module) and the associated circuits on the 4LO7 module generate the following signals.

Fetch one word flag (F1F)	F1F sets whenever the instruction stack requires a word from CM and the request has not yet entered the SAS.
Fetch two words flag (F2F)	F2F sets whenever the instruction stack requires two words from CM and neither request has entered the SAS. F1F always sets when F2F sets.
Marker one word flag (M1F)	M1F sets whenever the instruction stack requires a word from CM which has not yet arrived at the IWS.
Marker two words flag (M2F)	M2F sets whenever the instruction stack requires two words from CM which have not yet arrived at the IWS. M1F always sets when M2F sets.

Store exit flag (SXF)

SXF sets at the same time that RJF clears and continues the return jump sequence. It remains set until CMC access control has accepted the address for writing the RJX register data into CM. SXF then clears and JCF sets if the return jump sequence was not initiated by an error exit condition.

Jump out of stack flag (JOF)

JOF sets when a branch instruction is executed which requires jumping to a program address not currently held in the IAS. The flag remains set until P is entered in NSA.

Enable P to NSA-1, -2

These signals are used to enter P into the NSA register after a branch to an address not in the IAS.

Return jump flag (RJF)

RJF sets when the CIW register contains a return jump instruction and GJF sets or when the error 3 flag sets. This flag remains set until all words requested from CM for the IWS have arrived at the IWS. SXF then sets and RJF clears.

Out of stack flag (OSF)

OSF sets whenever the CIW register is ready for the next word from the IWS and no word is available. The flag remains set until it is specifically cleared.

P to NSA

This condition signals the arrival of the last fetch word requested prior to a program branch. The NSA register then sets to the new program sequence address (content of P register) and marker flags then set in anticipation of the instruction words for the new program sequence. P to NSA also clears JOF.

Enter 1FA-1, -2

These conditions are ORed together in the IFA register module to become a single condition. This condition occurs on a program branch resulting in a jump out of stack. It clears the IFA register and fills it with the content of the P register plus RAC.

Void stack

This condition is ORed with XSF, an 02 jump instruction, or an 02 jump out of stack instruction to clear the IAS. This condition occurs when RJF sets or when any jump to an address not in IAS occurs.

X REGISTERS DATA PATHS

The eight X registers are the principal operating registers for the CPU. These 60-bit registers serve as the source and destination for operands during arithmetic instruction execution. Data remains in an X register until a control condition generated in the X register access control (CPU 3.6) enters new data. Only one X register can be cleared and entered with new data during any given clock period.

The X registers receive data from all nine of the functional units and from CMC. The X register input control (5CB7 module) and access control circuits regulate the entry of data into the X registers. The contents of the X registers are sent to all nine of the functional units, branch control, and CMC. The merge and distribution functions are performed in 60-bit static networks preceding and following the X registers.

X REGISTER INPUT CONTROL

X register input control uses the special-case information from the floating-add, multiply, and divide functional units to generate an overflow signal. The complement of the overflow signal sets bit 59 of the X register input path to a zero when an overflow condition exists. This is the first step in generating the special bit pattern of the overflow condition exponent.

The X register input control translator merges the special case and sign treatment information from the functional units into complement control signals. These signals are used by complement control (4RC7 and 4RD7 modules) to generate the bit patterns for the special case floating-point formats or to complement the entire word when the result from the functional unit should be negative.

INPUT TO X NETWORK

Every functional unit has at least one result data path, and several of the floating-point units have multiple 60-bit result data paths to the X registers. Merging of the input to X data is performed in 60-bit static networks preceding the X registers.

A functional unit outputs ones to the X registers during all clock periods except those in which a meaningful result is ready for entry into the destination X register. Instruction issue control (CPU 3.2) ensures that only one of the nine functional units sends a result to the X registers during any given clock period.

The data flow from the functional units is treated in two groups. One group, consisting of the multiply, divide, shift, and normalize units, is treated in a static merge network. Data from these units flows through the complement control network before entering the X registers. The second group, consisting of the remaining functional units and CMC, merges with the data from the complement control network in a second static merge network. The 60 bits from this last network are delivered to all eight X registers. The data is entered into the selected register when the enter X condition is present.

COMPLEMENT CONTROL

The complement control network complements all, part, or none of the 60 bits of input data, depending upon the complement control signals received from the X register input control translator. The complementing network performs the following functions.

- Complements the multiply or divide unit output if the result coefficient is negative.
- Complements the normalize or shift unit output if the result is positive. The result received from each of these units is the complement of the true result.

- Packs a complete overflow, complete underflow, or indefinite quantity into the destination X register if the corresponding error condition was detected by any of the floating-point units during instruction execution in these units.
- Complements the ones input from the multiply, divide, shift, and normalize units to zeros during clock periods in which CMC delivers data to the X registers. Succeeding merges of these zero bits with ones from the remaining functional units result in zeros from the nine functional units, allowing the entry of data from CMC.

Overflow Quantity

When an overflow condition is detected by a floating-point functional unit, the functional unit sends all ones, and the X register input control (5CB7 module) causes bit 59 of the normalize unit input data to be a zero. Bits 0 through 58 from the normalize unit and bits 0 through 59 of the remaining functional units are ones. Thus, the data entering complement control from the static merge network for any overflow situation consists of a zero for bit 59 and ones for bits 0 through 58. Complement control operates on this input to form the required positive or negative overflow quantity. The positive overflow quantity (37770----0) is formed when the complement 0 through 47 signal complements bits 0 through 47 to zeros. The negative overflow quantity (40000----0) is formed when the complement 48 through 57, complement 58 and 59, and complement 0 through 47 signals complement bits 0 through 59.

Underflow Quantity

When an underflow condition is detected by a floating-point functional unit, the functional unit sends all ones. Thus, a word of all ones enters complement control. The underflow quantity (00000----0) is formed when the complement 0 through 47, complement 48 through 57, and complement 58 and 59 signals complement the entire word to zero bits.

Indefinite Quantity

When an indefinite condition is detected by a floating-point functional unit, the unit sends all ones. Thus, a word of all ones enters complement control. The indefinite quantity (17770----0) is formed when the complement 0 through 47 and complement 58 and 59 signals complement bits 0 through 47, 58, and 59.

DATA OUTPUT NETWORK

The contents of all eight X registers enter a static distribution network along with the i, j, and k designators from the CIW register (CPU 3.1). The distribution network decodes the i, j, and k designators and gates the contents of the selected Xi, Xj, and Xk registers to the nine functional units, branch control, and CMC.

Certain functional units require multiple sign bit inputs. Consequently, bit 59 of both Xj and Xk quantities is entered into fanout networks that distribute these sign bits.

The Xi and Xk quantities also enter a static selection network along with bit 1 of the m designator. This network gates Xk to the shift unit when m bit 1 is a one (22 and 23 instructions). It gates Xi when m bit 1 is a zero (20 and 21 instructions).

The lower 24 bits of the X0 register are unconditionally sent to the ECS instruction control (CPU 3.9).

A AND B REGISTERS DATA PATH

A REGISTERS

The eight A registers are used to address CM for operands and store results. These 18-bit registers are individually identified by the symbols A0 through A7. The A0 register is not used in CM addressing for operands. A0 holds the starting relative CM address for ECS block copy instructions. A0 may also be used for some indexing functions in a manner similar to the B register.

The A1 through A5 registers are used to address CM in reading data from CM to an X register. A read CM reference is initiated whenever one of these A registers is the destination in an increment instruction execution. The data from the relative CM address is delivered to the corresponding X register. For example, an increment instruction whose destination is the A2 register causes a CM reference with the CM data sent to the X2 register. Such an increment instruction reserves both the A register and the corresponding X register. The A register busy flag clears when the increment unit data arrives at the A register. The X register busy flag clears when the data arrives from CMC at the X register.

The A6 and A7 registers are used to address CM in writing data into CM from an X register. A write CM reference is initiated whenever one of these A registers is the destination in an increment instruction execution. The data from the corresponding X register is delivered to CMC to be written into the relative CM address specified in the A register. Such an increment instruction reserves the A register but not the X register. The X register data is copied into the SWS in CMC in the same clock period that the increment instruction issues from the CIW register. The X register is then free in the following clock period. The A register busy flag clears when the increment unit data arrives at the A register.

Data remains in an A register until a control condition generated in the A register access control (CPU 3.6) specifically gates a clock pulse to clear the register and enter new data. Only one A register may be cleared and entered with data during any given clock period. The control condition which causes this entry is the enter A condition. The selection of the proper A register is speci-

fied by A selection bits 0 through 2 originating in the A register access control.

INPUTS

The two sources of data for the A register are the increment unit and CMC. Since only one of these sources may transmit data in a given clock period, these data paths are merged (4RA7 module) without selection in the static network prior to the A registers. The static merge network delivers the merged 18 bits of data to all of the A registers, but the data is entered into the selected A register only when the control condition enter A is present.

OUTPUTS

Data from the A registers is distributed by the 4RI7 modules. These modules receive 18 bits of data from each of the A registers plus three bits from the j designator portion of the CIW register. The data in the A register specified by the j designator is delivered to the increment unit and CMC each clock period. The data in the A0 register is delivered to CMC and ECS instruction control (CPU 3.9) each clock period. The A0 register contains the starting CM address for ECS block copy instructions. The CMC data path is not used for transmitting addresses for CM references. The path is used for storing the A register data in the exchange package in CM during an exchange sequence. The addresses for CM references are directly from the increment unit.

B REGISTERS

The B registers are intended primarily for indexing functions in program execution. These 18-bit registers are individually identified by the symbols B0 through B7. The B0 register does not physically exist in the hardware. In the execution of instructions, this register appears to contain all zero bits. Information to be stored in the B0 register is, in effect, discarded.

Data remains in a B register until a control condition generated in B register access control specifically gates a clock pulse to clear the register and enter new data. Only one B register can be entered with new data during any given clock period.

Communication between the B registers and other parts of the system involves a merging and distribution of 18-bit data paths. These functions are performed by static networks preceding and following the actual B registers.

INPUT TO B NETWORK

The B registers have five inputs. All five data paths are merged (4RB7 modules) without selection at the B register end of the data paths; only one source may transmit data in a given clock period. The data is entered in a selected B register only when the control condition enter B is present. The selection of the proper B register is specified by the B selection bits 0 through 2 originating in B register access control (CPU 3.6).

OUTPUT DATA NETWORK

The data from the B registers is distributed by the 4RI7 modules. These modules receive 18 bits of data from each of the B registers plus 9 bits of control information from the CIW register. This control information corresponds to the i, j, and k designators in the CIW register. Data paths from this distribution network to other parts of the system are divided into three groups. Each group carries information from a single B register in any given clock period. The B register for each group is specified by the i, j, or k designator appropriate for that group.

X, A, B REGISTERS ACCESS CONTROL AND SELECTION

The X, A, and B registers access control and selection functions determine the timing and the register selection for each word entered into an X, A, or B register. In addition, the functions release the register reservations at the proper time to correspond with information arrival at the X, A, or B registers.

X REGISTER ACCESS CONTROL

X register access control contains four 4-bit registers (4RQ7 module). These registers are cleared and entered with new data every clock period. Information flows from register number 4 through the other three registers at a rate of one register each clock period. The four registers constitute a delay mechanism for the destination X register specified during an instruction execution. A delay of 0 to 4 clock periods may be obtained depending upon which register is entered with the data.

Preceding each register is a data switch (4RP7 module) which selects the data from the preceding register in the chain or data from outside X register access control.

Each data switch is controlled by a functional unit flag for the unit with the corresponding execution time. Data flows from register to register unless the controlling functional unit flag is set. When the flag is set, the data relative to that functional unit is entered into a register and processed down the chain.

No two functional units may deliver data to an X register in the same clock period. Conflicts of this type are prevented by instruction issue control (CPU 3.2) rather than X register access control. Register numbers 2 through 4 flags allow instruction issue control to sense this type of conflict.

An instruction is not allowed to issue if the data to the destination X register would conflict with data from another functional unit which is already in process. As a result, the functional unit control of the data switches in the access control chain never discards useful data from a higher-order register in the chain.

The functional unit with the longest execution time is the divide unit. The unit is not segmented to the degree of the other units so only one divide operation may be in process at one time. When the divide instruction issues from the CIW register, the *i* designator is captured in a three-bit register (4RO7 module). It is cleared and entered with the current value of the *i* designator every clock period until the divide busy flag sets. The divide busy flag sets when a divide instruction (44 or 45) issues and remains set for 17 clock periods. Consequently, the three-bit register retains the divide instruction *i* designator for 18 clock periods following issue of the divide instruction.

At divide time 15 (T15) in the divide sequence, the data in the three-bit register is gated into the X register access chain. At the same time, divide T15 becomes the flag that indicates the presence of a valid three-bit code. The four bits then move from register to register and control the data entry into the X register 4 clock periods later. The divide unit instructions cannot issue when an increment unit read to X instruction is in progress. This prevents conflicts at the input to the X registers.

CM access time is longer than any functional unit execution time other than the divide unit. A CM reference which results in reading a word into a destination X register is treated the same as a functional unit. CM to X tag bits 0 through 2 are transmitted from CMC 6 clock periods before the CM data is available at the input of the X registers. The tag bits enter into a three-bit register which clears every clock period. Read to X is generated in CMC during increment read to X instructions. Read to X is delayed and transmitted from CMC when the CM to X tag is in the three-bit register. Read to X gates the tag along with a flag into register number 4. The four bits then move from register to register and control the data entry into the proper X register 4 clock periods later.

The *i* designator may change to a new value when an instruction issues. Since the go unit flag (for example, go normalize) also sets at this time, the *i* value must be saved or delayed for 1 clock period to get the correct value into an access control register. This function is performed by a three-bit register which is cleared and entered with the *i* designator value every clock period. The *i* delayed output is also used in the X, A, and B register selection circuits.

A multiply instruction requires 5 clock periods for execution. The multiply busy flag sets during the first of these clock periods, and the data is transmitted from the multiply unit to the destination X register in the last clock period. At the end of the second clock period, i delayed is entered into register number 3 along with a flag. Three clock periods later, this code controls the entry of the multiply result into the appropriate X register.

The floating-add instructions require 4 clock periods for execution. The go floating add flag sets during the first clock period of instruction execution. During the second clock period, i delayed is entered into register number 2 along with a flag. Two clock periods later, this code controls the entry of the floating-add result into the appropriate X register.

The normalize unit requires 3 clock periods to execute an instruction. The go normalize flag sets during the first clock period of instruction execution. During the second clock period, i delayed is entered into register number 1 along with a flag. One clock period later, this code controls the entry of the normalize unit result into the appropriate X register.

X REGISTER SELECTION

The X register selection function consists of the circuits on the 4RK7 modules and the translator on the 4RL7 module. It selects a source of X selection bits and gates them to the X registers and instruction issue control simultaneously. The X selection bits, gated by enter X, enable the appropriate X register input data path, and gated by clear X reservation, clear the related register busy flag.

The gates on the 4RK7 module choose selection bits from four sources. The content of register number 1 is selected whenever the flag sets in that register. XJC bits 0 through 2 are selected when XJ to X is present. This occurs when the last eight words of the exchange package are available from CMC for entry into the X registers. The i delayed bits are selected whenever the go two cycle to Xi flag sets. This flag sets whenever an instruction issues from the CIW register which delivers a result to an X register in the following clock period. These three selections do not conflict because of their mutually exclusive origins. If none of the three selections is present, j delayed bits 0 through 2 are selected. These bits are not used.

60420310 A

A REGISTER ACCESS CONTROL

The A register access control function is performed in the 4RN7 and 4RL7 modules. It selects a source of A selection bits and gates that choice to the A registers and to instruction issue control simultaneously. The A selection bits, gated by enter A, enable the appropriate A register input data path, and gated by a clear A reservation, clear the related register busy flag.

The gates on the 4RN7 module choose selection bits from two sources. The i delayed bits are selected whenever the go two cycle to Ai flag is set. This flag sets when a 50 through 57 instruction issues. These instructions deliver a result to an A register in the clock period following issue. XJC delayed bits 0 through 2 are selected when XJ to A, B is present. This occurs when the first eight words of the exchange package are available from CMC for entry in the A registers.

B REGISTER ACCESS CONTROL

The B register access control and selection functions are performed by circuits on five modules. B register access control on modules 4RO7, 4RP7, and 4RQ7 determines the timing for 3-clock-period instructions. The circuits on the 4RN7 and 4RL7 modules provide the selection bits and associated gating signals.

The B register access control consists of two small registers and associated gating. One register provides j delayed for use in register selection. This is a three-bit register which is cleared every clock period. The j designator value is received every clock period and delayed for use during the following clock period.

The second register is the Bj access control register. It is a four-bit register which is cleared every clock period. This register holds a three-bit register designation code plus a flag to indicate a valid reference. The j delayed bits enter this register during normalize (24 and 25) instructions.

The normalize unit is the only functional unit with a 3-clock-period execution time that delivers results to a B register. The go normalize flag sets when the instruction issues. If j is not 0, go normalize gates j delayed into the Bj access control register and becomes the flag that indicates the presence of a valid three-bit code. If j is 0, the gating function is inhibited. In this case, the destination B register is B0 and the data is discarded.

The selection function consists of the gates associated with the B registers on the 4RN7 and 4RL7 modules. It selects a source of B selection bits and gates that choice to the B registers and to instruction issue control simultaneously. The B selection bits, gated by enter B, enable the appropriate B register input data path, and gated by clear B reservation, clear the related register busy flag.

The gates on the 4RN7 module choose selection bits from four sources. The i delayed bits are selected whenever the go 2 cycle to Bi flag is set. This flag sets when a 60 through 67 instruction with i not equal to 0 issues. These instructions deliver a result to a B register in the clock period following issue. XJC delayed bits 0 through 2 are selected when XJ to A, B is present. This occurs when the first eight words of the exchange package are available from CMC for entry in the B registers. The 3 cycle to Bj bits 0 through 2 are selected whenever the 3 cycle to Bj flag sets. The j delayed bits are selected whenever the go 2 cycle to Bj flag sets. This flag sets when an unpack instruction (26) with j not equal to 0 issues. This instruction delivers a result to a B register in the clock period following issue. These selections do not conflict because of their mutually exclusive origins.

BRANCH CONTROL; EXCHANGE SEQUENCE CONTROL

BRANCH CONTROL

The branch portion of the CPU contains the P register and the control flags required to sequence the entry of a new address into it as well as the RJX register and control flags required to execute the return jump instruction. A general description of the branch function and detailed descriptions of the P and RJX registers are given in the CPU 3.0 text.

BRANCH TRANSLATOR

Advance P Flag (APF)

APF is used to advance the content of the P register by one count when a new word is read from the IWS to the CIW register. The flag is set when the data is transmitted to the CIW register. The content of the P register is advanced in the following clock period.

Go Jump Flag (GJF)

GJF is set whenever a branch instruction is ready for execution. If the jump to a new instruction sequence occurs, RJF or JCF is set in the following clock period. If the jump is not taken, the branch instruction in the upper parcel of the CIW register is issued as a pass.

Jump Completed Flag (JCF)

JCF sequences the last step in either a normal branch instruction or a return jump instruction. Its presence allows the branch instruction to issue from the CIW register and the new instruction sequence to begin. JCF also sets during an exchange sequence to begin execution of the new program sequence.

Instruction in CIW

This signal is ANDed with $fm=0$ and CEJ/MEJ to set the error-1 flag in error exit control (CPU 3.10). A 00 instruction with the CEJ/MEJ switch enabled is treated as an illegal instruction.

ENTER P CONTROL AND BRANCH TRANSLATORS

Enter P

Enter P gates the selected address into the P register. Enter P-1 enables the content of the P register into the P buffer register for use in a peripheral processor read P instruction.

Fall Thru

This control condition occurs whenever a branch instruction is executed in which the jump is not taken. When this condition exists, the program continues with the current instruction sequence.

Jump Completed

This condition sets JCF. It is generated during a normal branch instruction or a return jump instruction.

fm=0

This signal indicates that an error exit instruction (00) is in the upper parcel of the CIW register. If the CEJ/MEJ switch is in the ENABLE position, an error exit is initiated.

02 Instruction

This signal indicates that an unconditional jump instruction (02) is in the upper parcel of the CIW register. It is used to clear the IAS (void the IWS).

02 Enable

02 enable is ANDed with GJF to gate K+Bi into the P register during an unconditional jump sequence.

Sample P

Sample P gates the content of the P register into the P buffer register whenever APF is set or the enter P-1 signal (delayed 1 clock period) is present.

BRANCH TEST

The X_j coefficient and exponent inputs to the enter P control and branch translators are used as follows to determine branch conditions for the fall thru translation. They generate the following conditions and their complements every clock period.

$X_j = 0$: bits 0 through 59 equal zero

X_j negative: X_j bit 59 equals one

X_j not in range: X_j bits 48 through 59 equal 3777 or 4000

X_j indefinite: X_j bits 48 through 59 equal 1777 or 6000

The B_i and B_j comparison results (equal, enable, and borrow) determine branch conditions for the fall thru translation as follows:

$B_i = B_j$: equal is a one

$B_i \geq B_j$: enable or borrow, but not both, is a one

$B_i < B_j$: enable and borrow are both ones or zeros

In the B register tests, a quantity consisting of all zeros is greater than a quantity consisting of all ones. Both quantities are considered as signed integers.

EXCHANGE SEQUENCE CONTROL

The exchange sequence involves the interchange of data between the operating registers in the CPU and a package of 16 consecutive storage locations in CM. Information from the operating registers is written into the same 16 storage locations which provide the new operating register information. The data is sequenced from the operating registers into CM by the exchange sequence counter (XSK) and the new data is sequenced from CM to the operating registers by the exchange jump count (XJC). The exchange of data in this sequence terminates the execution of one CPU program and begins the execution of the new CPU program.

The package of data read from CM into the CPU operating registers in the exchange sequence is called the exchange package (Figure 4-1) for the associated CPU program. The exchange package resides in the CPU registers throughout a period of time called the execution interval. The execution interval is terminated by another exchange sequence which returns the exchange package to CM and initiates the execution interval for another exchange package.

The following instructions initiate exchange jumps and select the exchange package that begins execution.

CPU instruction 013

PP instruction 2600, 2610, and 2620

The above instructions are influenced by the CEJ/MEJ switch located on the deadstart panel. A deadstart is required following each change to the position of this switch.

CEJ/MEJ switch in DISABLED position:

013 instruction	Illegal instruction.
2600, 2610, and 2620 PP instructions	Exchange jump to A .

CEJ/MEJ switch in ENABLED position:

013 CPU instruction	If monitor flag (MF) is clear, starting address of exchange package is MA, and MF sets. If MF sets, starting address for exchange package is K+(Bj) and MF clears.
2600 PP instruction	Exchange jump to (A).
2610 PP instruction	If MF is clear, starting address of exchange package is A, and MF sets. If MF is set, this instruction acts as a pass instruction.
2620 PP instruction	If MF is clear, starting address of exchange package is MA in the CPU, and MF sets. If MF is set, this instruction acts as a pass instruction.

EXCHANGE JUMP ADDRESS (XJA)

The XJA circuits consist of the K+Bj adder on the 4LV7 modules and the XJA register on the 3HV7 modules. The K+Bj addition is performed in an 18-bit ones complement mode. The XJA register clears and new data enters at the beginning of each sequence. The XJA register holds the initial address of the exchange sequence.

EXCHANGE SEQUENCE COUNTER (XSK)

The XSK consists of the registers, adder, and associated circuits on the 6HK7 module. Advance XSF from CMC gates XSK+1 into the XSK register as each word of register data enters CMC to be written into CM. This steps the count from 0 through 17 (octal) to zero (once for each word).

XSK bits 0 through 2 sequentially gate the P register and support register contents to the bit 36 through 59 inputs of the SWA. The support registers involved and the order in which they are gated are shown on CPU 3.8.

XSK bit 3 becomes T0 through T7 in CMC. This signal is a one during the first eight counts of XSK, and combined with XSF (CMC 3.1), gates the A, B, P, and support registers into SWA. It changes to a zero at a count of 10 (octal) and gates the contents of the eight X registers into the SWA.

EXCHANGE JUMP COUNT (XJC)

The XJC is generated by the registers, adder, and associated circuits on the 4HU7 module. Advance XJC from CMC gates XJC+1 into the XJC register as exchange package data words arrive from CM. This steps the count from 0 through 17 (octal) to zero (once for each word). The XJC delayed register is incremented 1 clock period later than the XJC register. The XJC delayed register outputs gate the first through eighth words of the exchange package from CMC to the registers. For P, the XJC count is actually delayed by the XJ enter P flag. The XJC register outputs gate the ninth (X0) through sixteenth (X7) words of the exchange package to the X registers.

Advance XJC arrives from CMC after the SAS has accepted the first exchange word. It sets the gate exchange data flag, gates the translated zero from the XJC buffer register into the XJ enter P flag, and gates a count of one into the XJC register. The following clock period, as the P register is being filled, a count of one is gated into the XJC delayed register, and enter RAC is translated from it. Also, at this time, gate exchange data enables the CMC read data output, and XJC delayed bits 0 through 2 are routed through A and B registers access control by XJ to A, B to fill A0. The XJC delayed bits continue to select the appropriate A, B, and support registers for the next 7 clock periods.

When the XJC register content equals 10 (octal), the following conditions occur.

XJ to X is generated to route XJC bits 0 through 2 through X register access control to fill X0 during the next clock period.

The gate exchange data flag is cleared by the next clock pulse inhibiting any further CMC read data outputs and further selection of A, B, or support registers.

The count of seven in the XJC delayed register combined with the next clock pulse gates the eighth word into the respective registers.

The XJC bits continue to select the appropriate X register for the next 7 clock periods. Advance XJC then becomes a zero as XJC is incremented from 17 (octal) to 0.

EXCHANGE JUMP TRANSLATOR AND FLAGS

The exchange jump translator and flags initiate, control, and select the address for the exchange sequence. The exchange sequence begins with the request interrupt flag (RIF). RIF prevents the current program sequence from advancing beyond the current program instruction word. When the current program instruction word has completed executing, the exchange sequence flag (XSF) sets to begin the interchange of data between the operating registers and CM. When the exchange sequence has been completed, JCF sets to begin execution of the new program sequence.

Request Interrupt Flag (RIF)

RIF is a register which sets whenever a control condition exists which requires an exchange sequence. It remains set until the currently executing program has completed the current program instruction word. It then clears, and XSF sets to continue the exchange sequence. Once cleared, RIF cannot set again until XSF clears.

Exchange Sequence Flag (XSF)

XSF is a register which sets at the same time that RIF clears. It indicates that the execution of the current program instruction word is completed and that all previously issued instructions have been completed. XSF remains set until XSF equals 17 (octal). It then clears, and JCF sets to begin execution of the new program sequence.

Program Exit Flag (PXF)

PXF is a register which sets simultaneously with the setting of RIF when the cause of the interruption is a central exchange jump instruction (013).

Exchange Issue Flag (XIF)

XIF is a register which sets 1 clock period after XSF sets. This flag requests CM references for the exchange sequence using an address generated by the BAK register and counter.

SUPPORT REGISTERS

The support registers (4HQ7 and 4HR7 modules) assist the operating registers during the execution of programs. These registers are entered with CM read data during the exchange sequence. The data is used during the execution interval for an exchange package. When the execution interval has been completed, the data in these registers is sent back to CMC (via 4LT7 modules).

REFERENCE ADDRESS - CM (RAC) REGISTER

The RAC register is entered with read data bits 36 through 53 when enter RAC is present. This occurs when the second word in an exchange sequence is read from CM. Absolute CM addresses are formed by adding RAC to relative addresses which are determined by the instructions.

FIELD LENGTH- CM (FLC) REGISTER

The FLC register is entered with read data bits 36 through 53 when enter FLC is present. This occurs when the third word in an exchange sequence is read from CM. Relative CM addresses are compared with FLC. If a relative address equals or exceeds FLC, the address range error bit sets in the exit condition register.

REFERENCE ADDRESS - ECS (RAE) REGISTER

The RAE register is entered with read data bits 36 through 56 when enter RAE is present. This occurs when the fifth word in an exchange sequence is read from CM. An absolute ECS address is formed by adding RAE to the relative address which is determined by the instruction. The lower six bits of the RAE register are always zero.

FIELD LENGTH - ECS (FLE) REGISTER

The FLE register is entered with read data bits 36 through 59 when enter FLE is present. This occurs when the sixth word in an exchange sequence is read from CM. Relative ECS addresses are compared with FLE. If a relative address equals or exceeds FLE, the address range error bit sets in the exit condition register.

MONITOR ADDRESS (MA) REGISTER

The MA register is entered with read data bits 36 through 53 when enter MA is present. This occurs when the seventh word in an exchange sequence is read from CM. This register contains an absolute address that specifies the starting address of an exchange package. Only 18 bits of this 24-bit register are used.

EXIT MODE (EM) REGISTER

The EM register is entered with read data bits 48 through 50 and 57 through 59 when enter EM is present. This occurs when the fourth word in an exchange sequence is read from CM. This register contains the exit mode selections for a program. The exit mode bits control CPU error processing. Only six bits of this 24-bit register are used.

EXCHANGE SEQUENCE DATA MERGE

When an exchange interval is terminated, data in the P and support registers enters SWA in CMC through the 4LT7 modules. The registers transfer data, one register at a time, under control of the XSK. The transfer process for these registers requires 7 clock periods. The order of transfer is as follows: P, RAC, FLC, EM, RAE, FLE, and MA.

ECS INSTRUCTION CONTROL

The ECS instruction control interfaces the CPU and the ECS coupler. The ECS block copy instructions transfer data from CM to ECS (012 instruction) and from ECS to CM (011 instruction). These instructions specify the number of 60-bit words to be transferred as $K+Bj$. The initial CM address is $A0 + RAC$; the initial ECS address is $X0 + RAE$.

ECS RANGE TEST

At the beginning of an ECS to CM or CM to ECS block copy instruction, a test is made (4MP7 and 4HY7 modules) to ensure that all requested addresses are in the assigned ECS field. The sum of $K+Bj$ is added to the lower 24 bits of $X0$. The result is compared with FLE . If the result is greater than FLE , indicating that the block copy will exceed the assigned ECS field, bit 48 sets in the exit condition register (4SY7 module) and an ECS end of transfer (EOT) signal is sent to error exit control (CPU 3.10). This allows the ECS instruction to issue without transferring any words.

CM RANGE TEST

At the beginning of an ECS to CM or CM to ECS block copy instruction, a test is made (4MP7 and 4HY7 modules) to ensure that all requested addresses are in the assigned CM field. The sum of $K+Bj$ is added to $A0$. The result is compared with FLC . If the result is greater than FLC , indicating that the block copy will exceed the assigned CM field, bit 48 sets in the exit condition register (3SY7 module) and an ECS end of transfer (EOT) signal is sent to error exit control (CPU 3.10). This allows the ECS instruction to issue without transferring any words.

CPU TO ECS SEQUENCE

Upon receipt of the coupler accept signal from the ECS coupler, ECS instruction control transmits a 24-bit word plus a parity bit back to the ECS coupler. Under normal block transfer conditions, the 3HX7 modules initially send $K+Bj$ bits 0 through 16 (word count), bits 17 through 23 consisting of zeros, and a parity bit. The second word transferred includes bits 0 through 5 consisting of zeros, $RAE + X0$ bits 6 through 23 (starting ECS address), and a parity bit. If ECS is to perform a flag register operation, FLE and $X0$ bits 23 are both set. This enables the path for $X0$ bits 0 through 23 and a parity bit to the ECS coupler instead of $RAE + X0$.

ERROR EXIT CONTROL

The error exit control monitors and responds to error conditions as shown in Tables 5-2 through 5-4. Depending upon the type of error and the exit mode bits, the program in execution may be interrupted. If the error is an illegal instruction, breakpoint, or address range error on RNI or branch, the program interruption is unconditional. For other types of errors, the exit mode selection bits determine whether or not the program is interrupted. If the exit mode bit is set and the corresponding condition is detected, the program is interrupted. The exit mode bits are contained in word N+3 of the exchange package and are selected as follows:

<u>Condition Bit</u>	<u>Exit Mode Bit</u>	<u>Description</u>
48	48	Address range error
49	49	Infinite result
50	50	Indefinite result
51	57	ECS flag register parity error
52	58	CM input error
53	59	CM parity or double error

Any error condition detected after an exchange jump instruction has started execution is treated as an error for the new program. Tables 5-2 through 5-4 indicate what happens when the various kinds of errors occur.

Each table lists the same error conditions. The error response depends upon the setting of the CEJ/MEJ switch and the state of the monitor flag (MF). The table headings specify the three combinations. The following breakpoint notes are referenced in the tables.

1. Since breakpoint is for an address request to CM, a breakpoint does not occur for an instruction executed from the instruction stack if the instruction entered the instruction stack before selecting breakpoint.
2. The value of P plus RAC when the CPU stops for a breakpoint may not correspond with the value of the breakpoint address because the CPU normally requests two words ahead of P on an RNI.
3. The value of P plus RAC when the CPU stops for a breakpoint on an increment address may not correspond with the value of P plus RAC of the increment instruction. Advancing P is based upon the 60-bit word of instructions entering CIW instead of any given parcel of CIW being executed.

TABLE 5-2. ERROR RESPONSE WITH CEJ/MEJ ENABLED, MF SET

Error Condition	Error Response	
	Exit Mode Selected	Exit Mode Not Selected
Illegal instruction	<ol style="list-style-type: none"> 1. Execute the illegal instruction as if it were a pass. 2. Stop CPU. 3. Store P and exit condition bits at RAC. 4. Clear P. 	<ol style="list-style-type: none"> 1. Execute the illegal instruction as if it were a pass. 2. Stop CPU. 3. Store P and exit condition bits at RAC. 4. Clear P.
Exit condition bit 48 set by an increment read of an address out of range	<ol style="list-style-type: none"> 1. Read all zeros to the selected X register. 2. Stop CPU. 3. Store P and exit condition bits at RAC. 4. Clear P. 	<ol style="list-style-type: none"> 1. Read all zeros to the selected X register. 2. Continue execution.
Exit condition bit 48 set by an increment write of an address out of range	<ol style="list-style-type: none"> 1. Block write operation, content of CM is unchanged. Force read data to all ones on read operation. 2. Stop CPU. 3. Store P and exit condition bits at RAC. 4. Clear P. 	<ol style="list-style-type: none"> 1. Block write operation, content of CM is unchanged. Force read data to all ones on read operation. 2. Continue execution.
Exit condition bit 48 set by RNI or branch address out of range	<ol style="list-style-type: none"> 1. Stop CPU. 2. Store P and exit condition bits at RAC. 3. Clear P. 	<ol style="list-style-type: none"> 1. Stop CPU. 2. Store P and exit condition bits at RAC. 3. Clear P.
Exit condition bit 48 set by an ECS address range check	<ol style="list-style-type: none"> 1. Force ECS instruction to execute as a pass instruction. 2. Stop CPU. 3. Store P and exit condition bits at RAC. 4. Clear P. 	<ol style="list-style-type: none"> 1. Force ECS instruction to execute as a pass instruction. 2. Exit to next 60-bit word. 3. Continue execution with next 60-bit word.

TABLE 5-2. ERROR RESPONSE WITH CEJ/MEJ ENABLED MF SET (Cont'd)

Error Condition	Error Response	
	Exit Mode Selected	Exit Mode Not Selected
Infinite condition (bit 49) Indefinite condition (bit 50) ECS flag register parity (bit 51) CMC input error condition (bit 52) CM data error condition (bit 53)	<ol style="list-style-type: none"> 1. Stop CPU. 2. Store P and exit condition bits at RAC. 3. Clear P. 	<ol style="list-style-type: none"> 1. Continue execution.
CMC input error condition (bit 52)	<ol style="list-style-type: none"> 1. Block write operation, content of CM is unchanged. 2. Stop CPU. 3. Store P and exit condition bits at RAC. 4. Clear P. 	<ol style="list-style-type: none"> 1. Block write operation, content of CM is unchanged. 2. Continue execution.
00 instruction	<ol style="list-style-type: none"> 1. Stop CPU. 2. Store P and exit condition bits at RAC. 3. Clear P. 	<ol style="list-style-type: none"> 1. Stop CPU. 2. Store P and exit condition bits at RAC. 3. Clear P.
Breakpoint signal from CMC (refer to breakpoint notes)	<ol style="list-style-type: none"> 1. Execute remaining parcels of 60-bit word currently executing. 2. Stop CPU. 3. Store P and exit condition bits at RAC. 4. Clear P. 	<ol style="list-style-type: none"> 1. Execute remaining parcels of 60-bit word currently executing. 2. Stop CPU. 3. Store P and exit condition bits at RAC. 4. Clear P.

TABLE 5-3. ERROR RESPONSE WITH CEJ/MEJ ENABLED, MF CLEAR

Error Condition	Error Response	
	Exit Mode Selected	Exit Mode Not Selected
Illegal instruction	<ol style="list-style-type: none"> 1. Execute the illegal instruction as if it were a pass. 2. Stop CPU. 3. Store P and exit condition bits at RAC. 4. Clear P. 5. Exchange jump to MA and set MF. 	<ol style="list-style-type: none"> 1. Execute the illegal instruction as if it were a pass. 2. Stop CPU. 3. Store P and exit condition bits at RAC. 4. Clear P. 5. Exchange jump to MA and set MF.
Exit condition bit 48 set by an increment read of an address out of range	<ol style="list-style-type: none"> 1. Read all zeros to the selected X register. 2. Stop CPU. 3. Store P and exit condition bits at RAC. 4. Clear P. 5. Exchange jump to MA and set MF. 	<ol style="list-style-type: none"> 1. Read all zeros to the selected X register. 2. Continue execution.
Exit condition bit 48 set by an increment write of an address out of range	<ol style="list-style-type: none"> 1. Block write operation, content of CM is unchanged. 2. Stop CPU. 3. Store P and exit condition bits at RAC. 4. Clear P. 5. Exchange jump to MA and set MF. 	<ol style="list-style-type: none"> 1. Block write operation, content of CM is unchanged. 2. Continue execution.
Exit condition bit 48 set by an RNI or branch address out of range	<ol style="list-style-type: none"> 1. Stop CPU. 2. Store P and exit condition bits at RAC. 3. Clear P. 4. Exchange jump to MA and set MF. 	<ol style="list-style-type: none"> 1. Stop CPU. 2. Store P and exit condition bits at RAC. 3. Clear P. 4. Exchange jump to MA and set MF.
Exit condition bit 48 set by an ECS address range check	<ol style="list-style-type: none"> 1. Force ECS instruction to execute as a pass instruction. 2. Stop CPU. 3. Store P and exit condition bits at RAC. 4. Clear P. 5. Exchange jump to MA and set MF. 	<ol style="list-style-type: none"> 1. Force ECS instruction to execute as a pass instruction. 2. Continue execution with next 60-bit word.

TABLE 5-3. ERROR RESPONSE WITH CEJ/MEJ ENABLED, MF CLEAR (Cont'd)

Error Condition	Error Response	
	Exit Response Selected	Exit Mode Not Selected
Infinite condition (bit 49) Indefinite condition (bit 50) ECS flag register parity (bit 51) CMC input error condition (bit 52) CM data error condition (bit 53)	1. Stop CPU. 2. Store P and exit condition bits at RAC. 3. Clear P. 4. Exchange jump to MA and set MF.	1. Continue execution
CMC input error condition (bit 52)	1. Block write operation, content of CM is unchanged. Force read data to all ones on read operation. 2. Stop CPU. 3. Store P and exit condition bits at RAC. 4. Clear P. 5. Exchange jump to MA and set MF.	1. Block write operation, content of CM is unchanged. Force read data to all ones on read operation. 2. Continue execution.
00 instruction	1. Stop CPU. 2. Store P and exit condition bits at RAC. 3. Clear P. 4. Exchange jump to MA and set MF.	1. Stop CPU. 2. Store P and exit condition bits at RAC. 3. Clear P. 4. Exchange jump to MA and set MF.
Breakpoint signal from CMC (refer to breakpoint notes)	1. Execute remaining parcels of 60-bit word currently executing. 2. Stop CPU. 3. Store P and exit condition bits at RAC. 4. Clear P. 5. Exchange jump to MA and set MF.	1. Execute remaining parcels of 60-bit word currently executing. 2. Stop CPU. 3. Store P and exit condition bits at RAC. 4. Clear P. 5. Exchange jump to MA and set MF.

TABLE 5-4. ERROR RESPONSE WITH CEJ/MEJ DISABLED

Error Condition	Error Response	
	Exit Mode Selected	Exit Mode Not Selected
Illegal instruction	<ol style="list-style-type: none"> 1. Execute the illegal instruction as if it were a pass. 2. Stop CPU. 3. Store P and exit condition bits at RAC. 4. Clear P. 	<ol style="list-style-type: none"> 1. Execute the illegal instruction as if it were a pass. 2. Stop CPU. 3. Store P and exit condition bits at RAC. 4. Clear P.
Exit condition bit 48 set by an increment read of an address out of range	<ol style="list-style-type: none"> 1. Read all zeros to the selected X register. 2. Stop CPU. 3. Store P and exit condition bits at RAC. 4. Clear P. 	<ol style="list-style-type: none"> 1. Read all zeros to the selected X register. 2. Continue execution.
Exit condition bit 48 set by an increment write of an address out of range	<ol style="list-style-type: none"> 1. Block write operation, content of CM is unchanged. 2. Stop CPU. 3. Store P and exit condition bits at RAC. 4. Clear P. 	<ol style="list-style-type: none"> 1. Block write operation, content of CM is unchanged. 2. Continue execution.
Exit condition bit 48 set by RNI or branch address out of range	<ol style="list-style-type: none"> 1. Stop CPU. 2. Store P and exit condition bits at RAC. 3. Clear P. 	<ol style="list-style-type: none"> 1. Stop CPU.
Exit condition bit 48 set by ECS address range check	<ol style="list-style-type: none"> 1. Force ECS instruction to execute as a pass. 2. Stop CPU. 3. Store P and exit condition bits at RAC. 4. Clear P. 	<ol style="list-style-type: none"> 1. Force ECS instruction to execute as a pass. 2. Continue execution with next 60-bit word.

TABLE 5-4. ERROR RESPONSE WITH CEJ/MEJ DISABLED (Cont'd)

Error Condition	Error Response	
	Exit Mode Selected	Exit Mode Not Selected
Infinite condition (bit 49) Indefinite condition (bit 50) ECS flag register parity (bit 51) CMC input error condition (bit 52) CM data error condition (bit 53)	<ol style="list-style-type: none"> 1. Stop CPU. 2. Store P and exit condition bits at RAC. 3. Clear P. 	<ol style="list-style-type: none"> 1. Continue execution.
CMC input error condition (bit 52)	<ol style="list-style-type: none"> 1. Block write operation, content of CM is unchanged. Force read data to all ones on read operation. 2. Stop CPU. 3. Store P and exit condition bits at RAC. 4. Clear P. 	<ol style="list-style-type: none"> 1. Block write operation, content of CM is unchanged. Force read data to all ones on read operation. 2. Continue execution.
00 instruction	<ol style="list-style-type: none"> 1. Stop CPU. 	<ol style="list-style-type: none"> 1. Stop CPU.
Breakpoint signal from CMC (refer to breakpoint notes)	<ol style="list-style-type: none"> 1. Execute remaining parcels of 60-bit instruction word. 2. Stop CPU. 3. Store P and exit condition bits at RAC. 4. Clear P. 	<ol style="list-style-type: none"> 1. Execute remaining parcels of 60-bit instruction word. 2. Stop CPU. 3. Store P and exit condition bits at RAC. 4. Clear P.

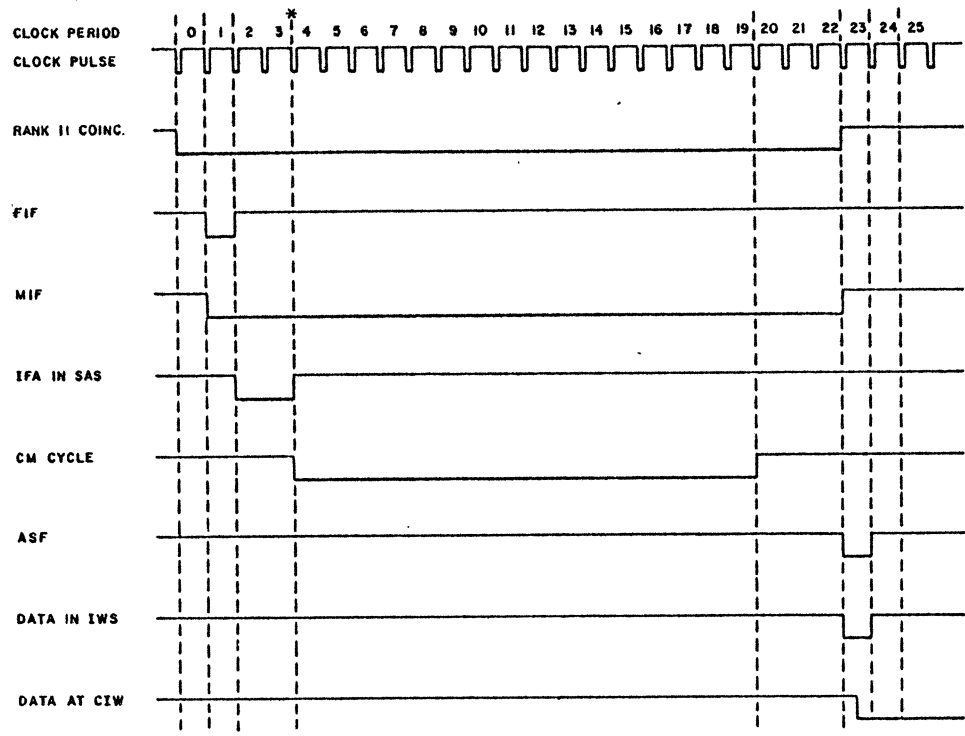
4

3



2

1



* ALL TIMES FROM TIME FOUR MAY BE DELAYED ONE CLOCK PERIOD
IF MEMORY ENABLE IS NOT UP WHEN FIF IS SET.

D

C

B

A

D

C

B

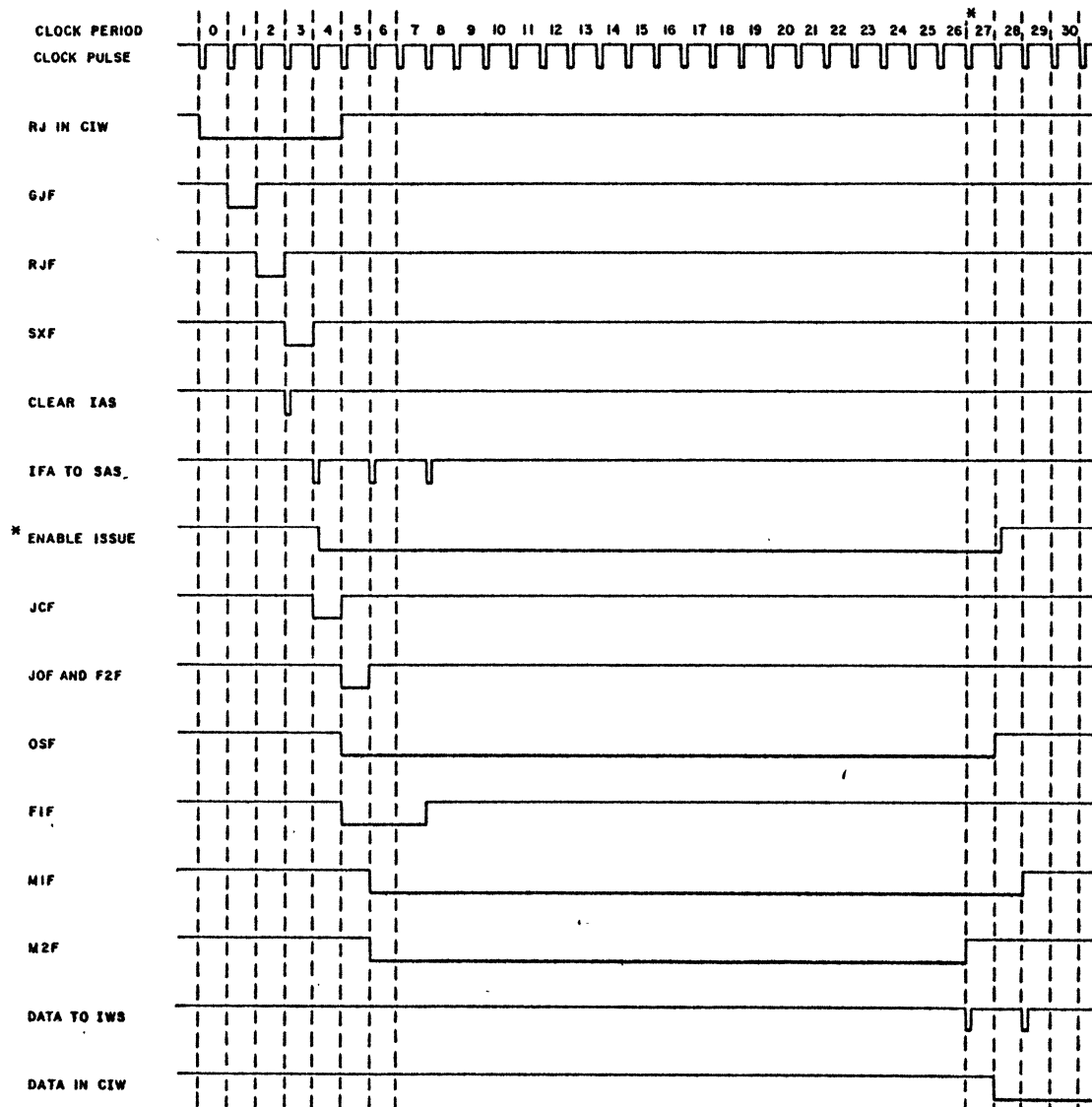
A

4

3

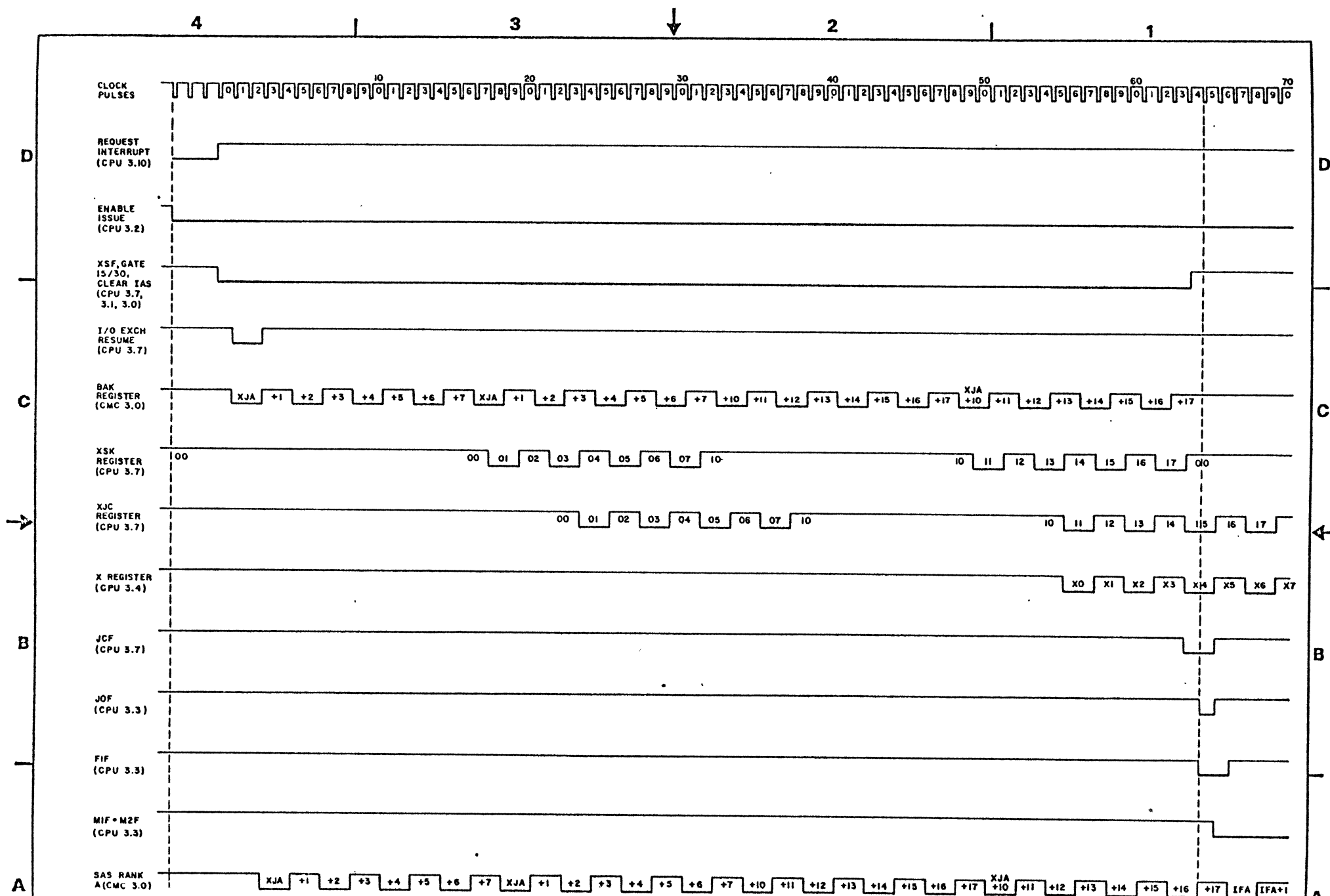
2

1



* ALL TIMES FROM TIME 27 MAY BE DELAYED ONE CLOCK PERIOD
IF MEMORY ENABLE IS UP WHEN SXF IS SET, TIMING ASSUMES NO I/O
REQUESTS TO CMC AND NO CONFLICTS.

CONTROL DATA DEVELOPMENT DIVISION	TIMING DIAGRAM RETURN JUMP	CODE IDENT 34010	DWG NO 60420310	REV A
	C	SHEET	PAGE 5-2-157	



NOTES:
 1. EXCHANGE JUMP DESCRIPTION IS LOCATED IN BOTH CPU AND CMC PARTS OF THIS VOLUME.

TIMING CHARTS

Part 6 of the manual contains command timing information for all CP instructions. The instructions are presented in numerical order. The timing charts list the sequence of events during execution of the instructions. Delays that may occur during execution are explained at the end of each chart. The events are keyed to the detailed-modules diagrams.

0100K RETURN JUMP TO K

EXECUTION TIME

The minimum execution time for a return jump instruction is 28 clock periods. The return jump sequence begins as soon as the instruction enters the upper parcel of the CIW register. The sequence does not wait for the completion of previously issued instructions. The sequence of events for this instruction is:

- CP00 010 instruction in upper parcel of CIW register. (CPU 3.1)
 Instruction does not issue.
 Set GJF. (CPU 3.7)
- CP01 010 instruction in upper parcel of CIW register.
 Instruction does not issue.
 Transmit (P) to RJX register. (CPU 3.0)
 Transmit K to P register. (CPU 3.0)
 Set RJF. (CPU 3.3)
 Clear GJF. (CPU 3.7)
- CP02 010 instruction in upper parcel of CIW register.
 Instruction does not issue.
 Void (IWS). Clear (IAS). (CPU 3.0)
 Transmit (P) + (RAC) to IFA register. (CMC 3.0)
 Set SXF. (CPU 3.3)
 Clear RJF. (CPU 3.3)

- CP03 010 instruction in upper parcel of CIW register.
Instruction does not issue.
Transmit (IFA) to SAS. (CMC 3.0)
Tag for return jump exit. (CMC 3.1)
Transmit (RJX) to SWS. (CMC 3.1)
Advance (P). (CPU 3.0)
Set JCF. (CPU 3.7)
Clear SXF. (CPU 3.3)
- CP04 010 instruction in upper parcel of CIW register.
Instruction issues.
No coincidence in IAS.
Read a blank word from IWS to CIW register.
Set JOF. (CPU 3.3)
Set OSF. (CPU 3.3)
Transmit (P) + (RAC) to IFA register. (CMC 3.0)
Set F1F, F2F. (CPU 3.3)
Clear JCF. (CPU 3.7)
- CP05 No instruction in upper parcel of CIW register.
Transmit (P) to NSA register. (CPU 3.0)
Set M1F, M2F. (CPU 3.3)
OSF remains set. (CPU 3.3)
Transmit (IFA) to SAS. (CMC 3.0)
Tag for read to IWS. (CMC 3.1)
Advance (IFA). (CMC 3.0)
Clear JOF. (CPU 3.3)
Clear F2F. (CPU 3.3)
- CP06 No instruction in upper parcel of CIW register.
No coincidence in IAS.
Read a blank word from IWS to CIW register.
OSF remains set. (CPU 3.3)
- CP07 No instruction in upper parcel of CIW register.
No coincidence in IAS.
Read a blank word from IWS to CIW register.
OSF remains set. (CPU 3.3)
Transmit (IFA) to SAS. (CMC 3.0)
Tag for read to IWS. (CMC 3.1)
Advance (IFA). (CMC 3.0)

First fetch address leaves SAS for CM. (CMC 3.0)
Clear F1F. (CPU 3.3)

- CP08 No instruction in upper parcel of CIW register.
No coincidence in IAS.
Read a blank word from IWS to CIW register.
OSF remains set. (CPU 3.3)
First fetch CM read cycle begins.
- CP09 No instruction in upper parcel of CIW register.
No coincidence in IAS.
Read a blank word from IWS to CIW register.
OSF remains set. (CPU 3.3)
Second fetch address leaves SAS for CM. (CMC 3.0)
- CP10 No instruction in upper parcel of CIW register.
No coincidence in IAS.
Read a blank word from IWS to CIW register.
OSF remains set. (CPU 3.3)
Second fetch CM read cycle begins.
- CP26 No instruction in upper parcel of CIW register.
No coincidence in IAS.
Read a blank word from ISW to CIW register.
OSF remains set. (CPU 3.3)
Transmit first fetch instruction word to IWS. (CPU 3.0)
Transmit (NSA) to IAS. (CPU 3.0)
Shift IWS and IAS one word position. (CPU 3.0)
Advance (NSA). (CPU 3.0)
Clear M2F. (CPU 3.3)
- CP27 No instruction in upper parcel of CIW register.
Coincidence in IAS.
Read a valid instruction word from IWS to CIW register.
Clear OSF. (CPU 3.3)
- CP28 Next instruction in upper parcel of CIW register.
Instruction may issue.
Transmit second fetch instruction word to IWS. (CPU 3.0)
Transmit (NSA) to IAS. (CPU 3.0)
Shift IWS and IAS one word position. (CPU 3.0)
Advance (NSA). (CPU 3.0)
Clear M1F. (CPU 3.3)

EXECUTION DELAYS

A delay occurs in the return jump sequence if M1F sets during CP02 in the timing sequence listed previously. SXF does not set until M1F clears. This condition exists if the instruction stack fetch control has requested one or more fetch instruction words which have not arrived at the instruction stack by CP02. This may occur if the sequence of instructions prior to the return jump instruction were straight-line coding for several instruction words. Execution of the commands in CP02 is delayed by the number of clock periods required to complete the instruction fetches and clear M1F. All commands in later clock periods are delayed by the same amount.

A delay occurs in the return jump sequence if an SAS backup condition or memory enable exists during CP03. This condition may exist if CM references unrelated to the return jump sequence caused a bank conflict prior to CP03. Execution of the CP03 commands is delayed until NBF sets, indicating that the SAS backup has been resolved. All commands in later clock periods are delayed by the same amount.

A delay occurs in the return jump sequence if the content of IFA is not transmitted to the SAS during CP05 because of an SAS backup condition. A backup condition can exist in the SAS during CP05 because of a CM bank conflict between the store exit reference of the return jump sequence and an unrelated CM reference, or between two unrelated CM references. The remainder of the return jump sequence is delayed by the number of clock periods required to resolve the conflict or backup condition.

A delay occurs in the return jump sequence if a CM bank conflict exists during CP07. This occurs if the address of the first fetch instruction word requires a CM bank which is busy. The arrival of the first fetch instruction word at the IWS is delayed by the number of clock period required for the bank to complete the read cycle required in the previous reference.

011jK BLOCK COPY (Bj) + K WORDS FROM ECS TO CM

EXECUTION TIME

This instruction remains in the CIW register until the block copy has been completed. This prevents issue of the instruction, and therefore, all following instructions until the end of instruction execution.

The sequence of events at the start of a block copy instruction is:

- CP00 011 instruction in upper parcel of CIW register. (CPU 3.1)
 Instruction does not issue.
 All operating registers free.
 F1F not set. (CPU 3.3)
 ECS request. (CMC 3.5)

- CP01 011 instruction in upper parcel of CIW register.
 Instruction does not issue.
 Memory enable. (CMC 3.5)

- CP02 011 instruction in upper parcel of CIW register.
 Instruction does not issue.

- CP03 011 instruction in upper parcel of CIW register.
 Instruction does not issue.
 No ECS range error. (CPU 3.9)
 Request coupler. (CPU 3.9)

The sequence of events at the termination of a block copy instruction is:

- CP00 011 instruction in upper parcel of CIW register. (CPU 3.1)
 Instruction does not issue.
 ECS end of transfer. (CMC 3.5)
 ECS mode. (CMC 3.5)

- CP01 011 instruction in upper parcel of CIW register.
 Instruction does not issue.
 End of transfer. (CPU 3.9)
 No PPS exchange request. (CMC 3.5)
 Memory enable. (CMC 3.5)
- CP02 011 instruction in upper parcel of CIW register.
 Instruction does not issue.
- CP03 011 instruction in upper parcel of CIW register.
 Instruction does not issue.
 Enter P register with P+1. (CPU 3.0)
- CP04 011 instruction in upper parcel of CIW register.
 Instruction issues.
 Enter CIW register with new instruction word.
- CP05 Next instruction in upper parcel of CIW register.
 Instruction may issue.

The sequence of events in a block copy instruction terminated by a range error is:

- CP00 011 instruction in upper parcel of CIW register. (CPU 3.1)
 Instruction does not issue.
 All operating registers free.
 F1F not set. (CPU 3.3)
 ECS request. (CMC 3.5)
- CP01 011 instruction in upper parcel of CIW register.
 Instruction does not issue.
 Memory enable. (CMC 3.5)
- CP02 011 instruction in upper parcel of CIW register.
 Instruction does not issue.
- CP03 011 instruction in upper parcel of CIW register.
 Instruction does not issue.
 ECS range error. (CPU 3.9)
 ECS end of transfer. (CMC 3.5)

CP04 011 instruction in upper parcel of CIW register.
Instruction does not issue.

CP05 011 instruction in upper parcel of CIW register.
Instruction does not issue.
End of transfer. (CPU 3.9)
Address range error in CM register. (CPU 3.10)

CP06 011 instruction in upper parcel of CIW register.
Instruction does not issue.

CP07 011 instruction in upper parcel of CIW register.
Instruction does not issue.
Enter P register with P+1. (CPU 3.0)

CP08 011 instruction in upper parcel of CIW register.
Instruction issues.
Enter CIW register with new instruction word.

CP09 Next instruction in upper parcel of CIW register.
Instruction may issue.

EXECUTION DELAYS

The execution of this instruction does not begin until the following four conditions have been satisfied.

- All operating registers must be free. This implies that all previously issued instructions have delivered their results to the operating registers. This normally does not occur in the clock period in which this instruction first appears in the CIW register.
- The F1F must not be set. F1F is clear unless a CM bank conflict has caused an SAS backup condition which prevented an instruction fetch address from leaving the IFA register.
- ECS request must be present.
- Memory enable must be present during the next clock period. Memory enable is present every other clock period.

After the request coupler signal is sent, the transfer is controlled by the ECS coupler until an ECS end of transfer signal is sent to CMC.

Termination of the ECS transfer because of an ECS error is almost the same as a normal termination. The difference is that during CP04, the next instruction, instead of a new instruction word, is entered in the upper parcel of the CIW register.

012jK BLOCK COPY (Bj) + K WORDS FROM CM TO ECS

EXECUTION TIME

Same as 011 instruction.

EXECUTION DELAYS

Same as 011 instruction.

013jK EXCHANGE JUMP TO (Bj) + K

EXECUTION TIME

The minimum execution time for this instruction is 91 clock periods, the minimum time from the arrival of this instruction in the upper parcel of the CIW register until the arrival of the first instruction for the next program. This instruction issues from the CIW register in the second clock period of the sequence. A 60-bit word of all zeros is read into the CIW register at this time, voiding any following instructions in the current instruction word. The IAS is cleared to all zeros, which nullifies the content of the IWS. No further instructions can enter the CIW register until the exchange sequence has been completed and the IWS is loaded with a new sequence of instructions.

The sequence of events for this instruction is:

- CP00 013 instruction in upper parcel of CIW register. (CPU 3.1)
Instruction does not issue.
Bj register free. (CPU 3.2)
Set PXF. (CPU 3.7)
Set RIF. (CPU 3.7)
- CP01 013 instruction in upper parcel of CIW register.
Instruction issues.
Transmit a blank word to CIW register. (CPU 3.1)
Transmit K + (Bj) and (RAC) to XJA register. (CPU 3.7)
RIF remains set. (CPU 3.7)
Clear PXF. (CPU 3.7)
- CP02 No instruction in upper parcel of CIW register.
All operating registers free.
F1F not set. (CPU 3.3)
Set XSF. (CPU 3.7)
Clear RIF. (CPU 3.7)
- CP03 No instruction in upper parcel of CIW register.
Nullify (IWS). Clear (IAS). (CPU 3.0)
Transmit (XJA) to BAK register. (CMC 3.0)
XSF remains set. (CPU 3.7)
Set XIF. (CPU 3.7)
- CP04 No instruction in upper parcel of CIW register.
Transmit (BAK) to SAS. (CMC 3.0)
Tag for exchange sequence. (CMC 3.1)
Advance (BAK). (CMC 3.0)
XSF remains set. (CPU 3.7)
XIF remains set. (CPU 3.7)
- CP06 No instruction in upper parcel of CIW register.
Transmit (BAK) to SAS. (CMC 3.0)
Tag for exchange sequence. (CMC 3.1)
Advance (BAK). (CMC 3.0)
First address leaves SAS for CM. (CMC 3.0)
XSF remains set. (CPU 3.7)
XIF remains set. (CPU 3.7)

CP08 No instruction in upper parcel of CIW register.
 Transmit (BAK) to SAS. (CMC 3.0)
 Tag for exchange sequence. (CMC 3.1)
 Advance (BAK). (CMC 3.0)
 Begin read cycle for first CM reference.
 Second address leaves SAS for CM. (CMC 3.0)
 XSF remains set. (CPU 3.7)
 XIF remains set. (CPU 3.7)

CP10 No instruction in upper parcel of CIW register.
 Transmit (BAK) to SAS. (CMC 3.0)
 Tag for exchange sequence. (CMC 3.1)
 Advance (BAK). (CMC 3.0)
 Begin read cycle for second CM reference.
 Third address leaves SAS for CM. (CMC 3.0)
 XSF remains set. (CPU 3.7)
 XIF remains set. (CPU 3.7)

CP12 No instruction in upper parcel of CIW register.
 Transmit (BAK) to SAS. (CMC 3.0)
 Tag for exchange sequence. (CMC 3.1)
 Advance (BAK). (CMC 3.0)
 Begin read cycle for third CM reference.
 Fourth address leaves SAS for CM.
 XSF remains set. (CPU 3.7)
 XIF remains set. (CPU 3.7)

CP14 No instruction in upper parcel of CIW register.
 Transmit (BAK) to SAS. (CMC 3.0)
 Tag for exchange sequence. (CMC 3.1)
 Advance (BAK). (CMC 3.0)
 First CM reference enters read data holding register. (CMC 3.2)
 Begin read cycle for fourth CM reference.
 Fifth address leaves SAS for CM. (CMC 3.0)
 XSF remains set. (CPU 3.7)
 XIF remains set. (CPU 3.7)

- CP16 No instruction in upper parcel of CIW register.
 Transmit (BAK) to SAS. (CMC 3.0)
 Tag for exchange sequence. (CMC 3.1)
 Advance (BAK). (CMC 3.0)
 Begin read cycle for fifth CM reference.
 Sixth address leaves SAS for CM. (CMC 3.0)
 Advance (XSK) to 10 (octal). (CPU 3.7)
 XSF remains set. (CPU 3.7)
 XIF remains set. (CPU 3.7)
- CP18 No instruction in upper parcel of CIW register.
 Transmit (BAK) to SAS. (CMC 3.0)
 Tag for exchange sequence. (CMC 3.1)
 Advance (BAK). (CMC 3.0)
 Begin read cycle for sixth CM reference.
 Seventh address leaves SAS for CM. (CMC 3.0)
 XSF remains set. (CPU 3.7)
 XIF remains set. (CPU 3.7)
- CP19 Transmit (B0), (A0), and (P) to SWA register. (CMC 3.1)
 Reload BAK. (CMC 3.1)
- CP20 No instruction in upper parcel of CIW register.
 Transmit (BAK) to SAS. (CMC 3.0)
 Tag for exchange sequence. (CMC 3.1)
 Advance (BAK). (CMC 3.0)
 Begin read cycle for seventh CM reference.
 Eighth address leaves SAS for CM. (CMC 3.0)
 Advance (XSK) to 01 (octal). (CPU 3.7)
 XSF remains set. (CPU 3.7)
 XIF remains set. (CPU 3.7)
- CP21 Transmit (B1), (A1), and (RAC) to SWA register. (CMC 3.1)

- CP22 No instruction in upper parcel of CIW register.
 Transmit (BAK) to SAS. (CMC 3.0)
 Tag for exchange sequence. (CMC 3.1)
 Advance (BAK). (CMC 3.0)
 Begin read cycle for eighth CM reference.
 Ninth address leaves SAS for CM. (CMC 3.0)
 Advance (XSK) to 02 (octal). (CPU 3.7)
 XSF remains set. (CPU 3.7)
 XIF remains set. (CPU 3.7)
- CP23 Transmit (B2), (A2), and (FLC) to SWA register. (CMC 3.1)
- CP24 No instruction in upper parcel of CIW register.
 Transmit (BAK) to SAS. (CMC 3.0)
 Tag for exchange sequence. (CMC 3.1)
 Advance (BAK). (CMC 3.0)
 Begin write cycle for ninth CM reference.
 10th address leaves SAS for CM. (CMC 3.0)
 Advance (XSK) to 03 (octal). (CPU 3.7)
 XSF remains set. (CPU 3.7)
 XIF remains set. (CPU 3.7)
- CP25 Transmit (B3), (A3), and (EM) to SWA register. (CMC 3.1)
- CP26 No instruction in upper parcel of CIW register.
 Transmit (BAK) to SAS. (CMC 3.0)
 Tag for exchange sequence. (CMC 3.1)
 Advance (BAK). (CMC 3.0)
 Transmit read data to B0, A0, and P registers. (CPU 3.5, 3.0)
 Begin write cycle for 10th CM reference.
 11th address leaves SAS for CM. (CMC 3.0)
 Advance (XSK) to 04 (octal). (CPU 3.7)
 XSF remains set. (CPU 3.7)
 XIF remains set. (CPU 3.7)
- CP27 Transmit (B4), (A4), and (RAE) to SWA register. (CMC 3.1)

- CP28 No instruction in upper parcel of CIW register.
 Transmit (BAK) to SAS. (CMC 3.0)
 Tag for exchange sequence. (CMC 3.1)
 Advance (BAK). (CMC 3.0)
 Transmit read data to B1, A1, and RAC registers. (CPU 3.5, 3.8)
 Begin write cycle for 11th CM reference.
 12th address leaves SAS for CM. (CMC 3.0)
 Advance (XSK) to 05 (octal). (CPU 3.7)
 XSF remains set. (CPU 3.7)
 XIF remains set. (CPU 3.7)
- CP29 Transmit (B5), (A5), and (FLE) to SWA register. (CMC 3.1)
- CP30 No instruction in upper parcel of CIW register.
 Transmit (BAK) to SAS. (CMC 3.0)
 Tag for exchange sequence. (CMC 3.1)
 Advance (BAK). (CMC 3.0)
 Transmit read data to B2, A2, and FLC registers. (CPU 3.5, 3.8)
 Begin write cycle for 12th CM reference.
 13th address leaves SAS for CM. (CMC 3.0)
 Advance (XSK) to 06 (octal). (CPU 3.7)
 XSF remains set. (CPU 3.7)
 XIF remains set. (CPU 3.7)
- CP31 Transmit (B6), (A6), and (MA) to SWA register. (CMC 3.1)
- CP32 No instruction in upper parcel of CIW register.
 Transmit (BAK) to SAS. (CMC 3.0)
 Tag for exchange sequence. (CMC 3.1)
 Advance (BAK). (CMC 3.0)
 Transmit read data to B3, A3, and EM registers (CPU 3.5, 3.8)
 Begin write cycle for 13th CM reference.
 14th address leaves SAS for CM. (CMC 3.0)
 Advance (XSK) to 07 (octal). (CPU 3.7)
 XSF remains set. (CPU 3.7)
 XIF remains set. (CPU 3.7)
- CP33 Transmit (B7) and (A7) to SWA register. (CMC 3.1)

- CP34 No instruction in upper parcel of CIW register.
Transmit (BAK) to SAS. (CMC 3.0)
Tag for exchange sequence. (CMC 3.1)
Advance (BAK). (CMC 3.0)
Transmit read data to B4, A4, and RAE registers. (CPU 3.5, 3.8)
Begin write cycle for 14th CM reference.
15th address leaves SAS for CM. (CMC 3.0)
Advance (XSK) to 10 (octal). (CPU 3.7)
XSF remains set. (CPU 3.7)
XIF remains set. (CPU 3.7)
- CP36 No instruction in upper parcel of CIW register.
Transmit (BAK) to SAS. (CMC 3.0)
Tag for exchange sequence. (CMC 3.1)
Advance (BAK). (CMC 3.0)
Transmit read data to B5, A5, and FLE registers. (CPU 3.5, 3.8)
Begin write cycle for 15th CM reference.
16th address leaves SAS for CM. (CMC 3.0)
XSF remains set. (CPU 3.7)
XIF remains set. (CPU 3.7)
- CP38 No instruction in upper parcel of CIW register.
Transmit (BAK) to SAS. (CMC 3.0)
Tag for exchange sequence. (CMC 3.1)
Advance (BAK). (CMC 3.0)
Transmit read data to B6, A6, and MA registers. (CPU 3.5, 3.8)
Begin write cycle for 16th CM reference.
17th address leaves SAS for CM. (CMC 3.0)
XSF remains set. (CPU 3.7)
XIF remains set. (CPU 3.7)
- CP40 No instruction in upper parcel of CIW register.
Transmit (BAK) to SAS. (CMC 3.0)
Tag for exchange sequence. (CMC 3.1)
Advance (BAK). (CMC 3.0)
Transmit read data to B7 and A7 registers. (CPU 3.5, 3.8)
Begin read cycle for 17th CM reference.
18th address leaves SAS for CM. (CMC 3.0)
XSF remains set. (CPU 3.7)
XIF remains set. (CPU 3.7)

CP42 No instruction in upper parcel of CIW register.
 Transmit (BAK) to SAS. (CMC 3.0)
 Tag for exchange sequence. (CMC 3.1)
 Advance (BAK). (CMC 3.0)
 Begin read cycle for 18th CM reference.
 19th address leaves SAS for CM. (CMC 3.0)
 XSF remains set. (CPU 3.7)
 XIF remains set. (CPU 3.7)

CP44 No instruction in upper parcel of CIW register.
 Transmit (BAK) to SAS. (CMC 3.0)
 Tag for exchange sequence. (CMC 3.1)
 Advance (BAK). (CMC 3.0)
 Begin read cycle for 19th CM reference.
 20th address leaves SAS for CM. (CMC 3.0)
 XSF remains set. (CPU 3.7)
 XIF remains set. (CPU 3.7)

CP46 No instruction in upper parcel of CIW register.
 Transmit (BAK) to SAS. (CMC 3.0)
 Tag for exchange sequence. (CMC 3.1)
 Advance (BAK). (CMC 3.0)
 Begin read cycle for 20th CM reference.
 21st address leaves SAS for CM. (CMC 3.0)
 XSF remains set. (CPU 3.7)
 XIF remains set. (CPU 3.7)

CP48 No instruction in upper parcel of CIW register.
 Transmit (BAK) to SAS. (CMC 3.0)
 Tag for exchange sequence. (CMC 3.1)
 Advance (BAK). (CMC 3.0)
 Begin read cycle for 21st CM reference.
 22nd address leaves SAS for CM. (CMC 3.0)
 XSF remains set. (CPU 3.7)
 XIF remains set. (CPU 3.7)

CP50 No instruction in upper parcel of CIW register.
 Transmit (BAK) to SAS. (CMC 3.0)
 Tag for exchange sequence. (CMC 3.1)
 Reload (BAK). (CMC 3.0)
 Begin read cycle for 22nd CM reference.
 23rd address leaves SAS for CM. (CMC 3.0)
 XSF remains set. (CPU 3.7)
 XIF remains set. (CPU 3.7)

CP51 Transmit (X0) to SWA register. (CMC 3.1)

CP52 No instruction in upper parcel of CIW register.
 Transmit (BAK) to SAS. (CMC 3.0)
 Tag for exchange sequence. (CMC 3.1)
 Advance (BAK). (CMC 3.0)
 Begin read cycle for 23rd CM reference.
 24th address leaves SAS for CM. (CMC 3.0)
 Advance (XSK) to 11 (octal). (CPU 3.7)
 XSF remains set. (CPU 3.7)
 XIF remains set. (CPU 3.7)

CP53 Transmit (X1) to SWA register. (CMC 3.1)

CP54 No instruction in upper parcel of CIW register.
 Transmit (BAK) to SAS. (CMC 3.0)
 Tag for exchange sequence. (CMC 3.1)
 Advance (BAK). (CMC 3.0)
 Begin read cycle for 24th CM reference.
 25th address leaves SAS for CM. (CMC 3.0)
 Advance (XSK) to 12 (octal). (CPU 3.7)
 XSF remains set. (CPU 3.7)
 XIF remains set. (CPU 3.7)

CP55 Transmit (X2) to SWA register. (CMC 3.1)

CP56 No instruction in upper parcel of CIW register.
 Transmit (BAK) to SAS. (CMC 3.0)
 Tag for exchange sequence. (CMC 3.1)
 Advance (BAK). (CMC 3.0)
 Begin write cycle for 25th CM reference.
 26th address leaves SAS for CM. (CMC 3.0)
 Advance (XSK) to 13 (octal). (CPU 3.7)
 XSF remains set. (CPU 3.7)
 XIF remains set. (CPU 3.7)

CP57 Transmit (X3) to SWA register. (CMC 3.1)
 Transmit read data to X0 register. (CPU 3.4)

CP58 No instruction in upper parcel of CIW register.
 Transmit (BAK) to SAS. (CMC 3.0)
 Tag for exchange sequence. (CMC 3.1)
 Advance (BAK). CMC 3.0)
 Begin write cycle for 26th CM reference.
 27th address leaves SAS for CM. (CMC 3.0)
 Advance (XSK) to 14 (octal). (CPU 3.7)
 XSF remains set. (CPU 3.7)
 XIF remains set. (CPU 3.7)

CP59 Transmit (X4) to SWA register. (CMC 3.1)
 Transmit read data to X1 register. (CPU 3.4)

CP60 No instruction in upper parcel of CIW register.
 Transmit (BAK) to SAS. (CMC 3.0)
 Tag for exchange sequence. (CMC 3.1)
 Advance (BAK). (CMC 3.0)
 Begin write cycle for 27th CM reference.
 28th address leaves SAS for CM. (CMC 3.0)
 Advance (XSK) to 15 (octal). (CPU 3.7)
 XSF remains set. (CPU 3.7)
 XIF remains set. (CPU 3.7)

CP61 Transmit (X5) to SWA register. (CMC 3.1)
 Transmit read data to X2 register. (CPU 3.4)

CP62 No instruction in upper parcel of CIW register.
 Transmit (BAK) to SAS. (CMC 3.0)
 Tag for exchange sequence. (CMC 3.1)
 Advance (BAK). (CMC 3.0)
 Begin write cycle for 28th CM reference.
 29th address leaves SAS for CM. (CMC 3.0)
 Advance (XSK) to 16 (octal). (CPU 3.7)
 XSF remains set. (CPU 3.7)
 XIF remains set. (CPU 3.7)

CP63 Transmit (X6) to SWA register. (CMC 3.1)
 Transmit read data to X3 register. (CPU 3.4)

CP64 No instruction in upper parcel of CIW register.
 Transmit (BAK) to SAS. (CMC 3.0)
 Tag for exchange sequence. (CMC 3.1)
 Advance (BAK). (CMC 3.0)
 Begin write cycle for 29th CM reference.
 30th address leaves SAS for CM. (CMC 3.0)
 Advance (XSK) to 17 (octal). (CPU 3.7)
 Set JCF. (CPU 3.7)
 Clear XSF. (CPU 3.7)
 XIF remains set. (CPU 3.7)

CP65 Transmit (X7) to SWA register. (CMC 3.1)
 Transmit read data to X4 register. (CPU 3.4)

CP66 No instruction in upper parcel of CIW register.
 No coincidence in IAS.
 Transmit a blank word to CIW register.
 Set OSF. (CPU 3.3)
 Set JOF. (CPU 3.3)
 Transmit (P) + (RAC) to IFA register. (CMC 3.0)
 Set F1F, F2F. (CPU 3.3)
 Clear JCF. (CPU 3.7)
 Transmit (BAK) to SAS. (CMC 3.0)
 Tag for exchange sequence. (CMC 3.1)
 Advance (BAK). (CMC 3.0)
 Advance (XSK) to 00 (octal). (CPU 3.7)
 Begin write cycle for 30th CM reference.
 31st address leaves SAS for CM. (CMC 3.0)
 Clear XIF. (CPU 3.7)

CP67 Transmit read data to X5 register. (CPU 3.4)

CP68 No instruction in upper parcel of CIW register.
 Transmit (P) to NSA register. (CPU 3.0)
 Set M1F, M2F. (CPU 3.3)
 Clear JOF. (CPU 3.3)
 OSF remains set. (CPU 3.3)
 Transmit (IFA) to SAS. (CMC 3.0)
 Tag for read to instruction stack. (CMC 3.1)
 Advance (IFA). (CMC 3.0)
 Clear F2F. (CPU 3.3)
 Begin write cycle for 31st CM reference.
 32nd address leaves SAS for CM. (CMC 3.0)

CP69 Transmit read data to X6 register. (CPU 3.4)

CP70 No instruction in upper parcel of CIW register.
 No coincidence in IAS.
 Read a blank word from IWS to CIW register.
 OSF remains set. (CPU 3.3)
 Transmit (IFA) to SAS. (CMC 3.0)
 Tag for read to instruction stack. (CMC 3.1)
 Advance (IFA). (CMC 3.0)
 First fetch address leaves SAS for CM. (CMC 3.0)
 Clear F1F. (CPU 3.3)
 Begin write cycle for 32nd CM reference.

CP71 Transmit read data to X7 register. (CPU 3.4)

CP72 No instruction in upper parcel of CIW register.
 No coincidence in IAS.
 Read a blank word from IWS to CIW register.
 OSF remains set. (CPU 3.3)
 First fetch CM read cycle begins.
 Second fetch address leaves SAS for CM. (CMC 3.0)

CP74 No instruction in upper parcel of CIW register.
 No coincidence in IAS.
 Read a blank word from IWS to CIW register.
 OSF remains set. (CPU 3.3)
 Second fetch CM read cycle begins.

- CP75 No instruction in upper parcel of CIW register.
No coincidence in IAS.
Read a blank word from IWS to CIW register.
OSF remains set. (CPU 3.3)
- CP76 No instruction in upper parcel of CIW register.
No coincidence in IAS.
Read a blank word from IWS to CIW register.
OSF remains set. (CPU 3.3)
- CP89 No instruction in upper parcel of CIW register.
No coincidence in IAS.
Read a blank word from IWS to CIW register.
OSF remains set. (CPU 3.3)
Transmit first fetch instruction word to IWS. (CPU 3.0)
Transmit (NSA) to IAS. (CPU 3.0)
Shift IWS and IAS one word position. (CPU 3.0)
Advance (NSA). (CPU 3.0)
Clear M2F. (CPU 3.3)
- CP90 No instruction in upper parcel of CIW register.
Coincidence in IAS.
Read a valid instruction word from IWS to CIW register.
Clear OSF. (CPU 3.3)
- CP91 Next instruction in upper parcel of CIW register.
Instruction may issue.
Transmit second fetch instruction word to IWS. (CPU 3.0)
Transmit (NSA) to IAS. (CPU 3.0)
Shift IWS and IAS one word position. (CPU 3.0)
Advance (NSA). (CPU 3.0)
Clear M1F. (CPU 3.3)

EXECUTION DELAYS

The PXF does not set in this mode of the 013 instruction until the Bj register is free. The RIF sets without this condition, but the sequence does not proceed to the commands listed for CP01 until the PXF sets.

The XSF does not set in CP02 until two conditions are satisfied.

- All operating registers must be free, which implies that all previously issued instructions have delivered their results to the operating registers. This normally occurs during CP02 because of the delay from the preceding two clock periods.
- The F1F must not be set. F1F clears unless a CM bank conflict has caused an SAS backup condition which prevented an instruction fetch address from leaving the IFA register. This condition should not occur often enough to cause a significant average delay.

An exchange cannot start until all CM banks are free and SAS is empty. This condition is called memory dead. A CM bank conflict delay may occur between the exchange sequence addresses and the instruction fetch addresses. All I/O references and CM references are blocked during an exchange.

013jk EXCHANGE JUMP TO MA

EXECUTION TIME

The minimum execution time for this instruction is 91 clock periods, the minimum time from the arrival of this instruction in the upper parcel of the CIW register until the arrival of the first instruction for the next program. This instruction issues from the CIW register in the second clock period of the sequence. A 60-bit word of all zeros is read into the CIW register at this time, voiding any following instructions in the CIW register. The IAS is cleared to all zeros, which nullifies the content of the IWS. No further instructions can enter the CIW register until the exchange sequence has been completed and the IWS is loaded with a new sequence of instructions.

Timing considerations for this mode of the exchange jump instruction are essentially the same as for the alternate mode. The only differences in the command timing occur in the initial clock periods of the sequence. They are:

- CP00 013 instruction in upper parcel of CIW register. (CPU 3.1)
 Instruction does not issue.
 Set PXF. (CPU 3.7)
 Set RIF. (CPU 3.7)
- CP01 013 instruction in upper parcel of CIW register.
 Instruction issues.
 Transmit a blank word to CIW register.
 Transmit (MA) to XJA register. (CPU 3.7)
 RIF remains set. (CPU 3.7)
 Clear PXF. (CPU 3.7)
- CP02 No instruction in upper parcel of CIW register.
 All operating registers free.
 All instruction fetches completed.
 Set XSF. (CPU 3.7)
 Clear RIF. (CPU 3.7)

(Refer to alternate mode 013 instruction for remainder of listing.)

EXECUTION DELAYS

Same as 013jK except for PXF.

02i0K JUMP TO (Bi)+K

EXECUTION TIME

The execution time for this instruction is 26 clock periods. The sequence of events for this instruction is:

- CP00 02 instruction in upper parcel of CIW register. (CPU 3.1)
 Bi register free. (CPU 3.2)
 Instruction does not issue.
 Set GJF. (CPU 3.7)

CP01 02 instruction in upper parcel of CIW register.
Instruction does not issue.
Transmit (Bi) + K to P register. (CPU 3.0)
Set JCF. (CPU 3.7)
Clear GJF. (CPU 3.7)

CP02 02 instruction in upper parcel of CIW register.
Instruction issues.
No coincidence in IAS. (CPU 3.0)
Read a blank word from IWS to CIW register.
Set OSF. (CPU 3.3)
Set JOF. (CPU 3.3)
Transmit (P) + (RAC) to IFA register. (CMC 3.0)
Set F1F, F2F. (CPU 3.3)
Clear JCF. (CPU 3.7)

CP03 No instruction in upper parcel of CIW register.
Transmit (P) to NSA register. (CPU 3.0)
Set M1F, M2F. (CPU 3.3)
OSF remains set. (CPU 3.3)
Transmit (IFA) to SAS. (CMC 3.0)
Tag for read to IWS. (CMC 3.1)
Advance (IFA). (CMC 3.0)
Clear JOF. (CPU 3.3)
Clear F2F. (CPU 3.3)

CP04 No instruction in upper parcel of CIW register.
No coincidence in IAS.
Read a blank word from IWS to CIW register.
OSF remains set. (CPU 3.3)

CP05 No instruction in upper parcel of CIW register.
No coincidence in IAS.
Read a blank word from IWS to CIW register.
OSF remains set. (CPU 3.3)
Transmit (IFA) to SAS. (CMC 3.0)
Tag for read to IWS. (CMC 3.1)
Advance (IFA). (CMC 3.0)
First fetch address leaves SAS for CM. (CMC 3.0)
Clear F1F. (CPU 3.3)

- CP06 No instruction in upper parcel of CIW register.
No coincidence in IAS.
Read a blank word from IWS to CIW register.
OSF remains set. (CPU 3.3)
First fetch read cycle begins.
- CP07 No instruction in upper parcel of CIW register.
No coincidence in IAS.
Read a blank word from IWS to CIW register.
OSF remains set. (CPU 3.3)
Second fetch address leaves SAS for CM. (CMC 3.0)
- CP08 No instruction in upper parcel of CIW register.
No coincidence in IAS.
Read a blank word from IWS to CIW register.
OSF remains set. (CPU 3.3)
Second fetch read cycle begins.
- CP24 No instruction in upper parcel of CIW register.
No coincidence in IAS.
Read a blank word from IWS to CIW register.
OSF remains set. (CPU 3.3)
Transmit first fetch instruction word to IWS. (CPU 3.0)
Transmit (NSA) to IAS. (CPU 3.0)
Shift IWS and IAS one word position. (CPU 3.0)
Advance (NSA). (CPU 3.0)
Clear M2F. (CPU 3.3)
- CP25 No instruction in upper parcel of CIW register.
Coincidence in IAS. (CPU 3.0)
Read a valid instruction word from IWS to CIW register.
Clear OSF. (CPU 3.3)
- CP26 Next instruction in upper parcel of CIW register.
Instruction may issue.
Transmit second fetch instruction word to IWS. (CPU 3.0)
Transmit (NA) to IAS. (CPU 3.0)
Shift IWS and IAS one word position. (CPU 3.0)
Advance (NSA). (CPU 3.0)
Clear M1F. (CPU 3.3)

EXECUTION DELAYS

This instruction sequence does not begin until the Bi register is free.

The sequence is delayed at CP02 if F1F sets because JOF did not set. F1F clears at this time unless a CM bank conflict has caused an SAS backup condition which delayed an instruction fetch address in leaving the IFA register. If F1F sets in CP02, the rest of the sequence is delayed until JOF sets. JOF sets when all fetch instruction words have arrived at the IWS and M1F has cleared.

The transmission of the content of P to the NSA register, the setting of M1F and M2F, and the clearing of JOF do not occur in CP03 if an instruction fetch is still in process and M1F is set for this earlier reference. This does not delay the execution of the remainder of this instruction. These commands are issued and the functions performed when the previous fetch instruction word arrives at the IWS.

The transmission of (IFA) to the SAS is delayed in CP03 or in CP05 if an SAS backup or memory enable exists. The remainder of the sequence is delayed by the amount of time required to resolve the CM bank conflict and clear up the SAS backlog.

A delay in the arrival of the first fetch instruction word at the IWS occurs if a CM bank conflict prevents the first fetch address from leaving the SAS in CP05. This delays the completion of this instruction by the time required to resolve the CM bank conflict.

030jK BRANCH TO K IF (Xj) = 0

EXECUTION TIME

The three normal cases for this instruction are:

- Branch fall through (minimum of 2 clock periods)

If the jump condition is not met, this instruction is executed and issues from the CIW register in 2 clock periods. Execution is delayed if the Xj register is not free.

- Branch in stack (minimum of 3 clock periods)

If the jump condition is met and the destination address K is in the instruction stack, this instruction is executed in 3 clock periods. Execution is delayed if the Xj register is not free.

- Branch out of stack (minimum of 26 clock periods)

If the jump condition is met and the destination address K is not in the instruction stack, this instruction is executed in 26 clock periods. Execution is delayed if the Xj register is not free. Execution may also be delayed by bank conflicts in CM, CM refresh cycle, and memory enable.

The sequence of events in this instruction execution for the branch fall-through case is:

- CP00 Branch instruction in upper parcel of CIW register. (CPU 3.1)
 Instruction does not issue.
 Jump condition not satisfied. (CPU 3.7)
 Set GJF. (CPU 3.7)
- CP01 Branch instruction in upper parcel of CIW register.
 Instruction issues.
 Transmit next instruction to upper parcel of CIW register.
 Clear GJF. (CPU 3.7)
- CP02 Next instruction in upper parcel of CIW register.
 Instruction may issue.

The sequence of events in this instruction execution for the branch in stack case is:

- CP00 Branch instruction in upper parcel of CIW register. (CPU 3.1)
 Instruction does not issue.
 Jump condition satisfied. (CPU 3.7)
 Set GJF. (CPU 3.7)
- CP01 Branch instruction in upper parcel of CIW register.
 Instruction does not issue.
 Transmit K to P register. (CPU 3.0)
 Set JCF. (CPU 3.7)
 Clear GJF. (CPU 3.7)

CP02 Branch instruction in upper parcel of CIW register.
Instruction issues.
Destination address in instruction stack.
Coincidence in IAS. (CPU 3.0)
Read a valid instruction word from IWS to CIW register.
Clear JCF. (CPU 3.7)

CP03. Next instruction in upper parcel of CIW register.
Instruction may issue.

The sequence of events in this instruction execution for the branch out of stack case is:

CP00 Branch instruction in upper parcel of CIW register. (CPU 3.1)
Instruction does not issue.
Jump condition satisfied. (CPU 3.7)
Set GJF. (CPU 3.7)

CP01 Branch instruction in upper parcel of CIW register.
Instruction does not issue.
Transmit K to P register. (CPU 3.0)
Set JCF. (CPU 3.7)
Clear GJF. (CPU 3.7)

CP02 Branch instruction in upper parcel of CIW register.
Instruction issues.
No coincidence in IAS.
Read a blank word from IWS to CIW register. (CPU 3.1)
Transmit (P) + (RAC) to IFA register. (CMC 3.0)
Set OSF. (CPU 3.3)
Set JOF. (CPU 3.3)
Set F1F, F2F. (CPU 3.3)
Clear JCF. (CPU 3.7)

CP03 No instruction in upper parcel of CIW register.
 Transmit (P) to NSA register. (CPU 3.0)
 Set M1F, M2F. (CPU 3.3)
 OSF remains set. (CPU 3.3)
 Transmit (IFA) to SAS. (CMC 3.0)
 Tag for read to IWS. (CMC 3.1)
 Advance (IFA). (CMC 3.0)
 Clear JOF. (CPU 3.3)
 Clear F2F. (CPU 3.3)

CP04 No instruction in upper parcel of CIW register.
 No coincidence in IAS.
 Read a blank word from IWS to CIW register. (CPU 3.1)
 OSF remains set. (CPU 3.3)

CP05 No instruction in upper parcel of CIW register.
 No coincidence in IAS.
 Read a blank word from IWS to CIW register. (CPU 3.1)
 OSF remains set. (CPU 3.3)
 Transmit (IFA) to SAS. (CMC 3.0)
 Tag for read to IWS. (CMC 3.1)
 Advance (IFA). (CMC 3.0)
 First fetch address leaves SAS for CM. (CMC 3.0)
 Clear F1F. (CPU 3.3)
 First fetch read cycle begins.

CP06 No instruction in upper parcel of CIW register.
 No coincidence in IAS.
 Read a blank word from IWS to CIW register. (CPU 3.1)
 OSF remains set. (CPU 3.3)

CP07 No instruction in upper parcel of CIW register.
 No coincidence in IAS.
 Read a blank word from IWS to CIW register. (CPU 3.1)
 OSF remains set. (CPU 3.3)
 Second fetch address leaves SAS for CM. (CMC 3.0)

CP08 No instruction in upper parcel of CIW register.
 No coincidence in IAS.
 Read a blank word from IWS to CIW register. (CPU 3.1)
 OSF remains set. (CPU 3.3)
 Second fetch read cycle begins.

CP24 No instruction in upper parcel of CIW register.
 No coincidence in IAS.
 Read a blank word from IWS to CIW register. (CPU 3.1)
 OSF remains set. (CPU 3.3)
 Transmit first fetch instruction word to IWS. (CPU 3.0)
 Transmit (NSA) to IAS. (CPU 3.0)
 Shift IWS and IAS one word position. (CPU 3.0)
 Advance (NSA). (CPU 3.0)
 Clear M2F. (CPU 3.3)

CP25 No instruction in upper parcel of CIW register.
 Coincidence in IAS. (CPU 3.0)
 Read a valid instruction word from IWS to CIW register. (CPU 3.1)
 Clear OSF. (CPU 3.3)

CP26 Next instruction in upper parcel of CIW register.
 Instruction may issue.
 Transmit second fetch instruction word to IWS. (CPU 3.0)
 Transmit (NSA) to IAS. (CPU 3.0)
 Shift IWS and IAS one word position. (CPU 3.0)
 Advance (NSA). (CPU 3.0)
 Clear M1F. (CPU 3.3)

EXECUTION DELAYS (BRANCH OUT OF STACK)

A delay in the execution of the instruction occurs if F1F sets in CP02. JOF sets when F1F clears, indicating that all instruction fetch requests have been sent to the SAS. If the F1F sets in CP02, the rest of the sequence is delayed. JOF sets when all fetch instruction words have arrived at the IWS and the M1F has cleared.

The transmission of the content of P to the NSA register, the setting of M1F and M2F, and the clearing of JOF do not occur in CP03 if an instruction fetch is still in process and M1F is set for this earlier reference. This does not delay the execution of the remainder of this instruction. These commands are issued and the functions are performed when the previous fetch instruction word arrives at the IWS.

The transmission of (IFA) to the SAS is delayed in CP03 or CP05 if an SAS backup or memory enable exists. The remainder of the sequence is delayed by the amount of time required to resolve the CM bank conflict and clear up the SAS backlog.

A delay in the arrival of the first fetch instruction word at the IWS occurs if a CM bank conflict prevents the first fetch address from leaving the SAS in CP05. This delays the completion of this instruction by the time required to resolve the CM bank conflict.

031jK BRANCH TO K IF (Xj) ≠ 0

EXECUTION TIME

The three normal cases for this instruction are:

- Branch fall through (minimum of 2 clock periods)
- Branch in stack (minimum of 3 clock periods)
- Branch out of stack (minimum of 26 clock periods)

Details of the execution timing for this instruction are the same as the 030 instruction.

EXECUTION DELAYS

The execution delays for this instruction are the same as the 030 instruction.

032jK BRANCH TO K IF (Xj) POSITIVE

EXECUTION TIME

The three normal cases for this instruction are:

- Branch fall through (minimum of 2 clock periods)
- Branch in stack (minimum of 3 clock periods)
- Branch out of stack (minimum of 26 clock periods)

Details of the execution timing for this instruction are the same as the 030 instruction.

EXECUTION DELAYS

The execution delays for this instruction are the same as the 030 instruction.

033jK BRANCH TO K IF (Xj) NEGATIVE

EXECUTION TIME

The three normal cases for this instruction are:

- Branch fall through (minimum of 2 clock periods)
- Branch in stack (minimum of 3 clock periods)
- Branch out of stack (minimum of 26 clock periods)

Details of the execution timing for this instruction are the same as the 030 instruction.

EXECUTION DELAYS

The execution delays for this instruction are the same as the 030 instruction.

034jK BRANCH TO K IF (Xj) IN RANGE

EXECUTION TIME

The three normal cases for this instruction are:

- Branch fall through (minimum of 2 clock periods)
- Branch in stack (minimum of 3 clock periods)
- Branch out of stack (minimum of 26 clock periods)

Details of the execution timing for this instruction are the same as the 030 instruction.

EXECUTION DELAYS

The execution delays for this instruction are the same as the 030 instruction.

035jK BRANCH TO K IF (Xj) NOT IN RANGE

EXECUTION TIME

The three normal cases for this instruction are:

- Branch fall through (minimum of 2 clock periods)
- Branch in stack (minimum of 3 clock periods)
- Branch out of stack (minimum of 26 clock periods)

Details of the execution timing for this instruction are the same as the 030 instruction.

EXECUTION DELAYS

The execution delays for this instruction are the same as the 030 instruction.

036jK BRANCH TO K IF (Xj) DEFINITE

EXECUTION TIME

The three normal cases for this instruction are:

- Branch fall through (minimum of 2 clock periods)
- Branch in stack (minimum of 3 clock periods)
- Branch out of stack (minimum of 26 clock periods)

Details of the execution timing for this instruction are the same as the 030 instruction.

EXECUTION DELAYS

The execution delays for this instruction are the same as the 030 instruction.

037jK BRANCH TO K IF (Xj) INDEFINITE

EXECUTION TIME

The three normal cases for this instruction are:

- Branch fall through (minimum of 2 clock periods)
- Branch in stack (minimum of 3 clock periods)
- Branch out of stack (minimum of 26 clock periods)

Details of the execution timing for this instruction are the same as the 030 instruction.

EXECUTION DELAYS

The execution delays for this instruction are the same as the 030 instruction.

04ijK BRANCH TO K IF (Bi) = (Bj)

EXECUTION TIME

The three normal cases for this instruction are:

- Branch fall through (minimum of 2 clock periods)

If the jump condition is not met, this instruction is executed and issues from the CIW register in 2 clock periods. Execution is delayed if the Bi or Bj register is not free.

- Branch in stack (minimum of 3 clock periods)

If the jump condition is met and the destination address K is in the instruction stack, this instruction is executed in 3 clock periods. Execution is delayed if the Bi or Bj register is not free.

- Branch out of stack (minimum of 26 clock periods)

If the jump condition is met and the destination address K is not in the instruction stack, this instruction is executed 26 clock periods. Execution is delayed if the Bi or Bj register is not free. Execution may also be delayed by CM bank conflicts, CM refresh cycle, and memory enable.

The sequence of events in this instruction execution for the branch fall-through case is:

CP00	Branch instruction in upper parcel of CIW register. (CPU 3.1) Instruction does not issue. Jump condition not satisfied. (CPU 3.7) Bi register free. (CPU 3.2) Bj register free. (CPU 3.2) Set GJF. (CPU 3.7)
CP01	Branch instruction in upper parcel of CIW register. Instruction issues. Transmit next instruction to upper parcel of CIW register. Clear GJF. (CPU 3.7)
CP02	Next instruction in upper parcel of CIW register. Instruction may issue.

The sequence of events in this instruction execution for the branch in stack case is:

- CP00 Branch instruction in upper parcel of CIW register. (CPU 3.1)
Instruction does not issue.
Jump condition satisfied. (CPU 3.7)
Bi register free. (CPU 3.2)
Bj register free. (CPU 3.2)
Set GJF. (CPU 3.7)
- CP01 Branch instruction in upper parcel of CIW register.
Instruction does not issue.
Transmit K to P register. (CPU 3.0)
Set JCF. (CPU 3.7)
Clear GJF. (CPU 3.7)
- CP02 Branch instruction in upper parcel of CIW register.
Instruction issues.
Destination address in instruction stack.
Coincidence in IAS. (CPU 3.0)
Read a valid instruction word from IWS to CIW register. (CPU 3.1)
Clear JCF. (CPU 3.7)
- CP03 Next instruction in upper parcel of CIW register.
Instruction may issue.

The sequence of events in this instruction execution for the branch out of stack case is:

- CP00 Branch instruction in upper parcel of CIW register. (CPU 3.1)
Instruction does not issue.
Jump condition satisfied. (CPU 3.7)
Bi register free. (CPU 3.2)
Bj register free. (CPU 3.2)
Set GJF. (CPU 3.7)
- CP01 Branch instruction in upper parcel of CIW register.
Instruction does not issue.
Transmit K to P register. (CPU 3.0)
Set JCF. (CPU 3.7)
Clear GJF. (CPU 3.7)

- CP02 Branch instruction in upper parcel of CIW register.
Instruction issues.
No coincidence in IAS.
Read a blank word from IWS to CIW register. (CPU 3.1)
Transmit (P) + (RAC) to IFA register. (CMC 3.0)
Set OSF. (CPU 3.3)
Set JOF. (CPU 3.3)
Set F1F, F2F. (CPU 3.3)
Clear JCF. (CPU 3.7)
- CP03 No instruction in upper parcel of CIW register.
Transmit (P) to NSA register. (CPU 3.0)
Set M1F, M2F. (CPU 3.3)
OSF remains set. (CPU 3.3)
Transmit (IFA) to SAS. (CMC 3.0)
Tag for read to IWS. (CMC 3.1)
Advance (IFA). (CMC 3.0)
Clear JOF. (CPU 3.3)
Clear F2F. (CPU 3.3)
- CP04 No instruction in upper parcel of CIW register.
No coincidence in IAS.
Read a blank word from IWS to CIW register. (CPU 3.1)
OSF remains set. (CPU 3.3)
- CP05 No instruction in upper parcel of CIW register.
No coincidence in IAS.
Read a blank word from IWS to CIW register. (CPU 3.1)
OSF remains set. (CPU 3.3)
First fetch read cycle begins.
Transmit (IFA) to SAS. (CMC 3.0)
Tag for read to IWS. (CMC 3.1)
Advance (IFA). (CMC 3.0)
First fetch address leaves SAS for CM. (CMC 3.0)
Clear F1F. (CPU 3.3)
- CP06 No instruction in upper parcel of CIW register.
No coincidence in IAS.
Read a blank word from IWS to CIW register. (CPU 3.1)
OSF remains set. (CPU 3.3)

- CP07 No instruction in upper parcel of CIW register.
No coincidence in IAS.
Read a blank word from IWS to CIW register. (CPU 3.1)
OSF remains set. (CPU 3.3)
Second fetch address leaves SAS for CM. (CMC 3.0)
- CP08 No instruction in upper parcel of CIW register.
No coincidence in IAS.
Read a blank word from IWS to CIW register. (CPU 3.1)
OSF remains set. (CPU 3.3)
Second fetch read cycle begins.
- CP24 No instruction in upper parcel of CIW register.
No coincidence in IAS.
Read a blank word from IWS to CIW register. (CPU 3.1)
OSF remains set. (CPU 3.3)
Transmit first fetch instruction word to IWS. (CPU 3.0)
Transmit (NSA) to IAS. (CPU 3.0)
Shift IWS and IAS one word position. (CPU 3.0)
Advance (NSA). (CPU 3.0)
Clear M2F. (CPU 3.3)
- CP25 No instruction in upper parcel of CIW register.
Coincidence in IAS. (CPU 3.0)
Read a valid instruction word from IWS to CIW register. (CPU 3.1)
Clear OSF. (CPU 3.3)
- CP26 Next instruction in upper parcel of CIW register.
Instruction may issue.
Transmit second fetch instruction word to IWS. (CPU 3.0)
Transmit (NSA) to IAS. (CPU 3.0)
Shift IWS and IAS one word position. (CPU 3.0)
Advance (NSA). (CPU 3.0)
Clear M1F. (CPU 3.3)

EXECUTION DELAYS (BRANCH OUT OF STACK)

A delay occurs if F1F sets in CP02. JOF does not set unless F1F clears, indicating all instruction fetch requests have been sent to the SAS. If F1F sets in CP02, the rest of the sequence is delayed. JOF does not set until all fetch instruction words have arrived at the IWS and M1F has cleared.

The transmission of the content of P to the NSA register, the setting of M1F and M2F, and the clearing of JOF do not occur in CP03 if an instruction fetch is still in process and M1F is set for this earlier reference. This does not delay the execution of the remainder of this instruction. These commands are issued and the functions are performed when the previous fetch instruction word arrives at the IWS.

The transmission of (IFA) to the SAS is delayed in CP03 or CP05 if an SAS backup or memory enable exists. The remainder of the sequence is delayed by the amount of time required to resolve the CM bank conflict and clear up the SAS backlog.

A delay in the arrival of the first fetch instruction word at the IWS occurs if a CM bank conflict prevents the first fetch address from leaving the SAS in CP05. This delays the completion of this instruction by the time required to resolve the CM bank conflict.

05ijK BRANCH TO K IF (Bi) ≠ (Bj)

EXECUTION TIME

The three normal cases for this instruction are:

- Branch fall through (minimum of 2 clock periods)
- Branch in stack (minimum of 3 clock periods)
- Branch out of stack (minimum of 26 clock periods)

Details of the execution timing for this instruction are the same as the 04 instruction.

EXECUTION DELAYS

The execution delays for this instruction are the same as the 04 instruction.

06iJK BRANCH TO K IF (Bi) ≥ (Bj)

EXECUTION TIME AND EXECUTION DELAYS

Details of the execution timing and execution delays for this instruction are the same as the 04 instruction.

07iJK BRANCH TO K IF (Bi) < (Bj)

EXECUTION TIME AND EXECUTION DELAYS

Details of the execution timing and execution delays for this instruction are the same as the 04 instruction.

10iJ0 TRANSMIT (Xj) TO Xi

ISSUE CONDITIONS

Xi register is free.

Xj register is free.

X register input path is free in next clock period.

No SAS backup condition.

EXECUTION TIME

No execution delays are possible after this instruction issues from the CIW register. The result is delivered to the Xi register 1 clock period after the instruction issues. The Xi register is reserved for the clock period from issue to delivery of data. The sequence of events for this instruction is:

CP00 10 instruction in upper parcel of CIW register. (CPU 3.1)
 Instruction issues.
 Transmit next instruction to upper parcel of CIW register.
 Transmit (Xj) to boolean unit. (BOOL 3.0)

CP01 Next instruction in upper parcel of CIW register.
 Instruction may issue.
 Transmit data from boolean unit to Xi register. (CPU 3.4)
 Set Xi register busy flag. (CPU 3.2)
 Set go boolean flag. (CPU 3.2)

CP02 Clear Xi register busy flag. (CPU 3.2)
 Clear go boolean flag. (CPU 3.2)
 Result in Xi register.

11ijk LOGICAL PRODUCT OF (Xj) AND (Xk) TO Xi

ISSUE CONDITIONS

Xi register is free.
 Xj register is free.
 Xk register is free.
 X register input path is free in next clock period.
 No SAS backup condition.

EXECUTION TIME

No execution delays are possible after this instruction issues from the CIW register. The result is delivered to the Xi register 1 clock period after the instruction issues. The Xi register is reserved for the clock period from issue to delivery of data. The sequence of events for this instruction is:

CP00 11 instruction in upper parcel of CIW register. (CPU 3.1)
 Instruction issues.
 Transmit next instruction to upper parcel of CIW register.
 Transmit (Xj) to boolean unit. (BOOL 3.0)
 Transmit (Xk) to boolean unit. (BOOL 3.0)

CP01 Next instruction in upper parcel of CIW register.
 Instruction may issue.
 Transmit data from boolean unit to Xi register. (CPU 3.4)
 Set Xi register busy flag. (CPU 3.2)
 Set go boolean flag. (CPU 3.2)

CP02 Clear Xi register busy flag. (CPU 3.2)
 Clear go boolean flag. (CPU 3.2)
 Result in Xi register.

12ijk LOGICAL SUM OF (Xj) AND COMPLEMENT OF (Xk) TO Xi

ISSUE CONDITIONS

Xi register is free.
Xj register is free.
Xk register is free.
X register input path is free in next clock period.
No SAS backup condition.

EXECUTION TIME

No execution delays are possible after this instruction issues from the CIW register. The result is delivered to the Xi register 1 clock period after the instruction issues. The Xi register is reserved for the clock period from issue to delivery of data. The sequence of events for this instruction is:

CP00 12 instruction in upper parcel of CIW register. (CPU 3.1)
 Instruction issues.
 Transmit next instruction to upper parcel of CIW register.
 Transmit (Xj) to boolean unit. (BOOL 3.0)
 Transmit (Xk) to boolean unit. (BOOL 3.0)

CP01 Next instruction in upper parcel of CIW register. (CPU 3.1)
 Instruction may issue.
 Transmit data from boolean unit to Xi register. (CPU 3.4)
 Set Xi register busy flag. (CPU 3.2)
 Set go boolean flag. (CPU 3.2)

CP02 Clear Xi register busy flag. (CPU 3.2)
 Clear go boolean flag. (CPU 3.2)
 Result in Xi register.

13ijk LOGICAL DIFFERENCE OF (Xj) AND (Xk) TO Xi

ISSUE CONDITIONS

Xi register is free.
Xj register is free.
Xk register is free.
X register input path is free in next clock period.
No SAS backup condition.

EXECUTION TIME

No execution delays are possible after this instruction issues from the CIW register. The result is delivered to the Xi register 1 clock period after the instruction issues. The Xi register is reserved for the clock period from issue to delivery of data. The sequence of events for this instruction is:

CP00 13 instruction in upper parcel of CIW register. (CPU 3.1)
Instruction issues.
Transmit next instruction to upper parcel of CIW register.
Transmit (Xj) to boolean unit. (BOOL 3.0)
Transmit (Xk) to boolean unit. (BOOL 3.0)

CP01 Next instruction in upper parcel of CIW register. (CPU 3.1)
Instruction may issue.
Transmit data from boolean unit to Xi register. (CPU 3.4)
Set Xi register busy flag. (CPU 3.2)
Set go boolean flag. (CPU 3.2)

CP02 Clear Xi register busy flag. (CPU 3.2)
Clear go boolean flag. (CPU 3.2)
Result in Xi register.

14ijk TRANSMIT COMPLEMENT OF (Xk) TO Xi

ISSUE CONDITIONS

Xi register is free.
Xk register is free.
X register input path is free in next clock period.
No SAS backup condition.

EXECUTION TIME

No execution delays are possible after this instruction issues from the CIW register. The result is delivered to the Xi register 1 clock period after the instruction issues. The Xi register is reserved for the clock period from issue to delivery of data. The sequence of events for this instruction is:

- CP00 14 instruction in upper parcel of CIW register. (CPU 3.1)
 Instruction issues.
 Transmit next instruction to upper parcel of CIW register. (CPU 3.1)
 Transmit (Xk) to boolean unit. (BOOL 3.0)
- CP01 Next instruction in upper parcel of CIW register. (CPU 3.1)
 Instruction may issue.
 Transmit data from boolean unit to Xi register. (CPU 3.4)
 Set Xi register busy flag. (CPU 3.2)
 Set go boolean flag. (CPU 3.2)
- CP02 Clear Xi register busy flag. (CPU 3.2)
 Clear go boolean flag. (CPU 3.2)
 Result in Xi register.

15ijk LOGICAL PRODUCT OF (Xj) AND COMPLEMENT OF (Xk) TO Xi

ISSUE CONDITIONS

Xi register is free.
Xj register is free.
Xk register is free.
X register input path is free in next clock period.
No SAS backup condition.

EXECUTION TIME

No execution delays are possible after this instruction issues from the CIW register. The result is delivered to the Xi register 1 clock period after the instruction issues. The Xi register is reserved for the clock period from issue to delivery of data. The sequence of events for this instruction is:

- CP00 15 instruction in upper parcel of CIW register. (CPU 3.1)
 Instruction issues.
 Transmit next instruction to upper parcel of CIW register.
 Transmit (Xj) to boolean unit. (BOOL 3.0)
 Transmit (Xk) to boolean unit. (BOOL 3.0)
- CP01 Next instruction in upper parcel of CIW register.
 Instruction may issue.
 Transmit data from boolean unit to Xi register. (CPU 3.4)
 Set Xi register busy flag. (CPU 3.2)
 Set go boolean flag. (CPU 3.2)
- CP02 Clear Xi register busy flag. (CPU 3.2)
 Clear go boolean flag. (CPU 3.2)
 Result in Xi register.

16ijk LOGICAL SUM OF (Xj) AND COMPLEMENT OF (Xk) TO Xi

ISSUE CONDITIONS

Xi register is free.
Xj register is free.
Xk register is free.
X register input path is free in next clock period.
No SAS backup condition.

EXECUTION TIME

No execution delays are possible after this instruction issues from the CIW register. The result is delivered to the Xi register 1 clock period after the instruction issues. The Xi register is reserved for the clock period from issue to delivery of data. The sequence of events for this instruction is:

CP00 16 instruction in upper parcel of CIW register. (CPU 3.1)
 Instruction issues.
 Transmit next instruction to upper parcel of CIW register.
 Transmit (Xj) to boolean unit. (BOOL 3.0)
 Transmit (Xk) to boolean unit. (BOOL 3.0)

CP01 Next instruction in upper parcel of CIW register.
 Instruction may issue.
 Transmit data from boolean unit to Xi register. (CPU 3.4)
 Set Xi register busy flag. (CPU 3.2)
 Set go boolean flag. (CPU 3.2)

CP02 Clear Xi register busy flag. (CPU 3.2)
 Clear go boolean flag. (CPU 3.2)
 Result in Xi register.

17ijk LOGICAL DIFFERENCE OF (Xj) AND COMPLEMENT OF (Xk) TO Xi

ISSUE CONDITIONS

Xi register is free.
Xj register is free.
Xk register is free.
X register input path is free in next clock period.
No SAS backup condition.

EXECUTION TIME

No execution delays are possible after this instruction issues from the CIW register. The result is delivered to the Xi register 1 clock period after the instruction issues. The Xi register is reserved for the clock period from issue to delivery of data. The sequence of events for this instruction is:

CP00 17 instruction in upper parcel of CIW register. (CPU 3.1)
 Instruction issues.
 Transmit next instruction to upper parcel of CIW register.
 Transmit (Xj) to boolean unit. (BOOL 3.0)
 Transmit (Xk) to boolean unit. (BOOL 3.0)

CP01 Next instruction in upper parcel of CIW register.
 Instruction may issue.
 Transmit data from boolean unit to Xi register. (CPU 3.4)
 Set Xi register busy flag. (CPU 3.2)
 Set go boolean flag. (CPU 3.2)

CP02 Clear Xi register busy flag. (CPU 3.2)
 Clear go boolean flag. (CPU 3.2)
 Result in Xi register.

20ijk LEFT SHIFT (Xi) BY jk

ISSUE CONDITIONS

Xi register is free.
X register input path is free in next clock period.
No SAS backup condition.

EXECUTION TIME

No execution delays are possible after this instruction issues from the CIW register. The result is delivered to the Xi register 1 clock period after the instruction issues. The Xi register is reserved for the clock period from issue to delivery of data. The sequence of events for this instruction is:

- CP00 20 instruction in upper parcel of CIW register. (CPU 3.1)
 Instruction issues.
 Transmit next instruction to upper parcel of CIW register.
 Transmit (Xi) to shift unit. (SHIFT 3.0)

- CP01 Next instruction in upper parcel of CIW register. (CPU 3.1)
 Instruction may issue.
 Transmit data from shift unit to Xi register. (CPU 3.4)
 Set Xi register busy flag. (CPU 3.2)
 Set go shift flag. (CPU 3.2)

- CP02 Clear Xi register busy flag. (CPU 3.2)
 Clear go shift flag. (CPU 3.2)
 Result in Xi register.

21ijk RIGHT SHIFT (Xi) BY jk

ISSUE CONDITIONS

Xi register is free.
X register input path is free in next clock period.
No SAS backup condition.

EXECUTION TIME

No execution delays are possible after this instruction issues from the CIW register. The result is delivered to the Xi register 1 clock period after the instruction issues. The Xi register is reserved for the clock period from issue to delivery of data. The sequence of events for this instruction is:

- CP00 21 instruction in upper parcel of CIW register. (CPU 3.1)
 Instruction issues.
 Transmit next instruction to upper parcel of CIW register.
 Transmit (Xi) to shift unit. (SHIFT 3.0)

- CP01 Next instruction in upper parcel of CIW register.
 Instruction may issue.
 Transmit data from shift unit to Xi register. (CPU 3.4)
 Set Xi register busy flag. (CPU 3.2)
 Set go shift flag. (CPU 3.2)

- CP02 Clear Xi register busy flag. (CPU 3.2)
 Clear go shift flag. (CPU 3.2)
 Result in Xi register.

22ijk LEFT SHIFT (Xk) NOMINALLY (Bj) PLACES TO Xi

ISSUE CONDITIONS

Xi register is free.
Bj register is free.
Xk register is free.
X register input path is free in next clock period.
No SAS backup condition.

EXECUTION TIME

No execution delays are possible after this instruction issues from the CIW register. The result is delivered to the Xi register 1 clock period after the instruction issues. The Xi register is reserved for the clock period from issue to delivery of data. The sequence of events for this instruction is:

CP00 22 instruction in upper parcel of CIW register. (CPU 3.1)
 Instruction issues.
 Transmit next instruction to upper parcel of CIW register.
 Transmit (Xk) to shift unit. (SHIFT 3.0)
 Transmit (Bj) to shift unit. (SHIFT 3.0)

CP01 Next instruction in upper parcel of CIW register.
 Instruction may issue.
 Transmit data from shift unit to Xi register. (CPU 3.4)
 Set Xi register busy flag. (CPU 3.2)
 Set go shift flag. (CPU 3.2)

CP02 Clear Xi register busy flag. (CPU 3.2)
 Clear go shift flag. (CPU 3.2)
 Result in Xi register.

23ijk RIGHT SHIFT (Xk) NOMINALLY (Bj) PLACES TO Xi

ISSUE CONDITIONS

Xi register is free.
Bj register is free.
Xk register is free.
X register input path is free in next clock period.
No SAS backup condition.

EXECUTION TIME

No execution delays are possible after this instruction issues from the CIW register. The result is delivered to the Xi register 1 clock period after the instruction issues. The Xi register is reserved for the clock period from issue to delivery of data. The sequence of events for this instruction is:

CP00 23 instruction in upper parcel of CIW register.. (CPU 3.1)
 Instruction issues.
 Transmit next instruction to upper parcel of CIW register.
 Transmit (Xk) to shift unit. (SHIFT 3.0)
 Transmit (Bj) to shift unit. (SHIFT 3.0)

CP01 Next instruction in upper parcel of CIW register.
 Instruction may issue.
 Transmit data from shift unit to Xi register. (CPU 3.4)
 Set Xi register busy flag. (CPU 3.2)
 Set go shift flag. (CPU 3.2)

CP02 Clear Xi register busy flag. (CPU 3.2)
 Clear go shift flag. (CPU 3.2)
 Result in Xi register.

24ijk NORMALIZE (Xk) TO Xi AND Bj

ISSUE CONDITIONS

Xi register is free.
Xk register is free.
Bj register is free.
X register input path is free in 2 clock periods.
B register input path is free in 2 clock periods.
No SAS backup condition.

EXECUTION TIME

No execution delays are possible after this instruction issues from the CIW register. The results are delivered to the Xi and Bj registers 2 clock periods after the instruction issues. The Xi and Bj registers are reserved for the 2 clock periods from issue to delivery of data. The sequence of events for this instruction is:

CP00	24 instruction in upper parcel of CIW register. (CPU 3.1) Instruction issues. Transmit next instruction to upper parcel of CIW register. Transmit (Xk) to normalize unit. (NORM 3.0)
CP01	Next instruction in upper parcel of CIW register. Instruction may issue. Calculate normalize shift count. (NORM 3.0) Move operand from input register to internal register. (NORM 3.0) Set Xi register busy flag. (CPU 3.2) Set Bj register busy flag. (CPU 3.2) Set go normalize flag. (CPU 3.2)
CP02	Transmit data from normalize unit to Xi register. (CPU 3.4) Transmit data from normalize unit to Bj register. (CPU 3.5) Clear go normalize flag. (CPU 3.2)
CP03	Clear Xi register busy flag. (CPU 3.2) Clear Bj register busy flag. (CPU 3.2) Result in Xi and Bj registers.

25ijk ROUND NORMALIZE (Xk) TO Xi AND Bj

ISSUE CONDITIONS

Xi register is free.
Xk register is free.
Bj register is free.
X register input path is free in 2 clock periods.
B register input path is free in 2 clock periods.
No SAS backup condition.

EXECUTION TIME

No execution delays are possible after this instruction issues from the CIW register. The results are delivered to the Xi and Bj registers 2 clock periods after the instruction issues. The Xi and Bj registers are reserved for the 2 clock periods from issue to delivery of data. The sequence of events for this instruction is:

CP00	25 instruction in upper parcel of CIW register. (CPU 3.1) Instruction issues. Transmit next instruction to upper parcel of CIW register. Transmit (Xk) to normalize unit. (NORM 3.0)
CP01	Next instruction in upper parcel of CIW register. Instruction may issue. Calculate normalize shift count. Move operand from input register to internal register. (NORM 3.0) Set Xi register busy flag. (CPU 3.2) Set Bj register busy flag. (CPU 3.2) Set go normalize flag. (CPU 3.2)
CP02	Transmit data from normalize unit to Xi register. (CPU 3.4) Transmit data from normalize unit to Bj register. (CPU 3.5) Clear go normalize flag. (CPU 3.2)
CP03	Clear Xi register busy flag. (CPU 3.2) Clear Bj register busy flag. (CPU 3.2) Result in Xi and Bj registers.

26ijk UNPACK (Xk) TO Xi AND Bj

ISSUE CONDITIONS

Xi register is free.
Xk register is free.
Bj register is free.
X register input path is free in next clock period.
B register input path is free in next clock period.
No SAS backup condition.

EXECUTION TIME

No execution delays are possible after this instruction issues from the CIW register. The results are delivered to the Xi and Bj registers 1 clock period after the instruction issues. The Xi and Bj registers are reserved for the clock period from issue to delivery of data. The sequence of events for this instruction is:

CP00 26 instruction in upper parcel of CIW register. (CPU 3.1)
 Instruction issues.
 Transmit next instruction to upper parcel of CIW register.
 Transmit (Xk) to boolean unit. (BOOL 3.0)

CP01 Next instruction in upper parcel of CIW register.
 Instruction may issue.
 Transmit data from boolean unit to Xi register. (CPU 3.4)
 Transmit data from boolean unit to Bj register. (CPU 3.5)
 Set Xi register busy flag. (CPU 3.2)
 Set Bj register busy flag. (CPU 3.2)
 Set go boolean flag. (CPU 3.2)

CP02 Clear Xi register busy flag. (CPU 3.2)
 Clear Bj register busy flag. (CPU 3.2)
 Clear go boolean flag. (CPU 3.2)
 Result in Xi and Bj registers.

27ijk PACK (Xk) AND (Bj) TO Xi

ISSUE CONDITIONS

Xi register is free.
Xk register is free.
Bj register is free.
X register input path is free in next clock period.
No SAS backup condition.

EXECUTION TIME

No execution delays are possible after this instruction issues from the CIW register. The result is delivered to the Xi register 1 clock period after the instruction issues. The Xi register is reserved for the clock period from issue to delivery of data. The sequence of events for this instruction is:

CP00 27 instruction in upper parcel of CIW register. (CPU 3.1)
 Instruction issues.
 Transmit next instruction to upper parcel of CIW register. (CPU 3.1)
 Transmit (Xk) to boolean unit. (BOOL 3.0)
 Transmit (Bj) to boolean unit. (BOOL 3.0)

CP01 Next instruction in upper parcel of CIW register.
 Instruction may issue.
 Transmit data from boolean unit to Xi register. (BOOL 3.0)
 Set Xi registers busy flag. (CPU 3.2)
 Set go boolean flag. (CPU 3.2)

CP02 Clear go boolean flag. (CPU 3.2)
 Result in Xi register.

30ijk FLOATING SUM OF (Xj) AND (Xk) TO Xi

ISSUE CONDITIONS

Ni register is free.
Xj register is free.
Xk register is free.
X register input path is free in 3 clock periods.
No SAS backup condition.

EXECUTION TIME

No execution delays are possible after this instruction issues from the CIW register. The execution time is a constant for all cases of operands. The result is delivered to the Xi register 3 clock periods after the instruction issues. The Xi register is reserved for the 3 clock periods from issue to delivery of data. The sequence of events for this instruction is:

CP00 30 instruction in upper parcel of CIW register. (CPU 3.1)
 Instruction issues.
 Transmit next instruction to upper parcel of CIW register.
 Transmit (Xk) to floating add unit. (FAD 3.0)
 Transmit (Xj) to floating add unit. (FAD 3.0)

CP01 Next instruction in upper parcel of CIW register.
 Instruction may issue.
 Form difference of operand exponents. (FAD 3.0)
 Select reference operand (FAD 3.1)
 Select shifted operand. (FAD 3.1)
 Perform initial coefficient alignment. (FAD 3.1)
 Set Xi register busy flag. (CPU 3.2)
 Set go floating add flag. (CPU 3.2)

CP02 Complete coefficient alignment. (FAD 3.1)
 Form partial coefficient sum in double precision mode. (FAD 3.2)
 Clear go floating add flag. (CPU 3.2)

- CP03 Complete double precision coefficient sum. (FAD 3.2)
 Transmit result to destination X register. (CPU 3.4)
- CP04 Clear Xi register busy flag. (CPU 3.2)
 Result in Xi register.

31ijk FLOATING DIFFERENCE OF (Xj) MINUS (Xk) TO Xi

ISSUE CONDITIONS

- Xi register is free.
 Xj register is free.
 Xk register is free.
 X register input path is free in 3 clock periods.
 No SAS backup condition.

EXECUTION TIME

No execution delays are possible after this instruction issues from the CIW register. The execution time is a constant for all cases of operands. The result is delivered to the Xi register 3 clock periods after the instruction issues. The Xi register is reserved for the 3 clock periods from issue to delivery of data. The sequence of events for this instruction is:

- CP00 31 instruction in upper parcel of CIW register. (CPU 3.1)
 Instruction issues.
 Transmit next instruction to upper parcel of CIW register.
 Transmit (Xk) to floating add unit. (FAD 3.0)
 Transmit (Xj) to floating add unit. (FAD 3.0)
- CP01 Next instruction in upper parcel of CIW register.
 Instruction may issue.
 Form difference of operand exponents. (FAD 3.0)
 Select reference operand.(FAD 3.0)
 Select shifted operand. (FAD 3.1)
 Perform initial coefficient alignment. (FAD 3.1)
 Set Xi register busy flag. (CPU 3.2)
 Set go floating add flag. (CPU 3.2)

- CP02 Complete coefficient alignment. (FAD 3.1)
 Form partial coefficient difference in double precision mode. (FAD 3.2)
 Clear go floating add flag. (CPU 3.2)
- CP03 Complete double precision coefficient difference. (FAD 3.2)
 Transmit result to destination X register. (CPU 3.4)
- CP04 Clear Xi register busy flag. (CPU 3.2)
 Result in Xi register.

32ijk FLOATING DP SUM OF (Xj) AND (Xk) TO Xi

ISSUE CONDITIONS

- Xi register is free.
 Xj register is free.
 Xk register is free.
 X register input path is free in 3 clock periods.
 No SAS backup condition.

EXECUTION TIME

No execution delays are possible after this instruction issues from the CIW register. The execution time is a constant for all cases of operands. The result is delivered to the Xi register 3 clock periods after the instruction issues. The Xi register is reserved for the 3 clock periods from issue to delivery of data. The sequence of events for this instruction is:

- CP00 32 instruction in upper parcel of CIW register. (CPU 3.1)
 Instruction issues.
 Transmit next instruction to upper parcel of CIW register.
 Transmit (Xk) to floating add unit. (FAD 3.0)
 Transmit (Xj) to floating add unit. (FAD 3.0)

- CP01 Next instruction in upper parcel of CIW register.
 Instruction may issue.
 Form difference of operand exponents. (FAD 3.0)
 Select reference operand. (FAD 3.1)
 Select shifted operand. (FAD 3.1)
 Perform initial coefficient alignment. (FAD 3.1)
 Set Xi register busy flag. (CPU 3.2)
 Set go floating add flag. (CPU 3.2)
- CP02 Complete coefficient alignment. (FAD 3.1)
 Form partial coefficient sum in double precision mode. (FAD 3.2)
 Perform double precision exponent correction. (FAD 3.0)
 Clear go floating add flag. (CPU 3.2)
- CP03 Complete double precision coefficient sum. (FAD 3.2)
 Transmit result to destination X register. (CPU 3.4)
- CP04 Clear Xi register busy flag. (CPU 3.2)
 Result in Xi register.

33ijk FLOATING DP DIFFERENCE OF (Xj) MINUS (Xk) TO Xi

ISSUE CONDITIONS

- Xi register is free.
- Xj register is free.
- Xk register is free.
- X register input path is free in 3 clock periods.
- No SAS backup condition.

EXECUTION TIME

No execution delays are possible after this instruction issues from the CIW register. The execution time is a constant for all cases of operands. The result is delivered to the Xi register 3 clock periods after the instruction issues. The Xi register is reserved for the 3 clock periods from issue to delivery of data. The sequence of events for this instruction is:

- CP00 33 instruction in upper parcel of CIW register. (CPU 3.1)
 Instruction issues.
 Transmit next instruction to upper parcel of CIW register.
 Transmit (Xk) to floating add unit. (FAD 3.0)
 Transmit (Xj) to floating add unit. (FAD 3.0)
- CP01 Next instruction in upper parcel of CIW register.
 Instruction may issue.
 Form difference of operand exponents. (FAD 3.0)
 Select reference operand. (FAD 3.1)
 Select shifted operand. (FAD 3.1)
 Perform initial coefficient alignment. (FAD 3.1)
 Set Xi register/busy flag. (CPU 3.2)
 Set go floating add flag. (CPU 3.2)
- CP02 Complete coefficient alignment. (FAD 3.1)
 Form partial coefficient difference in double-precision mode. (FAD 3.2)
 Perform double-precision exponent correction. (FAD 3.0)
 Clear go floating add flag. (CPU 3.2)
- CP03 Complete double-precision coefficient difference. (FAD 3.2)
 Transmit result to destination X register. (CPU 3.4)
- CP04 Clear Xi register busy flag. (CPU 3.2)
 Result in Xi register.

34ijk ROUND FLOATING SUM OF (Xj) AND (Xk) TO Xi

ISSUE CONDITIONS

- Xi register is free.
- Xj register is free.
- Xk register is free.
- X register input path is free in 3 clock periods.
- No SAS backup condition.

EXECUTION TIME

No execution delays are possible after this instruction issues from the CIW register. The execution time is a constant for all cases of operands. The result is delivered to the Xi register 3 clock periods after the instruction issues. The Xi register is reserved for the 3 clock periods from issue to delivery of data. The sequence of events for this instruction is:

- CP00 34 instruction in upper parcel of CIW register. (CPU 3.1)
Instruction issues.
Transmit next instruction to upper parcel of CIW register.
Transmit (Xk) to floating add unit. (FAD 3.0)
Transmit (Xj) to floating add unit. (FAD 3.0)
- CP01 Next instruction in upper parcel of CIW register.
Instruction may issue.
Form difference of operand exponents. (FAD 3.0)
Select reference operand. (FAD 3.1)
Select shifted operand. (FAD 3.1)
Perform initial coefficient alignment. (FAD 3.1)
Set Xi register busy flag. (CPU 3.2)
Set go floating add flag. (CPU 3.2)
- CP02 Complete coefficient alignment. (FAD 3.1)
Form partial coefficient sum. (FAD 3.2)
Clear go floating add flag. (CPU 3.2)
- CP03 Complete rounded coefficient sum. (FAD 3.2)
Transmit result to destination X register. (CPU 3.4)
- CP04 Clear Xi register busy flag. (CPU 3.2)
Result in Xi register.

35ijk ROUND FLOATING DIFFERENCE OF (Xj) MINUS (Xk) TO Xi

ISSUE CONDITIONS

Xi register is free.
Xj register is free.
Xk register is free.
X register input path is free in 3 clock periods.
No SAS backup condition.

EXECUTION TIME

No execution delays are possible after this instruction issues from the CIW register. The execution time is a constant for all cases of operands. The result is delivered to the Xi register 3 clock periods after the instruction issues. The Xi register is reserved for the 3 clock periods from issue to delivery of data. The sequence of events for this instruction is:

CP00 35 instruction in upper parcel of CIW register. (CPU 3.1)
 Instruction issues.
 Transmit next instruction to upper parcel of CIW register.
 Transmit (Xk) to floating add unit. (FAD 3.0)
 Transmit (Xj) to floating add unit. (FAD 3.0)

CP01 Next instruction in upper parcel of CIW register.
 Instruction may issue.
 Form difference of operand exponents. (FAD 3.0)
 Select reference operand. (FAD 3.1)
 Select shifted operand. (FAD 3.1)
 Perform initial coefficient alignment. (FAD 3.1)
 Set Xi register busy flag. (CPU 3.2)
 Set go floating add flag. (CPU 3.2)

CP02 Complete coefficient alignment. (FAD 3.1)
 Form partial coefficient difference. (FAD 3.2)
 Clear go floating add flag. (CPU 3.2)

CP03 Complete rounded difference. (FAD 3.2)
 Transmit result to destination X register. (CPU 3.4)

CP04 Clear Xi register busy flag. (CPU 3.2)
Result in Xi register.

36ijk INTEGER SUM OF (Xj) AND (Xk) TO Xi

ISSUE CONDITIONS

Xi register is free.
Xj register is free.
Xk register is free.
X register input path is free in next clock period.
No SAS backup condition.

EXECUTION TIME

No execution delays are possible after this instruction issues from the CIW register. The result is delivered to the Xi register 1 clock period after the instruction issues. The Xi register is reserved for the clock period from issue to delivery of data. The sequence of events for this instruction is:

CP00 36 instruction in upper parcel of CIW register. (CPU 3.1)
Instruction issues.
Transmit next instruction to upper parcel of CIW register.
Transmit (Xj) to long add unit. (LG ADD 3.0)
Transmit (Xk) to long add unit. (LG ADD 3.0)

CP01 Next instruction in upper parcel of CIW register.
Instruction may issue.
Transmit result from long add unit to Xi register. (CPU 3.4)
Clear Xi register busy flag. (CPU 3.2)
Set Xi register busy flag. (CPU 3.2)
Set go long add flag. (CPU 3.2)

CP02 Clear go long add flag. (CPU 3.2)
Result in Xi register.

37ijk INTEGER DIFFERENCE OF (Xj) MINUS (Xk) TO Xi

ISSUE CONDITIONS

Xi register is free.
Xj register is free.
Xk register is free.
X register input path is free in next clock period.
No SAS backup condition.

EXECUTION TIME

No execution delays are possible after this instruction issues from the CIW register. The result is delivered to the Xi register 1 clock period after the instruction issues. The Xi register is reserved for the clock period from issue to delivery of data. The sequence of events for this instruction is:

CP00 37 instruction in upper parcel of CIW register. (CPU 3.1)
 Instruction issues.
 Transmit next instruction to upper parcel of CIW register.
 Transmit (Xj) to long add unit. (LG ADD 3.0)
 Transmit (Xk) to long add unit. (LG ADD 3.0)

CP01 Next instruction in upper parcel of CIW register.
 Instruction may issue.
 Transmit result from long add unit to Xi register. (CPU 3.4)
 Clear Xi register busy flag. (CPU 3.2)
 Set Xi register busy flag. (CPU 3.2)
 Set go long add flag. (CPU 3.2)

CP02 Clear go long add flag. (CPU 3.2)
 Result in Xi register.

40ijk FLOATING PRODUCT OF (Xj) AND (Xk) TO Xi

ISSUE CONDITIONS

Xi register is free.
Xj register is free.
Xk register is free.
Multiply unit is free.
X register input path is free in 4 clock periods.
No SAS backup condition.

EXECUTION TIME

No execution delays are possible after this instruction issues from the CIW register. The result is delivered to the Xi register 4 clock periods after the instruction issues. The Xi register is reserved for the 4 clock periods from issue to delivery of data. The multiply unit is free 2 clock periods after this instruction issues. The sequence of events for this instruction is:

CP00 40 instruction in upper parcel of CIW register. (CPU 3.1)
 Instruction issues.
 Transmit next instruction to upper parcel of CIW register. (CPU 3.1)
 Transmit (Xj) to multiply unit. (MULT 3.0)
 Transmit (Xk) to multiply unit. (MULT 3.0)

CP01 Next instruction in upper parcel of CIW register.
 Instruction may issue.
 Form first 24 by 48 bit product. (MULT 3.0)
 Begin exponent arithmetic. (MULT 3.5)
 Set Xi register busy flag. (CPU 3.2)
 Set multiply busy flag. (CPU 3.2)

CP02 Form second 24 by 48 bit product. (MULT 3.0)
 Partially sum data from previous clock period. (MULT 3.0)
 Sense special case exponent values. (MULT 3.5)
 Clear multiply busy flag. (CPU 3.2)

CP03 Merge two 24 by 48 bit products. (MULT 3.1)

- CP04 Form 96-bit coefficient value. (MULT 3.4)
 Transmit result from multiply unit to Xi register. (CPU 3.4)
- CP05 Clear Xi register busy flag. (CPU 3.2)
 Result in Xi register.

41ijk ROUND FLOATING PRODUCT OF (Xj) AND (Xk) TO Xi

ISSUE CONDITIONS

- Xi register is free.
 Xj register is free.
 Xk register is free.
 Multiply unit is free.
 X register input path is free in 4 clock periods.
 No SAS backup condition.

EXECUTION TIME

No execution delays are possible after this instruction issues from the CIW register. The result is delivered to the Xi register 4 clock periods after the instruction issues. The Xi register is reserved for the 4 clock periods from issue to delivery of data. The multiply unit is free 2 clock periods after this instruction issues. The sequence of events for this instruction is:

- CP00 41 instruction in upper parcel of CIW register. (CPU 3.1)
 Instruction issues.
 Transmit next instruction to upper parcel of CIW register. (CPU 3.1)
 Transmit (Xj) to multiply unit. (MULT 3.0)
 Transmit (Xk) to multiply unit. (MULT 3.0)
- CP01 Next instruction in upper parcel of CIW register. (CPU 3.1)
 Instruction may issue.
 Form first 24 by 48 bit product. (MULT 3.0)
 Begin exponent arithmetic. (MULT 3.5)
 Set Xi register busy flag. (CPU 3.2)
 Set multiply busy flag. (CPU 3.2)

CP02 Form second 24 by 48 bit product. (MULT 3.0)
 Partially sum data from previous clock period. (MULT 3.0)
 Sense special case exponent values. (MULT 3.5)
 Clear multiply busy flag. (CPU 3.2)

CP03 Merge two 24 by 48 bit products. (MULT 3.1)

CP04 Form 96-bit coefficient value. (MULT 3.4)
 Transmit result from multiply unit to Xi register. (CPU 3.4)

CP05 Clear Xi register busy flag. (CPU 3.2)
 Result in Xi register.

42ijk FLOATING DP PRODUCT OF (Xj) AND (Xk) TO Xi

ISSUE CONDITIONS

Xi register is free.
 Xj register is free.
 Xk register is free.
 Multiply unit is free.
 X register input path is free in 4 clock periods.
 No SAS backup condition.

EXECUTION TIME

No execution delays are possible after this instruction issues from the CIW register. The result is delivered to the Xi register 4 clock periods after the instruction issues. The Xi register is reserved for the 4 clock periods from issue to delivery of data. The multiply unit is free 2 clock periods after this instruction issues. The sequence of events for this instruction is:

CP00 42 instruction in upper parcel of CIW register. (CPU 3.1)
 Instruction issues.
 Transmit next instruction to upper parcel of CIW register.
 Transmit (Xj) to multiply unit. (MULT 3.0)
 Transmit (Xk) to multiply unit. (MULT 3.0)

- CP01 Next instruction in upper parcel of CIW register. (CPU 3.1)
 Instruction may issue.
 Form first 24 by 48 bit product. (MULT 3.0)
 Begin exponent arithmetic. (MULT 3.5)
 Set Xi register busy flag. (CPU 3.2)
 Set multiply busy flag. (CPU 3.2)
- CP02 Form second 24 by 48 bit product. (MULT 3.0)
 Partially sum data from previous clock period. (MULT 3.0)
 Sense special case exponent values. (MULT 3.5)
 Clear multiply busy flag. (CPU 3.2)
- CP03 Merge two 24 by 48 bit products. (MULT 3.1)
- CP04 Form 96-bit coefficient value. (MULT 3.4)
 Transmit result from multiply unit to Xi register. (CPU 3.4)
- CP05 Clear Xi register busy flag. (CPU 3.2)
 Result in Xi register.

43ijk FORM MASK OF jk BITS TO Xi

ISSUE CONDITIONS

Xi register is free.
 X register input path is free in next clock period.
 No SAS backup condition.

EXECUTION TIME

No execution delays are possible after this instruction issues from the CIW register. The result is delivered to the Xi register 1 clock period after the instruction issues. The Xi register is reserved for the clock period from issue to delivery of data. The sequence of events for this instruction is:

- CP00 43 instruction in upper parcel of CIW register. (CPU 3.1)
 Instruction issues.
 Transmit next instruction to upper parcel of CIW register.

CP01 Next instruction in upper parcel of CIW register.
 Instruction may issue.
 Transmit data from shift unit to Xi register. (CPU 3.4)
 Clear Xi register busy flag. (CPU 3.2)
 Set Xi register busy flag. (CPU 3.2)
 Set go shift flag. (CPU 3.2)

CP02 Clear go shift flag. (CPU 3.2)
 Result in Xi register.

44ijk FLOATING DIVIDE (Xj) BY (Xk) TO Xi

ISSUE CONDITIONS

Xi register is free.
Xj register is free.
Xk register is free.
Divide unit is free.
X register input path is free in 19 clock periods.
No SAS backup condition.

EXECUTION TIME

No execution delays are possible after this instruction issues from the CIW register. The result is delivered to the Xi register 19 clock periods after the instruction issues. The Xi register is reserved for the 19 clock periods from issue to delivery of data. The divide unit is free 17 clock periods after this instruction issues. The sequence of events for this instruction is:

CP00 44 instruction in upper parcel of CIW register. (CPU 3.1)
 Instruction issues.
 Transmit next instruction to upper parcel of CIW register.
 Transmit (Xj) to divide unit. (DIV 3.2)
 Transmit (Xk) to divide unit. (DIV 3.1)

CP01 Next instruction in upper parcel of CIW register. (CPU 3.1)
 Instruction may issue.
 Enter (Xj) coefficient in dividend register. (DIV 3.2)
 Begin translation of divisor multiples. (DIV 3.1)
 Begin exponent calculation. (DIV 3.0)
 Set Xi register busy flag. (CPU 3.2)
 Set divide busy (begin sequence) flag. (CPU 3.2)

CP02 First trial subtraction. (DIV 3.2)
 Sense quotient overflow. (DIV 3.0)

CP17 Clear divide busy flag. (DIV 3.0)
 Sixteenth trial subtraction. (DIV 3.2)

CP18 Last trial subtraction. (DIV 3.2)
 Enter last quotient bits in quotient register. (DIV 3.0)

CP19 Transmit result from divide unit to Xi register. (DIV 3.0)

CP20 Clear Xi register busy flag. (CPU 3.2)
 Result in Xi register.

45ijk ROUND FLOATING DIVIDE (Xj) BY (Xk) TO Xi

ISSUE CONDITIONS

Xi register is free.
 Xj register is free.
 Xk register is free.
 Divide unit is free.
 X register input path is free in 19 clock periods.
 No SAS backup condition.

EXECUTION TIME

No execution delays are possible after this instruction issues from the CIW register. The result is delivered to the Xi register 19 clock periods after the instruction issues. The Xi register is reserved for the 19 clock periods from issue to delivery of data. The divide unit is free 17 clock periods after this instruction issues. The sequence of events for this instruction is the same as the 44 instruction.

46ixx NO OPERATION (PASS)

EXECUTION TIME

Designator i = 0, 1, 2, 3

No functional unit is involved in the execution of this instruction. The instruction requires 1 clock period to issue. The sequence of events for this instruction is:

- CP00 46 instruction in upper parcel of CIW register. (CPU 3.1)
Instruction issues.
Transmit next instruction to upper parcel of CIW register.
- CP01 Next instruction in upper parcel of CIW register.
Instruction may issue.

47i0k POPULATION COUNT OF (Xk) to Xi

ISSUE CONDITIONS

- Xi register is free.
Xk register is free.
X register input path is free in next clock period.
No SAS backup condition.

EXECUTION TIME

No execution delays are possible after this instruction issues from the CIW register. The result is delivered to the Xi register 1 clock period after the instruction issues. The Xi register is reserved for the clock period from issue to delivery of data. The sequence of events for this instruction is:

- CP00 47 instruction in upper parcel of CIW register. (CPU 3.1)
Instruction issues.
Transmit next instruction to upper parcel of CIW register.
Transmit (Xk) to population count unit. (POP CT 3.0)

CP01 Next instruction in upper parcel of CIW register.
 Instruction may issue.
 Transmit data from population count unit to Xi register. (CPU 3.4)
 Set Xi register busy flag. (CPU 3.2)
 Set go population count flag. (CPU 3.2)

CP02 Clear Xi register busy flag. (CPU 3.2)
 Clear go population count flag. (CPU 3.2)
 Result in Xi register.

50ijK SET Ai TO (Aj) + K

ISSUE CONDITIONS

Xi register is free (i = nonzero).
 Aj register is free.
 Ai register is free.
 No SAS backup condition.
 Memory enable.

EXECUTION TIME

Designator i = 0

No execution delays are possible after this instruction issues from the CIW register if the i designator is zero. The result is delivered to the Ai register 1 clock period after the instruction issues. The Ai register is reserved for the clock period from issue to delivery of data. The sequence of events for this case is:

CP00 50 instruction in upper parcel of CIW register. (CPU 3.1)
 Instruction issues.
 Transmit next instruction to upper parcel of CIW register.
 Transmit K to increment unit. (INCR 3.0)
 Transmit (Aj) to increment unit. (INCR 3.0)

CP01 Next instruction in upper parcel of CIW register.
 Instruction may issue.
 Transmit data from increment unit to Ai register. (CPU 3.5)
 Set Ai register busy flag. (CPU 3.2)
 Set go increment flag. (CPU 3.2)

CP02 Clear Ai register busy flag. (CPU 3.2)
 Clear go increment flag. (CPU 3.2)
 Result in Ai register

Designator i = 1, 2, 3, 4, 5

These values for the i designator cause a CM read reference in addition to the result delivered to the Ai register. No delays are possible from issue to delivery of data to the Ai register and to the SAS. The delivery of data to the Xi register may be delayed in CM by bank conflicts and refresh cycle. The Ai register is reserved for the clock period from issue to delivery of data. The Xi register is reserved from issue to delivery of the word from CM. The sequence of events for this case is:

CP00 50 instruction in upper parcel of CIW register. (CPU 3.1)
 Instruction issues.
 Transmit next instruction to upper parcel of CIW register.
 Transmit K to increment unit. (INCR 3.0)
 Transmit (Aj) to increment unit. (INCR 3.0)

CP01 Next instruction in upper parcel of CIW register.
 Instruction may issue.
 Set Ai register busy flag. (CPU 3.2)
 Set Xi register busy flag. (CPU 3.2)
 Set go increment flag. (CPU 3.2)

CP02 Address leaves SAS for CM. (CMC 3.0)
 Transmit data from increment unit to Ai register. (CPU 3.5)
 Clear Ai register busy flag. (CPU 3.2)
 Transmit address plus (RAC) to SAS. (CMC 3.0)
 Tag for read CM to X. (CMC 3.1)
 Clear go increment flag. (CPU 3.2)

CP03 Begin CM read cycle.

CP22 Transmit CM read data to Xi register. (CPU 3.4)

CP23 Clear Xi register busy flag. (CPU 3.2)
 CM read data in Xi register.

Designator i = 6, 7

These values for the i designator cause a CM write reference in addition to the result delivered to the Ai register. No delays are possible from issue to delivery of data to the Ai register and to the SAS. There may be delays in initiating the CM reference to store the data from the Xi register. These delays do not affect the timing for this instruction because the data is read from the Xi register at issue time and is held in the SWS if a delay occurs. The Ai register is reserved for the clock period from issue to delivery of data. The Xi register is not reserved. The sequence of events for this case is:

CP00 50 instruction in upper parcel of CIW register. (CPU 3.1)
 Instruction issues.
 Transmit next instruction to upper parcel of CIW register.
 Transmit K to increment unit. (INCR 3.0)
 Transmit (Aj) to increment unit. (INCR 3.0)

CP01 Next instruction in upper parcel of CIW register.
 Instruction may issue.
 Set Ai register busy flag. (CPU 3.2)
 Transmit (Xi) to SWS. (CMC 3.1)
 Set go increment flag. (CPU 3.2)

CP02 Address leaves SAS for CMC. (CMC 3.0)
 Transmit data from increment unit to Ai register. (CPU 3.5)
 Clear Ai register busy flag. (CPU 3.2)
 Transmit address plus (RAC) to SAS. (CMC 3.0)
 Tag for write CM from X. (CMC 3.1)
 Clear go increment flag. (CPU 3.2)

CP03 Begin CM write cycle.

51ijK SET Ai TO (Bj) + K

ISSUE CONDITIONS

Xi register is free (i = nonzero).
Bj register is free.
Ai register is free.
No SAS backup condition.
Memory enable.

EXECUTION TIME

Designator i = 0

No execution delays are possible after this instruction issues from the CIW register if the i designator is zero. The result is delivered to the Ai register one clock period after the instruction issues. The Ai register is reserved for the clock period from issue to delivery of data. The sequence of events for this case is:

CP00 51 instruction in upper parcel of CIW register. (CPU 3.1)
 Instruction issues.
 Transmit next instruction to upper parcel of CIW register. (CPU 3.1)
 Transmit K to increment unit. (INCR 3.0)
 Transmit (Bj) to increment unit. (INCR 3.0)

CP01 Next instruction in upper parcel of CIW register. (CPU 3.1)
 Instruction may issue.
 Transmit data from increment unit to Ai register. (CPU 3.5)
 Set Ai register busy flag. (CPU 3.2)
 Set go increment flag. (CPU 3.2)

CP02 Clear Ai register busy flag. (CPU 3.2)
 Clear go increment flag. (CPU 3.2)
 Result in Ai register.

Designator i = 1, 2, 3, 4, 5

These values for the i designator cause a CM read reference in addition to the result delivered to the Ai register. No delays are possible from issue to delivery of data to the Ai register and to the SAS. The delivery of data to the Xi register may be delayed in CM by bank conflicts and refresh cycle. The Ai register is reserved for the clock period from issue to delivery of data. The Xi register is reserved from issue to delivery of the word from CM. The sequence of events for this case is:

- CP00 51 instruction in upper parcel of CIW register. (CPU 3.1)
Instruction issues.
Transmit next instruction to upper parcel of CIW register.
Transmit K to increment unit. (INCR 3.0)
Transmit (Bj) to increment unit. (INCR 3.0)

- CP01 Next instruction in upper parcel of CIW register. (CPU 3.1)
Instruction may issue.
Set Ai register busy flag. (CPU 3.2)
Set Xi register busy flag. (CPU 3.2)
Set go increment flag. (CPU 3.2)

- CP02 Address leaves SAS for CM. (CMC 3.0)
Transmit data from increment unit to Ai register. (CPU 3.5)
Clear Ai register busy flag. (CPU 3.2)
Transmit address plus (RAC) to SAS. (CMC 3.0)
Tag for read CM to X. (CMC 3.1)
Clear go increment flag. (CPU 3.2)

- CP03 Begin CM read cycle.

- CP22 Transmit CM read data to Xi register. (CPU 3.4)

- CP23 Clear Xi register busy flag. (CPU 3.2)
CM read data in Xi register.

Designator i = 6, 7

These values for the i designator cause a CM write reference in addition to the result delivered to the Ai register. No delays are possible from issue to delivery of data to the Ai register and to the SAS. There may be delays in initiating the CM reference to store the data from the Xi register. These delays do not affect the timing for this instruction because the data is read from the Xi register at issue time and is held in the SWS if a delay occurs. The Ai register is reserved for the clock period from issue to delivery of data. The Xi register is not reserved. The sequence of events for this case is:

- CP00 51 instruction in upper parcel of CIW register. (CPU 3.1)
Instruction issues.
Transmit next instruction to upper parcel of CIW register.
Transmit K to increment unit. (INCR 3.0)
Transmit (Bj) to increment unit. (INCR 3.0)
- CP01 Next instruction in upper parcel of CIW register.
Instruction may issue.
Set Ai register busy flag. (CPU 3.2)
Transmit (Xi) to SWS. (CMC 3.1)
Set go increment flag. (CPU 3.2)
- CP02 Address leaves SAS for CM. (CMC 3.0)
Transmit data from increment unit to Ai register. (CPU 3.5)
Clear Ai register busy flag. (CPU 3.2)
Transmit address plus (RAC) to SAS. (CMC 3.0)
Tag for write CM from X. (CMC 3.1)
Clear go increment flag. (CPU 3.2)
- CP03 Begin CM write cycle.

52ijk SET Ai TO (Xj) + K

ISSUE CONDITIONS

Xi register is free (i = nonzero).
Xj register is free.
Ai register is free.
No SAS backup condition.
Memory enable.

EXECUTION TIME

Designator i = 0

No execution delays are possible after this instruction issues from the CIW register if the i designator is zero. The result is delivered to the Ai register one clock period after the instruction issues. The Ai register is reserved for the clock period from issue to delivery of data. The sequence of events for this case is:

- CP00 52 instruction in upper parcel of CIW register. (CPU 3.1)
Instruction issues.
Transmit next instruction to upper parcel of CIW register.
Transmit K to increment unit. (INCR 3.0)
Transmit (Xj) to increment unit. (INCR 3.0)
- CP01 Next instruction in upper parcel of CIW register.
Instruction may issue.
Transmit data from increment unit to Ai register. (CPU 3.5)
Set Ai register busy flag. (CPU 3.2)
Set go increment flag. (CPU 3.2)
- CP02 Clear Ai register busy flag. (CPU 3.2)
Clear go increment flag. (CPU 3.2)
Result in Ai register.

Designator i = 1, 2, 3, 4, 5

These values for the i designator cause a CM read reference in addition to the result delivered to the Ai register. No delays are possible from issue to delivery of data to the Ai register and to the SAS. The delivery of data to the Xi register may be delayed in CM by bank conflicts and refresh cycle. The Ai register is reserved for the clock period from issue to delivery of data. The Xi register is reserved from issue to delivery of the word from CM. The sequence of events for this case is:

- CP00 52 instruction in upper parcel of CIW register. (CPU 3.1)
Instruction issues.
Transmit next instruction to upper parcel of CIW register.
Transmit K to increment unit. (INCR 3.0)
Transmit (Xj) to increment unit. (INCR 3.0)
- CP01 Next instruction in upper parcel of CIW register.
Instruction may issue.
Set Ai register busy flag. (CPU 3.2)
Set Xi register busy flag. (CPU 3.2)
Set go increment flag. (CPU 3.2)
- CP02 Address leaves SAS for CM. (CMC 3.0)
Transmit data from increment unit to Ai register. (CPU 3.5)
Clear Ai register busy flag.
Transmit address plus (RAC) to SAS. (CMC 3.0)
Tag for read CM to X. (CMC 3.1)
Clear go increment flag. (CPU 3.2)
- CP03 Begin CM read cycle.
- CP22 Transmit CM read data to Xi register. (CPU 3.4)
- CP23 Clear Xi register busy flag. (CPU 3.2)
CM read data in Xi register.

Designator i = 6, 7

These values for the i designator cause a CM write reference in addition to the result delivered to the Ai register. No delays are possible from issue to delivery of data to the Ai register and to the SAS. There may be delays in initiating the CM reference to store the data from the Xi register. These delays do not affect the timing for this instruction because the data is read from the Xi register at issue time and is held in the SWS if a delay occurs. The Ai register is reserved for the clock period from issue to delivery of data. The Xi register is not reserved. The sequence of events for this case is:

- CP00 52 instruction in upper parcel of CIW register. (CPU 3.1)
Instruction issues.
Transmit next instruction to upper parcel of CIW register.
Transmit K to increment unit. (INCR 3.0)
Transmit (Xj) to increment unit. (INCR 3.0)
- CP01 Next instruction in upper parcel of CIW register.
Instruction may issue.
Transmit data from increment unit to Ai register. (CPU 3.5)
Set Ai register busy flag. (CPU 3.2)
Transmit (Xi) to SWS. (CMC 3.1)
Set go increment flag. (CPU 3.2)
- CP02 Address leaves SAS for CM. (CMC 3.0)
Clear Ai register busy flag. (CPU 3.2)
Transmit address plus (RAC) to the SAS. (CMC 3.0)
Tag for write CM from X. (CMC 3.1)
Clear go increment flag. (CPU 3.2)
- CP03 Begin CM write cycle.

53ijk SET Ai TO (Xj) + (Bk)

ISSUE CONDITIONS

Xi register is free (i = nonzero).
Xj register is free.
Bk register is free.
Ai register is free.
No SAS backup condition.
Memory enable.

EXECUTION TIME

Designator i = 0

No execution delays are possible after this instruction issues from the CIW register if the i designator is zero. The result is delivered to the Ai register one clock period after the instruction issues. The Ai register is reserved for the clock period from issue to delivery of data. The sequence of events for this case is:

CP00 53 instruction in upper parcel of CIW register. (CPU 3.1)
 Instruction issues.
 Transmit next instruction to upper parcel of CIW register.
 Transmit (Xj) to increment unit. (INCR 3.0)
 Transmit (Bk) to increment unit. (INCR 3.0)

CP01 Next instruction in upper parcel of CIW register.
 Instruction may issue.
 Set Ai register busy flag. (CPU 3.2)
 Set go increment flag. (CPU 3.2)
 Transmit data from increment unit to Ai register. (CPU 3.5)

CP02 Clear Ai register busy flag. (CPU 3.2)
 Clear go increment flag. (CPU 3.2)
 Result in Ai register.

Designator i = 1, 2, 3, 4, 5

These values for the i designator cause a CM read reference in addition to the result delivered to the Ai register. No delays are possible from issue to delivery of data to the Ai register and to the SAS. The delivery of data to the Xi register may be delayed in CM by bank conflicts and refresh cycle. The Ai register is reserved for the clock period from issue to delivery of data. The Xi register is reserved from issue to delivery of the word from CM. The sequence of events for this case is:

- CP00 53 instruction in upper parcel of CIW register. (CPU 3.1)
Instruction issues.
Transmit next instruction to upper parcel of CIW register.
Transmit (Xj) to increment unit. (INCR 3.0)
Transmit (Bk) to increment unit. (INCR 3.0)
- CP01 Next instruction in upper parcel of CIW register.
Instruction may issue.
Transmit data from increment unit to Ai register. (CPU 3.5)
Set Ai register busy flag. (CPU 3.2)
Set Xi register busy flag. (CPU 3.2)
Set go increment flag. (CPU 3.2)
- CP02 Address leaves SAS for CM. (CMC 3.0)
Clear Ai register busy flag. (CPU 3.2)
Transmit address plus (RAC) to SAS. (CMC 3.0)
Tag for read CM to X. (CMC 3.1)
Clear go increment flag. (CPU 3.2)
- CP03 Begin CM read cycle.
- CP22 Transmit CM read data to Xi register. (CPU 3.4)
- CP23 Clear Xi register busy flag. (CPU 3.2)
CM read data in Xi register.

Designator i = 6, 7

These values for the i designator cause a CM write reference in addition to the result delivered to the Ai register. No delays are possible from issue to delivery of data to the Ai register and to the SAS. There may be delays in initiating the CM reference to store the data from the Xi register. These delays do not affect the timing for this instruction because the data is read from the Xi register at issue time and is held in the SWS if a delay occurs. The Ai register is reserved for the clock period from issue to delivery of data. The Xi register is not reserved. The sequence of events for this case is:

- CP00 53 instruction in upper parcel of CIW register. (CPU 3.1)
Instruction issues.
Transmit next instruction to upper parcel of CIW register.
Transmit (Xj) to increment unit. (INCR 3.0)
Transmit (Bk) to increment unit. (INCR 3.0)
- CP01 Next instruction in upper parcel of CIW register.
Instruction may issue.
Transmit data from increment unit to Ai register. (CPU 3.5)
Set go increment flag. (CPU 3.2)
Transmit (Xi) to SWS. (CMC 3.1)
Set Ai register busy flag. (CPU 3.2)
- CP02 Address leaves SAS for CM. (CMC 3.0)
Clear Ai register busy flag. (CPU 3.2)
Transmit address plus (RAC) to SAS. (CMC 3.0)
Tag for write CM from X. (CMC 3.1)
Clear go increment flag. (CPU 3.2)
- CP03 Begin CM write cycle.

54ijk SET Ai TO (Aj) + (Bk)

ISSUE CONDITIONS

Xi register is free (i = nonzero).
Aj register is free.
Bk register is free.
Ai register is free.
No SAS backup condition.
Memory enable.

EXECUTION TIME

Designator i = 0

No execution delays are possible after this instruction issues from the CIW register if the i designator is zero. The result is delivered to the Ai register one clock period after the instruction issues. The Ai register is reserved for the clock period from issue to delivery of data. The sequence of events for this case is:

- CP00 54 instruction in upper parcel of CIW register. (CPU 3.1)
 Instruction issues.
 Transmit next instruction to upper parcel of CIW register.
 Transmit (Aj) to increment unit. (INCR 3.0)
 Transmit (Bk) to increment unit. (INCR 3.0)
- CP01 Next instruction in upper parcel of CIW register.
 Instruction may issue.
 Transmit data from increment unit to Ai register. (CPU 3.5)
 Set Ai register busy flag. (CPU 3.2)
 Set go increment flag. (CPU 3.2)
- CP02 Clear Ai register busy flag. (CPU 3.2)
 Clear go increment flag. (CPU 3.2)
 Result in Ai register.

Designator i = 1, 2, 3, 4, 5

These values for the i designator cause a CM read reference in addition to the result delivered to the Ai register. No delays are possible from issue to delivery of data to the Ai register and to the SAS. The delivery of data to the Xi register may be delayed in CM by bank conflicts and refresh cycle. The Ai register is reserved for the clock period from issue to delivery of data. The Xi register is reserved from issue to delivery of the word from CM. The sequence of events for this case is:

- CP00 54 instruction in upper parcel of CIW register. (CPU 3.1)
Instruction issues.
Transmit next instruction to upper parcel of CIW register.
Transmit (Aj) to increment unit. (INCR 3.0)
- CP01 Next instruction in upper parcel of CIW register.
Instruction may issue.
Transmit data from increment unit to Ai register. (CPU 3.5)
Set Ai register busy flag. (CPU 3.2)
Set Xi register busy flag. (CPU 3.2)
Set go increment flag. (CPU 3.2)
- CP02 Address leaves SAS for CM. (CMC 3.0)
Clear Ai register busy flag. (CPU 3.2)
Transmit address plus (RAC) to SAS. (CMC 3.0)
Tag for read CM to X. (CMC 3.1)
Clear go increment flag. (CPU 3.2)
- CP03 Begin CM read cycle.
- CP22 Transmit CM read data to Xi register. (CPU 3.4)
- CP23 Clear Xi register busy flag. (CPU 3.2)
CM read data in Xi register.

Designator i = 6, 7

These values for the i designator cause a CM write reference in addition to the result delivered to the Ai register. No delays are possible from issue to delivery of data to the Ai register and to the SAS. There may be delays in initiating the CM reference to store the data from the Xi register. These delays do not affect the timing for this instruction because the data is read from the Xi register at issue time and is held in the SWS if a delay occurs. The Ai register is reserved for the clock period from issue to delivery of data. The Xi register is not reserved. The sequence of events for this case is:

- CP00 54 instruction in upper parcel of CIW register. (CPU 3.1)
Instruction issues.
Transmit next instruction to upper parcel of CIW register.
Transmit (Aj) to increment unit. (INCR 3.0)
Transmit (Bk) to increment unit. (INCR 3.0)
- CP01 Next instruction in upper parcel of CIW register.
Instruction may issue.
Set Ai register busy flag. (CPU 3.2)
Transmit (Xi) to SWS. (CMC 3.1)
Set go increment flag. (CPU 3.2)
- CP02 Address leaves SAS for CM. (CMC 3.0)
Transmit data from increment unit to Ai register. (CPU 3.5)
Clear Ai register busy flag. (CPU 3.2)
Transmit address plus (RAC) to SAS. (CMC 3.0)
Tag for write CM from X. (CMC 3.1)
Clear go increment flag. (CPU 3.2)
- CP03 Begin CM write cycle

55ijk SET Ai TO (Aj) - (Bk)

ISSUE CONDITIONS AND EXECUTION TIME

The issue conditions and execution time for this instruction are the same as the 54 instruction.

56ijk SET Ai TO (Bj) + (Bk)

ISSUE CONDITIONS

Xi register is free (i = nonzero).

Bj register is free.

Bk register is free.

Ai register is free.

No SAS backup condition.

Memory enable.

EXECUTION TIME

Designator i = 0

No execution delays are possible after this instruction issues from the CIW register if the i designator is zero. The result is delivered to the Ai register 1 clock period after the instruction issues. The Ai register is reserved for the clock period from issue to delivery of data. The sequence of events for this case is:

- CP00 56 instruction in upper parcel of CIW register. (CPU 3.1)
Instruction issues.
Transmit next instruction to upper parcel of CIW register.
Transmit (Bj) to increment unit. (INCR 3.0)
Transmit (Bk) to increment unit. (INCR 3.0)
- CP01 Next instruction in upper parcel of CIW register.
Instruction may issue.
Transmit data from increment unit to Ai register. (CPU 3.5)
Set Ai register busy flag. (CPU 3.2)
Set go increment flag. (CPU 3.2)
- CP02 Clear Ai register busy flag. (CPU 3.2)
Clear go increment flag. (CPU 3.2)
Result in Ai register.

Designator i = 1, 2, 3, 4, 5

These values for the i designator cause a CM read reference in addition to the result delivered to the Ai register. No delays are possible from issue to delivery of data to the Ai register and to the SAS. The delivery of data to the Xi register may be delayed in CM by bank conflicts and refresh cycle. The Ai register is reserved for the clock period from issue to delivery of data. The Xi register is reserved from issue to delivery of the word from CM. The sequence of events for this case is:

- CP00 56 instruction in upper parcel of CIW register. (CPU 3.1)
Instruction issues.
Transmit next instruction to upper parcel of CIW register.
Transmit (Bj) to increment unit. (INCR 3.0)
Transmit (Bk) to increment unit. (INCR 3.0)
- CP01 Next instruction in upper parcel of CIW register.
Instruction may issue.
Transmit data from increment unit to Ai register. (CPU 3.5)
Set Ai register busy flag. (CPU 3.2)
Set Xi register busy flag. (CPU 3.2)
Set go increment flag. (CPU 3.2)
- CP02 Address leaves SAS for CM. (CMC 3.0)
Clear Ai register busy flag. (CPU 3.2)
Transmit address plus (RAC) to SAS. (CMC 3.0)
Tag for read CM to X. (CMC 3.1)
Clear go increment flag. (CPU 3.2)
- CP03 Begin CM read cycle.
- CP22 Transmit CM read data to Xi register. (CPU 3.4)
- CP23 Clear Xi register busy flag. (CPU 3.2)
CM read data in Xi register.

Designator i = 6, 7

These values for the i designator cause a CM write reference in addition to the result delivered to the Ai register. No delays are possible from issue to delivery of data to the Ai register and to the SAS. There may be delays in initiating the CM reference to store the data from the Xi register. These delays do not affect the timing for this instruction because the data is read from the Xi register at issue time and is held in the SWS if a delay occurs. The Ai register is reserved for the clock period from issue to delivery of data. The Xi register is not reserved. The sequence of events for this case is:

- CP00 56 instruction in upper parcel of CIW register. (CPU 3.1)
Instruction issues.
Transmit next instruction to upper parcel of CIW register.
Transmit (Bj) to increment unit. (INCR 3.0)
Transmit (Bk) to increment unit. (INCR 3.0)
- CP01 Next instruction in upper parcel of CIW register.
Instruction may issue.
Transmit data from increment unit to Ai register. (CPU 3.5)
Set Ai register busy flag. (CPU 3.2)
Transmit (Xi) to SWS. (CMC 3.1)
Set go increment flag. (CPU 3.2)
- CP02 Address leaves SAS for CM. (CMC 3.0)
Clear Ai register busy flag. (CPU 3.2)
Transmit address plus (RAC) to SAS. (CMC 3.0)
Tag for write CM from X. (CMC 3.1)
Clear go increment flag. (CPU 3.2)
- CP03 Begin CM write cycle.

57ijk SET Ai TO (Bj) - (Bk)

ISSUE CONDITIONS AND EXECUTION TIME

The issue conditions and execution time for this instruction are the same as the 56 instruction.

60ijK SET Bi TO (Aj) + K

ISSUE CONDITIONS

Bi register is free.
Aj register is free.
B register input path is free in next clock period.
No SAS backup condition.

EXECUTION TIME

No execution delays are possible after this instruction issues from the CIW register. The result is delivered to the Bi register 1 clock period after the instruction issues. The Bi register is reserved for the clock period from issue to delivery of data. The sequence of events for this instruction is:

- CP00 60 instruction in upper parcel of CIW register. (CPU 3.1)
 Instruction issues.
 Transmit next instruction to upper parcel of CIW register.
 Transmit K to increment unit. (INCR 3.0)
 Transmit (Aj) to increment unit. (INCR 3.0)

- CP01 Next instruction in upper parcel of CIW register.
 Instruction may issue.
 Transmit data from increment unit to Bi register. (CPU 3.5)
 Set Bi register busy flag. (CPU 3.2)
 Set go increment flag. (CPU 3.2)

- CP02 Clear Bi register busy flag. (CPU 3.2)
 Clear go increment flag. (CPU 3.2)
 Result in Bi register.

61ijk SET BI TO (Bj) + K

ISSUE CONDITIONS

Bi register is free.

Bj register is free.

B register input path is free in next clock period.

No SAS backup condition.

EXECUTION TIME

The execution timing for this instruction is the same as the 60 instruction. The sequence of events is the same except that the Bj register is used for the operand rather than the Aj register.

62ijk SET Bi TO (Xj) + K

ISSUE CONDITIONS

Xj register is free.

Bi register is free.

B register input path is free in next clock period.

No SAS backup condition.

EXECUTION TIME

The execution timing for this instruction is the same as the 60 instruction. The sequence of events is the same except that the Xj register is used for the operand rather than the Aj register.

63ijk SET Bi TO (Xj) + (Bk)

ISSUE CONDITIONS

Xj register is free.
Bk register is free.
Bi register is free.
B register input path is free in next clock period.
No SAS backup condition.

EXECUTION TIME

No execution delays are possible after this instruction issues from the CIW register. The result is delivered to the Bi register 1 clock period after the instruction issues. The Bi register is reserved for the clock period from issue to delivery of data. The sequence of events for this instruction is:

- CP00 63 instruction in upper parcel of CIW register. (CPU 3.1)
 Instruction issues.
 Transmit next instruction to upper parcel of CIW register.
 Transmit (Xj) to increment unit. (INCR 3.0)
 Transmit (Bk) to increment unit. (INCR 3.0)

- CP01 Next instruction in upper parcel of CIW register.
 Instruction may issue.
 Transmit data from increment unit to Bi register. (CPU 3.5)
 Set Bi register busy flag. (CPU 3.2)
 Set go increment flag. (CPU 3.2)

- CP02 Clear Bi register busy flag. (CPU 3.2)
 Clear go increment flag. (CPU 3.2)
 Result in Bi register.

64ijk SET Bi TO (Aj) + (Bk)

ISSUE CONDITIONS

Aj register is free.
Bk register is free.
Bi register is free.
B register input path is free in next clock period.
No SAS backup condition.

EXECUTION TIME

The execution timing for this instruction is the same as the 63 instruction. The sequence of events is the same except that the Aj register is used for the operand rather than the Xj register.

65ijk SET Bi TO (Aj) - (Bk)

ISSUE CONDITIONS

Aj register is free.
Bk register is free.
Bi register is free.
B register input path is free in next clock period.
No SAS backup condition.

EXECUTION TIME

The execution timing for this instruction is the same as the 63 instruction. The sequence of events is the same except that the Aj register is used for the operand rather than the Xj register.

66ijk SET Bi TO (Bj) + (Bk)

ISSUE CONDITIONS.

Bj register is free.
Bk register is free.
Bi register is free.
B register input path is free in next clock period.
No SAS backup condition.

EXECUTION TIME

The execution timing for this instruction is the same as the 63 instruction. The sequence of events is the same except that the Bj register is used for the operand rather than the Xj register.

67ijk SET Bi TO (Bj) - (Bk)

ISSUE CONDITIONS

Bj register is free.
Bk register is free.
Bi register is free.
B register input path is free in next clock period.
No SAS backup condition.

EXECUTION TIME

The execution timing for this instruction is the same as the 63 instruction. The sequence of events is the same except that the Bj register is used for the operand rather than the Xj register.

70ijK SET Xi TO (Aj) + K

ISSUE CONDITIONS

Aj register is free.

Xi register is free.

X register input path is free in next clock period.

No SAS backup condition.

EXECUTION TIME

No execution delays are possible after this instruction issues from the CIW register. The result is delivered to the Xi register one clock period after the instruction issues. The Xi register is reserved for the clock period from issue to delivery of data. The sequence of events for this instruction is:

- CP00 70 instruction in upper parcel of CIW register. (CPU 3.1)
Instruction issues.
Transmit next instruction to upper parcel of CIW register.
Transmit K to increment unit. (INCR 3.0)
Transmit (Aj) to increment unit. (INCR 3.0)
- CP01 Next instruction in upper parcel of CIW register.
Instruction may issue.
Transmit data from increment unit to Xi register. (CPU 3.5)
Set Xi register busy flag. (CPU 3.2)
Set go increment flag. (CPU 3.2)
- CP02 Clear Xi register busy flag. (CPU 3.2)
Clear go increment flag. (CPU 3.2)
Result in Xi register.

71ijK SET Xi TO (Bj) + K

ISSUE CONDITIONS

Bj register is free.

Xi register is free.

X register input path is free in next clock period.

No SAS backup condition.

EXECUTION TIME

The execution timing for this instruction is the same as the 70 instruction. The sequence of events is the same except that the Bj register is used for the operand rather than the Aj register.

72ijK SET Xi TO (Xj) + K

ISSUE CONDITIONS

Xj register is free.

Xi register is free.

X register input path is free in next clock period.

No SAS backup condition.

EXECUTION TIME

The execution timing for this instruction is the same as the 70 instruction. The sequence of events is the same except that the Xj register is used for the operand rather than the Aj register.

73ijk SET Xi TO (Xj) + (Bk)

ISSUE CONDITIONS

Xj register is free.
Bk register is free.
Xi register is free.
X register input path is free in next clock period.
No SAS backup condition.

EXECUTION TIME

No execution delays are possible after this instruction issues from the CIW register. The result is delivered to the Xi register 1 clock period after the instruction issues. The Xi register is reserved for the clock period from issue to delivery of data. The sequence of events for this instruction is:

CP00 74 instruction in upper parcel of CIW register. (CPU 3.1)
 Instruction issues.
 Transmit next instruction to upper parcel of CIW register.
 Transmit (Xj) to increment unit. (INCR 3.0)
 Transmit (Bk) to increment unit. (INCR 3.0)

CP01 Next instruction in upper parcel of CIW register.
 Instruction may issue.
 Transmit data from increment unit to Xi register. (CPU 3.4)
 Set Xi register busy flag. (CPU 3.2)
 Set go increment flag. (CPU 3.2)

CP02 Clear Xi register busy flag. (CPU 3.2)
 Clear go increment flag. (CPU 3.2)
 Result in Xi register.

74ijk SET Xi TO (Aj) + (Bk)

ISSUE CONDITIONS

Aj register is free.
Bk register is free.
Xi register is free.
X register input path is free in next clock period.
No SAS backup condition.

EXECUTION TIME

The execution timing for this instruction is the same as the 73 instruction. The sequence of events is the same except that the Aj register is used for the operand rather than the Xj register.

75ijk SET Xi TO (Aj) - (Bk)

ISSUE CONDITIONS

Aj register is free.
Bk register is free.
Xi register is free.
X register input path is free in next clock period.
No SAS backup condition.

EXECUTION TIME

No execution delays are possible after this instruction issues from the CIW register. The result is delivered to the Xi register 1 clock period after the instruction issues. The Xi register is reserved for the clock period from issue to delivery of data. The sequence of events for this instruction is:

CP00 75 instruction in upper parcel of CIW register. (CPU 3.1)
 Instruction issues.
 Transmit next instruction to upper parcel of CIW register.
 Transmit (Aj) to increment unit. (INCR 3.0)
 Transmit (Bk) to increment unit. (INCR 3.0)

CP01 Next instruction in upper parcel of CIW register.
 Instruction may issue.
 Transmit data from increment unit to Xi register. (CPU 3.4)
 Set Xi register busy flag. (CPU 3.2)
 Set go increment flag. (CPU 3.2)

CP02 Clear Xi register busy flag. (CPU 3.2)
 Clear go increment flag. (CPU 3.2)
 Result in Xi register.

76ijk SET Xi TO (Bj) + (Bk)

ISSUE CONDITIONS

Bj register is free.
 Bk register is free.
 Xi register is free.
 X register input path is free in next clock period.
 No SAS backup condition.

EXECUTION TIME

No execution delays are possible after this instruction issues from the CIW register.
 The result is delivered to the Xi register 1 clock period after the instruction issues.
 The Xi register is reserved for the clock period from issue to delivery of data.
 The sequence of events for this instruction is:

CP00 76 instruction in upper parcel of CIW register. (CPU 3.1)
 Instruction issues.
 Transmit next instruction to upper parcel of CIW register.
 Transmit (Bj) to increment unit. (INCR 3.0)
 Transmit (Bk) to increment unit. (INCR 3.0)

CP01 Next instruction in upper parcel of CIW register.
 Instruction may issue.
 Transmit data from increment unit to Xi register. (CPU 3.4)
 Set Xi register busy flag. (CPU 3.2)
 Set go increment flag. (CPU 3.2)

CP02 Clear Xi register busy flag. (CPU 3.2)
 Clear go increment flag. (CPU 3.2)
 Result in Xi register.

77ijk SET Xi TO (Bj) - (Bk)

ISSUE CONDITIONS

Bj register is free.
Bk register is free.
Xi register is free.
X register input path is free in next clock period.
No SAS backup condition.

EXECUTION TIME

No execution delays are possible after this instruction issues from the CIW register.
The result is delivered to the Xi register 1 clock period after the instruction issues.
The Xi register is reserved for the clock period from issue to delivery of data.
The sequence of events for this instruction is:

CP00 77 instruction in upper parcel of CIW register. (CPU 3.1)
 Instruction issues.
 Transmit next instruction to upper parcel of CIW register.
 Transmit (Bj) to increment unit. (INCR 3.0)
 Transmit (Bk) to increment unit. (INCR 3.0)

CP01 Next instruction in upper parcel of CIW register.
 Instruction may issue.
 Transmit data from increment unit to Xi register. (CPU 3.2)
 Set Xi register busy flag. (CPU 3.2)
 Set go increment flag. (CPU 3.2)

CP02 Clear Xi register busy flag. (CPU 3.2)
 Clear go increment flag. (CPU 3.2)
 Result in Xi register.

PART 2

BOOLEAN UNIT

BOOLEAN UNIT

The boolean unit executes the CPU instructions requiring bit-by-bit data manipulation. This includes both the logical operations and the transmissive operations. The instructions providing logical operations are:

- 11 Logical product of X_j and X_k to X_i
- 12 Logical sum of X_j and X_k to X_i
- 13 Logical difference of X_j and X_k to X_i
- 15 Logical product of X_j and $\overline{X_k}$ to X_i
- 16 Logical sum of X_j and $\overline{X_k}$ to X_i
- 17 Logical difference of X_j and $\overline{X_k}$ to X_i

The instructions providing transmissive operations are:

- 10 Transmit X_j to X_i
- 14 Transmit $\overline{X_k}$ to X_i
- 26 Unpack X_k to B_j and X_i
- 27 Pack X_k and B_j to X_i

INPUT REGISTERS

Three data input registers exist for the B_j , X_j , and X_k operands. These registers are cleared and entered with new data each clock period. The contents of the B_j , X_j , and X_k registers are transmitted to the boolean unit each clock period, without regard to the instruction in the CIW register. These operands are then available in the boolean unit in the following clock period.

CONTROL

Several bits of control information enter the boolean unit. Instruction designators f bit 0 and m bits 0 through 2 are sent to the boolean unit from the CIW register each clock period. These bits define the particular boolean instruction being executed. The instruction control translator decodes the bits to determine the type of logical operation and to select the data paths through the boolean unit for the various instructions. The control signals generated by the instruction

control translator are:

- Extend X: 26 · X_k bit 59
- Gate X_k upper: 12 + 13 + 14 + 16 + 17
- Comp X_k : 14 + 15 + 17 + 17
- Gate X_j : 10 + 12 + 13 + 16 + 17
- Gate LP: 11 + 12 + 15 + 16
- Gate B: 27
- Comp B: 27 · X_k bit 59
- Gate X_k lower: 12 + 13 + 14 + 16 + 17 + 26 + 27

Go boolean is received by the boolean unit during the clock period following issue of a boolean instruction. Data is transmitted to the destination registers only during a clock period in which go boolean is set.

OPERATION

If go boolean is set during a given clock period, a boolean instruction issued during the previous clock period and the data in the boolean unit input registers corresponds with the data described by the j and k designators in that instruction. Data in the input registers is merged in a static network for transmission to the destination B and/or X registers. The type of logical operation and the selection of data paths in this static network are determined by the control information.

LOGICAL PRODUCT

The boolean unit forms the logical product (AND function) of 60-bit words from the X_j and X_k registers (or its complement) and places the product in the X_i register. The operation is controlled by the gate LP signal. Bits in X_i are set to one when the corresponding bits in X_j and X_k (or its complement) are one as in the following examples.

- $X_j = 0101$ $X_j = 0101$
- $X_k = \underline{1100}$ $X_k = \underline{0011}$
- $X_i = 0100$ $X_i = 0001$

The boolean unit forms the complemented result, which is recomplemented so that the true result is transmitted to the X register.

LOGICAL SUM

The boolean unit forms the logical sum (OR function) of 60-bit words from the Xj and Xk registers (or its complement) and places the logical sum in the Xi register. This operation is performed in two steps and is controlled by the gate LP, gate Xk upper, gate Xk lower, and gate Xj signals.

Bits in Xi are set to one if the corresponding bits in Xj and Xk (or its complement) are a one as in the following examples.

Xj = 0101	Xj = 0101
Xk = <u>1100</u>	Xk = <u>0011</u>
Xi = 1101	Xi = 0111

The boolean unit forms the complemented result, which is recomplemented so that the true result is transmitted to the X register.

LOGICAL DIFFERENCE

The boolean unit forms the logical difference (exclusive OR function) of the quantity from the Xj and Xk registers (or its complement) and places the difference in the Xi register. This operation is controlled by the gate Xk upper, gate Xk lower, and gate Xj signals.

Bits in Xi are set to one if the corresponding bits in Xj and Xk (or its complement) are unlike as in the following examples.

Xj = 0101	Xj = 0101
Xk = <u>1100</u>	Xk = <u>0011</u>
Xi = 1001	Xi = 0110

The boolean unit forms the complemented result, which is recomplemented so that the true result is transmitted to the X register.

TRANSMIT Xj or \overline{Xk}

Except that it complements Xk, the boolean unit executes both the transmit Xj and transmit \overline{Xk} instruction in essentially the same manner. Xj or \overline{Xk} enter an equivalence circuit in which the other operand is zero. The result is either \overline{Xj} or Xk. This result is recomplemented upon being sent to the Xi register. Transmit Xj is controlled by the gate Xk upper, gate Xk lower, and comp Xk signals.

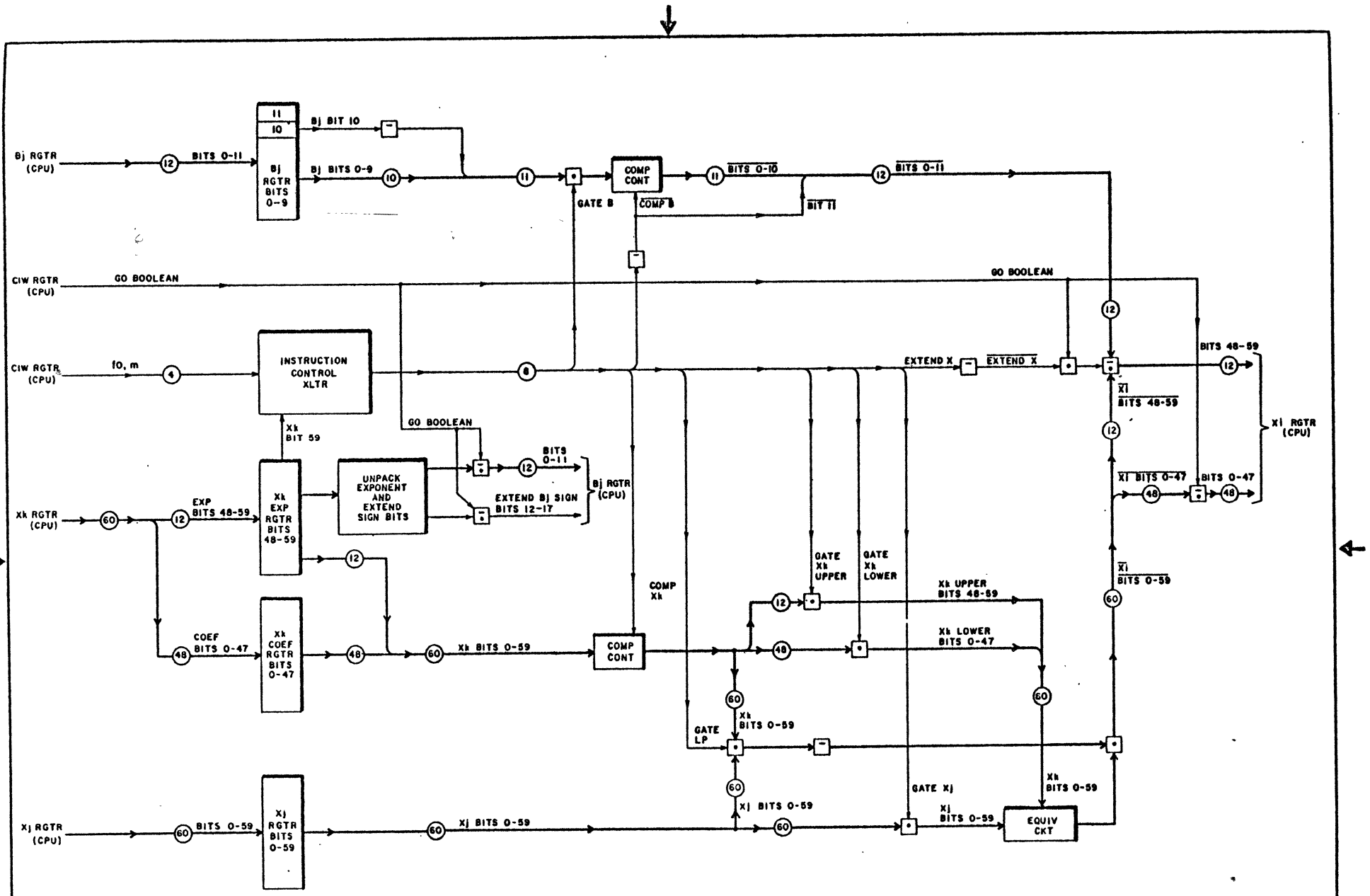
UNPACK

The boolean unit unpacks the floating-point quantity from the Xk register. It sends the 48-bit coefficient to the Xi register and the 11-bit exponent to the Bj register. The exponent bias is removed during the unpack operation so that the quantity in Bj is the true ones complement representation of the exponent. The sign bit of the exponent is extended to fill the 18-bit B register. The true exponent result is gated to the Bj register by the go boolean signal. B register access control allows entry from the boolean unit for only the unpack instructions.

Except that Xk is not complemented, the Xk coefficient bits (0 through 47) follow the same paths as for the transmit \overline{Xk} instruction. Gate Xk lower controls this part of the operation. The extend X signal causes the coefficient sign to be extended to bits 48 through 59 to fill the 60-bit result in the X register.

PACK

The boolean unit packs a floating-point number in Xi. The coefficient of the number is obtained from Xk and the exponent from Bj. The boolean unit adds bias to the exponent before merging it with the Xk operand. If Xk is positive, the packed exponent occupying positions 48 through 58 of Xi is obtained from bits 0 through 10 of Bj by complementing bit 10; if Xk is negative, bit 10 is not complemented, but bits 0 through 9 are complemented. The comp B, gate B, and gate Xk lower signals control the pack operation.



- BOOLEAN UNIT FUNCTION:**
- 10 TRANSMIT x_j TO x_l
 - 11 LOGICAL PRODUCT x_j AND x_h TO x_l
 - 12 LOGICAL SUM x_j AND x_h TO x_l
 - 13 LOGICAL DIFFERENCE x_j AND x_h TO x_l
 - 14 TRANSMIT \bar{x}_l TO x_l
 - 15 LOGICAL PRODUCT x_j AND \bar{x}_h TO x_l
 - 16 LOGICAL SUM x_j AND \bar{x}_h TO x_l
 - 17 LOGICAL DIFFERENCE x_j AND \bar{x}_h TO x_l
 - 26 UNPACK x_h TO b_j TO x_l
 - 27 PACK x_h AND b_j TO x_l

BOOLEAN UNIT

The boolean unit executes the CPU instructions requiring bit-by-bit data manipulation. This includes the logical operations for instructions 11, 12, 13, 15, 16, and 17 plus the transmissive operations for instructions 10, 14, 26, and 27.

INPUT REGISTERS

The three input registers in the boolean unit receive data from the Bj, Xj, and Xk registers each clock period, without regard to the instruction in the CIW register. Data is transmitted to the input registers concurrent with the instruction issue. During the following clock period, data moves from the input registers through the static selection network and back to the operating registers. Thus, each instruction is executed in 2 clock periods. The boolean unit is free to begin executing a new instruction every clock period. If a boolean-type instruction does not issue in a given clock period, the data in the input registers is not used. New data enters the input registers in the following clock period.

CONTROL

The boolean unit also receives bits of the f and m designators from the CIW register each clock period. Instruction designators f bit 0 and m bits 0 through 2 are held in registers and are then translated into control signals that determine the type of logical operation and select data paths required by the instruction.

The boolean unit receives the go boolean signal in the clock period following issue of a boolean instruction. The go boolean signal enables the output of the boolean unit to the destination registers.

The data path to the destination X register for bits 48 through 59 on the 4KN7 module is shared by the various boolean instructions. Control signals from the 4KM7 module prevent conflicts by ensuring that only one data path is active at any one time. The active data path, containing the complemented result, merges with all ones on the other two paths. The result is recomplemented upon being sent to the destination X register.

LOGICAL PRODUCT

The logical product for instructions 11 and 15 is formed on the 4KL7 and 4KN7 modules. Xk is complemented if this operation is being performed for a 15 instruction. Xj is then ANDed with the corresponding Xk bits and the gate LP signal.

The complement of the logical product goes to another circuit where it is ANDed with the results of an equivalence circuit. For instructions 11 and 15, the results of the equivalence operation are all ones because the Xj and Xk inputs to the equivalence circuit are not enabled. Thus, this second AND circuit for the logical product instructions acts like a merge. From the second AND circuit, the complemented logical product is recomplemented and sent to the destination X register.

LOGICAL SUM

The logical sum instructions (12 and 16) use the same circuitry as the logical product instructions except that the inputs to the equivalence circuit are enabled. Instead of ones, meaningful data is ANDed with the complement of the logical product.

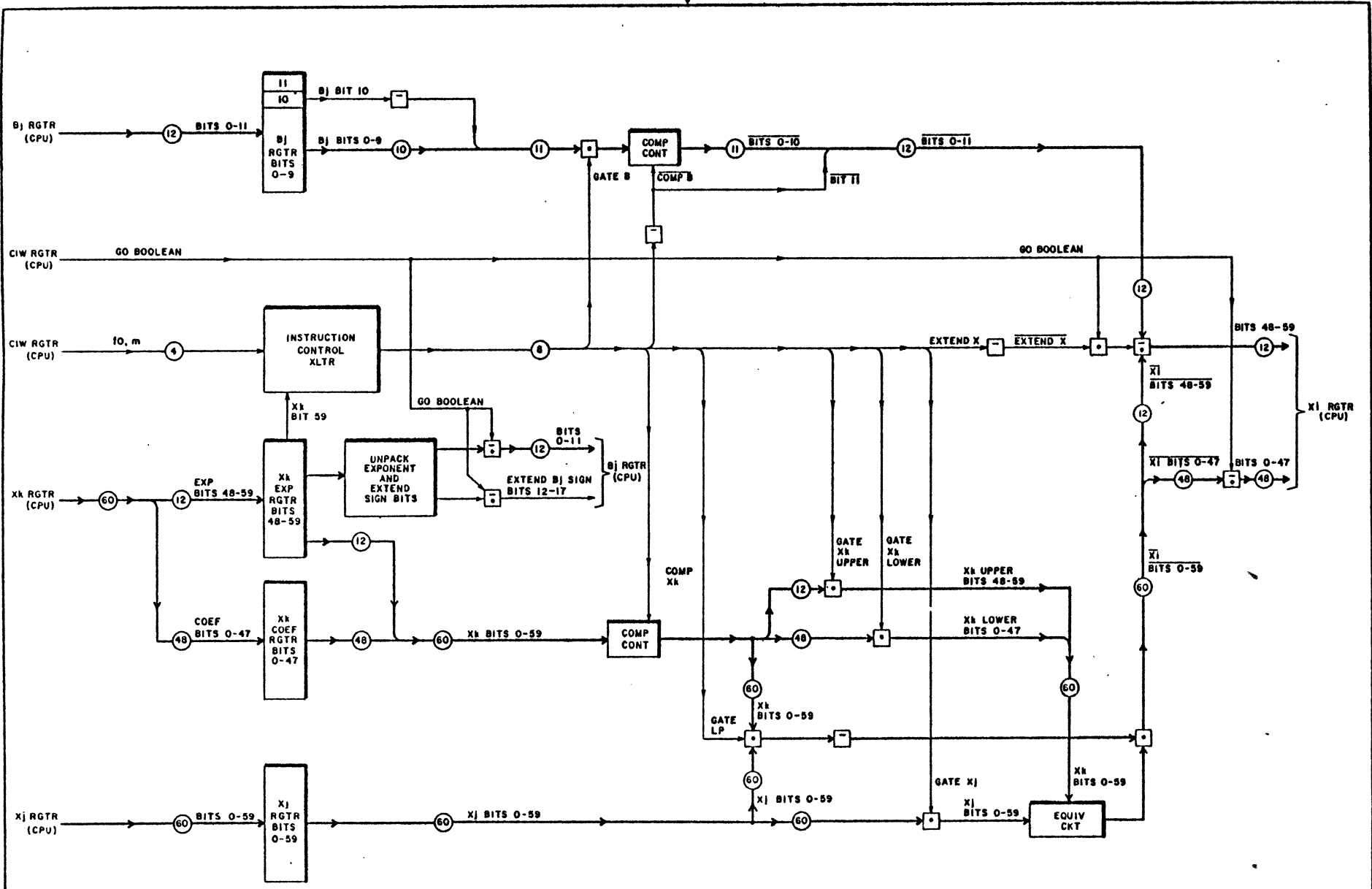
For example, if Xj equals 0101 and Xk equals 1100, the following logical operations take place.

1. The first AND circuit forms the logical product of Xj and Xk.

$$\begin{aligned} Xj &= 0101 \\ Xk &= \underline{1100} \\ LP &= 0100 \end{aligned}$$

2. Both the Xj and Xk inputs to the equivalence circuit are enabled by gate Xj, gate Xk upper, and gate Xk lower so the equivalence circuit produces:

$$\begin{aligned} Xj &= 0101 \\ Xk &= \underline{1100} \\ EQ &= 0110 \end{aligned}$$



BOOLEAN UNIT FUNCTION:

- 10 TRANSMIT x_j TO x_i
- 11 LOGICAL PRODUCT x_j AND x_k TO x_i
- 12 LOGICAL SUM x_j AND x_k TO x_i
- 13 LOGICAL DIFFERENCE x_j AND x_k TO x_i
- 14 TRANSMIT \bar{x}_i TO x_i
- 15 LOGICAL PRODUCT x_j AND \bar{x}_i TO x_i
- 16 LOGICAL SUM x_j AND \bar{x}_i TO x_i
- 17 LOGICAL DIFFERENCE x_j AND \bar{x}_i TO x_i
- 26 UNPACK x_k TO B_j TO x_i
- 27 PACK x_k AND B_j TO x_i

3. The complement of the logical product (\overline{LP}) is ANDED with EQ to produce:

LP = 1011

EQ = 0110

Xi = 0010

4. This result is recomplemented and sent to the destination X register as 1001.

LOGICAL DIFFERENCE

The logical difference instructions (13 and 17) use the equivalence circuit to form the complement of the result. The complemented result is ANDED with all ones from the logical product circuit, which is not enabled for these two instructions. The result of this merge operation is recomplemented and sent to the destination X register.

TRANSMIT X

The boolean unit executes both transmit instructions (10 and 14) in essentially the same way. Either Xj or Xk is enabled into the equivalence circuit with the other operand a zero. The result is the complement of whichever operand was input. The result is then recomplemented upon being sent to the destination X register.

UNPACK

The unpack operation for instruction 26 requires three separate data paths: one for the exponent to the destination B register, another for coefficient bits 0 through 47 to the destination X register, and a third for extending Xk sign to bits 48 through 59 of the destination X register.

Unpacking of the exponent takes place on the 4KM7 module. To remove bias and provide the proper sign, bits 48 through 57 of Xk enter a complement control circuit. This circuit complements bits 48 through 57 if the coefficient is positive

(bit 59 is not set). The output of the circuit is recomplemented to become bits 0 through 9 upon being gated to the destination B register. The complement of the sign of the exponent (Bj sign) is determined by the logical difference of Xk bits 58 and 59. Bj sign is complemented and sent to the B register as bits 10 and 11. This sign extension is repeated on the 4KN7 module for extending Bj sign to bits 12 through 17 of the B register. Bj and extend Bj sign bits are gated by the go boolean signal. B register access control prevents entry to the destination B register from the boolean unit for all instructions except 26.

Bits 0 through 47 of Xk are gated into the equivalence circuit on the three 4KL7 modules by the gate Xk lower signal. Since the Xj input to the equivalence circuit is not enabled, the result of the equivalence is the complement of Xk. This is recomplemented to become bits 0 through 47 in the destination X register.

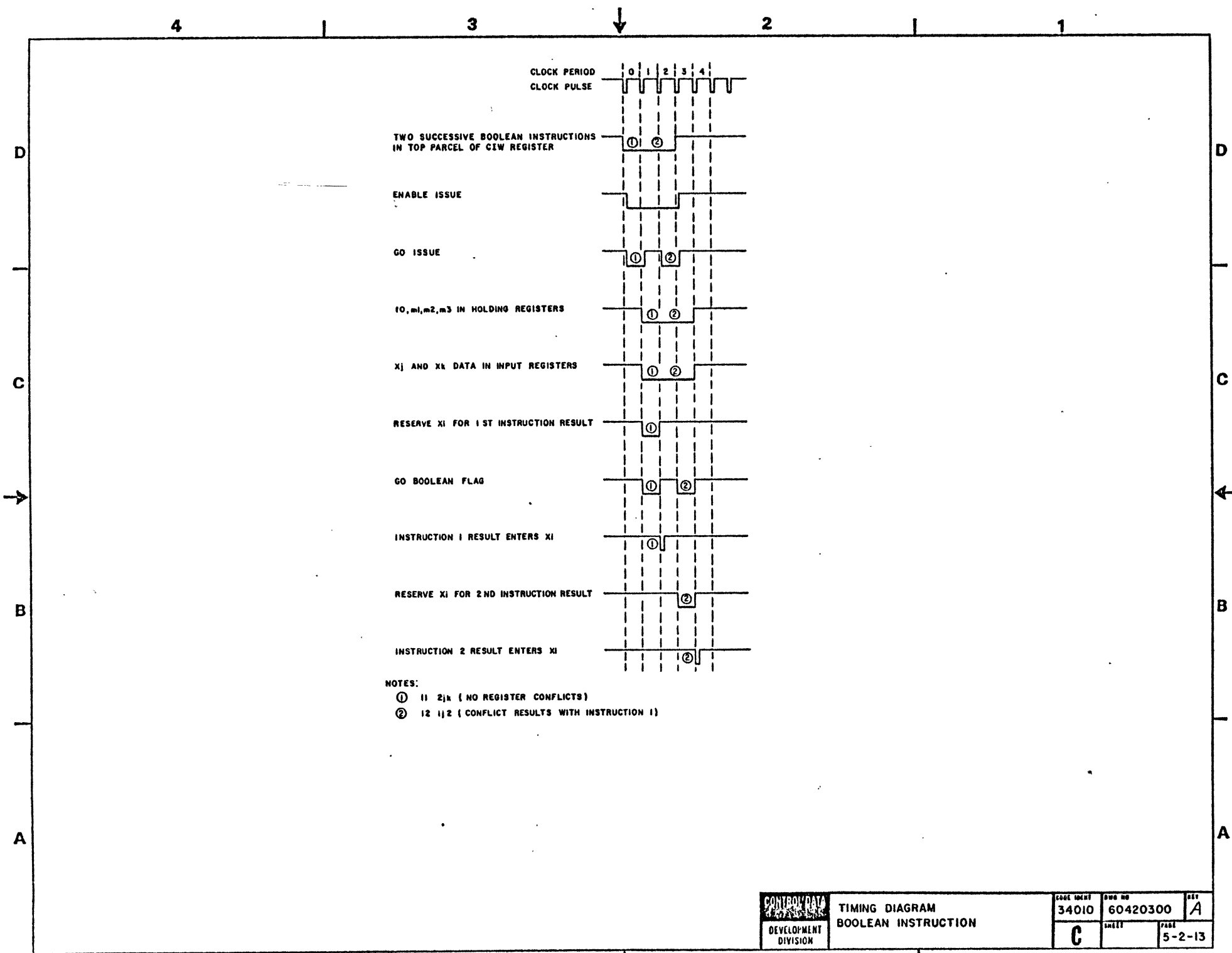
Note that gate Xk upper is not set for this instruction, thereby preventing transmission of Xk bits 48 through 59 to the destination X register. Instead, the coefficient sign is extended to bits 48 through 59 by the extend X signal on the 4KN7 module. If the coefficient is positive, extend X is set and is ANDED with the go boolean signal. When ANDED with all ones from the other two paths and recomplemented, the signal causes all zeros to enter bits 48 through 59 of the destination X register. When the coefficient is negative, extend X is clear, and the result is all ones to bits 48 through 59 of the destination X register.

PACK

Just as unpack requires separate data paths to the B and X registers, packing for instruction 27 requires separate data paths from the B and X registers.

Packing of the exponent takes place on the 4KN7 module. Bj bit 11 is discarded because it is merely an extension of the exponent sign bit (bit 10). Bit 10 is complemented to add bias, and bits 0 through 10 are then gated into a complement control circuit by the gate B signal. This circuit complements bits 0 through 10 if the coefficient is positive (Xk bit 59 is not set). The output of the circuit is Xi bits 48 through 58. These bits and the complement of the comp B signal (Xi bit 59) are ANDED with ones from the other paths and are recomplemented upon being sent to the destination X register.

Packing of the coefficient is a straightforward transmission of Xk bits 0 through 47 to the X register gated by gate Xk lower. Data paths for Xk bits 48 through 59 are blocked because gate Xk upper is not set for this instruction.



NOTES:

- ① 11 2jk (NO REGISTER CONFLICTS)
- ② 12 1j2 (CONFLICT RESULTS WITH INSTRUCTION 1)

PART 3

SHIFT UNIT

SHIFT UNIT

The shift unit executes CPU instructions 20, 21, 22, 23, and 43. These instructions shift the entire 60-bit field of data within the operand word. Input operands are of two types. A 60-bit word is read from either the Xi or the Xk register, depending upon the type of instruction. A second operand is read from either the CIW or the Bj register. This operand determines the shift count for the 60-bit word. Shifted operands are sent from the shift unit to the Xi register.

INSTRUCTIONS EXECUTED

LEFT SHIFT Xi BY jk (20ijk INSTRUCTION)

This instruction causes the shift unit to read one operand from the Xi register, shift the 60-bit word left circularly by jk bit positions, and write the resulting 60-bit word back into the same Xi register.

The shift unit does not actually shift data in a left circular mode. Instead, it simulates the shift. The shift count is complemented, a three-bit left circular shift correction is performed, and the shift is made in a right circular mode.

RIGHT SHIFT Xi BY jk (21ijk INSTRUCTION)

This instruction causes the shift unit to read one operand from the Xi register, shift the 60-bit word to the right with sign extension by jk positions, and write the resulting 60-bit word back into the same Xi register.

LEFT SHIFT Xk BY Bj TO Xi (22ijk INSTRUCTION)

This instruction normally causes the shift unit to read one operand from the Xk register, shift the 60-bit word left circularly by an amount determined by bits 0 through 5 of the Bj register, and write the resulting 60-bit word into the Xi register. However, if the 18-bit operand in the Bj register is negative (bit 17 is a one), the shift is to the right with sign extension, and the shift count is deter-

mined by the complement of Bj bits 0 through 5.

The shift unit always complements Bj bits 0 through 5 during the execution of this instruction. The left shift is simulated by making a three-bit left circular shift correction and shifting in a right circular mode.

RIGHT SHIFT Xk BY Bj TO Xi (23ijk INSTRUCTION)

This instruction normally causes the shift unit to read one operand from the Xk register, shift the 60-bit word to the right with sign extension by an amount determined by Bj bits 0 through 5, and write the resulting 60-bit word into the Xi register. However, if the 18-bit operand in the Bj register is negative (bit 17 is a one), the shift is left circular, and the shift count is determined by the complement of Bj bits 0 through 5.

The shift unit does not actually shift data in a left circular mode when Bj is negative; it simulates the shift. A three-bit left circular shift correction is performed and the shift is made in a right circular mode. The shift count is complemented because the Bj register contains a negative number.

MASK UPPER jk PLACES OF Xi (43ijk INSTRUCTION)

This instruction causes the shift unit to generate a 60-bit word containing ones in the upper jk bit positions and zeros in the remaining bit positions. The resulting 60-bit word is written into the Xi register.

INSTRUCTION TRANSLATION

The instruction translator determines what instruction has been issued by translating three bits from the instruction designators in the CIW register. These are f bit 1 and m bits 0 and 1.

OPERAND SELECTION

The operand holding register holds the data which is to be shifted. This 60-bit operand is selected in the CPU from either the Xi or Xk register. Before entering the holding register, the selected operand passes through a second selection network. For shift instructions which require a right shift of data, the operand enters the register unchanged. For shift instructions which require a left shift of data, the operand is left circularly shifted by three bit positions before entering the register. For the mask instruction, the operand is discarded and the register contains all zeros. Before entering the first shift rank, the operand is complemented in a complement control network if a 43 instruction (mask) has been issued or if the selected operand is negative.

SHIFT COUNT

SHIFT COUNT SELECTION

The amount of shift to be performed on the selected 60-bit operand is determined by translating a six-bit operand from one of two sources. For 20, 21, and 43 instructions, the j and k designators from the CIW register are used. These designators are treated as a single six-bit quantity. For 22 and 23 instructions, the lower six bits from the Bj register are used. If a 20 or 22 instruction has been issued, the selected shift count bits are complemented in a complement control circuit.

SHIFT COUNT TRANSLATORS AND FLAGS

The selected shift count bits are translated in three groups. Bits 0 and 1 determine which one of four rank 1 shift count flags will be set. These flags allow the 60-bit operand to be shifted zero, one, two, or three bit positions in the first shift rank. Bits 2 and 3 determine which one of four rank 2 shift count flags will be set. These flags allow the 60-bit operand to be shifted 0, 4, 8, or 12 bit positions in the second shift rank. Bits 4 and 5 determine which one of four rank 3 shift count flags will be set. These flags allow the 60-bit operand to be shifted 0, 16, 32, or 48 bit positions in the third shift rank.

SHIFT COUNT GREATER THAN 63 TEST

Prior to the execution of a 22 or 23 instruction which requires a right shift, the shift count greater than 63 test network determines if the shift will be greater than 63 bit positions. If so, the enable shift signal is not generated, causing none of the rank 3 shift count flags to be set. With no flags set, the third shift rank outputs all zeros, and the shift unit outputs all ones to the X registers. This causes all zeros to enter the Xi registers.

LEFT CIRCULAR SHIFT SIMULATION

For instructions requiring a left circular shift (20, 22 with Bj positive and 25 with Bj negative), the shift unit actually performs a right circular shift. The left circular shift signal enables the three right-shift ranks to shift circularly. The amount of right shift is determined by the complement of the shift count. For the left-shift instructions (20 and 22), the shift count is complemented as it enters the shift unit. For the right-shift instruction which results in a left shift (23 with Bj negative), the shift count is not complemented because it is already expressed as a negative number. When a left circular shift is simulated by doing a right circular shift using the complemented shift count, a three-bit position error is introduced. The error is caused because the maximum shift count specifies three bit positions more than the 60-bit word. This error is corrected by the three-bit left circular shift network which shifts the operand before it enters the operand holding register.

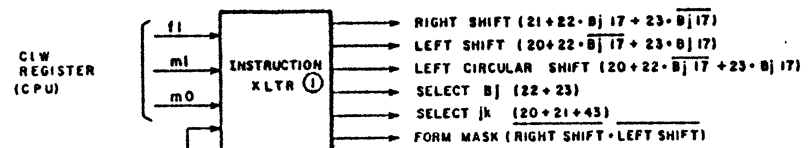
SHIFT RANKS

The three shift ranks shift the operand by an amount determined by the corresponding rank 1, 2, and 3 shift count flags. Each rank shifts the operand right circularly if the left circular shift signal is present. If this signal is absent, the shift is to the right with sign extension. Since the operand is always positive in the shift unit, the sign extension bits are always zeros.

A 43 instruction (mask) forces all ones into shift rank 1. These ones are right-shifted with zeros filling the vacated bit positions. The shift count is determined in the same manner as previously.

SHIFT UNIT OUTPUT

The third rank of the shift unit outputs the shifted operand (complemented) to the X registers whenever a go shift signal is present. If this signal is not present, all ones are sent to the X registers. If the original operand was positive, the shifted operand is recomplemented before entering the XI register. Negative operands are complemented twice in the shift unit. Therefore, they are not recomplemented.

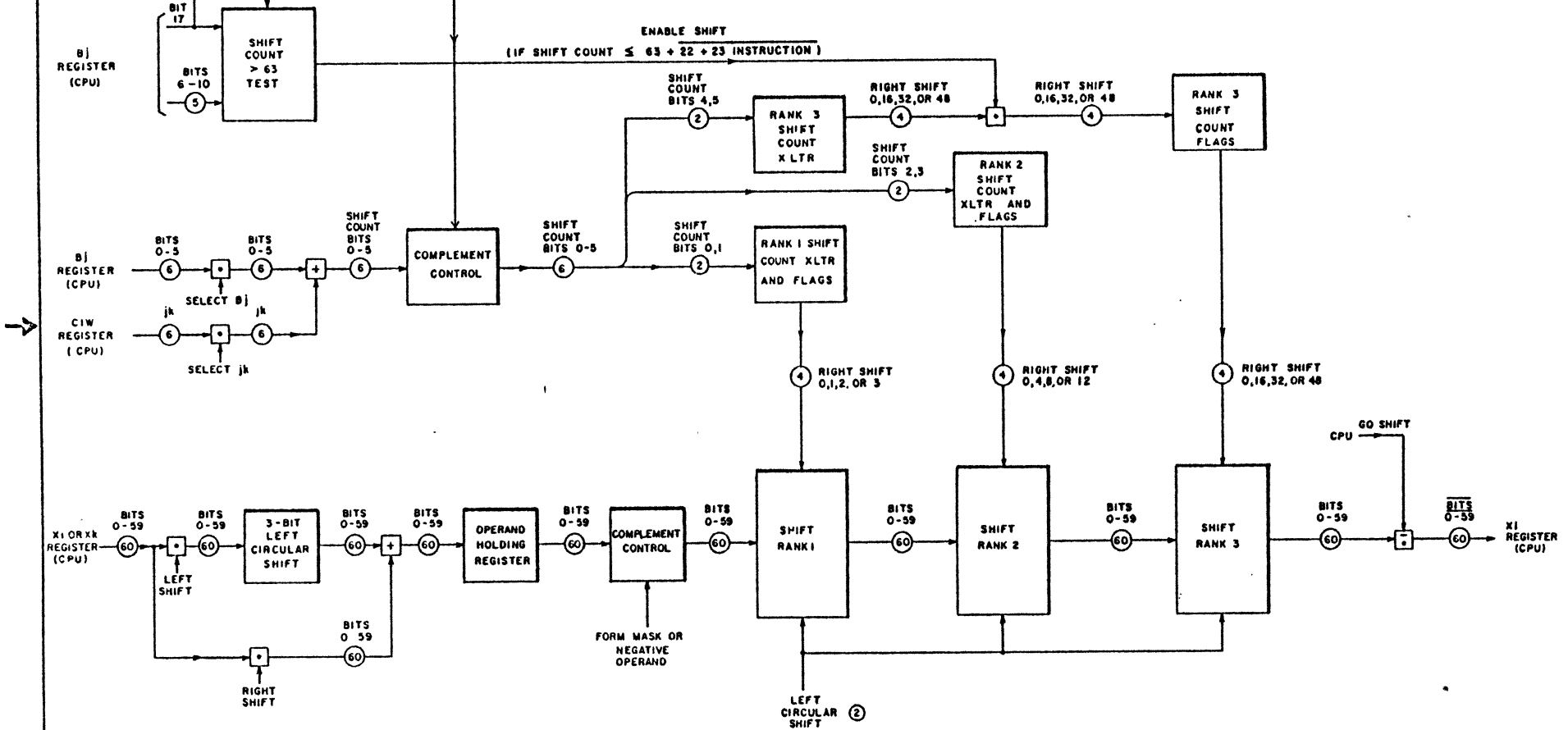


NOTES:

① THE SHIFT UNIT PERFORMS THE FOLLOWING FUNCTIONS:

INSTRUCTION	FUNCTION
20	SHIFTS 60 BITS IN X _i REGISTER j _k PLACES TO LEFT.
21	SHIFTS 60 BITS IN X _i REGISTER j _k PLACES TO RIGHT.
22	SHIFTS 60 BITS IN X _k REGISTER B _j 0-5 PLACES TO LEFT IF B _j IS POSITIVE OR B _j 0-5 PLACES TO RIGHT IF B _j IS NEGATIVE. RESULT IS PLACED IN X _i REGISTER.
23	SHIFTS 60 BITS IN X _k REGISTER B _j 0-5 PLACES TO RIGHT IF B _j IS POSITIVE OR B _j 0-5 PLACES TO LEFT IF B _j IS NEGATIVE. RESULT IS PLACED IN X _i REGISTER.
43	PUTS ONES IN UPPER j _k PLACES OF X _i REGISTER.

② LEFT CIRCULAR SHIFT ENABLES BITS TO BE RIGHT SHIFTED END-AROUND.



SHIFT UNIT

The shift unit executes CPU instructions 20, 21, 22, 23, and 43. These instructions shift the entire 60-bit field of data within the operand word. Input operands are of two types. A 60-bit word is read from either the Xi or the Xk register, depending upon the type of instruction. A second operand is read from either the CIW or the Bj register. This operand determines the shift count for the 60-bit word. Shifted operands are sent from the shift unit to the Xi register.

INSTRUCTIONS EXECUTED

LEFT SHIFT Xi BY jk (20ijk INSTRUCTION)

This instruction causes the shift unit to read one operand from the Xi register, shift the 60-bit word left circularly by jk bit positions, and then write the resulting 60-bit word back into the same Xi register.

The shift unit does not actually shift data in a left circular mode. Instead, it simulates the shift. The shift count is complemented, a three-bit left circular shift correction is performed, and the shift is made in a right circular mode.

RIGHT SHIFT Xi BY jk (21ijk INSTRUCTION)

This instruction causes the shift unit to read one operand from the Xi register, shift the 60-bit word to the right with sign extension by jk bit positions, and write the resulting 60-bit word back into the same Xi register.

LEFT SHIFT Xk BY Bj TO Xi (22ijk INSTRUCTION)

This instruction normally causes the shift unit to read one operand from the Xk register, shift the 60-bit word left circularly by an amount determined by Bj bits 0 through 5, and write the resulting 60-bit word into the Xi register. How-

ever, if the 18-bit operand in the Bj register is negative, the shift is to the right with sign extension, and the shift count is determined by the complement of Bj bits 0 through 5.

The shift unit always complements Bj bits 0 through 5 during the execution of this instruction. The left shift is simulated by making a three-bit left circular shift correction and shifting in a right circular mode.

RIGHT SHIFT Xk BY Bj TO Xi (23ijk INSTRUCTION)

This instruction normally causes the shift unit to read one operand from the Xk register, shift the 60-bit word to the right with sign extension by an amount determined by Bj bits 0 through 5, and write the resulting 60-bit word into the Xi register. However, if the 18-bit operand in the Bj register is negative, the shift is left circular and the shift count is determined by complemented bits 0 through 5.

The shift unit does not actually shift data in a left circular mode when Bj is negative; it simulates the shift. A three-bit left circular correction is performed and the shift is made in a right circular mode. The shift count is complemented because the Bj register contains a negative number.

MASK UPPER jk PLACES OF Xi (43ijk INSTRUCTION)

This instruction causes the shift unit to generate a 60-bit word containing ones in the upper jk bit positions and zeros in the remaining bit positions. The resulting 60-bit word is written into the Xi register.

INSTRUCTION TRANSLATION

The 3KO7 module determines what instruction has been issued by translating three bits from the instruction designators in the CIW register. These are f bit 1 and m bits 0 and 1.

OPERAND SELECTION

The operand holding register (4KP7 modules) holds the data which is to be shifted. This 60-bit operand is selected from either the Xi or Xk register. The selection is made in the CPU (4RF7 and 4RG7 modules) and is based on the value of m in the CIW register. The Xi data path is selected if the m designator has an octal value of 0, 1, 4, or 5. The Xk data path is selected when this value is 2, 3, 6, or 7.

Before entering the operand holding register, the selected operand passes through a second selection network (4KP7 modules). For 2X series instructions which require a right-shift of data, the operand enters the register unchanged. For 2X series instructions which require a left-shift of data, the operand is left circularly shifted by three bit positions before entering the register. For 43 instructions, the operand is discarded. In this case, the register contains all zeros.

Before entering the first shift rank, the operand is complemented in a complement control network (4KP7 modules) if the complement control flag is set. This flag is set when a 43 instruction has been issued or if the selected operand is negative (operand must be positive). A 43 instruction is indicated when the f designator from the CIW register has a zero in the middle bit (X0X). A negative operand is indicated when bit 59 (sign) of the selected operand is a one.

SHIFT COUNT

SHIFT COUNT SELECTION

The shift count to be performed on the selected 60-bit operand is determined by translating a six-bit operand from one of two sources. The selection of one of these sources is made in the 3KO7 module. For 20, 21, and 43 instructions, the j and k designators from the CIW register are used. These designators are treated as a single six-bit quantity. For 22 and 23 instructions, the lower six bits from the Bj register are used. If a 20 or 22 instruction has been issued, the selected shift count bits are complemented.

SHIFT COUNT TRANSLATORS AND FLAGS

The selected shift count bits are translated in three groups. Bits 0 and 1 are sent to a translator in the 4KP7 modules. This translator determines which one of four flags will be set. These flags allow the 60-bit operand to be shifted zero, one, two, or three bit positions in the first shift rank (4KP7 modules). Shift count bits 2 and 3 are translated in the 3KO7 module. This translator determines which one of four flags in the 4KP7 modules will be set. These flags allow the 60-bit operand to be shifted 0, 4, 8, or 12 bit positions in the second shift rank (4KQ7 and 4KW7 modules). Shift count bits 4 and 5 are also translated in the 3KO7 module. This translator determines which one of four flags in the 4KP7 modules will be set. These flags allow the 60-bit operand to be shifted 0, 16, 32, or 48 bit positions in the third shift rank (4KQ7 and 4KW7 modules).

SHIFT COUNT GREATER THAN 63 TEST

Prior to the execution of a 22 or 23 instruction, a network in the 3KO7 module determines if the right-shift is greater than 63 bit positions. If so, the enable shift signal is not generated. This causes none of the third shift rank flags to set. With no flags set, the third shift rank outputs all zeros and the shift unit outputs all ones to the X registers. This causes all zeros to enter the Xi register.

LEFT CIRCULAR SHIFT SIGNAL AND FLAG

For instructions requiring a left circular shift, the shift unit actually shifts the operand in a right circular mode. The left circular shift signal generated by the 3KO7 module enables the shift ranks to simulate this shift. This signal enables a three-bit left circular shift correction in the 4KP7 modules. It also sets the left circular shift flag (lower 4KP7 module). When this flag is set, the shift ranks perform a right circular shift.

SHIFT UNIT OUTPUT

The third rank of the shift unit (4KQ7 and 4KW7 modules) outputs the shifted operand (complemented) to the X registers whenever a go shift signal from the 4LE7 module is present. If this signal is not present, all ones are sent to the X registers. If the original operand was positive, the shifted operand is re-complemented before entering the Xi register under control of the X register sign control. Negative operands are complemented twice in the shift unit. Therefore, they are not re-complemented.

4

3



2

1

D

D

C

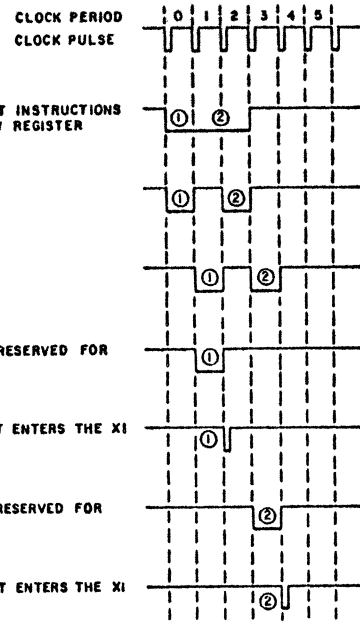
C

B

B

A

A



NOTES:

- ① 431_h (ASSUMING NO ISSUE CONFLICTS EXIST)
- ② 23_h (OPERAND REGISTER CONFLICTS WITH INSTRUCTION 1 DESTINATION REGISTER)

CONTROL DATA
CORPORATION
DEVELOPMENT
DIVISION

TIMING DIAGRAM
SHIFT INSTRUCTION

CODE IDENT 34010	DOC NO 60420300	REV A
C	SHEET	PAGE 5-3-13

PART 4

NORMALIZE UNIT

NORMALIZE UNIT

The normalize unit executes CPU instructions 24 and 25. These two instructions are identical except instruction 25 adds a round bit to the coefficient.

The normalize unit operates on operands in positive (true) form so it complements negative operands before operating on them. It then left-shifts the coefficient by an amount that causes a one bit to appear in the most significant position. The normalize unit adjusts the exponent by subtracting the shift count.

OPERATION

The normalize instructions require 3 clock periods for execution. Data moves from the operating registers to the normalize unit in the same clock period in which the instruction issues from the CIW register. Input registers hold the information necessary to complete the instruction for use during the second clock period. During the second clock period, the normalize unit complements the operand if the sign was negative, and a static network determines the number of shifts necessary to normalize the coefficient. The shift count determination network sends a six-bit shift count to the first stage of the exponent adder for exponent correction, to a register that holds the shift count for transmission to the B register, and to the shift count translator. The shift count translator translates the six-bit binary shift count into control signals: shift 0, 2, 4, 8, 16, 32, 48, and no shift. The decoded shift count register holds this decoded shift count for use during the third clock period.

The shift count determination network also sends bit zero of the binary shift count to shift rank 1 where it left-shifts the coefficient one position if bit 0 is a one. During the third clock period, shift ranks 2, 3, and 4 complete the shifting of the coefficient, and the second stage of the exponent adder completes the subtraction of the shift count from the exponent. Go normalize gates the shift count to the B register, and if there are no special cases, gates the corrected exponent and shifted coefficient to the destination X registers.

SPECIAL CASES

After the normalize unit corrects the exponent for sign, it tests for an overflow or indefinite quantity. If the exponent is overflow or indefinite, the unit blocks the shift count, the operand passes through the unit unaltered. The quantity delivered to the B register is zero and a bit is set in the exit condition register (CPU).

If the coefficient portion of the operand is all zeros after the correction for sign, the shift count is 48. If this situation occurs in execution of a 25 instruction, the round bit results in a normalized coefficient. If this situation occurs in execution of a 24 instruction, the shift count of 48 is a special case and the result entered in the destination X register is all zeros. The shift count delivered to the B register is 48 in either of these cases.

The subtraction of the six-bit shift count from the exponent may result in an underflow of the floating-point exponent range. This situation causes all zeros to enter the destination X register. The shift count delivered to the B register is not affected by this situation.

NORMALIZE UNIT

The normalize unit executes CPU instructions 24 and 25. These two instructions are identical except instruction 25 adds a round bit to the coefficient.

The normalize unit operates on operands in positive (true) form so it complements negative operands before operating on them. It then left-shifts the coefficient by an amount that causes a one bit to appear in the most significant position (bit 47). The normalize unit adjusts the exponent by subtracting the shift count.

EXPONENT AND CONTROL

4EE7 MODULE

Xk bits 48 through 59 and instruction designators m bit 0 enter registers in this module when the instruction issues from the CIW register. A static network complements bits 48 through 58 if bit 59 is a one to place the exponent in biased positive format. A ones check is then performed on exponent bit 0 through 9. Bits 0 through 9 are all ones if an exponent overflow condition (3777) or an exponent indefinite condition (1777) exists. If either of these conditions exists, the ones check detects it and blocks enable shift. Blocking enable shift causes the operand to pass unchanged through the normalize unit and a zero shift count to be sent to the B register.

This module also sends bit 59 (X sign) to the CPU X register sign control. This controls the complement of the result at the input to the destination X register.

If the instruction is a round normalize (25), m bit 0 is a one and sets the round flag. The round flag sends a round signal to the 3EF7 and 4EC7 modules. Round causes a bit to be added to the coefficient in a position immediately below the bit zero position of the coefficient before it is shifted. The shift ranks then the round bit along with the coefficient.

3EF7 MODULE

Exponent bits 0 through 10 enter this module from the 4EF7 module during the second clock period, where the first stage of subtraction of shift count bits 0 through 5 from the exponent takes place. Enable shift gates shift count bits 0 through 5 into the first stage of the adder. Registers hold the partial difference for use during the third clock period. The 3EF7 module also tests for a zero coefficient and a complete underflow.

In the case of a 24 instruction (no round bit) with a coefficient equal to zero, a normalized result is impossible. The zero coefficient results in a shift count of 48, and combined with round and enable shift, blocks gate output. Blocking gate output causes all ones to be sent to the X registers. A normalize underflow signal is sent to the X register input control (5CB7 module) which complements the word of ones causing all zeros to enter the destination X register.

If subtraction of the shift count causes the exponent to exceed its negative range, a complete underflow results. This is detected by testing the bit enables and bit borrows. A complete underflow blocks gate output which causes all ones to be sent to the X registers. A normalize underflow signal is sent to the X register input control (5CB7 module) which complements the word of ones causing all zeros to enter the destination X register.

Go normalize enters the module during the second clock period and the go normalize flag-2 holds it for use during the third clock period. Go normalize enables gate output under normal conditions and also causes the extension of zero bits in B register bits 6 through 17.

This module tests for indefinite or infinite operands. For indefinite operands (exponent equals 1777), the complement of exponent bit 10 and the complement of enable shift are ANDed with the go normalize signal. For infinite operands (exponent equals 3777), exponent bit 10 and the complement of enable shift are ANDed with the go normalize signal. The indefinite and infinite signals are sent to the CPU to set the corresponding bit in the exit condition register (CPU 2.10).

4EH7 MODULE

This module performs the second stage of subtraction of the shift count from the exponent. Gate output gates the output of the adder to the X registers. The result is complemented as it is sent to the X registers. If the result is positive, it is recomplemented by the X register input control (5CB7 module) as it enters the destination X register.

COEFFICIENT

4EA7 MODULE

Xk bits 0 through 47 enter this module when the instruction issues from the CIW register. The module contains the coefficient input register and complements the coefficient if bit 59 is a one. This changes the coefficient to positive format in the same way the 4EE7 module corrected the exponent for sign. The 48 coefficient bits go to the 4EC7 and 4EB7 modules.

4EB7 MODULE

This module contains the static shift count determination network. Its output is a six-bit binary shift count equal to the number of zero bits from the most significant one bit on the unshifted coefficient up to and including bit 47. All six bits of this shift count go to the 3EF7 and 4EG7 modules. Shift count bit 0 goes to the 4EC7 modules.

4EG7 MODULE

Enable shift gates the shift count into these modules during the second clock period. The shift count translator translates the six-bit shift count into shift control signals of shift 0, 2, 4, 8, 12, 16, 32, 48, and no shift. Registers hold them for use in the 4ED7 modules during the third clock period. A register also holds shift count bits 0 through 5 for use during the third clock period. Extend Bj gates the shift count to B register bits 0 through 5.

4EC7 MODULE

The coefficient bits 0 through 47 enter these modules during the second clock period. A left-shift of one takes place in shift rank 1 if shift count bit 0 and enable shift are both ones. A register holds the output of shift rank 1 for use in the 4ED7 modules during the third clock period.

4ED7 MODULE

Coefficient bits 0 through 47 enter these modules during the third clock period. Shift ranks 2, 3, and 4 complete the left shifts needed to normalize the coefficient. Gate output gates coefficient bits 0 through 47 to the destination X register. The result is complemented as it is sent to the X registers. If the result is positive, it is recomplemented by the X register input control as it enters the destination X register.

4

3

2

1

D

C

B

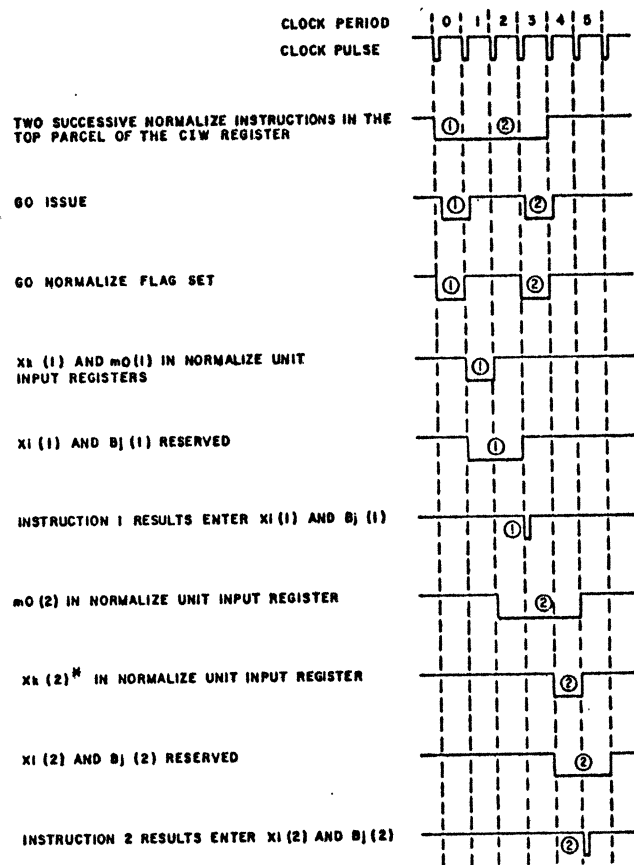
A

D

C

B

A



NOTES:

- ① INSTRUCTION 1 - ASSUME NO ISSUE CONFLICTS
 - ② INSTRUCTION 2 - ASSUME OPERAND REGISTER $X_k(2)$ CONFLICTS WITH INSTRUCTION 1 DESTINATION REGISTER $X_i(1)$. THAT IS, $k(2) = i(1)$.
THUS, ISSUE IS DELAYED TWO CLOCK PERIODS UNTIL THE $X_i(1)$ RESERVATION IS CLEARED.
- ^H THE $X_k(2)$ QUANTITY IS THE INSTRUCTION 1 RESULT THAT IS ENTERED INTO $X_i(1)$.

CONTROL DATA
DEVELOPMENT
DIVISION

TIMING DIAGRAM
NORMALIZE INSTRUCTION

CODE IDENT 34010	DWG NO 60420300	REV A
C	SHEET	PAGE 5-4-21

PART 5

FLOATING-ADD UNIT

FLOATING-ADD UNIT

DIAGRAM LAYOUT

The floating-add unit primary block diagram is drawn with the registers and flip-flops arranged in vertical columns according to clock periods.

There are three columns of registers. The first column on the left has the m_1 input flip-flop on top and the borrow register for the first stage of X_k minus X_j exponent on the bottom. The second column of registers has the m_1 flip-flop on top and the reference sign flip-flop on the bottom. The third column has the DP flip-flop on top and the borrow register for the first stage of shifted operand plus reference operand on the bottom.

Because of this arrangement, this diagram shows everything that happens in each clock period. Everything happening on the left side of the first column of registers, including the setting of the registers, occurs during the first clock period. Everything happening between the first and second columns of registers, including the setting of the second column of registers, occurs during the second clock period. Everything happening between the second and third columns of registers, including the setting of the third column of registers, occurs during the third clock period. Everything happening to the right of the third column of registers, including the setting of the result into the X_i register in the CPU, occurs during the fourth clock period.

The exponent takes a path from left to right across the upper half of the diagram, and the coefficient takes a path across the lower half of the diagram. Shift count determination and shift control are across the middle. The remaining control is across the top of the diagram.

GENERAL OPERATION

The floating-add unit performs CPU instructions 30 through 35. They are:

Floating sum (30)

Floating difference (31)

Floating double-precision sum (32)

Floating double-precision difference (33)

Round floating sum (34)

Round floating difference (35)

The m designator of the instruction controls the type of operation the unit performs. Bit 0 controls the mode of operation (add or subtract). Bit 1 controls single or double precision. Bit 2 controls the rounding operation.

Execution time is 4 clock periods.

The unit receives two 60-bit operands in floating-point format from the X registers specified by the j and k designators of the instruction.

If the instruction is a 31, 33, or 35 (subtract), m designator bit 0 causes the unit to complement the X_k operand. Thus, the unit subtracts by complementary addition.

The unit tests both exponents for overflow and indefinite. If a special case is detected, the output of the unit is blocked and the appropriate special case flag is sent to the CPU.

Before the two coefficients can be added, they must be aligned in a manner that causes the exponents to be equal. Thus, each bit in the adder has equal significance with the bit to which it is added. The unit aligns coefficient bits of equal significance by right-shifting the coefficient having the smaller exponent by an amount equal to the difference between the two exponents. The coefficient selected for shifting is called the shifted operand. The coefficient having the larger exponent is called the reference operand. The difference between the two exponents is determined by an 11-bit adder. The output of this adder translates into shift control signals. These signals control the shift ranks that shift the shift operand. This adder also determines which exponent is the larger and outputs a signal called sign of difference. Sign of difference is equal to one when X_j is the larger exponent. It controls the selection of the exponent and which coefficient becomes the shifted operand.

If the two exponents are equal so that no shifting is required, all the shift control signals are zero and the shifted operand passes through the shift ranks unchanged. Both coefficients occupy the upper half of the 97-bit shifted and reference operands with the signs of the two coefficients filling the lower half. The binary points are in the middle of the 97-bit operands (between bits 47 and 48). If the exponents are different so that alignment is required, the coefficient selected for shifting is right-shifted within the 97-bit shifted operand. The sign of this coefficient fills in above and below as it is shifted.

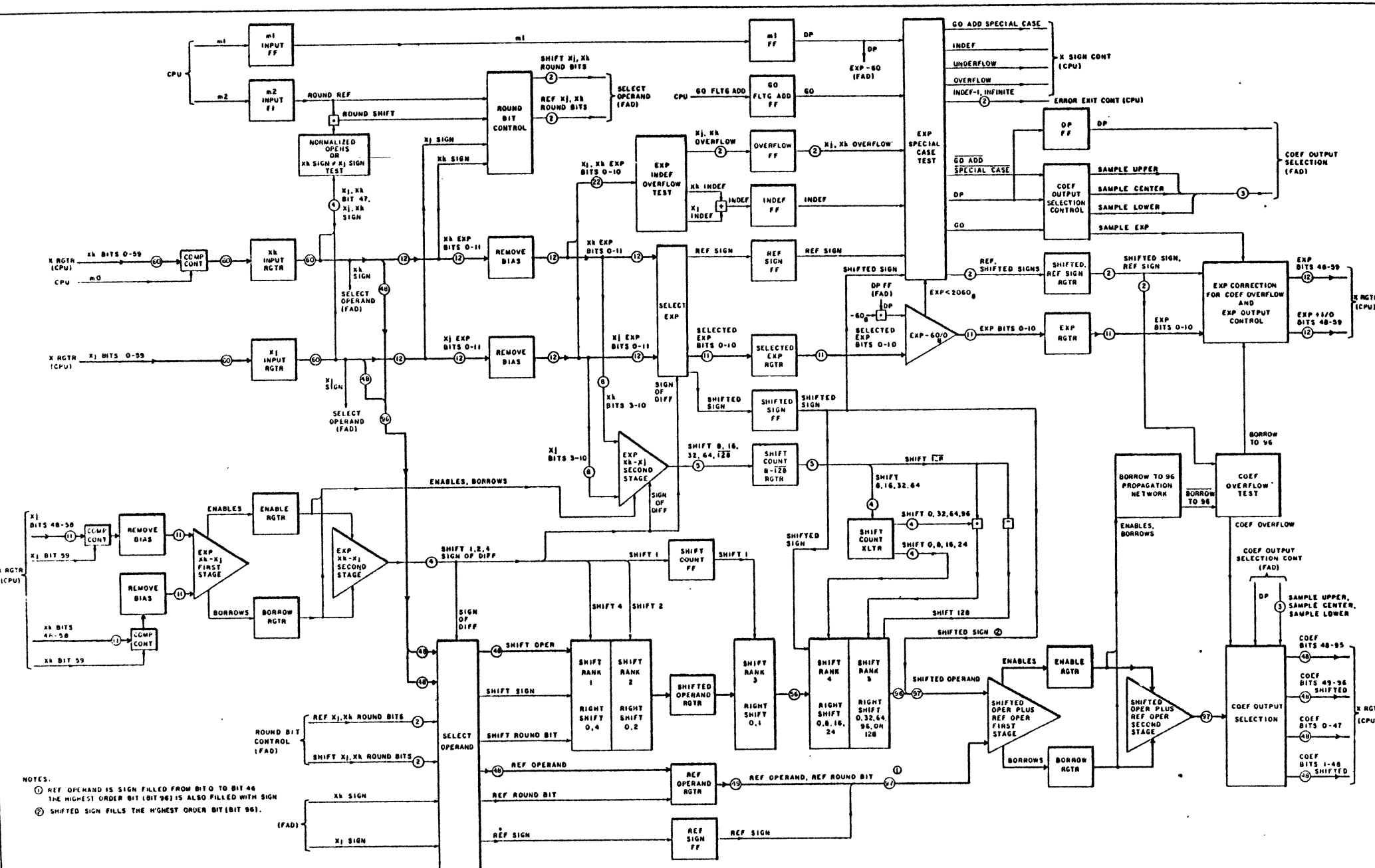
After the amount of the difference between the two exponents is determined, the smaller exponent is discarded, under control of sign of difference. The larger one is selected to continue through the unit to become the exponent of the result.

When the command is a 34 or 35 (round) instruction, m designator bit 2 causes the unit to add a round bit to one or both operands. If both operands are normalized or if the signs of the two operands are different, the round bit is added to both operands. If the above conditions are not met, the round bit is added only to the reference operand. The round bit added to the reference operand is always in bit position 47. This is immediately to the right of the binary point. If the round bit is also added to the shifted operand, it starts out in bit position 47 but is right-shifted with the rest of the coefficient. If no shifting takes place, the round bit remains in bit position 47. Round bits are equal to the complement of the sign of the operands to which they are added. After the alignment, the two operands are added in a 99-bit ones complement adder.

Instructions 30, 31, 34, and 35 (single precision) cause the unit to deliver the upper half of the 99-bit result to Xi location 0 through 47. The result exponent and sign bit are packed in floating-point format into Xi locations 48 through 59.

Instructions 32 and 33 (double precision) cause the unit to deliver the lower half of the 99-bit result to Xi locations 0 through 47. These 48 bits are to the right of the binary point, so the unit subtracts 48 from the result exponent to correct for double precision before sending it to the Xi register.

When the coefficient sum overflows the highest order bit of the result coefficient, the unit detects it by testing the signs of the two operands and determining whether or not there was a borrow (carry) into bit 96. If the unit determines that there was a coefficient overflow, the 48-bit coefficient delivered to Xi is taken from the 99-bit adder result one bit higher than if no overflow occurs. If overflow occurs, the exponent is increased by one prior to sending it to Xi.



NOTES:
 ① REF OPERAND IS SIGN FILLED FROM BIT 0 TO BIT 46
 THE HIGHEST ORDER BIT (BIT 96) IS ALSO FILLED WITH SIGN
 ② SHIFTED SIGN FILLS THE HIGHEST ORDER BIT (BIT 96).

**INPUT; SPECIAL CASE TEST; SHIFT COUNT DETERMINATION;
COEFFICIENT OUTPUT SELECTION CONTROL; EXPONENT SELECTION**

INPUT

The 60-bit Xj and Xk operands enter the 4FF7 module during the clock period in which the instruction issues from the CIW register. Instruction designator m bits 0 and 2 also enter the 4FF7 module at the same time. Designator m bit 1 enters the 4RD7 module (which will be discussed later).

Designator m bit 0 differentiates between an add and a subtract command. If m bit 0 is a one, the command is a subtract so it complements all 60 bits of the Xk operand. The 4FF7 module contains registers which hold both 60-bit operands for use during the second clock period.

ROUND BIT CONTROL

Designator m bit 2 controls the rounding of the coefficients. A flip-flop holds m bit 2 for use during the second clock period. The output of the m bit 2 input flip-flop is round reference. Round reference causes the round bit to be added to the operand with the largest exponent (reference operand) or the Xk operand if the exponents are equal. A round bit is also added to the operand with the smaller exponent (shift operand) if both operands are normalized or if the signs of the operands are not equal. The 4FF7 module tests for normalized operands by comparing bit 47 to the sign bit (59). The result of these tests is ANDed with round reference and causes the shift operand to be rounded.

The sign of the operand controls the binary state of the round bit for that operand. The state of the round bits is always different than the sign bit. This round bit preparation takes place on the 4FC7 modules.

The 4FC7 modules prepare round bits before the determination of which operand will be the reference and which will be the shift operand. Therefore, if the round reference signal is a one, the 4FC7 modules output a reference round bit for both the Xk and Xj operands. If the round shift signal is a one, shift round bits are output for both operands.

BIAS REMOVAL

The 4FC7 modules remove the bias, complement the exponent for negative coefficients, and test for overflow and indefinite. The sign bit (59) controls the removal of bias by complementing exponent bits 0 through 9 if the sign is negative. The sign bit controls bit 10 (bias bit) opposite to bits 0 through 9. For example, if the sign is negative, exponent bits 0 through 9 are complemented, and bit 10 is not complemented. If the sign is positive, exponent bits 0 through 9 are not complemented, and bit 10 is complemented.

OVERFLOW AND INDEFINITE TEST

The tests for overflow and indefinite take place after the removal of bias. Before removal of bias, an exponent of 3777 or 4000 indicates overflow. After removal of bias, an overflow is 1777. An indefinite exponent before bias removal is 1777 or 6000. Unbiased, indefinite is 3777.

SHIFT COUNT DETERMINATION

The exponents from the two floating-point format operands enter the 4FA7 module during the clock period that the instruction issues from the CIW register. Static networks remove the bias from both exponents and complement the Xk exponent. The exponents are also complemented if their coefficients are negative. The exponents then enter the first stage of an adder that subtracts the Xj exponent from the Xk exponent by complementary addition. Registers store the partial result for use in the 4FB7, 4FD7, and 3FE7 modules during the second clock period.

The 4FB7, 4FD7, and 3FE7 modules each contain a second stage adder which forms results that are shift control signals.

The second stage adder in the 4FB7 module receives enables and borrows from the 4FA7 module and forms a result which is shift 1, 2, 4, and sign of difference. Sign of difference is a one when the Xj exponent is larger than the Xk exponent. In this case, the shift 1, 2, and 4 signals are in complement form. The 4FB7 module sends shift 1, 2, and 4 signals to shift ranks 1, 2, and 3 where they control the shifting of the shift operand. Sign of difference is used for operand selection.

The second stage adder in the 4FD7 module receives group enables and group borrows from the 4FA7 module. The adder also receives Xk and Xj exponent bits 3 through 5 from the 4FC7 modules. From these inputs, it forms results which are shift 8, 16, and 32 control signals. These shift terms are always in true form. A register holds them for use during the third clock period on the 3FH7 modules.

The second stage adder in the 3FE7 module receives group enables and group borrows from the 4FA7 module. The adder also receives Xk and Xj exponent bits 6 through 10. From these inputs, it forms results which are shift 64 and 128 control signals. A register holds them for use during the third clock period on the 3FH7 module.

EXPONENT SELECTION

During the second clock period of execution, the unit makes a selection between the operands. One operand becomes the reference operand, and the other becomes the shift operand. The coefficient for this shift operand goes to a shift network where it is right-shifted by an amount equal to the difference between the two exponents. The unit discards the exponent of the shift operand. The sign of the shift operand becomes the shifted sign, and the sign of the reference operand becomes the reference sign.

The selection of which exponent will be the reference exponent takes place in the 4FD7 and 3FE7 modules. The sign of difference makes the choice and gates the larger exponent into the selected exponent register. Sign of difference is equal to one when Xj is the larger exponent.

Sign of difference gates the sign of the operand with the smaller exponent into the shifted sign flip-flop and the sign of the operand with the larger exponent into the reference sign flip-flop. These registers and flip-flops hold the selected exponent, shifted signs, and reference signs for use during the third clock period. The 4FD7 module handles the selection of exponent bits 0 through 5. The 3FE7 module handles the selection of exponent bits 6 through 10, shifted signs, and reference signs.

EXPONENT CORRECTION FOR DOUBLE PRECISION

Designator m bit 1 differentiates between a single-precision and a double-precision instruction. If m bit 1 is a zero, the instruction is single-precision and the unit selects the upper 48 bits of the 97-bit result as the result coefficient. If m bit 1 is a one, the instruction is double-precision and the unit selects the lower 48 bits of the 97-bit result register as the result coefficient. Since the binary point is effectively right-shifted 48 places, it is necessary to adjust the exponent by subtracting 48 from it.

Designator m bit 1 enters the 4FD7 module during the clock period in which the instruction issues from the CIW register. Flip-flops hold it for 2 clock periods for use during the third clock period. The output of the two flip-flops is double-precision and goes to the 3FM7 module.

During the third clock period, exponent bits 0 through 10 enter the 3FM7 module from the selected exponent register in the 4FD7 and 3FE7 modules. A static network in the 3FM7 module complements bits 0 through 10 and feeds them into an adder that subtracts 60 (octal) if DP is a one. If DP is a zero, exponent bits 0 through 10 pass through the adder unaltered. The exponent always leaves this adder in true form so it needs no complementing. A register holds the result for use during the fourth clock period.

SPECIAL CASE TEST

The floating-add unit senses overflow, underflow, and indefinite special cases. The 4FC7 modules test for overflow and indefinite for each operand and send the results to the 4FD7 module. The 4FD7 module stores Xk and Xj overflow for use during the third clock period. It also ORs Xk and Xj indefinite and stores the result in the indefinite flip-flop for use in the 3FM7 module during the third clock period.

Static networks in the 3FM7 module test these conditions and send the proper special case flag to the X sign control translator during the third clock period.

The following table shows which flag is sent to the X sign control translator for different conditions of overflow and indefinite. For example, if either operand is indefinite, the indefinite flag is set. If both operands are negative overflow, the overflow flag is set.

- W = Operand with no special cases
- ∞ = Overflow with negative sign
- +∞ = Overflow with positive sign
- IND = Indefinite flag
- OVF = Overflow flag

		Xk			
		W	+∞	-∞	IND
Xj	W		OVF	OVF	IND
	+∞	OVF	OVF	IND	IND
	-∞	OVF	IND	OVF	IND
	IND	IND	IND	IND	IND

Signs are reference and shifted.

An underflow occurs when the unbiased selected exponent is a number more negative than 2060 (octal) and the instruction is double-precision. The correction for double-precision subtracts 60 (octal) from the exponent causing it to exceed the most negative end of its range. If this happens, the 3FM7 module sends the underflow signal to the X sign control translator.

During the second clock period, the 4FD7 module receives the go floating-add signal from the CPU if the instruction code is 30 through 35. The go holding flip-flop holds it for use during the third clock period in the 3FM7 module. During the third clock period, the go signal enters a flip-flop in the 3FM7 module and enables output data to the destination X register during the fourth clock period. If go floating-add is not set, the data is discarded.

If the go floating-add signal is received and a special case flag is set, go add special case is set. This signal, together with the special case flag and the sign of the reference operand, goes to the X sign control translator and causes the X register input circuits to generate the proper special case result.

The 3FM7 module generates indefinite-1 and infinite signals. When these signals occur, the corresponding bit is set in the exit condition register (CPU). The indefinite-1 signal is generated when go and indefinite signals are present. The infinite signal is generated when go and either Xk or Xj overflow signals are present.

COEFFICIENT OUTPUT SELECTION CONTROL

During the fourth clock period, the floating-add unit transmits the results to the destination X register. There are four possible data paths from the coefficient portion of the unit to the X registers. The unit selects one of these four paths on the basis of instruction mode and if coefficient overflow occurred.

There are two possible output data paths from the 97-bit result for single-precision commands, one for normal single precision (bits 48 through 95) and one for coefficient overflow (bits 49 through 96).

There are also two possible output data paths from the 97-bit result for double-precision commands, one for normal double precision (bits 0 through 47) and one for coefficient overflow (bits 1 through 48).

The coefficient output selection control on the 3FM7 module controls the selection of which half of the 97-bit result is transmitted from the unit.

If the instruction is single precision with no special cases and a go floating-add is received from the CPU, the sample upper half and sample center signals are ones. These signals are held for use during the fourth clock period in the 4FK7 module.

If the instruction is double precision with no special cases and go floating-add is received from the CPU, the sample lower half and sample center signals are ones. These signals are held in the 3FM7 module for use during the fourth clock period by the 4FK7 module.

EXPONENT SAMPLE CONTROL

The exponent for the result coefficient is transmitted to the destination X register during the fourth clock period. Control for enabling the exponent output takes place on the 3FM7 module. If there are no special cases and go floating-add is received from the CPU, the sample exponent signal is sent to the 4FL7 module where it enables the output of the exponent.

COEFFICIENT SELECTION, SHIFT

COEFFICIENT SELECTION

Coefficient selection takes place during the second clock period on the 4FG7 and 4FO7 modules. The coefficient with the larger exponent becomes the reference operand and the other becomes the shifted operand. A register on the 4FG7 and 4FO7 modules holds the reference operand for use during the third clock period. The coefficient having the smaller exponent becomes the shift operand. The shift operand enters a shift network which right-shifts it by an amount equal to the difference between the two exponents.

Sign of difference gates the coefficient having the larger exponent into the reference operand register. Sign of difference is a one when X_j is the larger exponent. Sign of difference also gates the proper round bits. If the X_k coefficient becomes the shift operand, the shift X_k round bit is selected for the shift operand. If the X_k coefficient becomes the reference operand, the reference X_k round bit is selected for the reference operand. The round bit is inserted one bit position to the right of bit 0 of the original coefficient. Sign of difference gates the coefficient having the smaller exponent into shift rank 1.

Both coefficients expand into 96-bit operands before entering the adder. Since the reference operand is not shifted, the original coefficient becomes bits 48 through 95 of the 96-bit reference operand with the round bit in position 47. Reference sign fills bits 0 through 46 of the reference operand.

SHIFT

If the exponents are equal, there are no shifts so the shifted operand occupies bits 47 through 95 with bit 47 being the shift round bit. If the exponents are not equal, this coefficient is right-shifted by the amount of the difference. The bits on either end of the original coefficient in the 96-bit shifted operand are filled with shifted sign.

The shifting takes place in the 4FG7, 4FO7, and 3FH7 modules. The 4FG7 and 4FO7 modules contain shift ranks 1, 2, and 3. The control signals for these three shift ranks are shift 1, 2, and 4 which come from the 4FB7 module. If sign of difference is a one, the 4FB7 module sends these signals in complementary form. To correct for this, the 4FG7 and 4FO7 modules recomplement them if sign of difference is a one. Therefore, when the shift signals enter the shift ranks, they are always in true form.

Shift rank 1 performs a right shift of zero or four bit positions under the control of shift 4. If shift 4 is a one, the shift rank performs a right shift of four bit positions. If shift 4 is a zero, the operand moves through the shift rank unchanged. Shift ranks 2 and 3 function in the same way under control of shift 2 and shift 1.

These first three shift ranks perform any combination of right shifts up through seven bit positions. If a shift of seven is performed, the round bit which was inserted in bit position 47 is shifted to bit position 40. Therefore, the output of shift rank 3 is shifted operand bits 40 through 95.

Shift ranks 1 and 2 perform their shifts during the second clock period. The shifted operand register holds the result for use during the third clock period. Shift rank 3 performs its shift during the third clock period and sends the results to the 3FH7 modules for further shifting.

Shifted operand bits 40 through 95 enter the 3FH7 module during the third clock period where the remainder of the shifting takes place in shift ranks 4 and 5.

Control for these shift ranks comes from the 4FD7 and 3FE7 modules during the third clock period. These control signals are shift 8, 16, 32, 64, and 128. Shift 8, 16, 32, and 64 control signals enter the shift count translator on the 3FH7 module. The translator outputs two groups of shift control signals.

The first group of signals is shift 0, 8, 16, and 24. These signals go to shift rank 4 where they cause this shift rank to right-shift the shifted operand by 0, 8, 16, or 24 bit positions.

The second group of signals is shift 0, 32, 64, and 96. These signals are ANDed with shift 128 and go to shift rank 5. Shift 128 goes directly to shift rank 5.

The output of shift rank 5 is shifted operand bits 0 through 95. The 3FH7 module sends this shifted operand to the 4FJ7 and 4FI7 modules for the first stage of addition.

ADDER; COEFFICIENT OUTPUT SELECTION; COEFFICIENT OVERFLOW DETECTION;
EXPONENT OUTPUT CONTROL; OUTPUT

ADDER

Shifted operand bits 0 through 95 and reference operand bits 49 through 95 enter the 4FJ7 and 4FI7 modules during the third clock period. Reference sign fills bits 0 through 46 of the reference operand. Bit 96 of the reference operand is filled with reference sign, and bit 96 of the shifted operand is filled with shifted sign. This makes room for coefficient overflow and controls the section borrow from section 11 (end-around borrow). Since the upper two bits of the adder are sign bits, if both operands are positive the adder always has an end-around borrow. If both operands are negative, the adder never has an end-around borrow. If the signs are not alike, the remaining bits control the end-around borrow.

Each 4FJ7 module handles nine bits, and the 4FI7 module handles seven bits. Both operands are complemented as they enter the 4FJ7 and 4FI7 modules and then enter the first stage of the adder. The first stage of the adder divides the operands into bits, groups, and sections and forms a partial sum consisting of enables and borrows.

ADDER FORMAT

Each module is a section; therefore, sections 1 through 10 are nine bits long because they are on 4FJ7 modules, and section 11 is seven bits long because it is on a 4FI7 module. A section borrow sets when there is a borrow-out of that section. A section enable sets when all bit enables in that section are set.

A group is three bits long. There are two groups per section. These two groups form the lower six bits of each section. A group enable sets when all bit enables in that group set. A group borrow sets when there is a borrow-out of that group. This partial sum is held in registers for use during the fourth clock period.

A second stage of the adder is on 4FK7 modules. The second stage receives the partial sum inputs and forms the sum. The sum at this point is in ones complement form.

COEFFICIENT OVERFLOW DETECTION

The result of an addition or subtraction instruction may be one bit longer than the operand's input into the adder. Therefore, since the adder receives 96-bit operands, it is possible to overflow into the 97th bit. When this happens, it is detected in the 4FL7 module which generates a coefficient overflow signal. Coefficient overflow causes the result coefficient to be right-shifted one bit position to retain this overflow bit. Because of this right shift, the exponent is corrected by adding one to it.

The coefficient overflow detecting network is on the 4FL7 module and is divided into two parts. The first part receives section and group enables and borrows from the first stage of the adder. It tests to see if a borrow will be propagated to bit 96. The second part receives the output from the first part and the signs of the two operands. It determines if there was an overflow of the coefficient.

To have a coefficient overflow, the signs of the operands must first be alike. If the signs are not alike, they actually subtract and cause a smaller result.

If the signs of both operands are positive, the coefficient overflow determination network receives borrow to 96. If the signs are both negative, the network receives borrow to 96.

COEFFICIENT OUTPUT SELECTION

There are four coefficient data paths from the 97-bit result coefficient. The 4FK7 modules output the upper 48 bits for single-precision instructions and the lower 48 bits for double-precision instructions. If coefficient overflow occurred, the 4FK7 modules select a different set of outputs that perform a wired right-shift of one bit position to the upper or lower half of the 97-bit result.

Normal output for single precision is bits 48 through 95 of the 97-bit result. This data path is coefficient bits 48 through 95 and becomes bits 0 through 47 in the X register. If coefficient overflow occurred, the 4FK7 modules output bits 49 through

96 of the 97-bit result. This coefficient overflow data path is coefficient bits 49 through 96 shifted which becomes bits 0 through 47 in the X register.

Normal output for double precision is bits 0 through 47 of the 97-bit result which become bits 0 through 47 in the X register. If coefficient overflow occurred, the 4FK7 modules output coefficient bits 1 through 48 shifted, which become bits 0 through 47 in the X register.

Control of which half of the 97-bit result coefficient is sent to the X register comes from the coefficient output selection control on the 3FM7 module. These control signals are sample upper half, sample center, and sample lower half. The double precision (CP) also comes from the 3FM7 module.

The 4FK7 module in location 5E06 handles bits 45 through 53 of the 97-bit result. Because some of its bits are in the upper half and some in the lower half, this module is the only 4FK7 module that has four possible outputs.

The DP, sample center, and coefficient overflow signals control the output of bits 45 through 53 in the following manner.

If DP, sample center, and coefficient overflow signals are all present, the data path for coefficient bits 45 through 48 shifted is enabled.

If only sample center and coefficient overflow are present, the data path for coefficient bits 49 through 53 shifted is enabled.

If only DP and sample center are present, the data path for coefficient bits 45 through 47 is enabled.

If only sample center is present, the data path for coefficient bits 48 through 53 is enabled.

If sample center is not present, all ones are sent to the X register.

Bits 54 through 96 are controlled by sample upper half and coefficient overflow signals. The 4FK7 module inputs for DP and \overline{DP} are both forced to a constant one. Thus, if sample upper half is present, data is gated from the upper half

of the 97-bit result. Coefficient overflow determines which set of outputs (shifted or unshifted) is used.

Bits 0 through 44 are controlled by sample lower half and coefficient overflow signals in the same manner as the upper half.

The upper unused bits, coefficient bits 96 through 98 and coefficient bits 97, 98 shifted, are terminated. Coefficient bit 0 shifted is also terminated because the right shift for coefficient overflow is an end-off type shift and bit 0 is discarded.

EXPONENT CORRECTION FOR COEFFICIENT OVERFLOW, EXPONENT OUTPUT CONTROL

The exponent enters the 4FN7 module during the fourth clock period and has a choice of two paths through the module. The path taken by the exponent is under control of sample exponent, the signs of the two operands, and borrow to 96.

Sample exponent allows the 4FN7 module to output the exponent. If sample exponent is not present, both exponent paths are blocked and all ones are output to the X register.

If the signs of the two operands are alike, borrow to 96 determines if there was a coefficient overflow. If coefficient overflow occurred, the exponent is corrected by the addition of one. This correction is necessary because the coefficient was right-shifted one bit position. If the signs of the two operands are different, borrow to 96 indicates which operand was the larger so the correct sign can be given the result.

Three combinations of signs and borrow to 96 cause the exponent to take the path through output 2. They are listed with the sign given the result coefficient, the form (true or ones complement) of the result exponent, and if there was coefficient overflow.

Both Signs Positive	Both Signs Negative	Unlike Signs
Borrow to 96	<u>Borrow to 96</u>	<u>Borrow to 96</u>
+ sign of the result	- sign of the result	+ sign of the result
Output in true form	Output in ones complement form	Output in true form
No coefficient overflow	No coefficient overflow	No coefficient overflow

When both signs are positive and borrow to 96 is a one, there is no coefficient overflow. Since both signs are positive, the result is positive and expressed in true form. Complement control 2 controls the form of the result exponent and adds bias, sign 2 gives it its sign, and output control 2 gates the exponent to the X register.

When the signs of the two operands are not alike, borrow to 96 indicates which operand is larger. In the case of output 2, borrow to 96 is a zero so the positive operand is the larger of the two. Thus, the sign given the result exponent by sign 2 is positive, and complement control 2 causes the exponent to be output in true form.

Three combinations of signs and borrow to 96 cause the exponent to take the path through the exponent plus 1/0 adder and output 1. They are listed with the sign given the result coefficient, the form of the result exponent, and if there was coefficient overflow.

Both Signs Positive	Both Signs Negative	Unlike Signs
<u>Borrow to 96</u>	Borrow to 96	Borrow to 96
+ sign of the result	- sign of the result	- sign of the result
Output in true form	Output in ones complement form	Output in ones complement form
Coefficient overflow	Coefficient overflow	No coefficient overflow

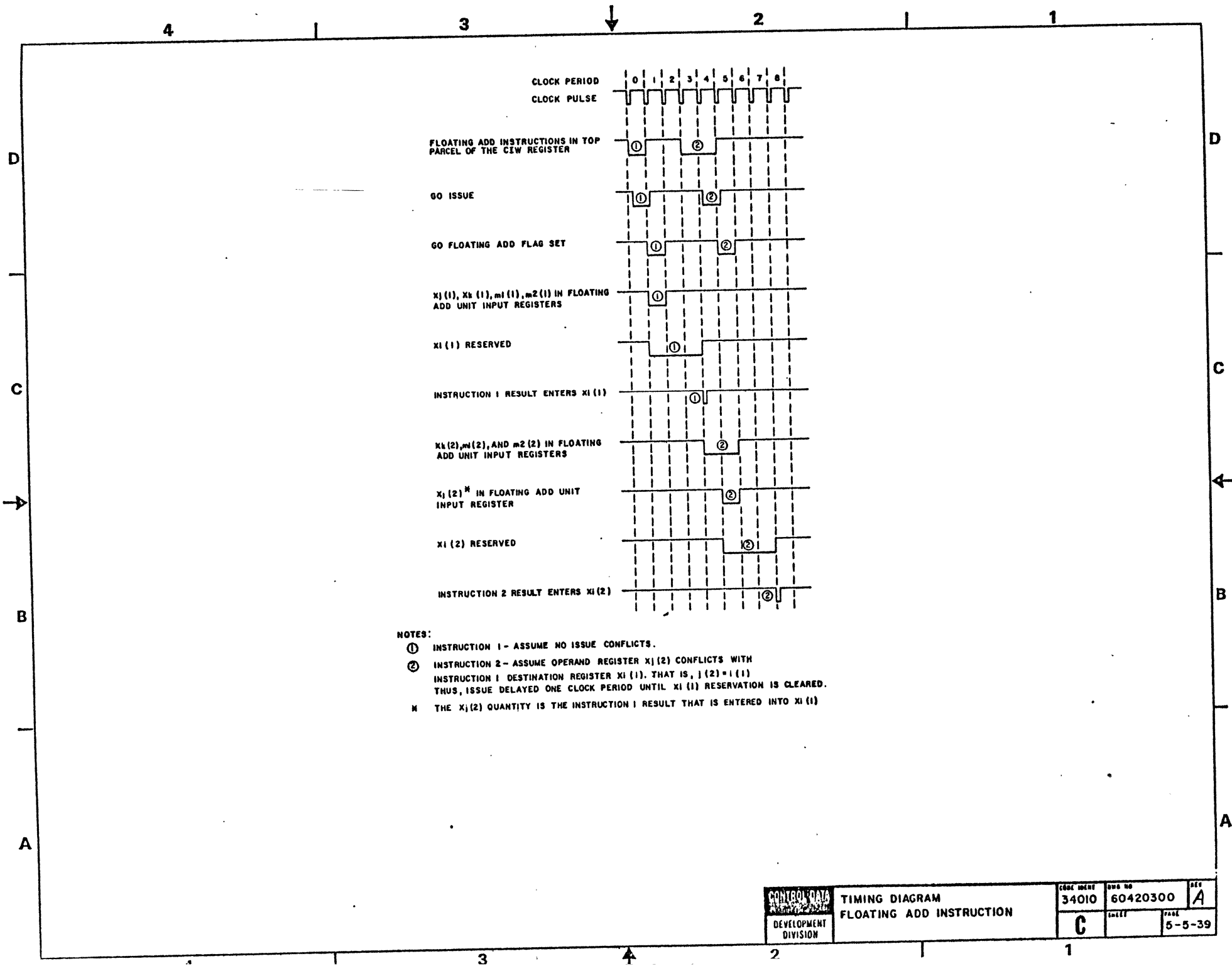
When both signs are positive and borrow to 96 is a zero, there is a coefficient overflow. The exponent takes the path through the adder where plus 1 causes the adder to add ones to the exponent. Complement control 1 causes the exponent to be output in true form and adds bias, sign 1 gives the result coefficient a positive sign, and output control 1 gates the result exponent to the X register.

Complement control 1 also ensures that the exponent correction for coefficient overflow does not result in the unit sending out a negative 0 exponent. If both signs are positive and the exponent takes the path through output 1, it is because coefficient overflow and the adder add one to the exponent. In this case, if the exponent is negative 1, the addition of one results in a negative 0 exponent. Complement control 1 senses this and causes the 4FN7 module to output a positive 0.

When the signs of the two operands are not alike and borrow to 96 is a one, there is no coefficient overflow. The exponent takes the path through the adder but plus 1 is a zero because the signs are not alike. This causes the exponent to pass through the adder unchanged. Complement control 1 causes the exponent to be output in ones complement form, and sign 1 gives a negative sign to the result coefficient. In this case, borrow to 96 indicates that the negative operand was the larger of the two.

The resultant leaving output 1 of the 4FN7 module is exponent positive 1/0 bits 48 through 59 and goes to the destination X register.

The result leaving output 2 of the 4FN7 module is exponent bits 48 through 59 and goes to the destination X register.



NOTES:

- ① INSTRUCTION 1 - ASSUME NO ISSUE CONFLICTS.
- ② INSTRUCTION 2 - ASSUME OPERAND REGISTER $X_1(2)$ CONFLICTS WITH INSTRUCTION 1 DESTINATION REGISTER $X_1(1)$, THAT IS, $J(2) = I(1)$ THUS, ISSUE DELAYED ONE CLOCK PERIOD UNTIL $X_1(1)$ RESERVATION IS CLEARED.
- * THE $X_1(2)$ QUANTITY IS THE INSTRUCTION 1 RESULT THAT IS ENTERED INTO $X_1(1)$

PART 6

LONG ADD UNIT

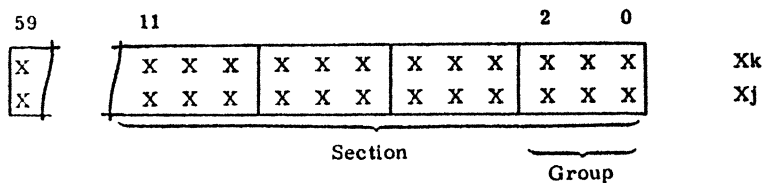
LONG ADD UNIT

The long add unit executes CPU instructions 36 and 37. It performs a 60-bit integer addition to X register operands specified by the j and k portions of the instruction and delivers the 60-bit sum to the X register specified by the l portion of the instruction. Prior to addition, the unit complements the Xk and Xj operands for the 36 instruction (integer sum) and only the Xj operand for the 37 instruction (integer difference).

The long add instructions require 2 clock periods for execution. The operands move from the X registers to the long add unit in the same clock period in which the instruction issues from the CIW register. The result moves from the long add unit to the destination X register during the following clock period. A new instruction may issue from the CIW register for execution in the long add unit each clock period.

The long add unit complements the operands as specified by the instruction code and forms a partial sum in the first stage of addition. The partial sum enters the second stage of the adder where the result is formed from the partial sum. At this point, the result is in ones complement form. The go long add flag, sent from the CPU, gates this sum to the result X register through a static network which complements it to return it to true form.

Adder Format:



FIRST STAGE

The first stage of addition forms a partial sum which consists of bit enables, bit borrow generates, group borrow generates, section borrow generates, and section enables. Group enables are also generated but are not sent to the second stage of the adder.

	0		1
Bit enables =	<u>1</u>	or	<u>0</u>
	1		1

	1
Bit borrow generates =	<u>1</u>
	0

Group borrow generates = borrow-out of three-bit group

Example:	110	101	
	<u>010</u>	<u>110</u>	
	000	or	011

Group enable = all bit enables in group equal to one

Section borrow generates = borrow-out of 12-bit section

Section enable = all group enables in section equal to one

SECOND STAGE

The second stage of the adder forms group borrow inputs and bit borrow inputs from the partial sum. An exclusive OR of the bit borrow inputs and the bit enables forms the sum. The sum at this point is in ones complement form.

Group borrow inputs = borrow into group

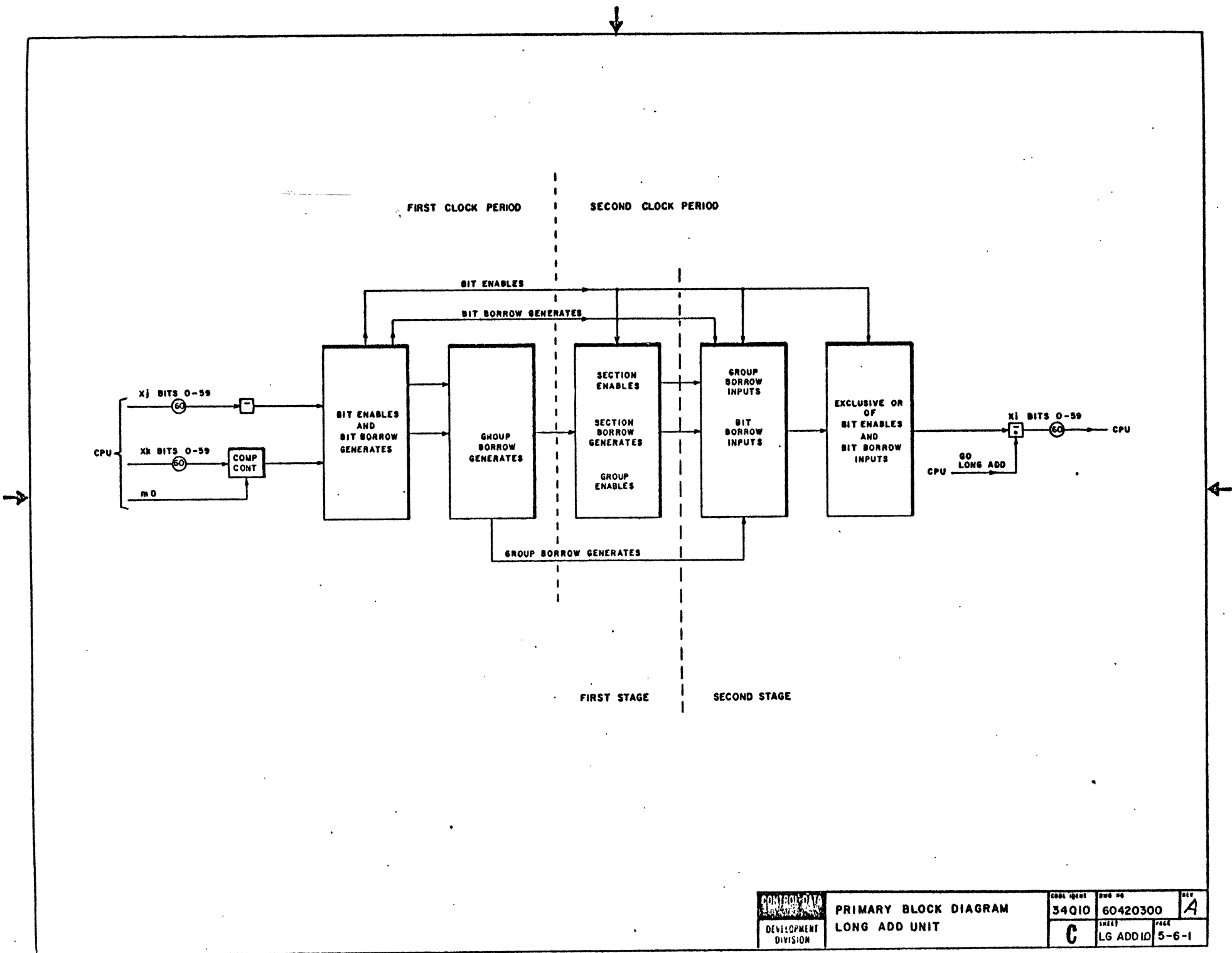
Bit borrow inputs = borrow into bit

Example:

The following example uses a 12-bit adder performing a 36 instruction. In this case, the section borrow generate is the end-around borrow. In the case of the long add unit, it is the section borrow from the highest-order section.

36 Instruction Example:

4470 → Comp → 3307 →	011	011	000	111	
<u>1532</u> → Comp → 6245 →	<u>110</u>	<u>010</u>	<u>100</u>	<u>101</u>	
6222	101	001	100	010	Bit enables
	010	010	000	101	Bit borrow generates
	1	0	0	1	Group borrow generates
	1				Section borrow generates
	0				Section enable
	0	0	1	1	Group borrow inputs
	100	100	001	111	Bit borrow inputs
	001	101	101	101	Exclusive OR of bit borrow inputs and bit enables
	110	010	010	010	Comp
	6	2	2	2	Sum (octal)



CONTROL DATA DEVELOPMENT DIVISION	PRIMARY BLOCK DIAGRAM LONG ADD UNIT		COOL '9401 34Q10	Dwg No 60420300	DES A
	C	SHEET LG ADD10	PAGE 5-6-1		

LONG ADD UNIT

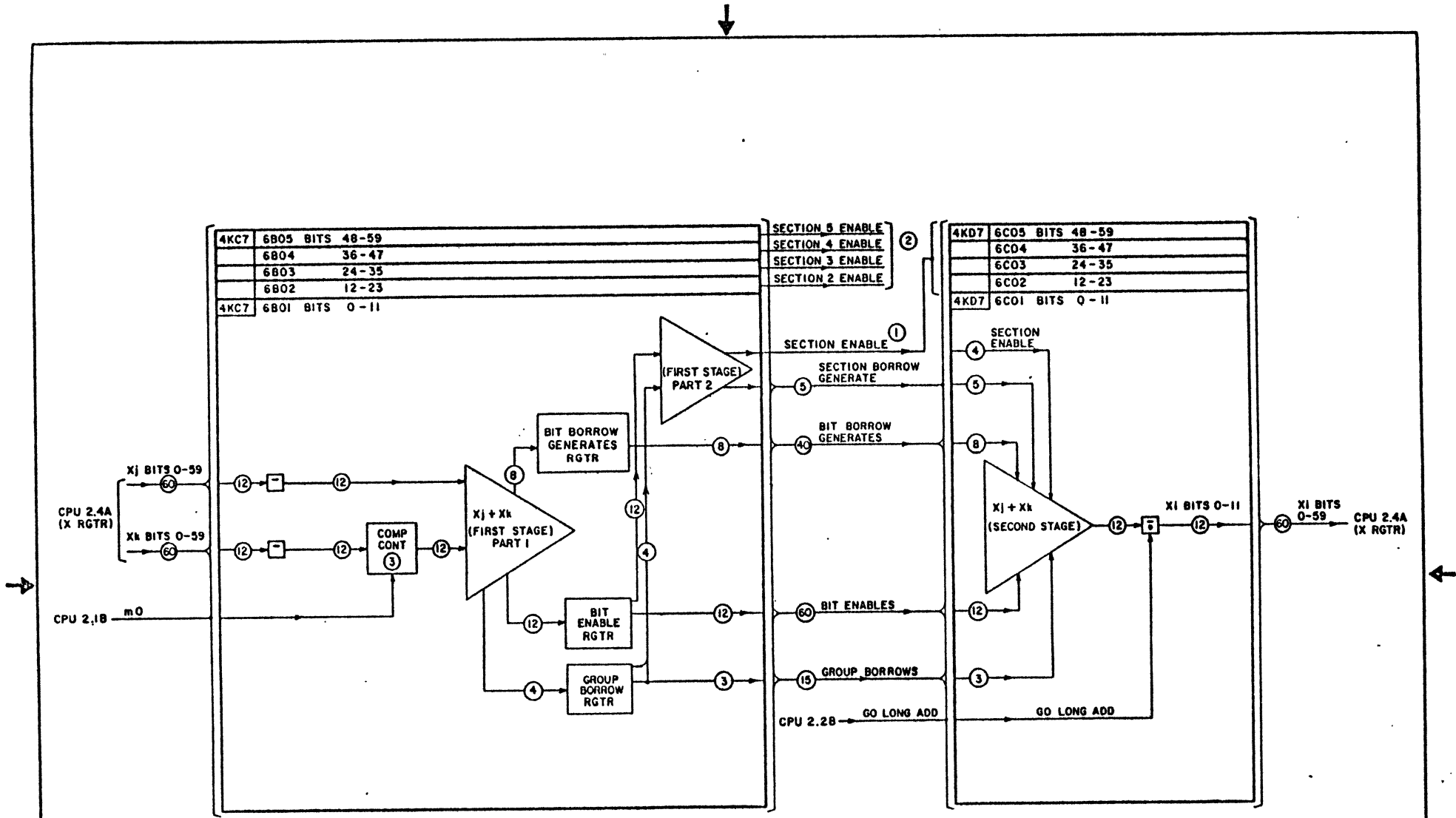
The long add unit executes CPU instruction 36 and 37. It performs a 60-bit integer addition to X register operands specified by the j and k portions of the instruction and delivers the 60-bit sum to the X register specified by the i portion of the instruction. Prior to addition, the unit complements the Xk and Xj operands for the 36 instruction (integer sum).

The long add instructions require 2 clock periods for execution. The operands move from the X registers to the long add unit in the same clock period in which the instruction issues from the CIW register. The result moves from the long add unit to the destination X register during the following clock period. A new instruction may issue from the CIW register for execution in the long add unit each clock period.

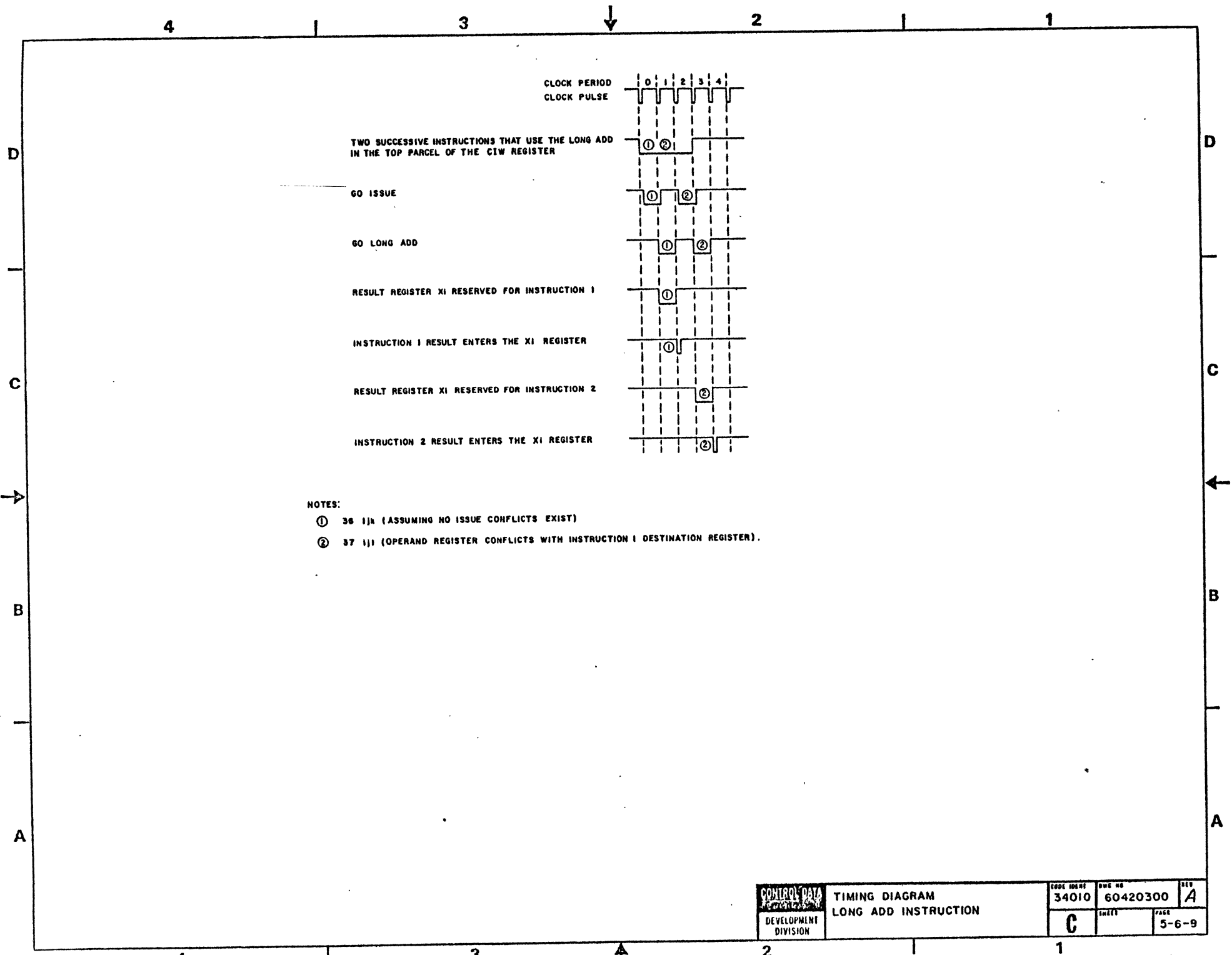
During the first clock period, the 4KC7 modules complement both operands. If the instruction is integer difference (37), m bit 0 is a one and the 4KC7 modules recomplement the Xk operand. After complementing, the operands enter the first stage of the adder.

The first stage of the adder is divided into two parts. The first part forms the bit enables, bit borrow generates, and group borrow generates. Registers hold them for use during the second clock period. The second part of the first stage of addition takes place during the second clock period. It forms the section borrow generates and section enables.

The 4KC7 modules send the partial sum to the 4KD7 modules where the second stage of addition takes place. Go long add gates the result from the second stage to the result X register.



- NOTE:
- ① SECTION 1 ENABLE GOES TO 6C02, 6C03, 6C04, 6C05
 - ② SECTION 2 ENABLE GOES TO 6C01, 6C03, 6C04, 6C05
SECTION 3 ENABLE GOES TO 6C01, 6C02, 6C04, 6C05
SECTION 4 ENABLE GOES TO 6C01, 6C02, 6C03, 6C05
SECTION 5 ENABLE GOES TO 6C01, 6C02, 6C03, 6C04
 - ③ m = 1 RECOMPLEMENT Xk OPERAND



PART 7

MULTIPLY UNIT

MULTIPLY UNIT

The floating-multiply unit executes the following three instructions.

- 40 Floating product of (Xj) times (Xk) to Xi
- 41 Rounded floating product of (Xj) times (Xk) to Xi
- 42 Floating double-precision product of (Xj) times (Xk) to Xi

The Xi coefficient and Xi exponent are formed separately with consideration taken for single or double precision. If single precision is specified, the upper half of the coefficient result and an exponent (corrected for single precision) are sent to the X register. For double precision, the lower half of the coefficient result and the exponent result are sent to the X register.

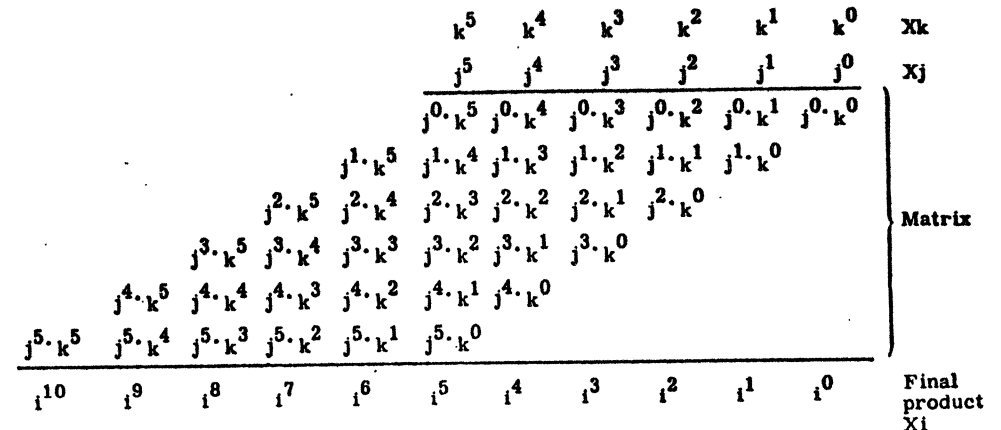
If both operands are normalized, the result is normalized. If rounding is specified, a round bit is generated and added to bit 46 of the coefficient result.

The multiply instructions require 5 clock periods for execution. Multiply instructions may enter the multiply unit every other clock period. Thus, during a 5-clock-period segment, more than one multiply instruction may be active in the unit.

COEFFICIENT MULTIPLY

Coefficient multiply is basically the same as a multiply done with paper and pencil. Every bit of the multiplier is multiplied by every bit of the multiplicand. In binary arithmetic, this is the AND (logical product) function.

As in pencil and paper multiplication, the bit-by-bit products are arranged in a matrix as shown below with each row offset by one bit. The columns of the matrix are then added to obtain the final product, Xi.



The multiply is performed in two steps. First, the lower 24 Xj bits are multiplied by all 48 Xk bits, and the columns of the resulting matrix are partially added. The upper 24 Xj bits are then multiplied by all 48 Xk bits and the resulting matrix is added to the partial sums and carries from the first pass to form the 96-bit double-precision product. Each step in the process involves forming 1152 binary products. These 1152 bits of data must then be added in the proper groupings to form a combined sum.

The multiply in the previous example appears as follows when executed in two steps.

First pass												
				k^5	k^4	k^3	k^2	k^1	k^0			
							j^2	j^1	j^0			
				$j^0.k^5$	$j^0.k^4$	$j^0.k^3$	$j^0.k^2$	$j^0.k^1$	$j^0.k^0$			
			$j^1.k^5$	$j^1.k^4$	$j^1.k^3$	$j^1.k^2$	$j^1.k^1$	$j^1.k^0$				
	$j^2.k^5$	$j^2.k^4$	$j^2.k^3$	$j^2.k^2$	$j^2.k^1$	$j^2.k^0$						
Partial sums	ps7	ps6	ps5	ps4	ps3	ps2	ps1	ps0	Results of first, second, and third level adds			
Partial carries	pc7	pc6	pc5	pc4	pc3	pc2						
Second pass												
				k^5	k^4	k^3	k^2	k^1	k^0			
							j^5	j^4	j^3			
				$j^3.k^5$	$j^3.k^4$	$j^3.k^3$	$j^3.k^2$	$j^3.k^1$	$j^3.k^0$			
		$j^4.k^5$	$j^4.k^4$	$j^4.k^3$	$j^4.k^2$	$j^4.k^1$	$j^4.k^0$					
	$j^5.k^5$	$j^5.k^4$	$j^5.k^3$	$j^5.k^2$	$j^5.k^1$	$j^5.k^0$						
	ps10	ps9	ps8	ps7	ps6	ps5	ps4	ps3	Results of first level add			
	pc10	pc9	pc8	pc7	pc6	pc5						
			ps7	ps6	ps5	ps4	ps3	ps2	ps1	ps0	Add loop bits from first pass	
			pc7	pc6	pc5	pc4	pc3	pc2				
	ps10	ps9	ps8	ps7	ps6	ps5	ps4	ps3	ps2	ps1	ps0	Results of second and third level adds
	pc10	pc9	pc8	pc7	pc6	pc5	pc4	pc3				
i^{11}	i^{10}	i^9	i^8	i^7	i^6	i^5	i^4	i^3	i^2	i^1	i^0	Final add

In a 24-by 48-bit multiply, in order to handle 1152 bit products in the matrix, the summing is performed in several more stages; there are several more levels of sums and carries. Because of the size of the matrix, there are also several sums or carries per bit position at some levels of the multiply. Since a complex carry network to resolve all the carries requires excessive time and hardware, the multiply unit employs a carry save adder for efficiency.

Using these inputs, the first half of the adder forms pseudo sums and pseudo carries.

CARRY SAVE ADDER

The carry save adder permits the addition of columns of numbers without using the time-consuming carry and satisfy checks typical of full adders. Instead, each level of add has two outputs, pseudo sums and pseudo carries.

	1	0	1	0
	0	1	1	0
Pseudo sums	1	1	0	0

	1	0	1	0	
	0	1	1	0	
Pseudo carries	0	0	1	0	(Pseudo carries are displaced one bit to the left because they affect the next significant bit position.)

The sum of these two numbers is the actual answer; however, the sum is not taken until the final add. Instead, both quantities are put back into the next level of the adder and are summed with other pseudo sum and carry bits that represent the same bit position of the final product (that is, sums of bits in the same column of the matrix or carries to that column).

The inputs to the first level add are the logical products produced by the matrix. The inputs to the second level add are the pseudo sums and carries from the first level add. The inputs to the third level add are the pseudo sums and carries from the second level add. The output from the third level add is a pseudo sum and carry bit for each bit position of the product.

After the first pass, the output from the third level add is right-shifted 24 places, fed back into the second level add, and added to the results of the second pass through the matrix. The second and third level adds are then repeated for the entire product until a pseudo sum and carry bit again represent each bit position of the product. This time the output of the third level add is fed to the final add network and is reduced to one bit for each bit position of the final product (96 bits).

EXPONENT FORMATION

Two adders are used for the formation of the final exponent. The first adder is a half adder that forms the sum of the complemented exponents of Xj and Xk. When single precision is selected, this adder adds 60 (octal) to the exponents to compensate for truncating the coefficient of the product in single-precision mode. The second adder subtracts one from the result of the first adder if the coefficient is left-shifted one place to normalize it.

The exponent logic also examines the incoming exponents for special cases involving overflow, underflow, or indefinite operands before performing the additions. Upon receiving the output from the first adder, the exponent logic checks the result for overflow or underflow of the floating-point range, summarizes the special cases, and sets the appropriate special case flags.

Xj + Xk + 60 (OCTAL) ADDER

The Xj + Xk + 60 adder is a static network that sums the two exponents in ones complement mode during the second and third clock periods of instruction execution. For purposes of this discussion, the exponent portion of the multiply operand (bits 48 through 58) is referred to as bits 0 through 10.

The add network for bits 0 through 3 and 6 through 10 is a two-input adder that sums Xj and Xk. Bits 4 and 5 of the add network, however, have three inputs to allow the single-precision correction to be made. The following example illustrates the adder inputs.

<u>Bit</u>	<u>10</u>	<u>9</u>	<u>8</u>	<u>7</u>	<u>6</u>	<u>5</u>	<u>4</u>	<u>3</u>	<u>2</u>	<u>1</u>	<u>0</u>
Xj	1	1	1	1	1	0	1	1	1	0	1
Xk	1	1	1	1	1	1	0	1	1	0	0
Correction						1	1				

Using these inputs, the first half of the adder forms pseudo sums and carries. The second half of the adder uses the pseudo sums and carries to determine enables and carries for three-bit groups. The add network resolves the enables and carries to form the complemented result.

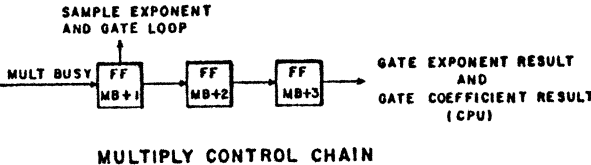
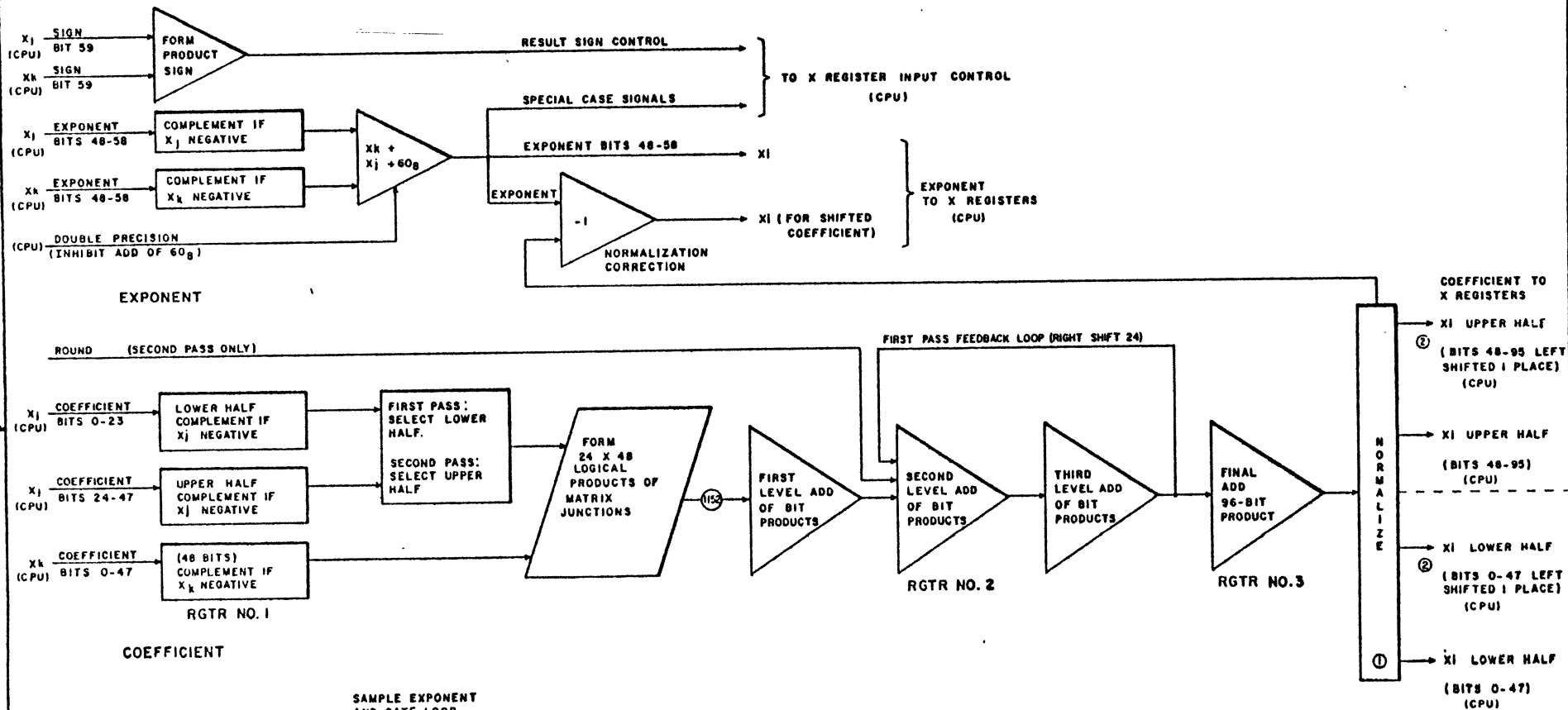
MINUS ONE NETWORK

The minus one network subtracts one from the output of the first adder by adding one to the complement. This add network takes into account any end carry condition and also makes carry propagation checks. Bit 10 is complemented to add the exponent bias value. The complemented output of this add network (true output) is gated to the X register when the coefficient has been left-shifted one place to normalize it.

An alternate network, with no minus one correction, is used when the coefficient is already normalized or cannot be normalized by a left-shift of one. This alternate network merely forms the exponent bias and complements the output so that the true exponent goes to the X register.

CONTROL

Multiply busy is the primary control signal in the multiply unit. It is set in the CPU at the end of the clock period in which a floating-multiply instruction issues from the CIW register. The flag is copied to other register ranks as data is processed by the unit. The flag blocks input to the multiply unit in the clock period immediately following instruction issue. This flag and copies of it control data movement through the multiply unit.



NOTE:

- ① ALL OPERANDS ARE TREATED AS POSITIVE. IF RESULT IS TO BE NEGATIVE, THE RESULT IS COMPLEMENTED INTO THE X REGISTER BY THE X REGISTER INPUT CONTROL.
- ② IF BOTH OPERANDS WERE NORMALIZED AND THE RESULT IS NOT NORMALIZED A LEFT SHIFT OF 1 IS PERFORMED. THIS WILL PRODUCE A NORMALIZED RESULT.

SECOND LEVEL ADD

The second level add modules receive 788 pseudo sum and pseudo carry bits from the first level adder and continue to sum groups of bits to form pseudo sum and pseudo carry bits for each bit position of the result. The resulting bits are held in registers for 1 clock period.

On the first pass through the second level add, the only bits summed are from the first pass through the 4GB7 and 4GC7 modules. However, on the second pass through the second level add, loop bits from the first pass through the 4GK7, 4GL7, 4GM7, and 4GN7 modules are merged with the pseudo sum and pseudo carry bits from the second pass through the 4GB7 and 4GC7 modules. The hardware for both passes through the second level adder is the same, except that the 4GJ7 modules are not used during the first pass. The modules containing logic for the second level add are 4GJ7, 4GI7, 4GD7, 4GE7, 4GF7, 4GG7, 4GH7, 4GK7, 4GL7, 4GO7, and 4GP7. The 4GK7, 4GL7, 4GO7, and 4GP7 modules also contain logic for the third level add.

On the 4GE7 and 4GF7 modules, reference is made to duplicate bits. These bits are received from fanouts on the 4GB7 and 4GC7 modules. The duplicate bits provide a means of generating a pseudo carry to bit N from bit N-1. The pseudo sum for bit N-1 is generated by the next lower module. Since separate modules generate the pseudo sum and pseudo carry bits, duplication of the input bit is necessary.

FIRST PASS

During the first pass (second clock period), the 4GD7, 4GE7, 4GF7, 4GG7, 4GH7, and 4GI7 modules receive pseudo sums and carries and sum these bits in groups to form pseudo sums and carries for each bit position of the lower part of the product (bits 0 through 70). The resulting pseudo sums and carries are held in registers for use during the third clock period. In the beginning of the third clock period, add logic on the same modules further sums the pseudo sums and carries before sending them to the 4GK7, 4GL7, 4GM7, 4GN7, 4GO7, and 4GP7 modules for the third level add.

The 4GJ7 modules are not used during the first pass.

SECOND PASS

During the second pass (third clock period), the 4GD7, 4GE7, 4GF7, 4GG7, 4GH7, 4GI7, and 4GJ7 modules merge 141 loop bits (bits 0 through 70) from the first pass with pseudo sums and carries of the upper half of the product (bits 24 through 94). The loop bits come from the 4GK7, 4GL7, 4GM7, 4GN7, 4GO7, and 4GP7 modules and are right-shifted 24 places so that the lower 24 bits enter the 4GJ7 modules and are properly positioned for the merge. Except for the 4GJ7 modules, the pseudo sums and carries that result from the second level add are held in registers for use during the fourth clock period. In the beginning of the fourth clock period, add logic on all except the 4GJ7 modules further sums the pseudo sums and carries before sending them to the 4GK7, 4GL7, 4GM7, 4GN7, 4GO7, and 4GP7 modules for the second pass through the third level add.

The 4GJ7 modules have two loop bits entering for each bit position of the final product. The two loop bits enter registers on the 4GJ7 modules during the third clock period and are held there during the fourth clock period when the final add is performed. This part of the final add generates an enable for each of 12 bit positions, a carry for each of 9 bit positions, and a group carry and group enable for each 4 bits of the final product. The output of the 4GJ7 modules goes directly to the 4GS7 and 4GT7 modules, where the final add is completed.

ROUNDING

If rounding is specified by the multiply instruction, two copies of the round bit signal enter at bit 46 of the 4GE7 module during the second pass. These two bits generate a carry into bit 47 of the product during the fourth clock period. The round bit signals come from the 3KV7 module of the multiply exponent logic during the third clock period of instruction execution.

THIRD LEVEL ADD

The third level add modules receive pseudo sum and pseudo carries from the second level adder and continue to sum groups of bits to form a pseudo sum and carry for each bit position of the result. The modules used for the third level add are 4GK7, 4GL7, 4GM7, 4GN7, 4GO7, and 4GP7. The 4GK7, 4GL7, 4GO7, and 4GP7 modules contain registers that hold some of the pseudo sum, pseudo carry, and loop bits. All the other pseudo sum and carries received by these modules were held in registers on the 4GD7, 4GE7, 4GF7, 4GG7, 4GH7, and 4GI7 modules.

On the 4GM7, 4GN7, 4GO7, and 4GP7 modules, reference is made to duplicate bits. These bits are received from fanouts on the second level add modules. The duplicate bits provide a means of generating a pseudo carry to bit N from bits N-1 and N-2. The pseudo sums for bit N-1 are generated on the next lower module as bit N + 2 pseudo sums. Since separate modules generate the pseudo sums and carries, duplication of the input bits is necessary.

FIRST PASS

The first pass through these modules takes place during the third clock period. The third level add further reduces the results of the first pass through the matrix to a pseudo sum and pseudo carry (loop bits) per bit position of the lower part of the product. These 141 bits loop back to the 4GJ7, 4GK7, 4GL7, 4GI7, 4GD7, 4GF7, 4GG7, and 4GH7 modules to be added with the results of the second pass through the matrix. An enable loop signal from the 3KV7 module enables the loop from the 4GM7, 4GN7, 4GO7, and 4GP7 modules during the third clock period and blocks the loop on all other clock periods.

SECOND PASS

The second pass through the third level add takes place during the fourth clock period. At this time, the third level add further sums the entire product, the results of both passes through the matrix, until only two bits of data remain in each bit position of the upper part of the double-precision sum (bits 24 through 95). These 141 pseudo sum and carry bits are delivered to the 4GS7, 4GR7, and 4GQ7 modules for the final add.

FINAL ADD - LOWER HALF

The final add is performed in two parts. The first half of the final add sums the pseudo sums and pseudo carries and sends the resulting carries and enables to the second half of the final add. The second half of the final add uses a standard pass and carry check method of carry propagation. Both halves of the final add constitute a full adder. A full adder is not used in the earlier stages of the multiply unit because of the extra time involved in checking enables and carries.

The first half of the final add is located on the 4GJ7, 4GS7, 4GR7, and 4GQ7 modules. The output of the 4GJ7 modules goes to the two 4GT7 modules, where the second half of the final add for bits 0 through 23 takes place during the fourth clock period.

During the fourth clock period, the 4GS7 module generates enables for bits 24 through 31 by performing an exclusive OR of the pseudo sum and carry bits input for each bit position. These enables are held in registers during the fifth clock period when they are used by the 4GV7 modules for the second part of the final add.

The 4GS7 module generates carries for bits 25 through 31 during the fourth clock period by performing an AND of the pseudo sum and carry bits input for each bit position. The enable and carry bits for bits 24 through 31 of the final product go to a 4GV7 module.

During the fifth clock period, the 4GS7 module uses the bit enables and carries to form a section carry and a section enable. The eight bits of the final product that are handled by each module constitute a section. The section carry and section enable generated are fanned out to the 4GU7, 4GV7, 4GW7, and 4GX7 modules of the coefficient hardware and to the 4KU7 module of the exponent hardware for coefficient left-shift determination.

FINAL ADD - UPPER HALF

FIRST HALF

The first half of the final add is located on the 4GJ7, 4GS7, 4GR7, and 4GQ7 modules. During the fourth clock period, the 4GR7 and 4GQ7 modules generate enables for bits 32 through 94 by performing an exclusive OR of the pseudo sum and pseudo carry bits input for each bit position. These enables are held in registers during the fifth clock period when they are used by the 4GW7 and 4GX7 modules for the second part of the final add.

The 4GR7 and 4GQ7 modules generate carries for bits 33 through 95 during the fourth clock period by performing an AND of the pseudo sum and carry bits input for each bit position. The enable and carry bits for bits 32 through 95 of the final product go to the 4GV7, 4GW7, and 4GX7 modules.

During the fifth clock period, the 4GR7 and 4GQ7 modules use the bit enables and carries to form a section carry and a section enable. The eight bits of the final product that are handled by each module constitute a section. The section carry and section enable generated are fanned out to the 4GU7, 4GV7, 4GW7, and 4GX7 modules of the coefficient hardware and to the 4KU7 module of the exponent hardware for coefficient left-shift determination.

The section carry generated by the 4GQ7 module includes special circuitry that prevents the final result from being left-shifted one place when neither of the source operands was normalized. One of the two bits entering the 4GQ7 module for bit 94 is the pseudo sum for bit 94. This bit is the result of the logical product of bit 47 of the two source operands. Neither the 4GC7 module that formed the logical product nor the 4GP7 module has modified this product

other than to call it pseudo sum 94. Assuming that the two source operands were normalized, this bit should be a one, since it would then be the logical product of two one bits. If it is not a one, the two source operands were not normalized. The 4GQ7 module uses the complement of pseudo sum 94 to set the section carry bit and thereby prevents left-shifting the final result when the two source operands are not normalized.

SECOND HALF

The second half of the final add is performed on the 4GT7, 4GU7, 4GV7, 4GW7, and 4GX7 modules. The 4GW7 and 4GX7 modules form the final product of bits 48 through 95. The input of these modules is an enable and a carry for each bit position and an enable and a carry for each section. The result is the complement of the final sum and is complemented again as it is gated to the X registers by the gate upper half signal. This recomplements the final product so that the X registers receive the positive value of the result. If the result is negative, it is complemented by the X register input control (5CB7 module) before it enters the destination X register.

LEFT SHIFT NETWORK

Before gating the final result, the 4GU7 and 4GV7 modules determine whether to left-shift the result by one to normalize it. If bit 95 is a one, shifting is not performed. Bit 95 is a one when the multiply process has generated a carry to it or it is forced to a one when one or both of the multiply source operands are normalized. If bit 95 is a zero, the result is not normalized and the hardware left-shifts it one to normalize it. Bit 95 is a zero only when both source operands are not normalized and there was no carry into bit 95.

EXPONENT AND CONTROL

EXPONENT ARITHMETIC

EXPONENT INPUT REGISTERS

The exponent input registers, located on the 4KR7 module, receive bits 48 through 59 of the multiply operands from the 4RE7 module during the first clock period. If the coefficient sign (bit 59) is negative, the associated exponent is complemented to obtain the true value. Bit 59 is then removed and is sent to the 3BA7 module. The two resulting 11-bit exponents are sent to the 4KS7 module. From this point on, bits 48 through 58 are referred to as bits 0 through 10.

ADD NETWORK

During the second clock period, the exponent logic on the 4KS7 module submits the complement of bits 0 through 9 of both exponents to the add network. Exponent bias is removed by not complementing bit 10.

The add network sums the two exponents in a 13-bit ones complement mode. Depending upon the instruction mode, this network also adds a third quantity. If the multiply double-precision flag is set, this third quantity is a zero. If the multiply double-precision flag is cleared, the third quantity is 60 (octal) (+48 decimal).

First Half

The first half of the add network forms pseudo sums for bits 0 through 3 and 6 through 10 by performing an equivalence of the two exponents. For these bits, a pseudo sum is generated when the corresponding bits of the two exponents are equal.

Bits 4 and 5 are handled by a three-input adder to allow the single-precision correction to be made. The add network performs an exclusive OR of the exponents for bits 4 and 5. Then, depending upon whether double precision is selected or not, it performs an exclusive OR of the result and the double-precision correction signal received from the 3KV7 module during the second clock period.

The 4KS7 module also generates pseudo carries. The add network forms pseudo carries to bits 1 through 4 and 7 through 11 by performing an OR of the corresponding bits of the two exponents. Pseudo carries to bits 5 and 6 are handled by a three-input network to allow the single-precision correction to be made. When the pseudo sums and pseudo carries are half-added, the result is 60 (octal) added to the exponent if single precision is selected.

Second Half

The 3BA7 module combines the pseudo sums and carries from the 4KS7 module to determine which stages are enables and which stages have carries coming into them.

A bit enable is produced by an exclusive OR of the pseudo sum and pseudo carry coming into that stage. Bit and group carries are produced by an AND of the pseudo sum and pseudo carry into a stage or an AND of an enable and carry generated by the next lower stage.

The add network uses the resulting bit enables, bit carries, and group carries to perform carry propagation checks until each bit of the sum is represented by an enable and a carry. An exclusive OR of these two bits produces the sum in complemented form. This complemented result is fed to the 4KU7 module.

PRODUCT SIGN

The 3BA7 module generates the product sign by performing an exclusive OR of the sign bits received from the 4KR7 module.

SPECIAL CASE CHECKS

The 4KS7 module receives the true values of the two operand exponents from the 4KR7 module during the second clock period and examines them for overflow, underflow, or indefinite conditions. Overflow of the floating-point range is indicated by an operand exponent value of 3777 in packed form, the largest exponent value that can be represented in floating-point format. Underflow of the floating-point range is indicated by 0000 in packed form, the smallest exponent value that can be represented in floating-point format. An indefinite condition is indicated by a minus zero exponent, 1777 in packed form. The exponent logic looks for each of these conditions and sends an overflow, underflow, or indefinite condition indicator for each operand exponent to the 3BA7 module.

The 3BA7 module summarizes the special cases and examines the final result for overflow or underflow of the floating-point range.

INDEFINITE

An indefinite signal is sent to the 3KV7 module when one of the following conditions exists.

One or both of the operands have an indefinite exponent.

One operand has an underflow exponent and the other operand has an overflow exponent.

OVERFLOW

An overflow signal is sent to the 3KV7 module when one of the following conditions exists.

One operand has an overflow exponent and the other operand has a normal exponent.

Both operands have overflow exponents.

The unpacked exponent of the result (sensed prior to reducing it by one when a left-shift is performed) is greater than +1777 (octal).

UNDERFLOW

An underflow signal is sent to the 3KV7 module when one of the following conditions exists.

One operand has an underflow exponent and the other operand has a normal exponent.

Both operands have underflow exponents.

The unpacked exponent of the result (sensed prior to reducing it by one when a left-shift is performed) is less than -1776 (octal).

SPECIAL CASE

A special case signal is sent to the 3KV7 module if any of the previous special conditions exist.

INDEFINITE-1

An indefinite-1 signal is sent to the 3KV7 module if one or both operands have an indefinite exponent. The indefinite bit is set in the exit condition register (CPU) for this case.

INFINITE

An infinite signal is sent to the 3KV7 module if one or both operands have an overflow exponent. The infinite bit is set in the exit condition register (CPU) for this case.

INTEGER MULTIPLY

If both operands have underflow exponents and the operand coefficients are not both normalized, an integer multiply is performed.

NOTE

If an integer multiply is performed with one coefficient normalized, an undetected overflow result may occur.

An integer multiply forces the second half and result bits 0, 4, and 5 to ones and forces bit 10 (bias bit) to a zero. This ensures that all zero bits are sent to the Xi register for the exponent portion of the result. The complement of the exponent portion is sent to the Xi register if the sign of the result is negative.

RESULT REGISTER

The exponent result register on the 4KU7 module receives the complemented result of the exponent add from the 3BA7 module during the fourth clock period. The result is used by two separate output networks.

One output network subtracts one from the output of the first adder by adding one to the complement. This network, called the minus one network, gates the exponent to the X register when the coefficient result has been left-shifted one place to normalize it. The other network, which does not have a minus one correction, gates the exponent when the coefficient result is not shifted. Both output networks complement bit 10 to add the exponent bias and then complement the entire output to form the true output.

The 4KU7 module determines whether or not the coefficient was shifted by checking for a carry into bit 95 using the section carries and enables from the 4GT7, 4GS7, 4GR7, and 4GQ7 modules. If no carry was generated into bit 95 (forced or otherwise), the coefficient was shifted and the network with the minus one correction is gated out.

Gating of the exponent from either network occurs upon receipt of the gate exponent signal from the 3KV7 module during the fifth clock period.

MULTIPLY CONTROL LOGIC

The multiply control logic for both the exponent and coefficient arithmetic of the floating-multiply unit is located on the 3KV7 module. The multiply busy flag, which is received from the 4LE7 module, controls data movement through the multiply unit. The multiply busy flag on the 4LE7 module sets at the end of the first clock period.

Consequently, multiply busy on the 3KV7 module sets during the second clock period and is clear on all others. Similarly, the multiply busy signal on the 3KV7 module is a one during the first clock period and a zero during the second clock period. A timing chain and the multiply busy signal control the output of following signals.

ROUND FLAG

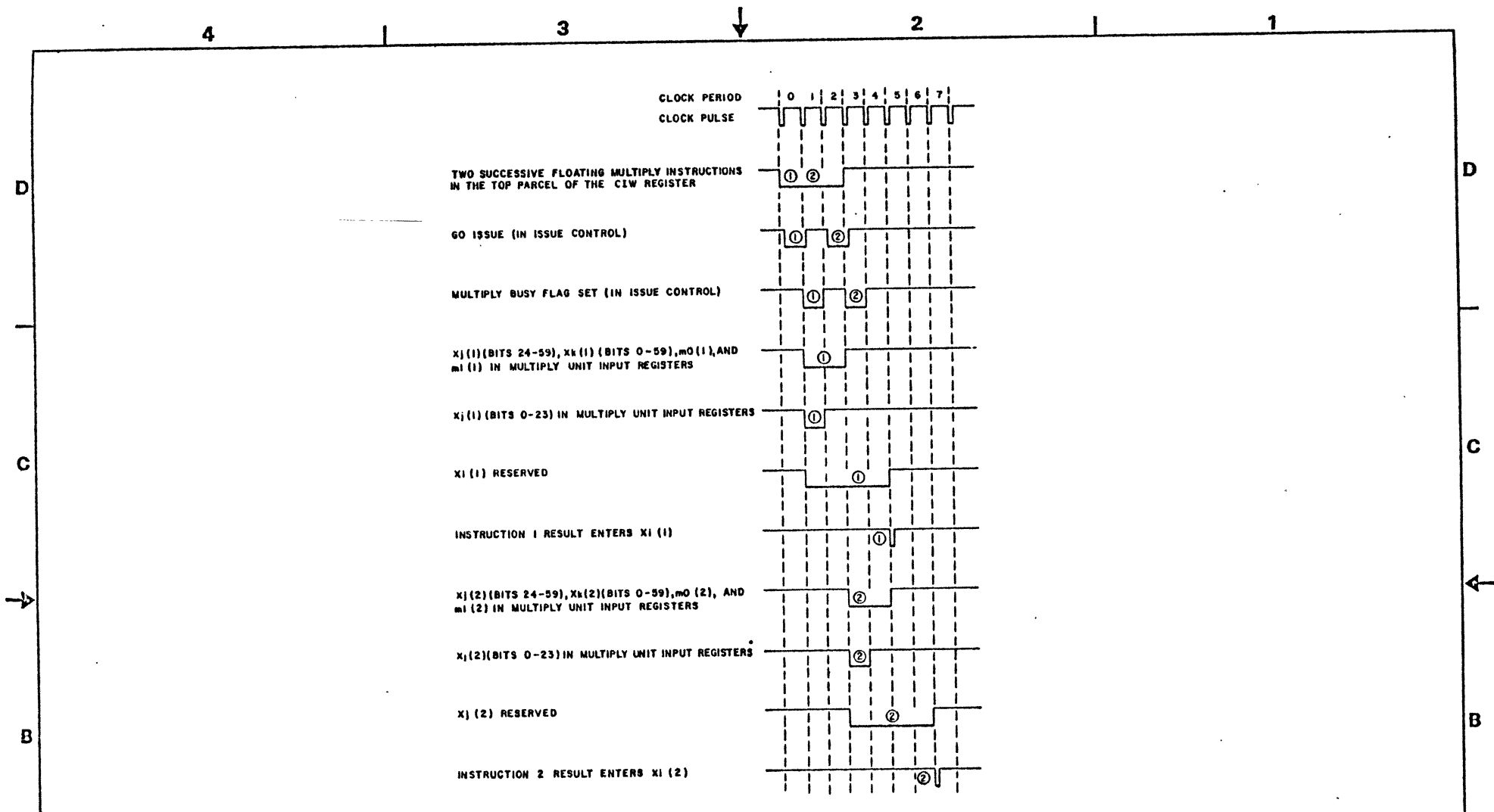
Two copies of the round flag cause the addition of a round bit to bit 49 of the double-precision coefficient during the third clock period of instruction execution. The round flag sets during the first clock period of execution when rounding is specified by the multiply instruction. This flag is copied to another register rank for use during the third clock period when two copies of the flag are sent to coefficient bit 45 on a 4GE7 module. A carry bit, generated by these two bits, rounds bit 46 of the coefficient.

DP CORRECTION

The DP correction flag sets during the first clock period when double precision is specified by the multiply instruction and sent immediately to the 4KS7 module to control the exponent arithmetic. This flag is also copied to other register ranks on the 3KV7 module until the fifth clock period when it gates the lower half of the double-precision coefficient. The complement of this signal gates the upper half of the double-precision coefficient.

ENABLE LOOP

The enable loop signal, conditioned by multiply busy, gates the loop bits from the first pass through the coefficient logic during the third clock period. Looping is blocked at all other times to prevent the possibility of bits from the second pass through the matrix from interfering with the execution of a new multiply instruction that may have entered the multiply unit.



NOTES:

- ① INSTRUCTION 1 - ASSUME NO ISSUE CONFLICTS.
- ② INSTRUCTION 2 - ASSUME THAT ONLY ISSUE CONFLICT IS THAT THE MULTIPLY UNIT IS BUSY. THUS, ISSUE IS DELAYED ONE CLOCK PERIOD.



TIMING DIAGRAM
MULTIPLY INSTRUCTION

CODE IDENT 34010	DOC NO 60420300	REV A
DEVELOPMENT DIVISION		PAGE 5-7-79

PART 8

DIVIDE UNIT

DIVIDE UNIT

The floating-divide unit executes the two CPU instructions, floating divide (44) and round floating divide (45). These instructions direct the computer to divide X_j by the divisor X_k and send the quotient to X_l . This unit involves a 17-step iterative process to form the quotient from the two operands. Only one divide instruction may be executed in the iterative portion of the divide unit at a given time.

The divide instructions require 20 clock periods for execution. Data moves from the operating registers to the divide unit input registers each clock period in which the divide busy flag is clear. The data is used for instruction execution only if a divide instruction issues from the CIW register and sets the divide busy flag. The data which arrives at the divide unit during the clock period of instruction issue is then used in the execution of the following divide sequence. The divide busy flag prevents the CIW from issuing another divide instruction for the 17 clock periods following instruction issue. However, in the 18th clock period after instruction issue, a second divide instruction may issue.

Bit 0 of the m designator is held in an input register along with the operand data in X_j and X_k . This bit is the divide round flag which distinguishes between the two instruction modes.

The divide unit operates on positive coefficient values only. Each coefficient for X_j and X_k is individually complemented in static networks if its sign is negative. The sign of the result is determined by the divide unit and is sent to X register input control in the CPU. This sign bit is the logical difference of the two operand sign bits.

The divide busy flag initiates a chain of divide sequence control flags which sequence the steps in the instruction execution. This chain controls the sequence of events for the 19 clock periods following the issue of the divide instruction. These static conditions then control the data movement within the divide unit and the data transmission of the X register input path at the end of the divide sequence.

The format of a floating-point number is $k2^n$, where k is a 48-bit integer coefficient and n is a 10-bit integer exponent.

Division of floating-point numbers requires the subtraction of the exponents and the division of two 48-bit coefficients. Double-precision division is not provided and a remainder cannot be retrieved. Since the divide hardware produces a quotient in the range, 1.7777 7777 7777 7777 through 0.0000 0000 0000 0000, the ratio of X_j to X_k must always be less than 2 to 1. If the divisor is normalized, this requirement is always met. If this requisite is not met, the resulting quotient is meaningless. If both the dividend X_j and the divisor X_k are normalized, the quotient X_l is also normalized.

The quotient coefficient is formed three bits per clock period in 17 iterative steps. The result of the first iteration is stored in the control module. The results of the 16 succeeding iterations (a total of 48 bits) are stored in the quotient shift register. If the quotient is of the form, 0.X-----X (the ratio of X_j to X_k is less than 1 to 1), the result of the first iteration is a zero, and the 48 bits in the quotient shift register are taken as the coefficient of the result.

Given: $X_l = X_j/X_k$
 $(X_j) = 2057\ 4400000000000015$
 $(X_k) = 2032\ 6000000000000000$

The divide coefficient logic forms:

$\frac{4400000000000015}{6000000000000000} = 0.6000000000000021$

In floating-point format, the coefficient must be an integer. Therefore, the binary point must be moved 48 places to the right. Since the exponent must be decreased by one for every place, the binary point is right-shifted, 60 (octal) is subtracted from the difference of the exponents, and the final result is:

$(X_l) = 1744\ 6000000000000021$

[Final exponent = 2057 - 2032 - 60 = -33 (unbiased) = 1744]

If the quotient is of the form, 1.X-----X (the ratio of Xj to Xk is 1 to 1 or greater, but less than 2 to 1), the result of the first iteration is a one, and the upper 47 bits in the shift register are interpreted as bits 0 through 46 of the quotient coefficient. The X register access control interprets bit 47 as a one.

Given: $X_l = X_j/X_k$
 $(X_j) = 2016\ 7000000000000000$
 $(X_k) = 2025\ 4000000000000000$

The divide logic forms:

$$\frac{7000000000000000}{4000000000000000} = 1.6000000000000000$$

In order to make this quantity an integer, the binary point must be moved 47 places to the right. Therefore, 57 (octal) is subtracted from the difference of the exponents and the final result is:

$(X_l) = 1711\ 7000000000000000$

[Final exponent = 2016 - 2025 - 57 = -66 (unbiased) = 1711]

If the ratio of Xj to Xk is 2 to 1 or greater, the result of the first iteration is forced to a two by the release remainder control signal. The control network then sends a special case signal (indefinite) to the X register input control network along with an operand consisting of all ones. The X register input control network then generates an indefinite result.

The quotient coefficient is formed three bits per clock period. The iteration network functions exactly like a pencil and paper octal divide. A multiplication network forms seven multiples of the divisor, Xk through 7Xk. The dividend is entered into the remainder register, which holds 51 bits (initially bits 48, 49, and 50 are all 0). The trial subtraction network simultaneously compares each of the seven divisor multiples with the contents of the remainder register. The largest multiple that is smaller than the remainder is subtracted from the remainder. The resulting quantity is left-shifted three binary (one octal) positions and entered into the remainder register. The number of the multiple chosen (pick number) becomes the first quotient digit. The pencil and paper method is:

	0.6000 0000 0000 0021
6000 0000 0000 0000	4400 0000 0000 0015.0000 0000 0000 0000
	0000 0000 0000 0000
	4400 0000 0000 0015 0
	4400 0000 0000 0000 0

The same division as performed by the functional unit:

Dividend (Xj) = 4400 0000 0000 0015

Divisor (Xk) = 6000 0000 0000 0000

	15 0000 0000 0000 000
	14 0000 0000 0000 000
	1 0000 0000 0000 0000
	6000 0000 0000 0000
	2000 0000 0000 0000

The multiplication network forms:

Xk = 6000 0000 0000 0000
2Xk = 1 4000 0000 0000 0000
3Xk = 2 2000 0000 0000 0000
4Xk = 3 0000 0000 0000 0000
5Xk = 3 6000 0000 0000 0000
6Xk = 4 4000 0000 0000 0000
7Xk = 5 2000 0000 0000 0000

The dividend enters the remainder register.

RMDR = 0 4400 0000 0000 0015

Initially, the quotient shift register has contents: XXXX XXXX XXXX XXXX

The 17 iterative operations are performed as follows:

1. Compare RMDR versus Xk through 7Xk:
RMDR is smaller than any of them.
Enter octal digit 0 in shift register:
Quotient = XXXX XXXX XXXX XXX0 (the quotient is left-shifted one octal digit each clock period).
Enter chosen digit (in this example, a 0) in control module this iteration only.
Enter RMDR-0 and left-shift three:
RMDR = 4 4000 0000 0000 0150 ← becomes a 2525 pattern if instruction 45
↑
enter 0 since left-shift three

2. Compare RMDR versus Xk through 7Xk: $6Xk \leq RMDR < 7Xk$
Enter second octal digit: QUOT = XXXX XXXX
XXXX XX06

Enter RMDR-6Xk and left-shift three: RMDR =
0 0000 0000 0000 1500

3 through 14 In the next 12 iterations, an octal digit 0 is picked, and the quotient and remainder are left-shifted three binary places each clock period.

RMDR = 0 1500 0000 0000 0000
QUOT = XX06 0000 0000 0000

15. Compare RMDR versus Xk through 7Xk: RMDR < Xk

Enter octal digit 0: QUOT = X060 0000 0000 0000
Enter RMDR-0: RMDR = 1 5000 0000 0000 0000

16.

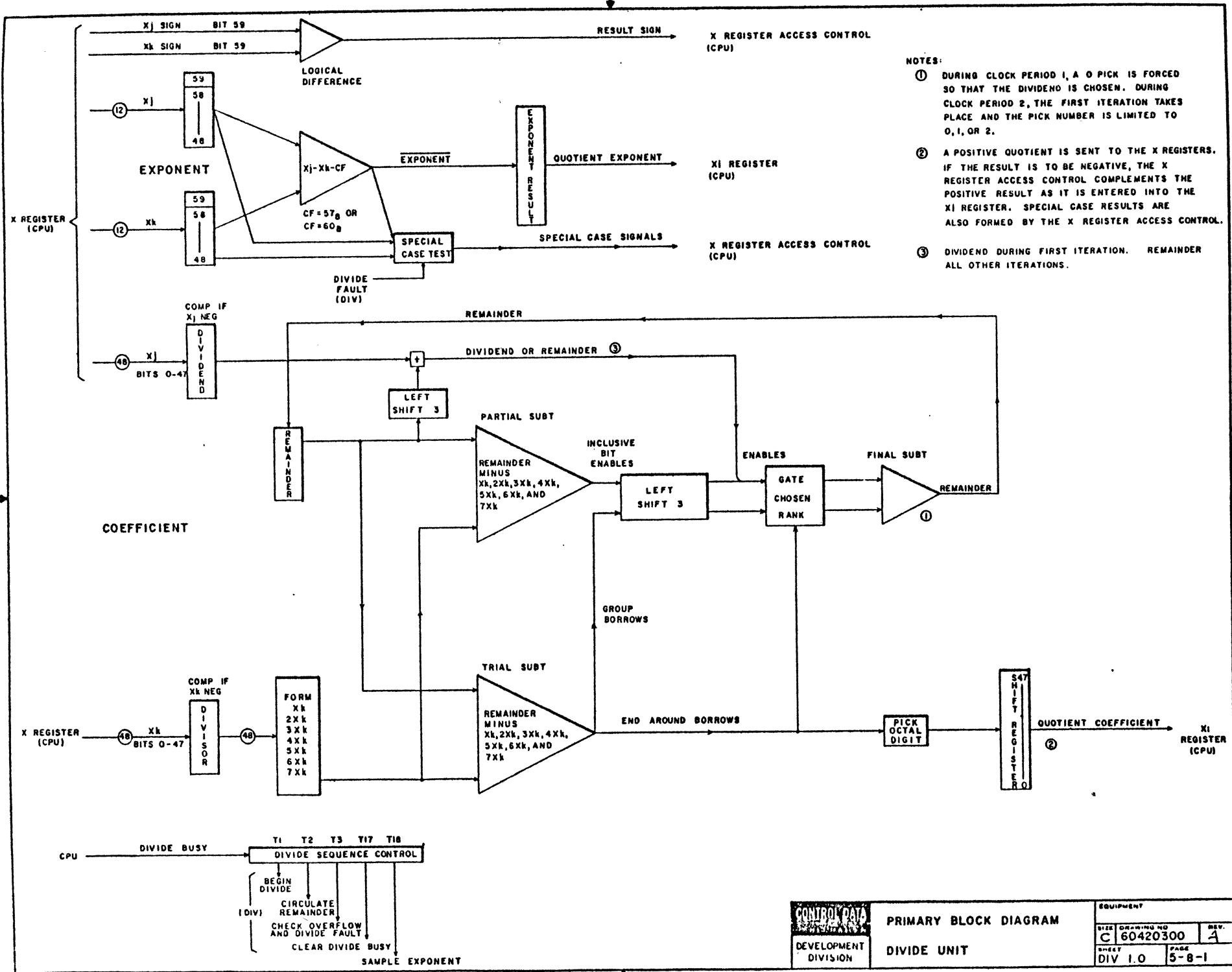
Compare RMDR versus Xk through 7Xk: $2Xk \leq RMDR < 3Xk$

Enter octal digit 2: QUOT = 0600 0000 0000 0002
Enter RMDR-2Xk: RMDR = 1 0000 0000 0000 0000

17.

Compare RMDR versus Xk through 7Xk: $Xk \leq RMDR < 2Xk$

Enter octal digit 1: QUOT = 6000 0000 0000 0021
Discard remainder and transmit quotient to Xi



EXPONENT, OUTPUT, AND CONTROL

EXPONENT MANIPULATION

INPUT REGISTERS

The data input registers for both operand exponents, bits 48 through 59, are on the 4KR7 module. The data that is in the registers when the divide busy flag sets is held for the 17 clock periods during which the flag remains set. Xj bit 59 and Xk bit 59 are sensed and the exponents (bits 48 through 58) are individually complemented if they are negative. The output goes to the 4DO7 module (bits 48 through 58) and the 3DP7 module (bit 59).

EXPONENT FORMATION

The exponent subtraction occurs while the coefficient is being formed. The quotient exponent is held until the quotient coefficient is completely assembled. The exponent subtraction network, comprised of the preliminary and final addition networks, is on the 4DO7 and 3DP7 modules. Instead of subtracting the divisor exponent and the correction factor from the dividend exponent, the dividend exponent is complemented and added to the divisor exponent and to the correction factor (CF). The CF is determined in CP04 by bit 0 (overflow bit) of the first quotient coefficient digit. The resulting sum (exponent) is complemented on output by the 4DS7 module to obtain the quotient exponent. In effect, the exponent network forms minus $(-X_j + X_k + CF)$ instead of $(X_j - X_k - CF)$.

The exponent addition is performed in two halves. The network that performs the preliminary exponent addition is on the 4DO7 module; the final exponent addition network is on the 3DP7 module. The bias is removed from both exponents on input to the 4DO7 module. The exponents and the correction factor are then added. If the overflow bit from the 4DQ7 module, bit 0 of the first iteration, is a zero, the quotient is of the form 0.XXXX XXXX XXXX XXXX, and the correction factor is 60 (octal). If bit 0 of the first iteration is a one, the quotient is of the form 1.XXXX XXXX XXXX XXXX, and the correction factor is 57 (octal).

The result of the first half addition is a bit carry and a bit enable for each position. These bits are transferred to the final exponent addition network on the 3DP7 module.

The final addition network combines the carries and enables in a 13-bit mode. The two extra bits are formed by sign-extending the exponents. The upper three bits of the 13-bit sum are sensed to detect special case conditions. Exponent bias is added, and the lower 11 bits of the sum are then transferred to the output network on the 4DS7 module.

The sign of the quotient is formed by a network on the 3DP7 module and sent to the X register input control. This sign bit is the logical difference of the two operand sign bits. A network on the 4DO7 module checks for operand exponent overflow, underflow, and indefinite exponents. The rest of the special case checks are performed on the 3DP7 module. If any special case exists, the special case signal to the 4DQ7 module is a one, in which case, a quotient of all ones is sent to the X register. The three signals to the X register input control specify which, if any, of the special case conditions exist.

SPECIAL CASE

A number of special cases are treated in the floating-divide unit. If any special case conditions exist, the special case signal from 3DP7 to the 4DQ7 module becomes a one. When the go divide special case signal from the 4DQ7 module to the X register input control becomes a one, the specific condition is indicated by the exponent indefinite, overflow, and underflow signal from the 3DP7 module. In any special case condition, the transmission of data from the divide unit output registers to the destination X register is blocked. The gate quotient signal from 4DQ7 to 4DS7 remains a zero, forcing the exponent output registers to transmit all ones. The gate output signal (which is the gate quotient signal delayed 1 clock period) forces the coefficient output registers to transmit all ones. Thus, a quotient of all ones is delivered to the X register and the input control forms the particular special case word. The special condition flags cause the appropriate bit to be set in the exit condition register.

If no special case conditions exist, the special case signal from 3DP7 to 4DQ7 remains a zero. The go divide special case signal to the X register input control is a zero and the gate divide and the gate quotient signals are ones. The gate quotient signal gates the quotient output.

SPECIAL CASE OPERANDS

One category of special case conditions involves overflow, underflow, or indefinite operand values. These situations are sensed in the 4DO7 module. The combination of special operand values which cause specific results are sensed on the 3DP7 module. If either operand is indefinite, or if both operands are indefinite, the result is indefinite. The operand coefficients are ignored in this case, and the resulting word delivered to the Xi register is positive indefinite with a zero coefficient. The indefinite bit is set in the exit condition register (CPU) for this case.

If either operand has an overflow exponent, or if both operands have overflow exponents, the result is infinite. The infinite bit is set in the exit condition register (CPU) for this case.

If Xj has an overflow exponent and Xk is in floating-point range or has an underflow exponent, the result is a complete overflow word delivered to the Xi register. The coefficients of the operands are ignored in this case, and the result is a zero coefficient. The sign of the result is calculated in the same manner as for operands in range.

If Xj has an underflow exponent and Xk is in floating-point range or has an overflow exponent, the result is a complete underflow word delivered to the Xi register. The coefficients of the operands are ignored in this case, and the result is a zero word.

If Xk has an overflow exponent and Xj is in floating-point range, the result is a complete underflow word delivered to the Xi register. The coefficients of the operand are ignored in this case, and the result is a zero word.

If Xk has an underflow exponent and Xj is in floating-point range, the result is a complete overflow word delivered to the Xi register. The coefficients of the operands are ignored in this case. The sign of the result is calculated in the same manner as for operands in range.

The combination of operand exponents of overflow divided by overflow and underflow divided by underflow results in a positive indefinite word delivered to the Xi register.

SPECIAL CASE QUOTIENT

A second category of special cases occurs if there is an underflow or an overflow of the floating-point exponent range during the exponent calculation. In these cases, the special case signal is sent to the X register input control, and the output from the divide unit is blocked in the same manner as for the special case operands.

A complete overflow occurs for this instruction whenever the exponent computation results in an exponent greater than plus 1777 (unbiased). If any combination of operand exponents causes an indefinite condition, the overflow situation is ignored. Otherwise, this situation is sensed as a special case, and a complete overflow word with proper sign is delivered to the Xi register. The coefficient calculation is ignored in this case.

A complete underflow occurs for this instruction whenever the exponent computation results in an exponent less than minus 1777 (unbiased). If any combination of operand exponents causes an indefinite condition, this underflow situation is ignored. Otherwise, this situation is sensed as a special case, and a complete zero word is delivered to the Xi register. The coefficient calculation is ignored in this case.

SPECIAL CASE: DIVIDE FAULT

A third special case category occurs if the dividend coefficient is larger than the divisor by a factor of two or more. This is a divide fault situation which can occur if the divisor is not normalized. The initial trial subtraction in the coefficient calculation results in an octal digit with a value of two. If an overflow or underflow condition does not exist, an indefinite condition result is indicated for this case, and it is treated in the same manner as the other special cases.

If an overflow or an underflow condition caused by an underflow or infinite operand does exist, the divide fault situation is ignored.

PARTIAL OVERFLOW OR UNDERFLOW

A partial overflow occurs for this instruction whenever the exponent computation results in exactly plus 1777 (unbiased). The result is delivered to the Xi register in a normal manner. Subsequent use of this result as an operand in a floating-point unit may, however, result in overflow detection.

A partial underflow occurs for this instruction whenever the exponent computation results in exactly minus 1777 (unbiased). The result is delivered to the Xi register in a normal manner. Subsequent use of this result as an operand in a floating-point unit may, however, result in underflow detection.

OUTPUT

The exponent output register is on the 4DS7 module. The complemented 11-bit biased quotient exponent is delivered from the 3DP7 final addition network. In CP18, a gate quotient signal from the 4DQ7 module enables a clock gate so that the exponent enters the output register. The exponent is complemented on output because the exponent network forms the complement of the quotient exponent,

Since a positive quotient is delivered to the X register, bit 59 is entered as a zero. The absence of a gate quotient signal in CP18 indicates that one of the special case conditions exists, in which case, an exponent of 12 ones is delivered to the X register.

The quotient coefficient is assembled three bits per clock period in the 48-bit shift register. In each of CP03 through CP19, an octal digit is delivered from 4DD7 to 4DS7. After 17 iterations, the 4DS7 module has bits 0 through 2 and 45 through 47; the two 4DR7 modules have bits 3 through 44. Every clock period, the quotient is left-shifted three places, three new quotient bits from 4DD7 enter from the iteration network, and the upper three bits are discarded. After 17 iterations, the shift register contains 48 bits of data. Bits 0 and 1 of the octal digit formed in the first iteration are stored in the 3DQ7 module. If bit 1 is a one, a divide fault condition (the dividend exceeds the divisor by a factor of two or more) exists. If this condition or any other special case condition exists, the gate quotient signal remains a zero and the delivery of the quotient is blocked. As a result, a coefficient of all ones is in the path to the X register in CP19. If bit 1 is a zero and no other special case conditions exist, the gate quotient signal becomes a one in CP18 and the quotient coefficient is delivered to the X register during CP19.

If bit 0 of the first iteration digit is a zero, the quotient is of the form 0.XXXX XXXX XXXX XXXX. In this case, a correction factor of 48 is subtracted from the exponent, the overflow bit is a zero, and the 48 bits in the shift register are delivered to the X register.

If bit 0 of the first iteration digit is a one, the quotient is of the form 1.XXXX XXXX XXXX XXXX. In this case, a correction factor of 47 is subtracted from the exponent. The overflow bit becomes a one, causing a one-bit right-shift of the coefficient output. The upper 47 bits in the shift register are delivered to the X register as bits 0 through 46. In this case, the bit 47 output is blocked. Therefore, bit 47 is entered as a one. All outputs are timed to occur in CP19.

CONTROL

The divide sequence control of the 4DQ7 module times the entire divide instruction. The clock periods on the left of the divide sequence control indicate the set times. The times on the right are the clear times.

The three remainder bits are part of the coefficient iteration network. They are used as rank 0 bit enables in the 4DD7 and 4DE7 modules.

The special case signal indicates a special case exponent. The gate divide and go divide signals go to the X register input control to gate in the appropriate data. The gate quotient signal causes the quotient to be delivered during CP19.

The begin sequence signal initiates the divide timing sequence. The divide fault signal is bit 1 of the first coefficient iteration digit. This bit is sent to the 3DP7 module where it forces a special case condition if it is a one.

The overflow bit is bit 0 of the first coefficient iteration digit. This bit goes to the 4DO7 module where it determines the correction factor. It also goes to the 4DR7 and 4DS7 modules where it determines which 48 bits of the coefficient are delivered to the X register.

The divide round flag (m bit 0) is set when a round floating divide instruction issues from the CIW register. This flag modifies the dividend in the third clock period of instruction execution. Octal digits with a value of 25252 are entered in the lowest order bits of the remainder register if this flag is set. This modification of the dividend increases the dividend value by one-third of the least significant bit in the original operand.

The circulate remainder signal goes to the four 4DI7 modules where it gates the current remainder to the final subtraction network. When this signal is a zero, the content of the dividend input register is gated to the final subtraction network.

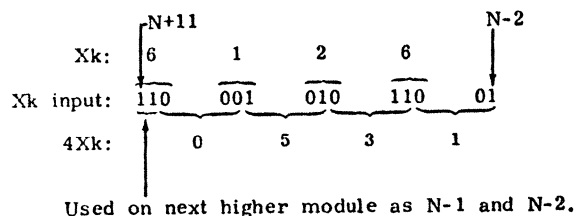
The release remainder signal goes to the 4DD7 and 4DE7 modules. It allows the iteration network to generate a digit of three through seven. When this signal is a zero, no digit greater than a two can be generated by the iteration network. This signal is a one on the 4DQ7 module from CP02 through CP17.

The divide 15 (T15) condition originates in the divide sequence control and is used in instruction issue control and in the X register access control. This condition exists during the 16th clock period of execution for a divide instruction. It is used in the instruction issue control to block issue of an instruction which would conflict with the delivery of data from the divide unit to the X register data input path. It is used in the X register access control to initiate the process of register access for the destination X register.

The clear divide busy condition originates in the divide sequence control and is used to clear the divide busy flag. This condition exists during the 18th clock period of execution for a divide instruction.

5Xk ADDER

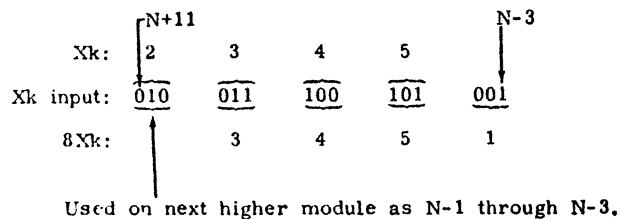
The network that forms 5Xk is on the 4DL7 and 4DN7 modules. By left-shifting Xk two places, 4Xk is formed, 5Xk is formed by adding Xk to this quantity. Thus, 5Xk is a 51-bit quantity. The 4DL7 modules generate bits 0 through 47; the 4DN7 module generates bits 48 through 50. Each 4DL7 module receives 14 Xk bits and outputs 12 5Xk bits. The 14Xk bits are left shifted two places to form bits N to N+11 or 4Xk.



Bits N-2 through N+9 are used as bits N through N+11 of 4Xk. These bits are added to bits N through N+11 to form 5Xk. Thus, 5Xk is formed by adding 4Xk to Xk. The section carry and enable bits are used in exactly the same manner as in forming 3Xk. The 4DN7 module is also used in the same manner as in forming 3Xk.

7Xk SUBTRACTER

The subtraction network that forms 7Xk is on the 4DM7 and 4DN7 modules. First, 10Xk is formed by left-shifting Xk three places. 7Xk is then formed by subtracting Xk from this quantity. The 4DM7 modules generate bits 0 through 47; the 4DN7 module generates bits 48 through 50. Each 4DM7 module receives 15 bits of Xk and outputs 12 bits of 7Xk. The 15Xk bits are left-shifted three places to form bits N through N+11 of 4Xk.



Bits N-3 through N+8 of Xk are left-shifted to become bits N through N+11 of 10Xk. Bits N through N+11 of Xk are then subtracted from these bits. Thus, 7Xk is formed by subtracting Xk from 10Xk. The section borrows and enables are used in the 4DN7 module in the same manner as the section carries and enables are used in forming 3Xk and 5Xk.

Xk, 2Xk, and 4Xk NETWORKS

The networks that form Xk, 2Xk, and 4Xk are on the 4DK7, 4DL7, and 4DM7 modules. These multiples are formed by left-shifting the 48 Xk bits. By left-shifting Xk one bit position, 2Xk is formed and has 49 bits. By left-shifting Xk two bit positions, 4Xk is formed and has 50 bits. These three multiples are supplied to the trial subtraction network and the partial subtraction network.

DIVISOR MULTIPLICATION NETWORK

The divisor (X_k) coefficient input registers are on the four 4DJ7 modules. Each module handles 12 bits. Every clock period, a 48-bit quantity from the X register arrives at the modules. The sign bit (bit 59) is delivered along with the coefficient. The registers are cleared and new data is entered every clock period in which the enter X_k control signal is a one. When the divide instruction issues in CP00, enter X_k is a one. The signal becomes a zero in CP01 and remains a zero until CP18. The data that arrives at the input register in the clock period of instruction issue (CP00) is entered into the register at the end of CP00 and held until the end of CP18 when a new quantity is gated into the register. This data is used in executing the divide instruction.

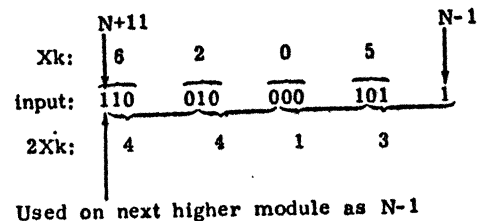
The divide unit uses the coefficient magnitude of the operands in executing the divide instructions. The coefficient is complemented (4DJ7 module) if it is negative. If bit 5 is a one, the operand is negative.

The X_k coefficient magnitude is sent to the multiplication network. The 4DK7, 4DL7, 4DM7, and 4DN7 modules form the seven multiples of the divisor coefficient from X_k to $7X_k$.

3X_k ADDER

A network on the 4DK7 and 4DN7 modules creates a 50-bit quantity $3X_k$ and shifts this quantity to form $6X_k$. Bits 0 through 47 of $3X_k$ are created on the four 4DK7 modules, 12 bits to a module. The 12 bits on a 4DK7 module are called a section and are referred to as bits N through $N+11$. Bits 48 and 49 are created in the 4DN7 module.

Each 4DK7 module receives 13 bits of X_k from the divisor input registers on the 4DJ7 modules. These bits are used as both X_k and $2X_k$. $2X_k$ is formed by left-shifting X_k one bit position. The 13 X_k bits received are bits $N-1$ through $N+11$. Of these bits, $N-1$ through $N+10$ are used as $2X_k$ bits N through $N+11$. Note that $N+11$ in one section is $N-1$ in the next higher section. For example:



The four 4DK7 modules form $3X_k$ by adding bits 0 through 47 of $2X_k$ to bits 0 through 47 of X_k . A preliminary adder on each module generates section carries and enables.

The 4DN7 module receives the four section carries and three section enables from the 4DK7 modules and performs carry propagation checks on each section. The carryout of section 4 (to bit 48) is added to X_k bit 47, which left shifted one place is $2X_k$ bit 48. The sum of this bit and the section 4 carry bit produces bits 48 and 49 of $3X_k$.

The other three section carries are fed back into the final add network on the 4DK7 modules to produce $3X_k$ bits 0 through 47.

6X_k NETWORK

$6X_k$ is formed from $3X_k$ by left-shifting one bit position. The multiples are sent to partial subtraction and trial subtraction networks. Each module in the two subtraction networks handles a three-bit group. Therefore, each module needs three bits of $3X_k$ and three bits of $6X_k$. The multiplication network sends four $3X_k$ bits to each module. These bits are labeled bits $N-1$ through $N+2$. Bits $N-1$ through $N+1$ of $3X_k$ are used by the 4DA7 and 4DH7 modules as bits N through $N+2$ of $6X_k$. This is equivalent to left-shifting one bit or multiplying by two.

ITERATION NETWORK

TRIAL SUBTRACTION NETWORK

The trial subtraction network determines which of the multiples of the divisor are larger than the current remainder. The trial subtraction network also generates a borrow into every three-bit group. The trial subtraction occurs in two stages.

FIRST-STAGE TRIAL SUBTRACTION NETWORK

The first-stage trial subtraction network generates a borrow and an enable for each three-bit group. The first-stage network is on the 4DA7, 4DB7, 4DC7, 4DD7, and 4DE7 modules. Each module in this network handles three bits (one octal digit). Each of the seven multiples of the divisor is subtracted from the remainder. The rank of a subtraction refers to the multiple being subtracted. Thus, the rank 5 subtraction network subtracts 5Xk from the remainder. The 4DC7 module handles group 0 (bits 0 through 2) of all seven ranks. The 15 4DA7 modules handle groups 1 through 15 (bits 3 through 47) of ranks 3, 6, and 7. The 4DD7 module handles group 16 (bits 48 through 50) of ranks 3, 6, and 7. The 15 4DB7 modules handle groups 1 through 15 (bits 3 through 47) of ranks 1, 2, 4, and 5. The 4DE7 module handles group 16 (bits 48 through 50) of ranks 1, 2, 4, and 5.

The data for the first-stage subtraction comes from two sources. The remainder is held within each module from the final subtraction on the previous iteration.

Each module in the first-stage trial subtraction network handles a three-bit group (one octal digit). The remainder digit is compared with each of the multiples on a module. The 4DC7 module does not generate any enables since no borrow can be propagated through bits 0 through 2.

The control signals into the 4DD7 and 4DE7 modules can force the rank 1 through 7 group 16 borrow-out bits to be ones. If these borrow bits are ones, the end-around borrow (EAB) bits for the corresponding ranks are ones, and rank 0 is selected for the final subtraction. The release remainder signal

from the 4DQ7 module into the 4DD7 and 4DE7 modules forces the rank 3 through 7 group 16 borrow-out bits to ones. The release rank 1 and 2 borrow signal into the 4DE7 module forces the rank 1 and 2 borrow-out bits to ones. These signals are used in CP01 and CP02.

In CP01, both signals are zeros; the rank 1 through 7 group 16 borrow-out bits are all ones; the rank 1 through 7 EAB bits are all ones; the final subtraction network selects rank 0 for the final subtraction; the dividend is selected by the 4DI7 modules as the rank 0 bit enables; and the dividend is gated into the remainder register.

In CP02, the 4DD7 and 4DE7 modules receive the release remainder signal from the 4DQ7 module. The 4DD7 and 4DE7 modules complement the release remainder signal and use it to force the rank 3 through 7 group 16 borrow-out bits to ones during CP01 and CP02. This causes the rank 3 through 7 EAB bits to be ones, and the final subtraction network selects one of ranks 0, 1, or 2 for the final subtraction. Thus, if the dividend exceeds the divisor by a factor of two or more, the 4DD7 module generates a two as the first quotient digit.

SECOND-STAGE TRIAL SUBTRACTION NETWORK

The outputs of the first-stage trial subtraction network go to the second-stage trial subtraction network on the 4DG7 and 4DF7 modules. The second-stage trial subtraction network determines which multiples of divisor are larger than the remainder and generates borrows into all groups for every rank. Each of the seven 4DF7 modules handles a different one of the seven ranks. The same is true of the seven 4DG7 modules. One 4DF7 module and one 4DG7 module together handle one rank.

The 4DF7 and 4DG7 modules receive identical inputs. Each module in the second-stage network receives 17 borrows, one from each of 17 groups, and 16 enables, one from each of 16 groups. All the inputs to one module pertain to the same rank.

The second-stage network determines which multiples are larger than the remainder. If the remainder is smaller than the multiple, that rank subtraction generates an EAB. Each 4DF7 and 4DG7 module generates an EAB bit for its rank. If the EAB bit for a particular rank is a one, the remainder is smaller than the multiple. If this EAB bit is a zero, the remainder is greater than or equal to the multiple. These bits are sensed in the final subtraction network to determine which multiple to use for the final subtraction. The 4DF7 EAB bit goes to every 4DA7 and 4DD7 module. The 4DG7 EAB bit goes to every 4DB7 and 4DE7 module. The EAB bit generated by the 4DF7 module is a duplicate of the EAB bit generated by the 4DG7 module. This duplication is simply an additional means of fanning out the EAB bit to all the destination modules.

The second-stage network (4DF7 and 4DG7 modules) generates borrows into groups 3 through 16 for each rank. Each 4DF7 module generates borrows into groups 3, 5, 7, 9, 11, 13, and 15 for its rank. Each 4DG7 module generates borrows into groups 4, 6, 8, 10, 12, 14, and 16 for its rank. The 4DF7 and 4DG7 modules generate the group borrow inputs for each group in the normal fashion. However, in transmitting the group borrow inputs to the destination 4DA7, 4DB7, 4DD7, and 4DE7 modules, the group borrow inputs are left-shifted one group to align them properly with the inclusive bit enables from the 4DH7 modules. Thus, instead of generating a group borrow into group 8, the following conditions generate a borrow into group 9: group 7 generates a borrow-out; group 6 generates a borrow-out; group 7 generates an enable; any of groups 0 through 5 generate a borrow; and all the higher groups (up to and including group 7) generate an enable.

PARTIAL SUBTRACTION NETWORK

The partial subtraction network provides the inclusive bit enables used in the final subtraction. The partial subtraction network is on the 16 4DH7 modules. Bits 0 through 47 are divided into 16 three-bit groups (that is, into 16 octal digits) with one digit per module.

The partial subtraction network receives the complemented remainder from the remainder register and seven multiples of the remainder from the multiplication network. The subtraction network combines the remainder bits and the multiple bits to generate inclusive bit enables. Each 4DH7 module adds the remainder digit to each of the seven multiples.

The outputs of the partial subtraction network go to the final subtraction network. All the outputs from the partial subtraction network are left-shifted one octal digit (that is, three binary bits). The remainder (rank 0 inclusive bit enables) is recomplemented to generate the true remainder bits. Remainder bits 45 through 57 are sent to the 4DQ7 module, and bits 0 through 44 are sent to four 4DI7 modules. Because of the three-bit left shift, these bits are labeled bits 48 through 50 at the 4DQ7 module and bits 3 through 47 at the 4DI7 modules. The results of the computation, seven octal digits on each module, are sent to the final subtraction network on the 4DA7 and 4DB7 modules. The seven octal digits from 4DH7-O16 are sent to the 4DD7 and 4DE7 modules. Because of the left shift, these are bits 3 through 50 enables.

The remainder digits (rank 0 inclusive bit enables) go through an intermediate network on the 4DI7 and 4DQ7 modules before they are transmitted to the final subtraction network. The 4DQ7 module gates bits 48 through 50 to the final subtraction network in the 4DD7 and 4DE7 modules in clock periods CP02 through CP17 only. The circulate remainder signal in the 4DI7 modules chooses either the remainder bits 3 through 47 or the quantity in the dividend register.

In CP01, the dividend from the divide sequence is in the dividend register. In CP01, therefore, the complemented dividend is gated to the final subtraction network. During CP02 through CP17, the complemented remainder is gated to the final subtraction network. In CP02 through CP17, bits 0 through 2 are always zeros with one exception; a 25252 pattern is entered in CP02 if the instruction is a round floating divide. All output bits (bits 0 through 47 from the 4DI7 modules and bits 48 through 50 from the 4DQ7 module) are complemented to become rank 0 inclusive bit enables before transmission to the final subtraction network.

NOTE

When a divide round instruction is executed, bits 0 through 2 are alternately entered with either a 2 or a 5. This effectively shifts a 25252..5 pattern into the bit positions below the least significant bit of the dividend.

During a round floating divide instruction, a round bit is added to the dividend which has the effect of increasing the dividend by one-third count. The effect this has on the quotient varies depending upon the value of the divisor and the truncation point in the quotient. If the dividend is smaller than the divisor, the quotient is truncated one bit position lower than if the dividend is equal to, or larger than, the divisor. These effects cause the rounding to vary in the quotient from a value of 1/6 of the least significant bit in the result to almost one. The average rounding bias over the entire range of coefficient values is zero.

When a divide instruction is executed, the m0 (round) bit is held in the 4DQ7 (control) module. If the instruction is a round floating divide instruction, this bit is a one. If the instruction is a floating divide instruction, this bit is a zero. The 4DQ7 module holds this round bit until CP02, when it is transmitted to the 4DI7-P10 module. This bit then becomes bits 0 through 2 of the rank 0 inclusive bit enables. Since bits 0 through 2 of the rank 0 bit enables are normally zero, the round bit has an effect only on round instructions (when the round bit is a one). The 4DC7 module enters the rank 0 bit 0 through 2 enables into the remainder register every clock period.

FINAL SUBTRACTION NETWORK

The final subtraction network picks one rank, completes that rank subtraction, and gates the result into the remainder register. An octal digit (pick number) corresponding to the chosen rank is gated into the quotient shift register. The remainder is then available to the first-stage trial subtraction network and the partial subtraction network.

The final subtraction network is on the 4DA7, 4DB7, 4DD7, and 4DE7 modules. The 4DA7 and 4DB7 modules have identical final subtraction networks, as do the 4DD7 and 4DE7 modules. The 4DA7 modules output remainder bits to the partial subtraction network. The 4DD7 module generates the pick number. Each module in this network handles a three-bit group. The 15 4DA7 and the 60420300 A

15 4DB7 modules handle groups 1 through 15, bits 3 through 47. The 4DD7 and 4DE7 modules handle group 16, bits 48 through 50. Group 0, bits 0 through 2 are handled in the 4DC7 module and will be discussed later.

The inputs to the subtraction network come from the second-stage trial subtraction network and the partial subtraction network. Each module in the final subtraction network receives seven EAB bits, one for each rank. Each 4DA7 or 4DD7 module receives one EAB from each of the seven 4DF7 modules. Each 4DB7 or 4DE7 module receives one EAB from each of the seven 4DG7 modules. Each module in the final network receives a group borrow input for each of ranks 1 through 7 from the second-stage trial subtraction network. 4DA7-M02 and 4DB7-N02 receive group 0 borrows (left-shifted one group) from the 4DC7 module. All other modules in this network receive inputs from a 4DF7 or 4DG7 module, depending upon the group handled. Each module in the network receives three enable bits for each of the eight ranks. The ranks 1 through 7 enables come from the 4DG7 modules, and the rank 0 enables come from the 4DI7 modules. The group 16 (bits 48 through 50) rank 0 enables come from the 4DQ7 module.

The data is combined to generate three remainder bits for each module. The EAB bits are sensed, and the largest rank that did not produce EAB is used for the final subtraction. The borrow bit for that rank is added to the octal digit and the sum is complemented. The result is entered into the remainder register.

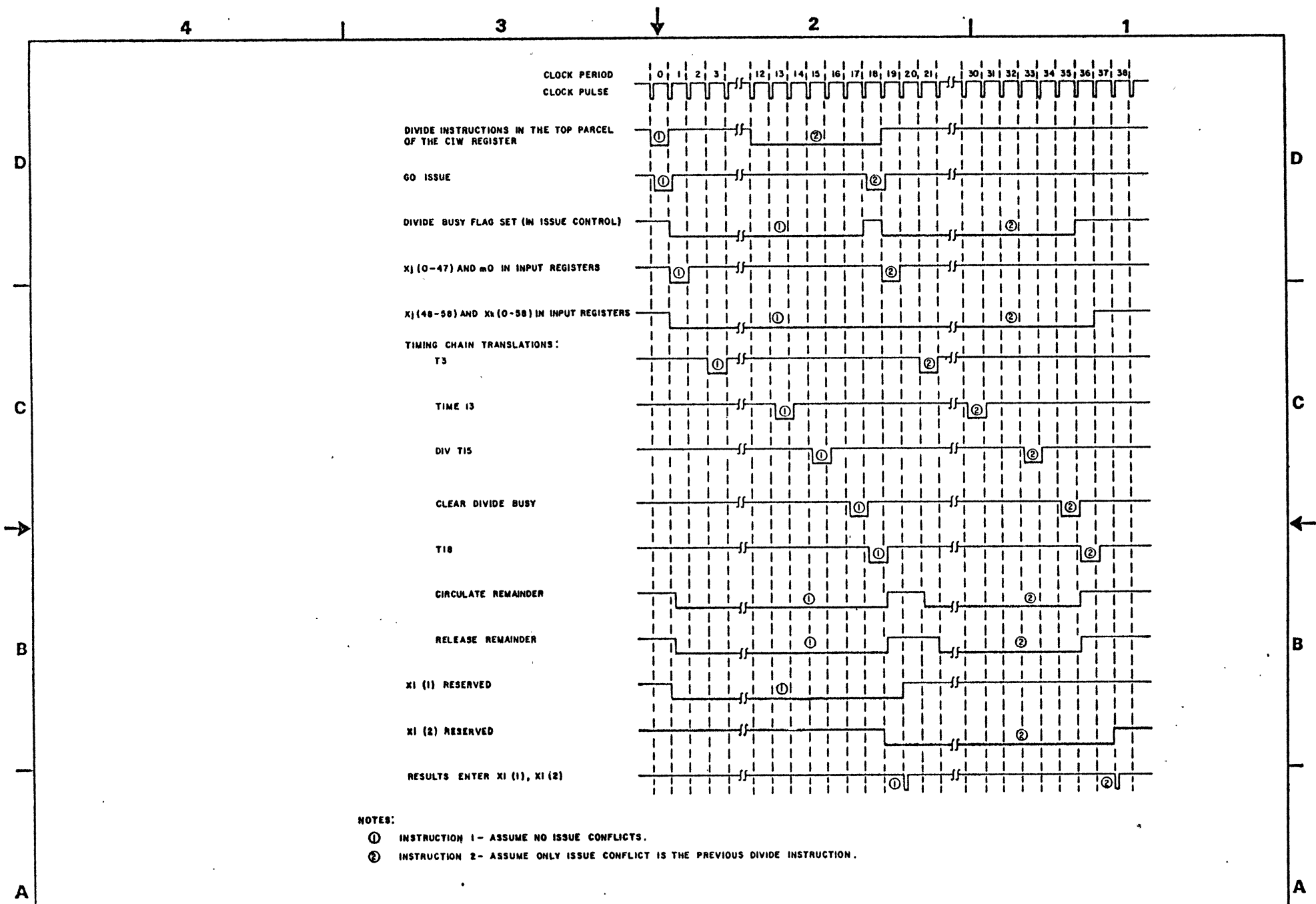
A network on the 4DD7 module senses the EAB bits and generates a pick number. The digit generated is the same as the rank chosen for the final subtraction. This digit is entered into a register and sent from there to the coefficient shift register on the 4DS7 module.

The 4DC7 module holds group 0, bits 0 through 2 of the remainder. In CP01, bits 0 through 2 of the dividend are entered into the remainder register. In CP02 through CP17, zero bits are entered into bits 0 through 2. During a round floating divide instruction, a 25252 pattern is entered in bits 0 through 2 in CP02 through CP17. These bits increase the value of the dividend by one-third count.

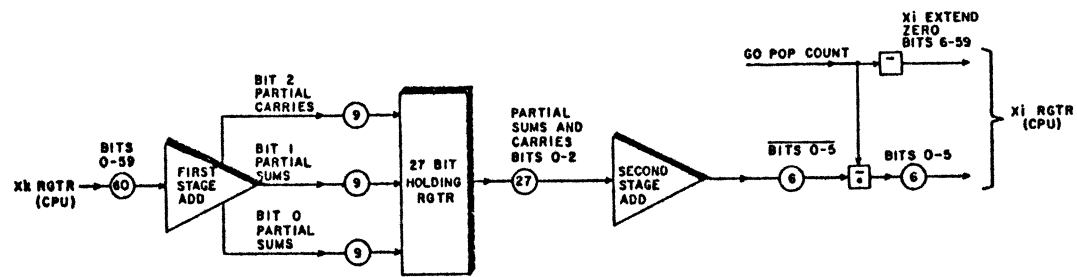
Bits 0 through 47 of the remainder are sent to the partial subtraction network. These bits are transmitted from the 4DC7 and 4DA7 modules to the 4DH7 modules.

PART 9

POPULATION COUNT UNIT



NOTES:
 ① INSTRUCTION 1 - ASSUME NO ISSUE CONFLICTS.
 ② INSTRUCTION 2 - ASSUME ONLY ISSUE CONFLICT IS THE PREVIOUS DIVIDE INSTRUCTION.



CONTROL DATA DEVELOPMENT DIVISION	PRIMARY BLOCK DIAGRAM POP. COUNT UNIT		CODE IDENT 34010	QNS NO 60420300	REV A
	C	POP CT 10	PAGE	5-9-1	

POPULATION COUNT UNIT

The population count unit executes CPU instruction 47 by counting the number of one bits in the Xk register and storing the results in the lower order six bits of the Xi register. Bits 6 through 59 are filled with zeros.

The population count instruction requires 2 clock periods for execution. Data moves from the Xk register to the population count unit in the same clock period in which the instruction issues from the CIW register. Data moves from the population count unit to the Xi register during the following clock period. A new instruction may be issued for execution in the population count unit each clock period.

The content of the Xk register is transmitted to the population count unit each clock period. This data enters a static network that partially sums the one bits and enters the partially reduced data into a 27-bit holding register. This register is cleared and reset with new data each clock period. The data is further summed in a second static network until six bits represent the complement of the result. When go pop count is received from CPU instruction control, the six-bit result is recomplemented and sent to the lower order six bits of the Xi register, and 54 zero bits are sent to the rest of the Xi register. If go pop count is not received, the data in the population count unit is discarded.

4

3

2

1

D

D

C

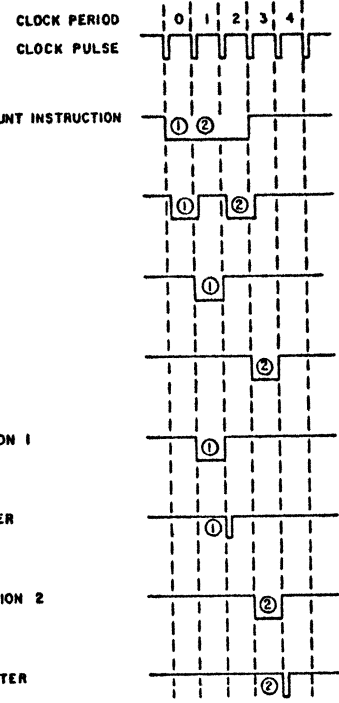
C

B

B

A

A



A BOOLEAN INSTRUCTION FOLLOWED BY A POP COUNT INSTRUCTION IN THE TOP PARCEL OF THE CIW REGISTER

GO ISSUE

GO BOOLEAN

GO POP COUNT

RESULT REGISTER X1 RESERVED FOR INSTRUCTION 1

INSTRUCTION 1 RESULT ENTERS THE X1 REGISTER

RESULT REGISTER X1 RESERVED FOR INSTRUCTION 2

INSTRUCTION 2 RESULT ENTERS THE X1 REGISTER

NOTES:

- ① $112 \mid n \ X1 + Xn \ TO \ X2$ (ASSUMING NO ISSUE CONFLICTS EXIST).
- ② $47 \mid 2 \ COUNT \ OF \ ONES \ IN \ X2 \ TO \ X1$ ($X2$ OPERAND REGISTER CONFLICTS WITH INSTRUCTION 2 DESTINATION REGISTER).

· PART 10

INCREMENT UNIT

INCREMENT UNIT

The increment unit executes CPU instructions 50 through 77. These instructions involve arithmetic operations on two selected 18-bit operands from the A, B, X, and CIW registers. During 50 through 57 instructions, the result is transmitted to an A register. The result plus RAC is also sent to the SAS when the A1 through A7 registers are used. These two arithmetic operations are performed independently and in parallel with each other. During 60 through 67 instructions, the result is transmitted to a B register. During 70 through 77 instructions, the result is transmitted to an X register.

INPUT OPERAND SELECTION

Data arrives at the increment unit from five different input paths. One group of data paths consists of inputs from the Aj, Bj, and Xj registers. One of these operands is selected as one of the two increment operands. The selection is based on the value of the m designator from the CIW register. The second increment operand is selected from another group of input data paths. This group consists of the K field in the CIW register, the Bk register, and the complement of the Bk register. This selection is also based on the m designator value. The two selected operands enter both parallel computation sections simultaneously.

COMPUTATION

One computation is performed in a two-operand adder. The two selected operands are partially added in the first stage of the adder. The resultant bit/group borrows and enables are stored in the partial sum holding register. The computation is completed in the second stage of the adder.

A second computation is performed in a three-operand adder. This happens simultaneously with the operation described previously. The two selected operands and bits 0 through 17 from the RAC register are partially added in the first stage of the adder. The resultant bit/group borrows and enables are stored in the partial sum holding register. The computation is completed in the second stage of the adder.

INCREMENT TEST HOLDING REGISTER

An 18-bit incremented operand is sent to the increment test holding register during every clock period. The content of this register is compared with FLC in the CPU. When the incremented operand is equal to or greater than FLC, CMC blocks the CM write reference and returns a zero word for a CM read reference. This occurs only when a 50 through 57 instruction causes a CM reference.

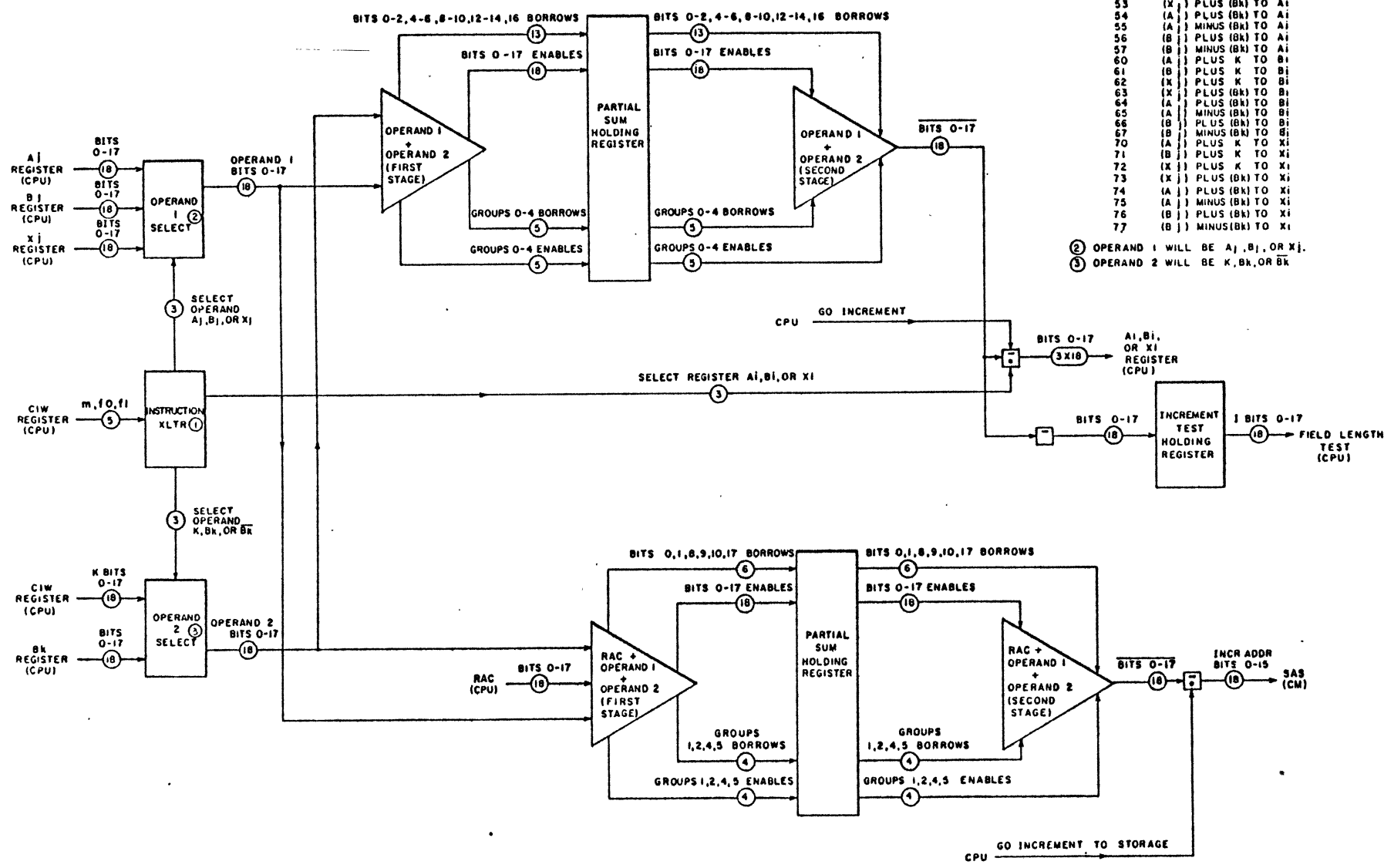
DESTINATION REGISTER SELECTION

The 18-bit incremented operand is gated to an A, B, or X register whenever the CPU generates a go increment signal. The destination register selection is based on the value of the lower two bits of the f designator from the CIW register.

The A1 register receives the incremented operand during 50 through 57 instructions (f designator bits equal X01). When these instructions occur, the CPU generates a go increment to storage signal. This signal gates the incremented operand (plus RAC) to the SAS. The A1 register and the SAS receive increment data simultaneously. If the l designator in the CIW register has an octal value of zero, the go increment to storage signal is not generated.

The B1 register receives the incremented operand during 60 through 67 instructions (f designator bits equal X10).

The X1 register receives the incremented operand during 70 through 77 instructions (f designator bits equal X11). When these instructions occur, the sign bit is extended in the X1 register.



NOTES:
 ① THE INCREMENT UNIT PERFORMS THE FOLLOWING FUNCTIONS:

INSTRUCTION	FUNCTION
50	(A) PLUS K TO Ai
51	(B) PLUS K TO Ai
52	(X) PLUS K TO Ai
53	(X) PLUS (Bk) TO Ai
54	(A) PLUS (Bk) TO Ai
55	(A) MINUS (Bk) TO Ai
56	(B) PLUS (Bk) TO Ai
57	(B) MINUS (Bk) TO Ai
60	(A) PLUS K TO Bi
61	(B) PLUS K TO Bi
62	(X) PLUS K TO Bi
63	(X) PLUS (Bk) TO Bi
64	(A) PLUS (Bk) TO Bi
65	(A) MINUS (Bk) TO Bi
66	(B) PLUS (Bk) TO Bi
67	(B) MINUS (Bk) TO Bi
70	(A) PLUS K TO Xi
71	(B) PLUS K TO Xi
72	(X) PLUS K TO Xi
73	(X) PLUS (Bk) TO Xi
74	(A) PLUS (Bk) TO Xi
75	(A) MINUS (Bk) TO Xi
76	(B) PLUS (Bk) TO Xi
77	(B) MINUS (Bk) TO Xi

② OPERAND 1 WILL BE A_i, B_i, OR X_i.
 ③ OPERAND 2 WILL BE K, B_k, OR B_k

4

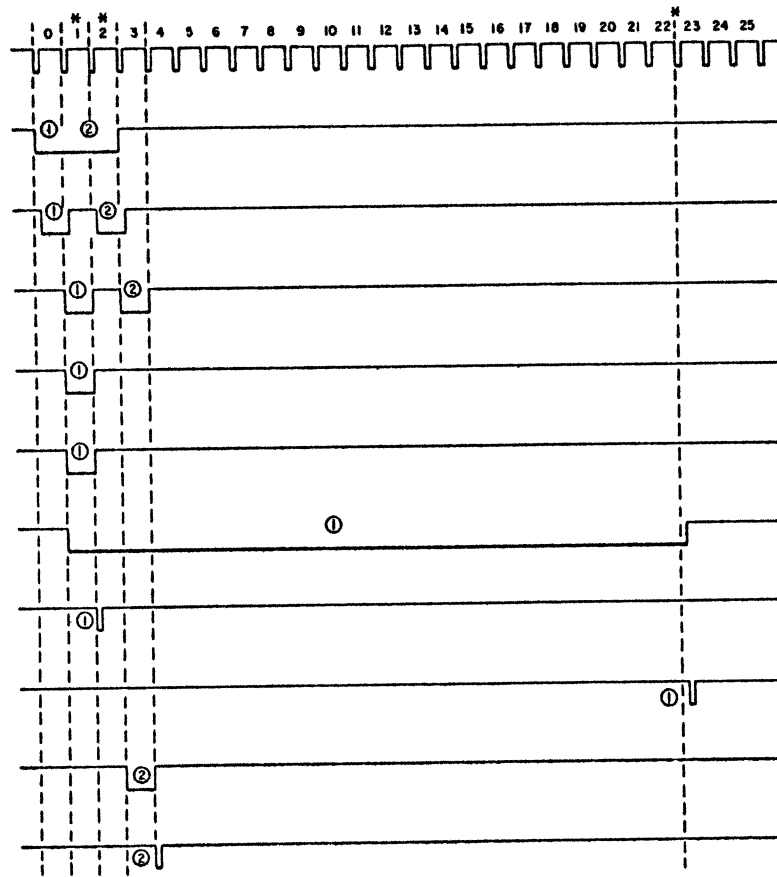
3

↓

2

1

CLOCK PERIOD
CLOCK PULSE



* GO ISSUE FOR INCREMENT MAY BE DELAYED ONE CLOCK PERIOD IF MEMORY ENABLE IS NOT UP.

D

D

C

C

→

←

B

B

A

A

ECS Subsystem

GENERAL DESCRIPTION

1

The 7030-1XX Extended Core Storage (ECS) Subsystem is an on-line, semirandom access, magnetic core memory system which augments central memory (CM). It is a fixed-word-length system capable of two-way communication. The ECS subsystem is primarily used with the CDC CYBER 170 computer systems.

The ECS subsystem consists of core storage, an ECS controller, and a distributive data path (DDP). An ECS coupler interfaces the ECS subsystem with the central computer. The ECS coupler is part of the mainframe. Figure 1-1 shows an ECS subsystem used with a model 175 central computer.

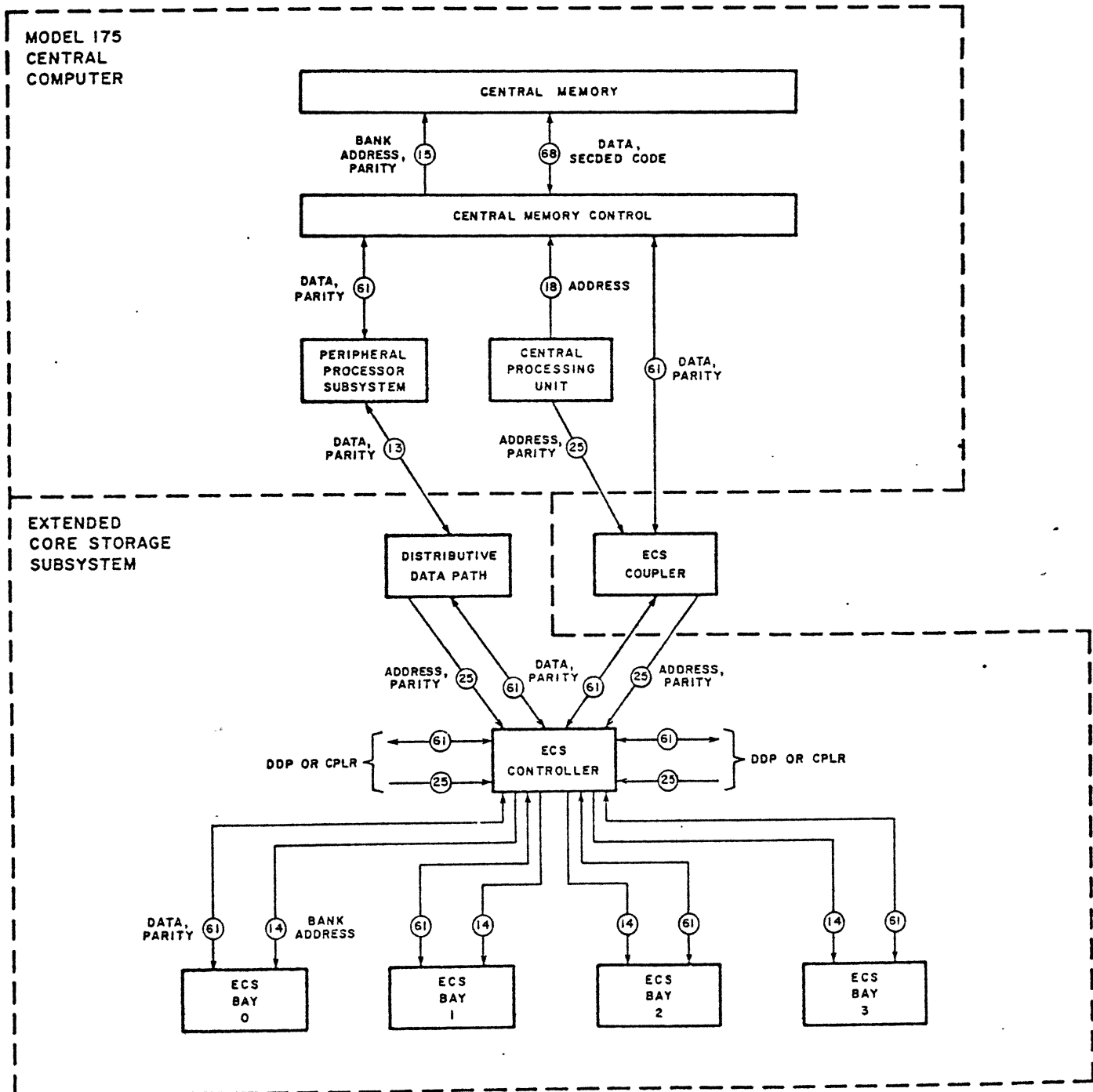


Figure 1-1. ECS Subsystem with Model 175 Central Computer

EXTENDED CORE STORAGE

ECS consists of 1, 2, 4, 8, or 16 memory banks, each capable of storing 131,072 60-bit words. A cabinet (called a bay) holds up to four memory banks. Each ECS bank address stores one ECS record. An ECS record contains eight 60-bit words. Assembly/disassembly of 60/480-bit words/records during data transfers is performed in the ECS banks. References as low as one 60-bit word are possible. ECS is capable of a maximum data transfer rate of one 60-bit word per 100 nanoseconds.

ECS CONTROLLER

The ECS controller regulates access to the ECS banks. The controller has four 60-bit access channels and four 60-bit ECS interfaces. Each access channel connects to one ECS coupler or one DDP. The controller performs time-sharing on the four access channels. Each of the four ECS interfaces connects to an ECS bay.

DISTRIBUTIVE DATA PATH

The DDP controls data flow between ECS and a peripheral processor (PP). The DDP contains one PP interface (called a port) and is expandable to four ports. Each port connects to one I/O channel in the peripheral processor subsystem (PPS). A scanner monitors the DDP ports and sequentially connects requesting ports to one access chan-

nel in the ECS controller. The DDP assembles 12-bit words into 60-bit words during a write operation and disassembles 60-bit words into 12-bit words during a read operation.

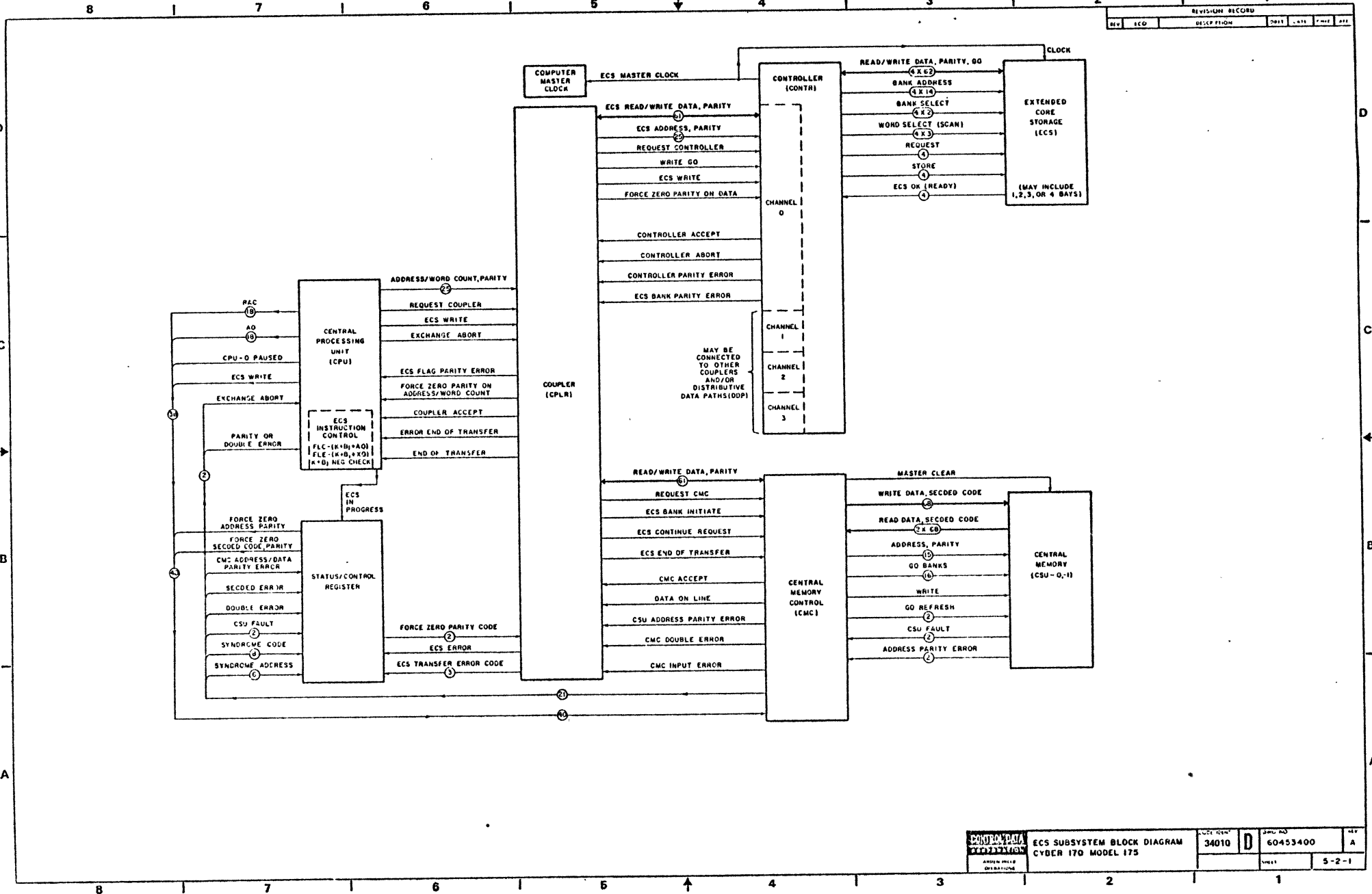
NOTE

Training and maintenance information for the DDP is not provided in this manual. Comparable information is contained in the DDP Hardware Maintenance Manual.

ECS COUPLER

The ECS coupler controls the transfer of 60-bit data words from CM to ECS (write operation) or from ECS to CM (read operation). The read or write operation is initiated by instructions in the central processing unit (CPU). Central memory control (CMC) controls data flow between CM and the ECS coupler. The ECS coupler controls and monitors the data to ensure a successful transfer operation. If errors occur during a transfer operation, the ECS coupler detects the errors or passes the error signals detected by the ECS controller to the CPU.

REVISION RECORD				
REV	ECO	DESCRIPTION	DATE	BY



THEORY OF OPERATION

This section contains signal definitions and sequence charts for the ECS subsystem used with the CDC CYBER 170 model 175 computer system. Descriptions are keyed to the diagrams in section 5 and to block or logic diagrams in other manuals.

SIGNAL DEFINITION

The ECS subsystem block diagram (section 5, part 2) shows all data and control paths between functional components. These signals are described in the following paragraphs.

CENTRAL PROCESSING UNIT

Request Coupler

This signal is sent to the coupler to initiate a data transfer. ECS instruction control sends the request when the following conditions exist.

- ECS instruction in CIW register
- Word count positive
- ECS addresses in range
- CM addresses in range

Coupler Accept

This signal enables the ECS address and word count from the CPU to the coupler.

Address/Word Count, Parity

This 24-bit (plus parity bit) path carries the 24-bit ECS address and 17-bit word count from the CPU to the coupler. Word count is the number of 60-bit words to be transferred.

ECS Write

This signal is sent to CMC and the coupler to enable data transfer from CM to ECS. The absence of this signal enables data transfer from ECS to CM.

Exchange Abort

This signal from CMC informs the CPU and the coupler that an exchange jump has been requested during a data transfer.

Error End of Transfer

This signal from the coupler informs the CPU that a transfer error occurred.

End of Transfer

This signal from the coupler informs the CPU that transfer concluded without errors.

ECS Flag Parity Error

This signal from the coupler informs the CPU that an address parity error occurred during an ECS flag operation.

A0

This 18-bit path carries the starting relative CM address to CMC.

RAC

This 18-bit path carries the reference CM address to CMC. RAC is added to A0 to form the absolute starting CM address.

Force Zero Parity on Address/Word Count

This signal from the coupler forces the address/word count parity bit to zero.

CPU-0 Paused

This signal is sent to CMC to enable ECS mode.

Parity or Double Error

This signal informs the CPU that CMC has detected a parity error (parity mode) or a double error (SECEDED mode).

ECS in Progress

This signal sets bit 194 in the status and control register during data transfer.

COUPLER

Request Controller

This signal is sent to the controller to initiate a data transfer.

Controller Accept

This signal informs the coupler that the controller has honored a request.

ECS Address, Parity

This 24-bit (plus parity bit) path carries the ECS address or flag function data from the coupler to the controller.

Force Zero Parity on Data

This signal forces the ECS read data parity bit from the controller to the coupler to zero.

Controller Abort

This signal informs the coupler that the controller has detected a transfer error.

Controller Parity Error

This signal informs the coupler that the controller has detected an address or data parity error during transfer between the coupler and the controller.

ECS Bank Parity Error

This signal informs the coupler that the controller has detected a data parity error during transfer between the controller and ECS.

ECS Read Data, Parity

This 60-bit (plus parity bit) path carries data from the controller to the coupler.

ECS Write Data, Parity

This 60-bit (plus parity bit) path carries data from the coupler to the controller.

Go

This signal informs the controller that the coupler is sending valid write data.

ECS Write

This signal is sent to the controller to enable data transfer from CM to ECS. The absence of this signal enables data transfer from ECS to CM.

Request CMC

This signal is sent to CMC to initiate a data transfer.

CMC Accept

This signal informs the coupler that CMC is in ECS mode, ready for ECS transfer.

ECS Bank Initiate

This signal is sent to CMC to request a CM reference.

ECS Continue Request

This signal is sent to CMC to reserve CM banks.

ECS End of Transfer

This signal is sent to CMC to disable ECS mode after a data transfer.

CSU Address Parity Error

This signal from CMC informs the coupler that CM has detected an address parity error during transfer from CMC to CM.

CMC Double Error

This signal informs the coupler that CMC has detected multiple data bit errors in a word read from CM.

CMC Input Error

This signal informs the coupler that CMC has detected a data parity error during transfer between the coupler and CMC.

Read Data, Parity

This 60-bit (plus parity bit) path carries data from CMC to the coupler.

Data on Line

This signal informs the coupler that CMC is sending valid read data.

Write Data, Parity

This 60-bit (plus parity bit) path carries data from the coupler to CMC.

Force Zero Parity Code

This 2-bit code from bits 132 and 133 in the status and control register sets parity bit to zero during selected data and address transfers.

<u>Code</u>	<u>Force Zero Parity On</u>
0	No zero parity
1	Address/word count at CPU
2	ECS address to controller
3	Data at controller

ECS Error

This signal sets bit 11 in the status and control register when a transfer error occurs.

ECS Transfer Error Code

This 3-bit code sets bits 136 through 138 in the status and control register to define the type of transfer error that occurred.

<u>Code</u>	<u>Error</u>
7	No error (if status bit 11 not set)
1	CPU (address/word count) parity error
2	CMC double error
3	CSU address parity error
4	CMC data parity error
5	ECS bank parity error
6	ECS controller data parity error
7	ECS controller address parity error

CONTROLLER

Request

This signal is sent to ECS to initiate a memory reference.

Bank Select

This 2-bit code is sent to ECS to select one of four banks in a bay.

Bank Address

This 14-bit address is sent to a selected ECS bank to select one location of eight 60-bit (plus parity bit) words.

Word Select (Scan)

This 3-bit code is sent to ECS to select one of eight words from the addressed location.

Store

This signal is sent to ECS to initiate a write operation. The absence of this signal initiates a read operation.

ECS OK (Ready)

This signal from ECS informs the controller that bank power is applied and is not in maintenance mode.

Read Data, Parity

This 60-bit (plus parity bit) path carries data from the ECS bay to the controller.

Read Go

This signal informs the controller that ECS is sending valid data.

Write Data, Parity

This 60-bit (plus parity bit) path carries data from the controller to ECS.

Write Go

This signal informs ECS that the controller is sending valid data.

ECS Master Clock

This signal overrides the computer system master clock and causes it to become a slave to the master clock generated by the controller.

CENTRAL MEMORY CONTROL

Address, Parity

This 14-bit address (plus parity bit) is sent to a selected CM bank to select one 60-bit (plus 8-bit SECDED code) word location.

Go Banks

These 16 signals are sent to CM to initiate bank references.

Go Refresh

These two signals are sent to CM to initiate CSU refresh cycles.

Master Clear

This signal sets all CM locations to zero.

Write

This signal is sent to CM to initiate a write operation. The absence of this signal initiates a read operation.

Address Parity Error

These two signals inform CMC that a CSU has detected an address parity error during transfer between CMC and CM.

Write Data, SECDED Code

This 60-bit (plus 8-bit SECDED code or parity bit) path carries data from CMC to CM.

Read Data, SECDED Code

This 60-bit (plus 8-bit SECDED code or parity bit) path carries data from CM to CMC.

Force Zero Address Parity

This signal from bit 129 in the status and control register forces the address parity bit from CMC to CM to zero.

Force Zero SECDED Code, Parity

This signal from bit 128 in the status and control register forces the write data SECDED code or parity bit from CMC to CM to zero. It also forces the read data parity bit from CMC to the coupler to zero.

CMC Address/Data Parity Error

This signal sets bit 139 in the status and control register when CMC detects an address parity error. If bit 139 is clear and bit 5 is set, CMC detected a data parity error.

SECDED Error

This signal sets bit 3 in the status and control register when CMC detects a CM double error or corrects a CM single error.

Double Error

This signal sets bit 183 in the status and control register when CMC detects a CM double error.

CSU Fault

These two signals set bits 8 and/or 9 in the status and control register when CSU-0 and/or CSU-1 detects a CM refresh problem.

Syndrome Code

This 8-bit code is sent to bits 40 through 47 in the status and control register when CMC detects a SECDED error. This code identifies which bit failed in a single-error failure.

Syndrome Address

These six bits are sent to bits 48 through 53 in the status and control register when CMC detects a SECDED error. These bits provide information to isolate a CM failure.

<u>Address Bit</u>	<u>Status Bit</u>	<u>Description</u>
0	48	
1	49	CSU bank
2	50	
3	53	CSU chassis
16	51	
17	52	CM quadrant

ECS READ/WRITE SEQUENCES

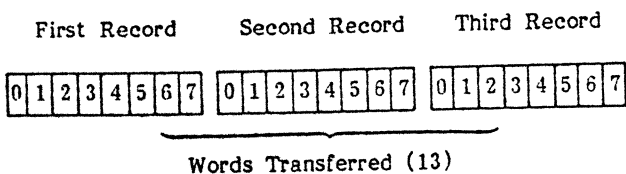
The following paragraphs describe the sequence of events for ECS read and write operations. Sequences are keyed to the ECS subsystem block diagram in section 5, part 2.

ECS READ SEQUENCE

The following sequence describes the signal flow between ECS subsystem components during an 011 read ECS to CM instruction. This example assumes that no errors or exchange requests occur during the transfer. Parameters for this example are:

<u>Parameter</u>	<u>Value (octal)</u>
Instruction (CIW)	011000015
Word count (K+Bj)	000015
CM starting address (A0)	002000
CM reference address (RAC)	010000
CM field length (FLC)	020000
ECS starting address (X0)	00100006
ECS reference address (RAE)	00200000
ECS field length (FLE)	00500000

Three records are used in this example. The first and last records are incomplete. For a transfer of more than three records, the second record sequence repeats through the next-to-last record, and the last record sequence then begins.



- CPU transmits A0 (002000₈) and RAC (010000₈) to A0+RAC adder in CMC.
- CIW contains read ECS to CM instruction (0110000015₈) in upper two parcels (CPU).
- CPU checks FLC-(K+Bj+A0) for CM range error, FLE-(K+Bj+X0) for ECS range error, and K+Bj for negative word count. If no range errors are detected and word count is positive, CPU transmits request coupler signal to coupler.
- Coupler transmits coupler accept signal to CPU.
- CPU transmits word count (000015₈) and parity bit to coupler.
- CPU transmits ECS address (X0+RAE=00300006₈) and parity bit to coupler.
- Coupler checks for zero word count and address/word count parity error. If word count is not zero and no parity error, coupler transmits request CMC signal to CMC.
- CMC gates CM address (A0+RAC=012000₈) to BAK register.
- CMC waits for all CM banks to be free and SAS to be empty, and transmits CMC accept signal to coupler.
- Coupler determines that two 60-bit words are to be transferred in first record.
- Coupler transmits request controller signal and ECS address (00300006₈) with parity bit to controller for first record.
- When controller request is accepted, controller transmits controller accept signal to coupler and transmits request signal, bank select (0), bank address (06000₈), and word select (6) to ECS for first record.
- Coupler starts read delay to wait for data from ECS, decrements word count (000013₈), and increments ECS address (00300010₈) for each of two 60-bit words to be transferred in first record.
- ECS starts read reference for one 60-bit word and parity bit for each word select count (6 and 7) in first record.
- Coupler transmits request controller signal and ECS address (00300010₈) with parity bit to controller for second record.
- When controller request is accepted, controller transmits controller accept signal to coupler and transmits request signal, bank select (1), bank address (06000₈), and word select (0) to ECS for second record.
- ECS transmits two 60-bit words, parity bits, and read go signals to controller (first record).

18. Controller transmits two 60-bit words and parity bits to coupler (first record).
19. Coupler starts read delay to wait for data from ECS, decrements word count (000003₈), and increments ECS address (00300020₈) for each of eight 60-bit words to be transferred in second record.
20. ECS starts read reference for one 60-bit word and parity bit for each word select count (0 through 7) in second record.
21. Coupler completes read delay for first record and transmits ECS continue request signal to CMC to reserve CM banks 0 through 4.
22. After 400-nanosecond delay, coupler transmits two 60-bit words, parity bits, and ECS bank initiate signals to CMC (first record).
23. Coupler transmits request controller signal and ECS address (00300020₈) with parity bit to controller for third record.
24. When controller request is accepted, controller transmits controller accept signal to coupler and transmits request signal, bank select (2), bank address (06000₈), and word select (0) to ECS for third record.
25. CMC transmits address (00500₈) with parity bit, go bank (0) signal, write signal, and first 60-bit word (6) of first record with 8-bit SECEDED code to CM. After 100-nanosecond delay, CM transmits address (00500₈) with parity bit, go bank (1) signal, write signal, and second 60-bit word (7) of first record with 8-bit SECEDED code to CM.
26. ECS transmits eight 60-bit words, parity bits, and read go signals to controller (second record).
27. Controller transmits eight 60-bit words and parity bits to coupler (second record).
28. Coupler starts read delay to wait for data from ECS, decrements word count (000000₈) for each of three 60-bit words to be transferred in third record, and increments ECS address (00300030₈).
29. ECS starts read reference for one 60-bit word and parity bit for each word select count (0 through 7) in third record.
30. When word count reaches 000000₈, coupler starts end time sequence.
31. Coupler completes read delay for second record and transmits ECS continue request signal to CMC to reserve CM banks 2 through 6.
32. After 400-nanosecond delay, coupler transmits eight 60-bit words, parity bits, and ECS bank initiate signals to CMC (second record).
33. CMC transmits address (00500₈) with parity bit, go bank (2) signal, write signal, and first 60-bit word (0) of second record with 8-bit SECEDED code to CM. After every 100 nanoseconds, CMC transmits address (00500₈) with parity bit, next go bank (3 through 18) signal, write signal, and next 60-bit word (1 through 7) of second record with 8-bit SECEDED code to CM.

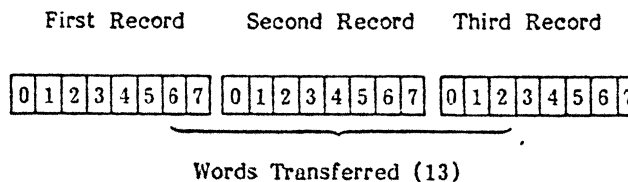
34. ECS transmits eight 60-bit words, parity bits, and read go signals to controller (third record).
35. Controller transmits eight 60-bit words and parity bits to coupler (third record).
36. Coupler completes read delay for third record and transmits ECS continue request signal to CMC to reserve CM banks 12₈ through 16₈.
37. After 400-nanosecond delay, coupler transmits three 60-bit words, parity bits, and ECS bank initiate signals to CMC (third record).
38. CMC transmits address (00500₈) with parity bit, go bank (128) signal, write signal, and first 60-bit word (0) of third record with 8-bit SECEDED code to CM. After every 100 nanoseconds, CMC transmits address (00500₈) with parity bit, next go bank (138 and 148) signal, write signal, and next 60-bit word (1 and 2) of third record with 8-bit SECEDED code to CM.
39. Coupler transmits ECS end of transfer signal to CMC; this clears ECS code of operation. Coupler transmits end of transfer signal to CPU; this causes full exit to next instruction which starts at address P+1.

ECS WRITE SEQUENCE

The following sequence describes the signal flow between ECS subsystem components during an 012 write ECS from CM instruction. This example assumes that no errors or exchange requests occur during the transfer. Parameters for this example are:

<u>Parameter</u>	<u>Value (octal)</u>
Instruction (CIW)	0120000015
Word count (K+Bj)	000015
CM starting address (A0)	002000
CM reference address (RAC)	010000
CM field length (FLC)	020000
ECS starting address (X0)	00100006
ECS reference address (RAE)	00200000
ECS field length (FLE)	00500000

Three records are used in this example. The first and last records are incomplete. For a transfer of more than three records, the second record sequence repeats through the next-to-last record, and the last record sequence then begins.



1. CPU transmits A0 (00200_g) and RAC (010000_g) to A0+RAC adder in CMC.
2. CIW contains write ECS from CM instruction (0120000015_g) in upper two parcels (CPU).
3. CPU transmits ECS write signal to coupler and CMC.
4. CPU checks FLC-(K+Bj+A0) for CM range error, FLE-(K+Bj+X0) for ECS range error, and K+Bj for negative word count. If no range errors are detected and word count is positive, CPU transmits request coupler signal to coupler.
5. Coupler transmits accept signal to CPU.
6. CPU transmits word count (000015_g) and parity bit to coupler.
7. CPU transmits ECS address (X0+RAE=00300006_g) and parity bit to coupler.
8. Coupler checks for zero word count and address/word count parity error. If word count is not zero and no parity error, coupler transmits request CMC signal to CMC.
9. CMC gates CM address (A0+RAC=012000_g) to BAK register.
10. CMC waits for all CM banks to be empty and transmits CMC accept signal to coupler.
11. Coupler determines that two 60-bit words are to be transferred in first record.
12. Coupler clears input/output address register.
13. Coupler transmits ECS continue request signal to CMC to reserve CM banks 0 through 4 for first record.
14. Coupler decrements word count (000013_g) for each of two 60-bit words to be transferred in first record.
15. 400 nanoseconds after ECS continue request signal, coupler transmits two 60-bit words, parity bits, and ECS bank initiate signals to CMC (first record).
16. Coupler transmits request controller signal, ECS write signal, and ECS address (00300006_g) with parity bit to controller for first record.
17. CMC transmits address (00500_g) with parity bit and go bank (0) signal to CM. After 100-nanosecond delay, CMC transmits address (00500_g) with parity bit and go bank (1) signal to CM. This initiates a CM read cycle for words 6 and 7 in first record.
18. Coupler determines that eight 60-bit words are to be transferred in second record.
19. When controller request is accepted, controller transmits controller accept signal to coupler and transmits request signal, store signal, bank select (0), bank address (06000_g), and word select (6) to ECS for first record.
20. ECS starts write reference for each word select count (6 and 7) in first record.
21. When controller accept for first record is received, coupler starts write delay and increments ECS address (00300010_g).
22. Coupler transmits ECS continue request signal to CMC to reserve CM banks 2 through 6 for second record.
23. Coupler decrements word count (000003_g) for each of eight 60-bit words to be transferred in second record.
24. Coupler transmits request controller signal, ECS write signal, and ECS address (00300010_g) with parity bit to controller for second record.
25. CM transmits two 60-bit words (6 and 7) with 8-bit SECDED code to CMC (first record).
26. CMC transmits two 60-bit words, parity bits, and data on line signals to coupler. Coupler stores these first record words 6 and 7 in RAM.
27. When controller request is accepted, controller transmits controller accept signal to coupler and transmits request signal, store signal, bank select (1), bank address (06000_g), and word select (0) to ECS for second record.
28. ECS starts write reference for each word select count (0 through 7) in second record.
29. When controller accept for second record is received, coupler starts write delay and increments ECS address (00300020_g).
30. 400 nanoseconds after ECS continue request signal, coupler transmits eight ECS bank initiate signals to CMC (second record).
31. CMC transmits address (00500_g) with parity bit and go bank (2) signal to CM. After every 100 nanoseconds, CMC transmits address (00500_g) with parity bit and next go bank (3 through 118) signal to CM. This initiates a CM read cycle for words 0 through 7 in second record.
32. Coupler completes write delay for first record and transmits two 60-bit words, parity bits, and go signals from RAM to controller.
33. Controller transmits two 60-bit words, parity bits and write go signals to ECS. ECS writes first record words 6 and 7.
34. Coupler determines that three words are to be transferred in third record.
35. Coupler transmits ECS continue request signal to CMC to reserve CM banks 12_g through 16_g for third record.
36. Coupler decrements word count (000000_g) for each of three 60-bit words to be transferred in third record.
37. When word count reaches 000000_g, coupler starts end time sequence.

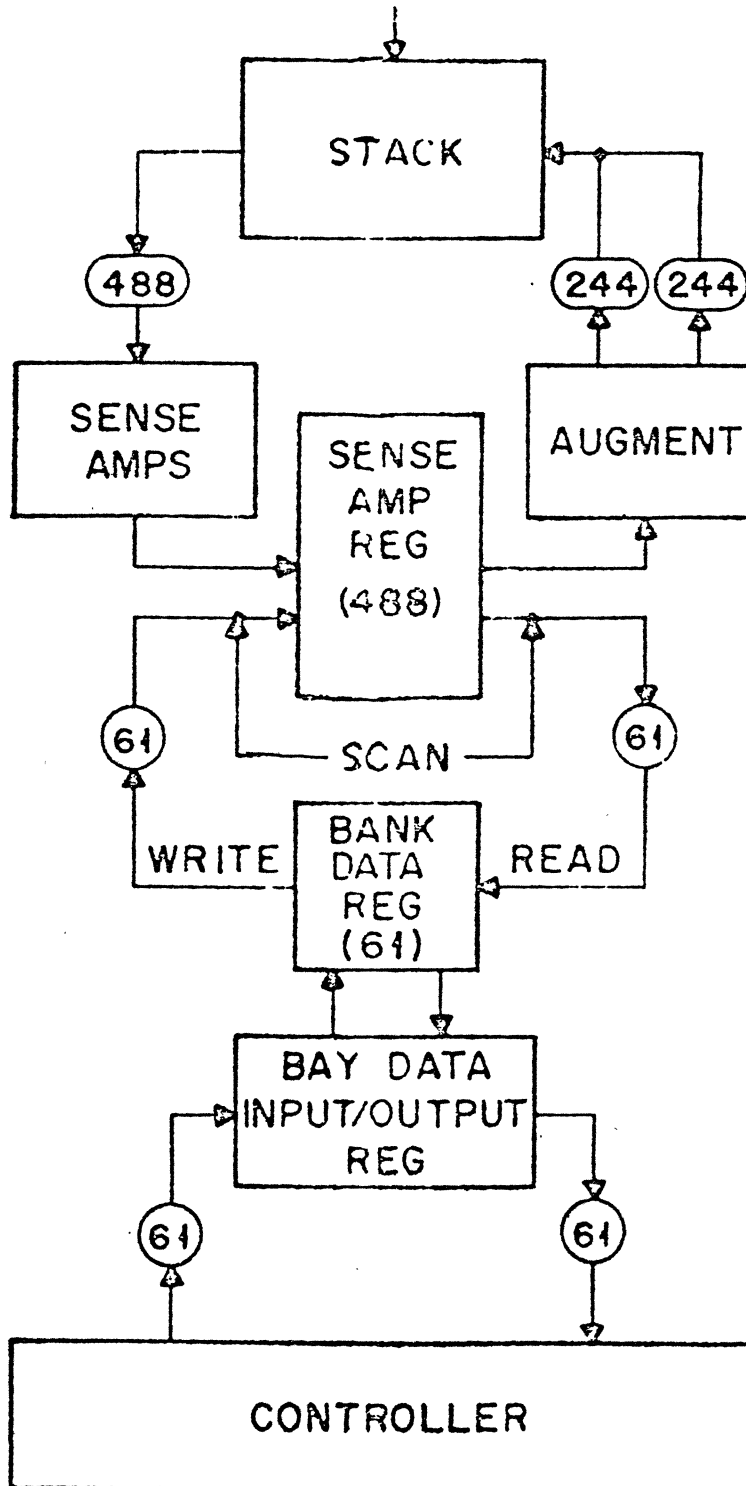
38. 400 nanoseconds after ECS continue request signal, coupler transmits three ECS bank initiate signals to CMC (third record).
39. Coupler transmits request controller signal, ECS write signal, and ECS address (00300020₈) with parity bit to controller for third record.
40. CMC transmits address (00500₈) with parity bit and go bank (12₈) signal to CM. After every 100 nanoseconds, CMC transmits address (00500₈) with parity bit and next go bank (138 and 148) signal to CM. This initiates a CM read cycle for words 0 through 2 in third record.
41. When controller request is accepted, controller transmits controller accept signal to coupler and transmits request signal, store signal, bank select (2), bank address (06000₈), and word select (0) to ECS for third record.
42. ECS starts write reference for each word select count (0 through 2) in third record.
43. When controller accept for third record is received, coupler starts write delay and increments ECS address (00300030₈).
44. CM transmits eight 60-bit words (0 through 7) with 8-bit SECDED code to CMC (second record).
45. CMC transmits eight 60-bit words, parity bits, and data on line signals to coupler. Coupler stores these second record words 0 through 7 in RAM.
46. Coupler completes write delay for second record and transmits eight 60-bit words, parity bits, and go signals from RAM to controller.
47. Controller transmits eight 60-bit words, parity bits, and write go signals to ECS. ECS writes second record words 0 through 7.
48. CM transmits three 60-bit words (0 through 2) with 8-bit SECDED code to CMC (third record).
49. CMC transmits three 60-bit words, parity bits, and data on line signals to coupler. Coupler stores these third record words 0 through 2 in RAM.
50. Coupler completes write delay for third record and transmits three 60-bit words, parity bits, and go signals from RAM to controller.
51. Controller transmits three 60-bit words, parity bits, and write go signals to ECS. ECS writes third record words 0 through 2.
52. Coupler transmits ECS end of transfer signal to CMC; this clears ECS mode of operation. Coupler transmits end of transfer signal to CPU; this causes full exit to next instruction which starts at address P+1.

SEQUENCE TABLES

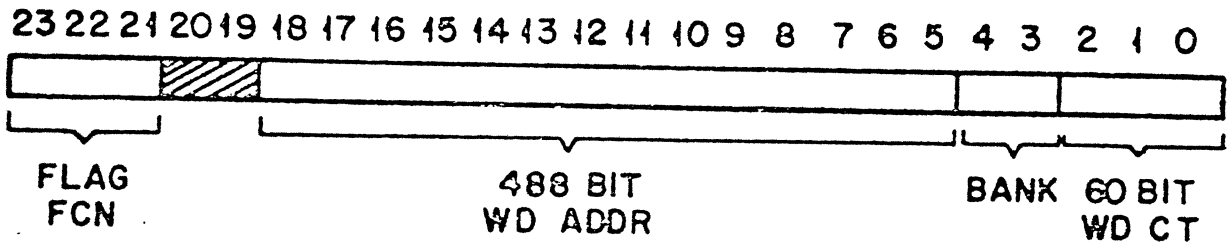
Tables 4-2-1 through 4-2-10 list the sequence of events for the ECS subsystem operations. These tables are keyed to the functional diagrams in section 5, parts 2 and 3. In addition, the tables reference specific diagrams in the individual hardware maintenance manuals listed in the preface. References to diagrams in this manual are to sheet numbers and zones. References to diagrams in other manuals are to sheet numbers (CPU, CMC, coupler, ECS) or page numbers (controller).

DATA FLOW, ECS BANK

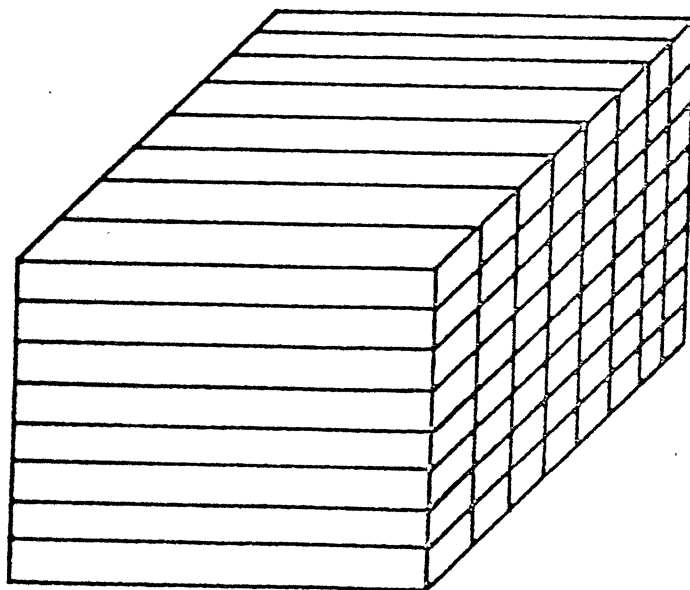
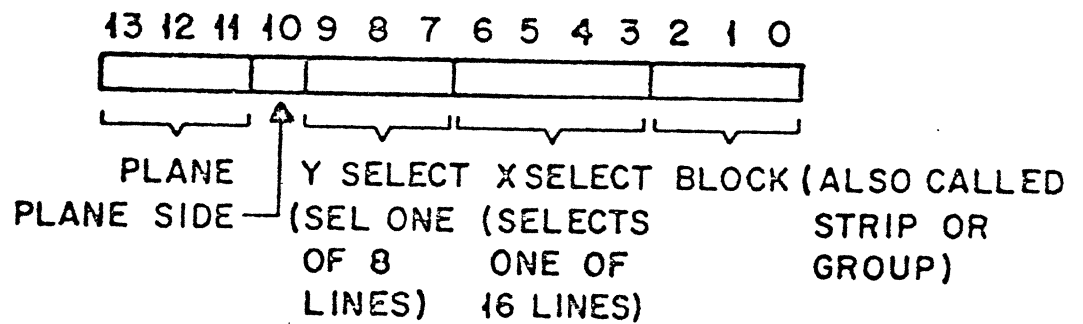
ADDRESS AND CONTROLS



524K ADDRESS FORMAT



488 BIT WD ADDR FORMAT



131K STACK (BANK)

- ◀ 131K 60 BIT WORDS
- ◀ 16 K 488 BIT WORDS

171 - 174

INTRODUCTION

SYSTEM BLOCK DIAGRAM (figure 5-1-1)

The ECS coupler is located within the mainframe cabinet. It serves as an interface between the mainframe and the ECS subsystem (ECS controller, ECS bays) by processing, monitoring and controlling data between the systems. The coupler:

- Receives a composite ECS address from the CPU. The 3 least significant bits are used to disassemble a 488-bit ECS word (block) into eight 60-bit data words (plus a parity bit for each). Remaining bits are used to locate the bay, bank and block in ECS.
- Relays this address, along with a read or write and request signal, to the ECS controller.
- Receives the word count (number of individual words to be transferred) from the CPU. This value is decremented for each word transferred to determine when the transfer has ended.
- Utilizes the complement of the 3 least significant bits from the address to determine if the last word in a block has been reached. If it has, and the word count is still greater than zero, the ECS address will be incremented by 10_8 to address the next block.
- During a read operation, generates a continue request signal to CMC to set bank reservations for CSU. Data is read out of ECS via the ECS controller.
- During a write operation, generates a bank initiate signal to the ECS controller for depositing data from CSU via CMC to the ECS via the ECS controller.
- When a transfer has been completed normally, shuts down to prepare for another transfer, and informs the CPU of this condition.
- If an error condition (parity error, illegal address, etc.) exists, terminates transfer and informs the CPU of this condition.

Other subsidiary functions of the coupler include:

- Checking parity of the word count and address from the CPU.
- Generating a parity bit for addresses sent to the ECS controller.
- Receiving flag functions from the CPU and relaying them to the ECS controller. This allows the CPU to interrogate the operating status of the ECS controller.

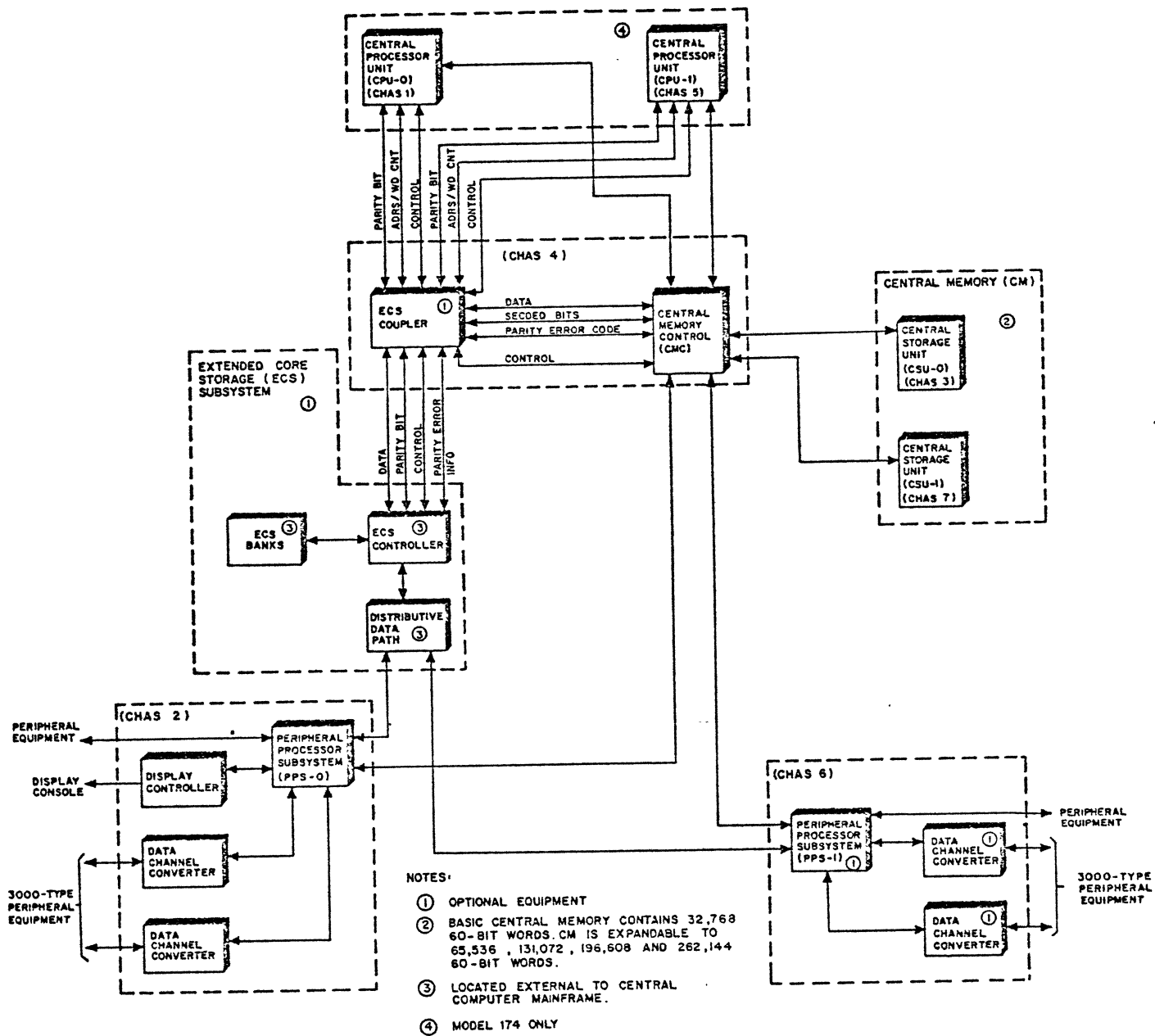
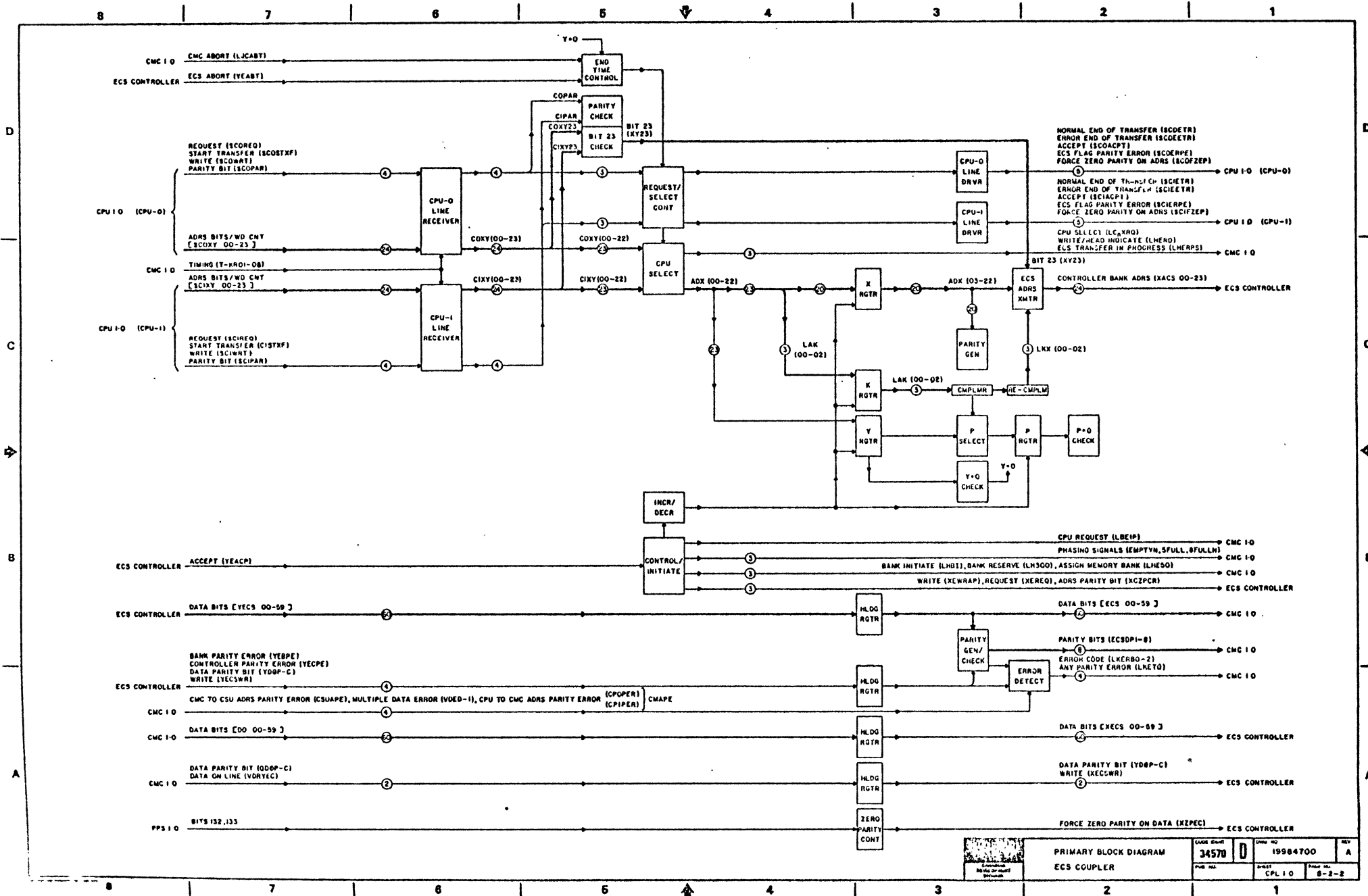


Figure 5-1-1. System Block Diagram



GENERAL DESCRIPTION

INTRODUCTION

The computer system master clock and the extended core storage (ECS) coupler option logic paks are located in bay 1 chassis 4 (1A4) of the computer mainframe. The ECS coupler option is installed only when an ECS system is purchased in addition to the computer system. Cabling required to connect the coupler between the ECS system and the central computer is provided as part of the coupler. The coupler may be installed at the factory or on site at the customer's location. The computer system provides power for both the master clock and the ECS coupler. The information contained in this section supplements the information in the hardware reference manual. Refer to the System Publication Index in the preface of this manual for the publication number.

MASTER CLOCK

PURPOSE OF EQUIPMENT

The master clock provides 25-nanosecond pulses at a 10-MHz rate to synchronously run all components of the computer system. The master clock contains a 10-MHz crystal oscillator to generate the internal master reference frequency. If the computer system has the ECS option installed, a master clock within the ECS controller generates the external master reference frequency and the master clock is slaved to this frequency. The master clock, when slaved to the ECS clock, compares the internal reference frequency against the external reference frequency to check for a failure of the external reference frequency. If a failure occurs, the internal reference frequency automatically becomes the master reference frequency.

The master clock operating in the internal mode is capable of generating frequency margins for diagnostics. The frequency margins are controlled by the bits received from the status and control (S/C) register of the peripheral processor subsystem in the computer system. Frequency margin switches, located on the master clock paks, allow manual operation of the frequency margins.

DESCRIPTION

The logic circuits for the master clock are mounted on plug-in paks located in row B of chassis 1A4. There are three toggle switches and a pushbutton switch located on two of the paks for the manual control of the margin frequency checks. The use of these switches for checkout or maintenance is covered in the computer system maintenance manual.

ECS COUPLER

PURPOSE OF EQUIPMENT

A computer system that contains the optional ECS subsystem to augment the central memory (CM) requires an ECS coupler connected as the interface between the computer system and the ECS subsystem (refer to figure 1-1). The ECS coupler controls the transfer of information from CM to the ECS subsystem memory banks (write operation) or from the ECS memory banks to CM (read operation). The read or write operation is initiated by instructions in the central processing unit (CPU) of the computer system. The central memory control (CMC) of the computer system controls the data in and out of CM and the ECS coupler. The ECS coupler controls and monitors the information to ensure a successful transfer operation. If errors occur during a transfer operation, the ECS coupler detects the errors or passes the error signal detected by other equipments to the CPU. The computer system master clock is slaved to the ECS controller clock of the ECS subsystem to provide synchronization of the computer system and the optional equipment.

The ECS coupler performs the following internal functions.

- Generates and sends the ECS address, request controller signal, and the read or write signals to the ECS controller after receiving the initial request coupler signal and the starting ECS address from the CPU.
- Checks for odd parity on the word count and ECS starting address received from the CPU.
- Generates and sends the continue request signal to the CMC 400 nanoseconds ahead of the bank initiate signal for the central memory bank reservation operation.
- Generates and sends the bank initiate signal to the CMC with each central memory word to start a memory cycle.
- Receives the starting ECS address from the CPU. Increments the ECS address from the starting address for each ECS record (eight 60-bit words/record) transferred.
- Generates and sends an odd parity bit for the ECS address to the ECS controller.
- Compares the number of words transferred with the word count received from the CPU to ensure a complete transfer of data.

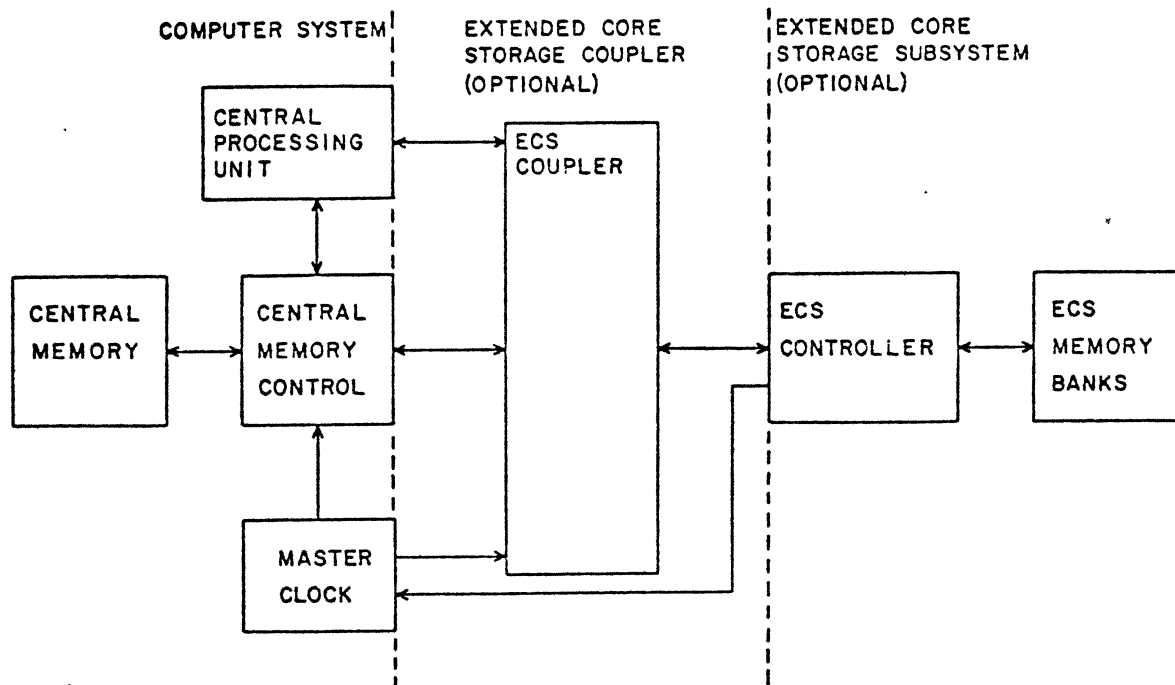


Figure 1-1. ECS Coupler Interface Configuration

- Generates and sends the end of transfer signal to the CPU when the transfer is completed without an error.
- Generates and sends the error end of transfer signal to the CPU if an error is detected during a transfer. The particular error detected determines if the transfer is terminated immediately or allowed to continue until all words are transferred.
- Generates and sends the ECS transfer error signal and a 3-bit error code to the CPU to identify the error when an error is detected during a transfer. This signal and code are routed by the CPU to the S/C register in the peripheral processor subsystem (PPS) of the computer system.
- Interprets a flag operation instruction from the CPU and sends the instruction to the ECS controller.
- Generates and sends the OK exchange CPU signal to the CPU when the coupler has completed a transfer. A CPU exchange may proceed upon receipt of the signal by the CPU.

DESCRIPTION

The logic circuits for the ECS coupler are mounted plug-in paks located in rows C through F of chassis 1A4.

PRIMARY BLOCK DIAGRAM

ECS COUPLER

The ECS coupler is the connecting link between the computer system and the optional ECS subsystem that augments the central memory (CM) of the computer system. The following paragraphs describe the interface signals used by the ECS coupler to complete a transfer operation.

CENTRAL PROCESSOR UNIT INTERFACE SIGNALS

REQUEST COUPLER

The CPU generates this signal when it has decoded either a read ECS (011 instruction) or a write ECS (012 instruction) and needs the coupler to transfer information. The coupler uses the signal to prepare its logic circuits for the receipt of the transfer word count and the starting ECS address for the information.

COUPLER ACCEPT

The coupler generates this signal if it is not busy with a previous request and is ready to receive the word count and starting ECS address to start the transfer.

WORD COUNT

The word count bits (0 through 17) plus one odd parity bit (bit 24) indicate the number of 60-bit words in the transfer operation.

ADDRESS

The address bits (0 through 23) plus one odd parity bit (bit 24) indicate the starting ECS address for the transfer operation.

ECS WRITE

The CPU sends this signal to the coupler when an 012 instruction is decoded and data is to be written into ECS. The absence of this signal indicates an ECS read (011 instruction).

CPU EXCHANGE REQUEST

The CPU generates this signal if the CMC receives an exchange request from the peripheral processor subsystem or the CPU. The receipt of this signal in the coupler terminates ECS transfer to permit the CPU to exchange if: ECS is doing a flag register operation, CMC is in ECS mode, or the last record of the transfer is being or has been transmitted to ECS.

ECS TRANSFER ERROR AND CODE 0 THROUGH 2

The coupler error code generator generates this signal if an error is detected during a transfer operation. The coupler sends the transfer error signal with the 3-bit error code for the error detected to the CPU along with the error end-of-transfer signal.

5-3-0.0

FORCE ZERO PARITY ON ADDRESS/WORD COUNT

The coupler sends this signal to the CPU to force zero parity on the address and word count for the coupler. This is used to check the capability of the coupler to detect a parity error on the address and word count.

ERROR END OF TRANSFER

The coupler sends this signal to the CPU when an error was detected during the transfer operation. Certain errors cause the transfer operation to abort at the end of the record while others allow the transfer to be completed.

Errors aborting the transfer at the end of the record are:

- Word count or address parity error (CPU to CPLR).
- Address parity error (CPLR to ECS controller).
- ECS bank addressed is not available due to maintenance mode or loss of power. (This error does not cause an abort during an ECS read operation.)

Errors allowing the transfer to complete are:

- Address parity error (CMC to CSU).
- Data parity error (CMC to ECS controller).
- Data parity error (ECS controller to CMC).
- Data parity error (ECS controller to ECS memory).
- Double error detected in data read from CM.

END OF TRANSFER

The coupler sends this signal to the CPU when the transfer operation is completed normally.

FORCE ZERO PARITY CODE 0 AND 1

The CPU sends these code bits to the coupler for translation. The code is translated for zero parity instructions to the ECS controller or the CPU.

OK EXCHANGE CPU

The coupler has terminated transfer; CPU exchange may proceed.

CENTRAL MEMORY CONTROL INTERFACE SIGNALS

ECS REQUEST CMC

The coupler sends this signal to request CM access after the coupler has received the address and word count from the CPU.

CMC ACCEPT

The CMC sends this signal to the coupler when the CM access request (ECS request CMC) has been accepted. The coupler must send the ECS continue request signal and bank initiate signal after the CMC accept to start the CM memory cycle.

ECS CONTINUE REQUEST

The coupler sends this signal to the CMC to reserve the CM banks for the transfer operation.

ECS BANK INITIATE

The coupler sends this signal to the CMC 400 nanoseconds after the ECS continue request signal to cause the CMC to send the address and a go signal to CM and start a CM memory cycle.

ECS END OF TRANSFER.

The coupler sends this signal to the CMC when the last record has been received. The signal clears the ECS in progress status in the CMC.

ECS FLAG PARITY ERROR

The coupler sends this signal to the CMC when a parity error is detected in the address received by the ECS controller during a flag register operation.

DATA ON LINE (GO ECS)

This signal accompanies the CM read data to the coupler to indicate the data lines contain valid information.

CSU ADDRESS PARITY ERROR

The coupler receives this signal when an address parity error occurs on the CMC address to the CSU for an ECS related CM reference.

CMC DOUBLE ERROR

The coupler receives this signal when the single error correction double error detection (SECDED) circuit detects a double error on data read from CM during an ECS transfer operation.

CMC INPUT ERROR

The coupler receives this signal when a parity error is detected on the data received by the CMC from ECS.

DATA (CM READ) BITS 0 THROUGH 59, PARITY

These are the 60 data bits read from CM and their odd parity bit for an ECS write operation.

DATA (CM WRITE) BITS 0 THROUGH 59, PARITY

These are the 60 data bits to be written into CM and their odd parity bit for an ECS read operation.

ECS CONTROLLER INTERFACE SIGNALS

REQUEST CONTROLLER

The coupler receives this signal prior to each record of data to be transferred. The first and last records may not be full records but they still require request controller signals.

CONTROLLER ACCEPT

The controller sends this signal to the coupler when it is ready for a transfer of one record.

ECS ADDRESS BITS 0 THROUGH 23

The coupler sends these bits containing the bank address to the controller for the ECS record to be transferred. The address bits are transmitted at the same time as the request controller signal.

ECS ADDRESS PARITY BIT

The coupler sends the odd parity bit with the address bits to the controller.

ECS WRITE

The coupler sends this signal to the controller to indicate the data transfer is an ECS write operation. The ECS write is transmitted at the same time as the request controller signal. The absence of this signal indicates the transfer operation is an ECS read operation.

CONTROLLER ABORT.

The controller sends this signal to the coupler to indicate the data transfer was aborted. The transfer is aborted by the controller due to one or more of the following reasons.

- ECS bank address referenced does not exist.
- ECS bank is in maintenance mode.
- ECS bank does not have power.
- Address parity error has been detected at the controller.

FORCE ZERO PARITY ON DATA

The coupler sends this signal to the controller to force zero parity on data transferred to the CMC. This tests the capability of the parity error detection circuit in the CMC to detect parity errors.

CONTROLLER PARITY ERROR

The controller sends this signal to the coupler when the controller detects a parity error on the address or data received from CMC. If an address parity error is detected, the controller sends an abort signal at the same time as the parity error signal.

ECS BANK PARITY ERROR

The controller sends this signal to the coupler when a parity error is detected on data going to or from the ECS banks.

DATA (ECS WRITE) BITS 0 THROUGH 59

The coupler sends 60 bits of data to the controller during an ECS write operation.

DATA (ECS READ BITS 0 THROUGH 59)

The controller sends 60 bits of data to the coupler during an ECS read operation.

DATA (ECS READ) PARITY.

The controller sends an odd parity bit to the coupler with the data transferred during an ECS read operation.

DATA (ECS WRITE) PARITY, GO

The coupler sends an odd parity bit to the controller with the data transferred during an ECS write operation. A go signal to the controller referred to as a data-on-the-line signal, indicates valid transfer data.

GENERAL DESCRIPTION

The ECS coupler is activated when the central processor initiates a read or write ECS operation and issues a request coupler signal to the coupler. When the coupler receives the request, it sends a coupler accept signal to the CPU. The CPU sends the word count 350 nanoseconds after it receives the coupler accept signal and 50 nanoseconds later it sends the ECS starting address. The word count enters the word count control Y register and the address enters the X and K registers. The CPU sends an ECS write signal at the same time as the request coupler signal for an ECS write operation (012 instruction). The absence of the write signal or a not write signal indicates an ECS read operation (011 instruction).

The coupler transmits the request controller signal, ECS starting address, and the ECS write signal (if present), to the ECS controller. If the ECS is not busy performing another ECS operation, the ECS controller sends a controller accept signal to the coupler. If this is an ECS write operation, the coupler responds to the controller accept signal by issuing an ECS continue request signal and 400 nanoseconds later issues an ECS bank initiate signal to CMC. In an ECS read operation, the ECS continue request signal is not issued for 1300 nanoseconds and then 400 nanoseconds later the ECS bank initiate is issued. In an ECS write operation, data is passed from the CMC through the 16-word random access memory (RAM) data buffer, the data registers and then transmitted to the ECS controller. During an ECS read operation, data passes directly through the coupler via the data registers. In either mode of operation the transmission of data continues until the transfer is completed or an error signal is detected by the coupler. If an error is detected during an ECS read operation and an error signal is generated, the remaining words transferred to the CMC are all zeros. If the error is detected in an ECS write operation and an error signal generated, the request controller signal is blocked to end the data transfer and prevent data from possibly being written into an incorrect address and destroying other data.

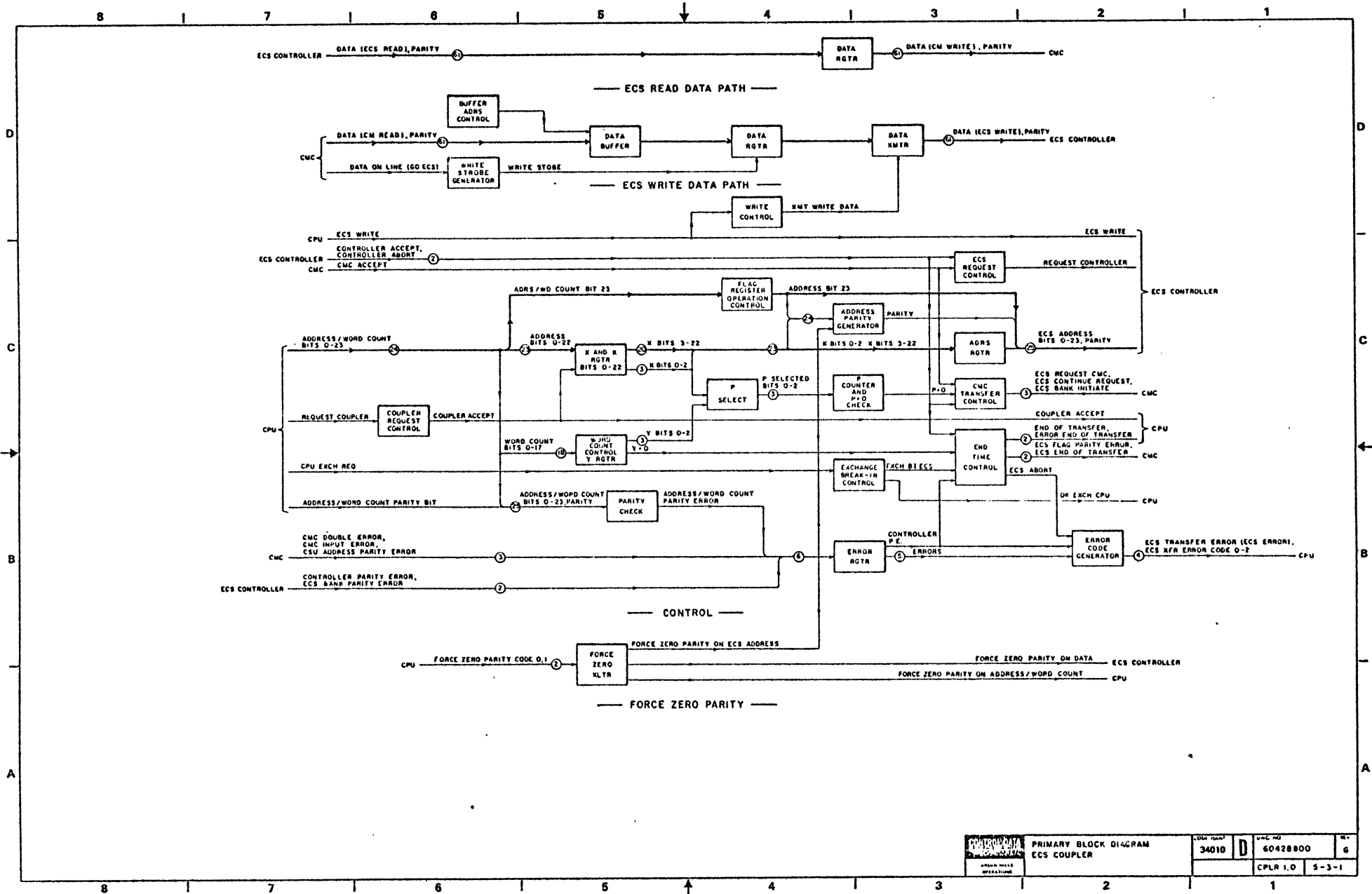
The 18 bit Y register keeps track of the word count and decrements each time a word is received. The decrementing of the Y register continues until Y is reduced to zero and the end time control circuit activates to end the transfer operation.

The CPU provides the 23-bit ECS starting address to the X and K registers for the transfer operation. The address register is incremented in octal addresses of ten for each location of a record in ECS. The 24th address bit is not used by the X register as part of the address but is routed through the coupler and used by the ECS controller for a flag operation.

The Y and K registers provide the P select with the lower 3 bits of both the address and the word count. The P select circuit output switches from the K bits to the Y bits when the coupler transmits the last record. The P counter and P equals zero check circuit controls the number of words transferred during each record.

The error register receives external error signals from the CMC, CPU, and ECS controller and internally a parity error signal if the address/word count from the CPU has a parity error. The error signals are converted to an error code that is transmitted to the status and control register via the CPU.

The status and control register via the CPU sends the coupler a force zero parity code during diagnostics to force zero parity to the ECS controller and the CPU to create a parity error. The error detection circuits should detect the parity error and return an error signal to the error register in the coupler.



DETAILED-PAK DIAGRAM

CPU INTERFACE FOR COUPLER REQUEST CONTROL, ADDRESS/WORD COUNT, AND ECS WRITE

COUPLER REQUEST CONTROL

An ECS operation begins when the CPU sends a request coupler signal to the ECS coupler. For the request for a write ECS operation (012 instruction), the request coupler signal is accompanied by an ECS write signal. The absence of the ECS write signal (not ECS write) indicates a read ECS operation (011 instruction).

The request coupler signal sets a latch register (3KR0 pak) which in turn sets the request hold flip-flop (5LB0 pak). The request hold flip-flop holds the request coupler signal if the request flip-flop is not in a reset mode (clear). The request flip-flop is cleared by the master clear signal or the end-of-transfer clear signal generated at the completion of each ECS operation. If the request flip-flop is clear, the output signal of the request hold flip-flop sets the request flip-flop at time T1, 3. The output signal of the request flip-flop disables its own clock input, clears the request hold flip-flop in preparation for the next request, and sets the ECS IP (in progress) flip-flop. The request flip-flop and the ECS IP flip-flop are connected in a one-shot pulse-forming combination. In this combination, the request flip-flop sets on the trailing edge of a clock pulse and forms the leading edge of the coupler accept signal. The ECS IP flip-flop sets 50 nanoseconds later on the trailing edge of the next clock pulse. Since the inverted output signal of the ECS IP flip-flop is ANDed with the output signal of the request flip-flop, the setting of the ECS IP flip-flop breaks the AND gate and forms the trailing edge of the 50-nanosecond coupler accept pulse. The request flip-flop and the ECS IP flip-flop remain set until cleared by the end-of-transfer clear signal at the completion of the transfer operation or the master clear signal.

The coupler accept pulse clears the error register (CPLR 3.7) of any errors detected in the previous transfer operation and indicates to the CPU, via a latch register and transmitter (4KT0 pak), that the coupler accepts the request for the transfer operation. The accept delay chain (5KS0 pak) delays the coupler accept pulse 350 and 550 nanoseconds. The register enable and request CMC circuits use the two delayed coupler accept pulses as timing signals.

ADDRESS/WORD COUNT

The CPU sends the transfer operation word count and its odd parity bit to the coupler 350 nanoseconds after receiving the coupler accept signal. The word count enters the latch registers which in turn transfer the count at time T1, 3 to the Y register (CPLR 3.1). The parity checker (3KR0 pak) checks the word count in the latch registers in three 8-bit groups for parity errors. The parity checker generates a parity bit for each of the three groups. These parity bits, designated P1 through P3, are merged with the parity bit that accompanied the word count from the CPU in a second parity check circuit (4KQ0 pak). When the 4KQ0 pak parity check circuit detects a parity error in the parity bits, it generates a word count parity error signal for the error register (CPLR 3.7). The parity error signal transfers to the error register via a timing select circuit on the register enable pak (CPLR 3.1).

The CPU sends the transfer operation ECS starting address and its parity bit to the coupler 50 nanoseconds after the word count, which is 400 nanoseconds after the CPU receives the coupler accept signal. The address enters the latch registers (3KR0 pak) which in turn transfer the lower address bits 0 through 22 to the X and K registers (CPLR 3.1). The two parity checkers (3KR0 and 4KQ0 paks) check the address and the address parity bit for parity errors in the same manner they check

the word count for parity errors. The parity check circuit (4KQ0 pak) transfers an address parity error signal via a timing select circuit on the register enable pak (CPLR 3.1) to the error register (CPLR 3.7).

The CPU sends the ECS starting address with bit 23 set when it wants the ECS controller to perform a flag register operation. The latch register (3KR0 pak) transfers address bit 23 at time T1, 3 to the flag register operation flip-flop (CPLR 3.3). The flag register operation flip-flop merges address bit 23 back with the lower 23 bits of the address in the address register (CPLR 3.1) and performs control functions on coupler circuits for the duration of the flag register operation. The ECS controller performs the flag register function using bits 20 through 22 of the address as the instructions. The controller accomplishes the flag register operation without further instructions or information.

CPLR 3.0 TEST POINTS

Module	Location	Test Point	Description
3KR0	4E19-21	3 (T)	Adrs/Wd Count Bit N
3KR0	4E19-21	5 (T)	Adrs/Wd Count Bit N + 1
3KR0	4E19-21	2 (T)	Adrs/Wd Count Bit N + 2
3KR0	4E19-21	6 (T)	Adrs/Wd Count Bit N + 3
3KR0	4E19-21	13 (T)	Adrs/Wd Count Bit N + 4
3KR0	4E19-21	9 (T)	Adrs/Wd Count Bit N + 5
3KR0	4E19-21	12 (T)	Adrs/Wd Count Bit N + 6
3KR0	4E19-21	8 (T)	Adrs/Wd Count Bit N + 7
3KR0	4E19-21	7 (T)	Clock Time T1, 3
3KR0	4E18	3 (T)	Adrs/Wd Count Parity Bit
3KR0	4E18	1 (F)	Adrs/Wd Count Parity Bit
3KR0	4E18	5 (T)	ECS Write
3KR0	4E18	6 (T)	Req Cplr
3KR0	4E18	7 (T)	Clock Time T1, 3
4KQ0	4E34	10 (T)	Parity Bit P1
4KQ0	4E34	9 (T)	Parity Bit P2
4KQ0	4E34	8 (T)	Parity Bit P3
5LB0	4D36	3 (T)	Req Cplr
5LB0	4D36	2 (F)	MC + EOT CLR
5LB0	4D36	1 (T)	Clock Time T1, 3
5KS0	4E33	5 (F)	Dly Cplr Acpt (350NS)
5KS0	4E33	4 (T)	Dly Cplr Acpt (550NS)
5KS0	4E33	13 (T)	Clock Time T1, 3
4KT0	4F20	9 (T)	Cplr Acpt
4KT0	4F20	2 (T)	Clock Time T1, 3

DETAILED-PAK DIAGRAM

CPU INTERFACE AND EXCHANGE BREAK-IN CONTROL

COUPLER REQUEST CONTROL

An ECS operation begins when the CPU sends a request coupler signal to the ECS coupler. If the request is for a write ECS operation (012 instruction), the request coupler signal is accompanied by an ECS write signal. The absence of the ECS write signal (not ECS write) indicates a read ECS operation (011 instruction).

The request coupler signal sets a latch register (3KR0 pak) which in turn sets the request hold flip-flop (5LB0 pak). The request hold flip-flop holds the request coupler signal if the request flip-flop is not in a reset mode (clear). The request flip-flop is cleared by the master clear signal or the end-of-transfer clear signal generated at the completion of each ECS operation. If the request flip-flop is clear, the output signal of the request hold flip-flop sets the request flip-flop at time T1, 3. The output signal of the request flip-flop disables its own clock input, clears the request hold flip-flop in preparation for the next request, and sets the ECS IP (in progress) flip-flop. The request flip-flop and the ECS IP flip-flop are connected in a one-shot pulse-forming combination. In this combination, the request flip-flop sets on the trailing edge of a clock pulse and forms the leading edge of the coupler accept signal. The ECS IP flip-flop sets 50 nanoseconds later on the trailing edge of the next clock pulse. Since the inverted output signal of the ECS IP flip-flop is ANDed with the output signal of the request flip-flop, the setting of the ECS IP flip-flop breaks the AND gate and forms the trailing edge of the 50-nanosecond coupler accept pulse. The request flip-flop and the ECS IP flip-flop remain set until cleared by the end-of-transfer clear signal at the completion of the transfer operation or the master clear signal.

The coupler accept pulse clears the error register (CPLR 3.7) of any errors detected in the previous transfer operation and indicates to the CPU, via a latch register and transmitter (4KT0 pak), that the coupler accepts the request for the transfer operation. The accept delay chain (5KS0 pak) delays the coupler accept pulse 350 and 550 nanoseconds. The register enable and request CMC circuits use the two delayed coupler accept pulses as timing signals.

ADDRESS/WORD COUNT

The CPU sends the transfer operation word count and its odd parity bit to the coupler 350 nanoseconds after receiving the coupler accept signal. The word count enters the latch registers which in turn transfer the count at time T1, 3 to the Y register (CPLR 3.1). The parity checker (3KR0 pak) checks the word count in the latch registers in three 8-bit groups for parity errors. The parity checker generates a parity bit for each of the three groups. These parity bits, designated P1 through P3, are merged with the parity bit that accompanied the word count from the CPU in a second parity check circuit (4KQ0 pak). When the 4KQ0 pak parity check circuit detects a parity error in the parity bits, it generates a word count parity error signal for the error register (CPLR 3.7). The parity error signal transfers to the error register via a timing select circuit on the register enable pak (CPLR 3.1).

The CPU sends the transfer operation ECS starting address and its parity bit to the coupler 50 nanoseconds after the word count, which is 400 nanoseconds after the CPU receives the coupler accept signal. The address enters the latch registers (3KR0 pak) which in turn transfer the lower address bits 0 through 22 to the X and K registers (CPLR 3.1). The two parity checkers (3KR0 and 4KQ0 paks) check the address and the address parity bit for parity errors in the same manner they check

the word count for parity errors. The parity check circuit (4KQ0 pak) transfers an address parity error signal via a timing select circuit on the register enable pak (CPLR 3.1) to the error register (CPLR 3.7).

The CPU sends the ECS starting address with bit 23 set when it wants the ECS controller to perform a flag register operation. The latch register (3KR0 pak) transfers address bit 23 at time T1, 3 to the flag register operation flip-flop (CPLR 3.3). The flag register operation flip-flop merges address bit 23 back with the lower 23 bits of the address in the address register (CPLR 3.1) and performs control functions on coupler circuits for the duration of the flag register operation. The ECS controller performs the flag register function using bits 20 through 22 of the address as the instructions. The controller accomplishes the flag register operation without further instructions or information.

EXCHANGE BREAK-IN CONTROL

The CPU sends a CPU exchange request signal to the coupler to initiate an exchange break in ECS. The signal is initiated to perform an exchange and to halt ECS transfer if possible. The CPU exchange request sets the request exchange break-in (BI) flip-flop via the one-shot pulse forming network. The stop transfer flip-flop sets if:

- ECS not doing a flag register operation, or
- CMC is in ECS mode, or
- The last record of the transfer is being (near end of transfer) or has been transmitted to ECS.

If one of the above conditions is present, the transfer is allowed to complete before the CPU exchange is performed.

The stop transfer flip-flop blocks further requests to the ECS controller and blocks controller accepts from being received by the coupler. This stops the transfer at the end of the current record.

The exchange break-in flip-flop sets if there are no unanswered controller requests. (The controller request outstanding flip-flop sets with a request ECS controller and it clears with a controller accept or controller abort signal.) The exchange break-in flip-flop sends an exchange break-in signal to the end time control to terminate the transfer.

The end of transfer clear signal then causes the CPU exchange abort flip-flop to set. The OK exchange CPU signal is transmitted to the CPU to indicate that the ECS transfer is terminated and that the CPU exchange may proceed.

DETAILED-PAK DIAGRAM

REGISTER ENABLE K, X, Y REGISTERS AND P SELECT

Y REGISTER (WORD COUNT)

The CPU sends the 23-bit word count for the data transfer operation 350 nanoseconds after it receives the coupler accept signal. The Y register must be in a preset mode at that time to load the data word count. The Y register monitors and outputs the data word count to the coupler control circuits for the duration of the transfer operation.

The input signals to functional control inputs S1 and S2 of the Y register (3LA0 pak) control its mode of operation. The 50-nanosecond coupler accept signal that was delayed 350 nanoseconds is passed directly through the register enable pak (1PACpak) to input S2 of the Y register. The delayed coupler accept (350 nanoseconds) signal is delayed an additional 50 nanoseconds by the 50-nanosecond delay flip-flop (1PAGpak) before providing an input to S1 of the Y register. Since the output signal of the 50-nanosecond delay flip-flop (that goes to input S1 of the Y register) is inverted, the Y register is in a preset mode until the flip-flop sets. The Y register remains in the preset mode for 50 nanoseconds (350 to 400 nanoseconds after the coupler accept signal). The data transfer operation word count bits 0 through 22 arrive at the Y register 350 nanoseconds after the coupler accept signal and enter the register. The data word count is contained in the lower 19 bits. The upper 4 bits are ignored.

The 50-nanosecond delay flip-flop sets 400 nanoseconds after the coupler accept signal, forcing the Y register into a hold mode. The Y register remains in the hold mode for 50 nanoseconds while the X register loads the ECS starting address. When the 50-nanosecond delay flip-flop clears, it forces the Y register into a decrement mode. Since the Y register contains the number of data words in the ECS transfer operation and is in a decrement mode, it is ready to monitor the data transfer count when enabled. The enable signal to the Y register is the result of the ECS controller accepting the first transfer operation request or the previous transfer of eight 60-bit data words (record). The coupler generates the go central signal from the ECS controller accept signal. The go central signal is ANDed with the 100-nanosecond square wave and generates the enable register clock signal. The clock signal is synchronized with the clock pulse (3AA0 pak) to enable the clock input of the Y register. The clock synchronized register enable signal enables the Y register at the transfer rate of once every 100 nanoseconds unless stopped by the loss of the go central signal. The enable signal decrements the data word count in the Y register at the data transfer rate until the count equals zero.

The 3CM0 and 6LC0 paks monitor the data word count from the Y register for a word count of zero. The complemented Y register bits 4 through 18 are ANDed in the 3CM0 pak which generates the Y register count is less than 16 signal. The Y bits 0 through 2 equal zero test circuit (6LC0 pak) generates an output signal when the bits equal zero. The Y is less than 16 signal and the Y bits 0 through 2 equals zero signal are ANDed with the complement of bit 3 from the Y register. When the word count in the Y register is at zero, all these signals are present, and the AND gate generates the Y equals zero signal that sets the word count equals zero flip-flop. During an ECS write operation, the controller accept signal gates the word count equals zero flip-flop output signal to the end time circuits that start the timing sequence to end the transfer operation. The coupler receives the controller accept signal after the transfer of the last data word that caused the Y register to go to a zero word count. This ensures that the controller receives the last word of the transfer operation before the end time circuits receive the Y equals zero signal to start the timing sequence and end the transfer operation.

During an ECS read operation, the last word of the transfer operation has left the ECS controller when the word count reaches zero. Therefore, the ECS read signal gates the Y equals zero signal to start the timing sequence and end the transfer operation.

X AND K REGISTERS (ADDRESS)

The coupler X and K registers receive the ECS starting address for the transfer operation from the CPU. The register control circuits increment the X and K registers from the starting address to generate the additional ECS addresses for the data of the transfer operation.

The ECS memory is organized in records (eight 60-bit words) of data. Memory references to ECS normally require a specific address for each record. The exception is the first record of a transfer operation which could contain less than the normal eight words and is referred to as a partial record. The X register provides the normal record address while the K register provides the address of a word within the first record.

The CPU sends the 23-bit starting address 400 nanoseconds after it receives the coupler accept signal. The X and K registers (3LA0 pak) must be in a preset mode 400 nanoseconds after the coupler accept signal to load the ECS starting address.

The coupler accept signal that was delayed 350 nanoseconds is delayed an additional 50 nanoseconds by the 50-nanosecond delay flip-flop (1PAGpak). The delay flip-flop provides the delayed coupler accept signal (400 nanoseconds) to functional control input S1 of the X register for 50 nanoseconds. Since the complemented delayed coupler accept signal (350 nanoseconds) to input S2 of X register has ended, S1 and S2 are in the proper state to preset the register.

The same delayed coupler accept signal (400 nanoseconds) provided to S1 of the X register is provided to S1 of the K register. The delayed coupler accept signal (400 nanoseconds) from the 50-nanosecond delay flip-flop is ANDed with the 100-nanosecond square wave signal to provide an input to S2 of the K register. Inputs S1 and S2 of the K register are in the proper state to preset the K register at the same time the X register presets.

The X and K registers are in the preset mode for 50 nanoseconds (400 to 450 nanoseconds after the coupler accept signal). During this time, the address bits 0 through 22 arrive and load into the X and K registers. The ECS address uses all 23 bits.

When the 50-nanosecond delay flip-flop clears, the X register goes into the increment mode and the K register goes into the hold mode. The registers remain in their respective modes until the ECS controller returns a controller accept signal to indicate it is ready for the transfer of data and requires the ECS address. After receiving the controller accept signal, the coupler generates the go ECS and go central signals. The go ECS is necessary to start the 100-nanosecond square wave signal. The go central and the 100-nanosecond square wave combine to form the K register S2 signal that alternates the K register from the hold to the increment mode.

With both registers in the increment mode, the register enable signal to the clock input increments the address in the registers. The go central signal is ANDed with the 100-nanosecond square wave and generates the enable register clock signal. The clock signal is synchronized with the clock pulse (3AA0 pak) to enable the clock input of the X and K registers. The clock synchronized register enable signal increments the X and K registers at the transfer rate of once every 100 nanoseconds unless stopped by the loss of the go central signal. The incrementing continues until the transfer is complete and the end time circuits stop the go central signal.

The K register contains zeros except when the first record of a transfer operation is a partial record. The K register allows references to specific 60-bit word locations for the number of words contained in the partial record. The delayed coupler accept signal (550 nanoseconds) gates the first record K register bits (lower 3 bits of the address) into the LAK register (5LK0 pak). The ECS accept signal from the ECS controller clears the LAK register. The ECS accept signal indicates the ECS controller received the first address of the transfer. Since the delayed coupler accept signal (550 nanoseconds) occurs only once every transfer operation, only the lower address bits of the first address from the K register are gated into the LAK register. All remaining addresses will have zeros in the lower 3 bits. Therefore, each time the address registers increment one, the address to ECS increments ten (octal).

The request controller signal gates the address bits into the address register (4AP0 pak). The 100-nanosecond square wave and the T25 clock pulse (25-nanosecond square wave) gate the address bits in the address registers to the transmitter. The T25 clock pulse synchronizes the address bits within the 100-nanosecond square wave for transfer to the ECS controller. The clock time T3 resets the address register in preparation for the next address from the X and K registers.

PARITY GENERATOR

The address bits 0 through 23 provided to the address register (4AP0 pak) for transmission to the ECS controller are also provided to the parity generator (3LD0 pak). The parity generator generates an odd parity bit for each address and transfers the parity bit via the zero parity circuits (CPLR 3.3) to the ECS controller.

ADDRESS/WORD COUNT PE ENABLE

The address/word count parity error (if detected) requires an enable signal synchronized with the appropriate address or word count to pass the error signal on to the error register. The coupler accept signal (350 nanoseconds) gates a word count parity error signal (1PAG pak) through the select circuit. The delayed coupler accept signal (400 nanoseconds) from the 50-nanosecond delay flip-flop gates an address parity error through the other half of the select circuit. These gating signals perform the synchronizing function that differentiates a word count parity error from an address parity error for the error register.

P SELECT

The P counter (CPLR 3.4) keeps track of the number of words in the record transferred. The P select circuit (6LC0 pak) provides the P counter with the appropriate bits to accomplish that function. The K register (3LA0 pak) provides the complemented lower 3 bits of the address to the Y minus 1 is greater than K test circuit and the P select circuit. The K register contains the lower three address bits which are normally zero except in the case of a partial first record. Since the K register is normally zero, the complemented bits provide a seven to the Y minus 1 is greater than K test circuit and the P select circuit. The Y register (3LA0 pak) provides the 3 bits of the word count to the Y minus 1 circuit (6LC0 pak) which in turn provides its output to the Y minus 1 is greater than K test circuit and the P select circuit. The Y minus 1 is greater than K test circuit controls the P select flip-flop that selects the appropriate bits for the P counter. The go ECS signal that indicates the CMC is ready for a data transfer gates the P select flip-flop. Assuming that Y (data word count) is greater than eight, there is at least part of another record to be transferred. The P select flip-flop selects the K bits (seven) for the P register. The P counter decrements from the number provided by the selected bits at the data transfer rate of one a clock period (100 nanoseconds) until P equals zero indicating the end of one record. The count and bit selections are repeated until Y minus 1 is less than K. When Y minus 1 is less than K, the last record is ready for transfer and the P select flip-flop selects the Y minus 1 value as the selected P register bits. When the P counter decrements to zero this time, the Y count equals zero signal sets the word count equals zero flip-flop. The word count equals zero flip-flop generates the WC=0 signal and the Y equals zero signal.

DETAILED-PAK DIAGRAM TRANSFER CONTROL

The basic control center of the coupler during a transfer operation is the transfer control circuit (1PAG pak). When the CPU initiates a data transfer operation, the coupler must generate a request for the CMC. To generate an ECS request CMC signal, the flag register operation signal must not be present, the Y register must contain a word count, and a parity error must not have been detected in the word count or starting ECS address. When these conditions are available, the Y not equal to zero signal, the not error signal, and the complemented not flag register operation signal are ANDed with the delayed coupler accept (550 nanoseconds) to generate the ECS request CMC signal. The ECS request CMC signal is a 50-nanosecond pulse that occurs once during each transfer operation, 550 nanoseconds after the coupler accepts a transfer operation request from the CPU. The ECS request CMC signal is transmitted to the CMC via a latch register (4KT0 pak) and transmitter at time T1, 3.

The ECS request CMC is not generated if the flag register operation signal is present, Y is equal to zero, or an address/word count parity error signal is present, but an end data transfer signal is generated. The end data transfer signal, 550 nanoseconds after the coupler accepts a transfer operation, sets the end time flip-flop (CPLR 3.7) to end the transfer operation before a data transfer occurs. The ending of the transfer operation before a transfer of data occurs does not affect a flag register operation as it does not require a transfer of data. The word count in the Y register is zero for a flag register operation. The ending of a transfer operation by a starting address/word count parity error prevents data from transferring to an incorrect address in ECS and possibly destroying other data.

If a flag register operation is requested by the CPU, the address received by coupler has bit 23 set. Address bit 23 sets the flag register operation flip-flop (CPLR 3.3) that generates the flag register operation signal. The ECS controller requires only the address with bit 23 set to perform a flag register operation. The flag register operation signal is ANDed (1PAG pak) with the coupler accept signal that was delayed 550 nanoseconds and synchronized by time T1, 3 to set the accept go flip-flop. The accept go flip-flop generates the go ECS signal. The go ECS signal sets the go ECS flip-flop (6LG0 pak) that is connected with the request controller flip-flop in a pulse forming combination. The go ECS flip-flop sets on the trailing edge of a clock pulse and forms the leading edge of the request controller signal. The request controller flip-flop sets 50 nanoseconds later on the trailing edge of another clock pulse. The setting of the request controller flip-flop breaks the pulse forming AND gate and forms the trailing edge of the request controller signal. The request controller signal is required during a flag register operation since the ECS controller performs the flag operation internally. The Y register is equal to zero indicating no word count since a transfer of data is not required. The complemented flag register operation signal (1PAG pak) prevents making the pulse forming AND function that generates the ECS request CMC signal since the CMC is not required.

The CMC sends the CMC accept signal when central memory is ready for the requested data transfer operation. The CMC accept signal proceeds through the receiver and latch register (3KR0 pak) and unconditionally sets the accept go flip-flop (1PAG pak). The accept go flip-flop generates the go ECS signal that gates the P select circuit (CPLR 3.1), enables the buffer address circuit (CPLR 3.6), and sets the go ECS flip-flop (6LG0 pak).

The go ECS flip-flop sets at time T1 and sends its output signal to the request controller flip-flop. The request controller flip-flop is initially in a reset mode at the start of each transfer operation. The output signal of the go ECS flip-flop is ANDed with the complemented output signal of the request controller flip-flop in a pulse forming network that generates the request controller signal, unless an abort block signal is present. The request controller flip-flop sets at the next time T3 after the go ECS signal is available. The setting of the request controller flip-flop breaks the AND gate and ends the request controller signal. Since the go ECS signal is generated at time T1 and the request flip-flop set at the next time T3, the request coupler accept signal duration is 50 nanoseconds. The request controller pulse gates the address bits from the X and K registers into the address register (CPLR 3.0) for transmission to the ECS controller. The go ECS signal is also used by the buffer control (CPLR 3.4) and the end time control (CPLR 3.7) circuits.

The request controller flip-flop is initially in the reset mode as the start of a transfer operation if the block request ECS signal is not present and Q does not equal two. The Q counter is in synchronization with the number of words written or read in ECS. When two words have been transferred, Q equals 2 and the request controller flip-flop is reset in preparation for the next go ECS signal. The not block request ECS signal gates the Q equals 2 signal to the delay circuit which resets the request controller flip-flop. During a read operation when the Y register (word count) indicates the last record is being read, the block request ECS signal is generated (CPLR 3.1). The block request ECS signal prevents the Q equals 2 signal from resetting the request controller flip-flop. Since the indication is that the last record of the transfer operation is being read, the ECS controller is no longer required and another request controller signal is not necessary.

The ECS controller sends an ECS abort signal to the coupler if an error is detected during a data transfer operation. The ECS abort signal is ANDed with the go ECS signal, inverted, and ANDed with the block req controller signal to form the abort block signal (1PAG). The abort block prevents the generation of another request controller signal. If the ECS abort signal occurs during an ECS read operation, the ECS abort signal with the ECS read and Y=0 signals generate a fake read signal. The fake read signal along with the Q counter keeps the go central memory flip-flop (CPLR 3.3) set long enough to transfer the data out of ECS. When the transfer operation is complete, the word count in the Y register is zero and the Y equals zero signal breaks the AND gate where the fake read signal was generated ending the fake read signal. The ending of the fake read signal causes the go central memory flip-flop to reset.

The CPU ECS write signal via the coupler interface circuits (CPLR 3.1) sets the ECS write flip-flop (1PAG pak) for an ECS write operation. The ECS write signal from the ECS write flip-flop is fanned out (3CA0 pak) to appropriate circuits within the coupler requiring notification of an ECS write mode of operation. The ECS write signal is transmitted to the ECS controller via an ECS control register (4AP0 pak) and transmitters. A request controller signal gates the ECS write signal into the ECS control register. Requiring the request controller signal as a gate for the ECS write signal ensures the ECS controller that the request for this particular transfer operation is a write operation. The absence of the ECS write signal to the ECS write flip-flop indicates an ECS read operation. Therefore, the absence of the ECS write signal into the ECS control register at the time of the request coupler signal indicates to the ECS controller that the request is for a read operation. The ECS write flip-flop is cleared at the end of each transfer operation or by a master clear signal.

DETAILED-PAK DIAGRAM

ZERO PARITY CONTROL, FLAG REGISTER OPERATION AND GO CENTRAL

ZERO PARITY CONTROL

The CPU sends a force zero parity code that originates in the status and control register of the central processor when a test is required of the parity error detection networks. The code bits are received and passed through the latch register (3KR0 pak) at time T1, 3 to the force zero parity decoder (5LK0 pak). The decoder checks the bits and carries out the coded instructions. Normally both of the bits are zeros and an output signal from the decoder is not required.

If the force zero parity on address/word count at CPU is decoded, the force zero parity signal is transmitted via a latch register and transmitter (4KT0 pak) at the next time T1, 3 to the CPU. If a force zero parity on address to controller is detected, the decoder sends the force zero parity signal to a gating select circuit (4KQ0 pak). The force zero parity signal makes the AND gate in the select circuit that gates a logical zero as the ECS address parity bit through the select circuit to the ECS control register (4AP0 pak). The complemented force zero parity on ECS address signal breaks the AND gate in the select circuit and blocks the address parity bit that accompanied the address from the CPU. This address parity bit path through the select circuit is the path that the address parity bit takes for a normal transfer operation. The request coupler signal gates the ECS address parity bit, real or logical zero, into the ECS control register (4AP0 pak) to ensure the ECS address parity bit is associated with the proper address and request. The ECS control register transmits the ECS address parity bit to the ECS controller at the 100-nanosecond data transfer rate under control of the timing pulses. The ECS control register is reset each clock period to prepare it for the next ECS address parity bit. If the force zero parity decoder (5LK0 pak) detects a force zero parity on data at the controller, the signal is transmitted via an ECS control register (4AP0 pak) that does not require an input gating signal to the ECS controller. The ECS controller generates zero parity for the ECS parity check circuits.

ECS RESPONSE REGISTER

The ECS controller sends a controller accept signal to the coupler for every successful transfer of a record and also to indicate that it is ready for another record. The controller accept signal is received and loaded into the ECS response register (4AP0 pak). The ECS response register sends the controller accept signal to various coupler locations including the go central circuit (5LB0 pak). The coupler accept signal along with the not stop go central signal (CPLR 3.1) and the not flag register operation signal set the go central memory flip-flop. The go central memory flip-flop generates the go central signal that is necessary for the coupler circuits to perform any data transfer operation.

The ECS controller sends a controller abort signal to the coupler if the controller detects an error during the data transfer operation. The controller abort signal is received and loaded into the ECS response register. The ECS response register sends a controller abort signal to the error register (CPLR 3.7) for determination to immediately end the data transfer operation in the case of an ECS write operation or to continue the data transfer operation to its normal conclusion in the case of an ECS read operation. The response register also sends a controller abort signal to the break-in control circuit.

GO CENTRAL AND FLAG REGISTER OPERATION

The controller accept and block ECS accept signals set the go central memory flip-flop (5LB0 pak) if the flag register operation flip-flop is not set and the word count equals zero flip-flop (CPLR 3.1) that generates the WC=0 signal is not set. The go central signal, indicating that the ECS controller has accepted the transfer request, fans out to appropriate coupler circuits. The go central signal allows data transfer to begin between the ECS and CMC via the coupler. The go central memory flip-flop is also set when the initial go ECS and ECS write signals result in the generation of the force set go central signal. This initial go central signal results in a bank initiate signal that transfers to CMC before the ECS controller responds with its first controller accept signal. This is required since the ECS controller is ready to accept data immediately with its acceptance of the transfer request while it takes longer for the CMC to get data out of central memory. The P equals zero signal, indicating that all the words of the active record have transferred, resets the go central memory flip-flop. This requires the ECS controller to generate a controller accept signal after receipt of each record to set the go central memory flip-flop again.

The 17.5-nanosecond delayed Q equals 7 signal and the fake read signal are ANDed to set the go central memory flip-flop should an ECS abort occur. The ECS abort causes the generation of the stop go central signal resulting in the clearing of the go central memory flip-flop at the end of the record. The fake read signal is generated only during a read ECS operation. The go central memory flip-flop remains set allowing the transfer of all zeros to CMC for the remaining words of the ECS read transfer operation.

The controller accept signal is delayed, and then ORed with the master clear or end of transfer clear signal to generate a reset signal for the LAK register (CPLR 3.1). This signal resets the LAK register containing the lower 3 bits of the address after each successful transfer of a record.

The flag register operation flip-flop is set if address bit 23 is set at the time the flip-flop is clocked by the flag register operation enable signal. This signal coincides with the time the address arrives from the CPU. The flag register operation flip-flop output signal requests the ECS controller and passes the flag operation function on to the ECS controller. Address bits 20 through 22 of the associated address are interpreted by the ECS controller as to what is required during the flag register operation function. The ECS controller requires no information other than the address with bit 23 set. The coupler sends no request for information to the CMC. The output signal of the flag register operation flip-flop is inverted and prevents the controller accept signal from setting go central memory flip-flop and allowing a data transfer to take place. Since no data transfer takes place during a flag register operation, the flag register operation flip-flop depends on the clear signal for the next transfer operation request to reset the flip-flop.

DETAILED-PAK DIAGRAM

CMC TRANSFER CONTROL, P AND Q COUNTERS AND WRITE BUFFER CONTROL

TRANSFER CONTROL (WRITE)

The ECS controller sends an accept signal to the coupler to indicate that the initial request has been accepted. When the coupler receives the accept signal, it generates the go central signal (CPLR 3.3). The go central signal is fanned out (3CA0 pak) to the appropriate coupler circuits. One of the go central signals is ANDed with the ECS write signal (4LH7 pak) to set the 300-nanosecond shift register and the ECS continue request flip-flop at time T3. The ECS continue request flip-flop transmits the ECS continue request signal (4KT0 pak) to the CMC to reserve a memory bank in central memory 400 nanoseconds ahead of each successive bank initiate signal. The output of the 300-nanosecond shift register sets the bank initiate flip-flop at the next time T3 (300-nanosecond shift plus 100-nanosecond clock period equals 400 nanoseconds). The 100-nanosecond square wave that gates the data into the data holding register in preparation for transmission to the ECS controller also gates the bank initiate signal to the latch register (4KT0 pak) for transmission to the CMC. The bank initiate signal arrives at the CMC 400 nanoseconds after the continue request signal to initiate the addressed bank in central memory to send its data to the coupler buffer.

TRANSFER CONTROL (READ)

During an ECS read operation, a go central signal from the fanout (3CA0 pak) is gated into the 1100-nanosecond delay chain (5KS0 pak) by the ECS read (not ECS write) signal. A further delay occurs as the signal is passed through two 100-nanosecond delay flip-flops (4LH7 pak) to the 300-nanosecond shift register and the ECS continue request flip-flop. The ECS continue request signal transfers to the CMC while the 300-nanosecond shift register delays the setting of the bank initiate flip-flop for an additional 300 nanoseconds plus one clock period. The total delay in an ECS read operation before a bank initiate signal reaches the CMC is at least 1700 nanoseconds. This amount of time is required for the ECS to get the data out of memory after it has accepted the transfer operation request. The CMC is not required until the memory cycle is complete and data transfer starts. Once the transfer starts, it continues at the transfer rate of one word every 100 nanoseconds.

P COUNTER

The selected P register bits 0 through 2 are selected by the P select circuit (CPLR 3.1) for the P counter. The P counter (6LG0 pak) keeps track of the number of words in the record being transferred. The selection circuit therefore provides the K register output (lower 3 bits of address) which is normally a 7 as the input to the P counter until the Y register output (word count) is equal to or less than the K register output. When the last record is in the process of being transferred, the Y minus 1 signal is supplied to the P counter as the number of words in the record.

The P counter is loaded with the selected P register bits 0 through 2 when the counter is in the preset mode. The go central signal, indicating an ECS controller accept, sets the one-shot flip-flop at clock time T1, 3. This forces the P counter into a decrement mode and allows the count to be decremented by one every clock period. The output of the P counter is monitored by the P test circuit until P equals 0. When P equals 0, the last word of the record has transferred and the P test circuit sends the P equals 0 signal to reset the go central flip-flop (CPLR 3.3). The go central

flip-flop requires the ECS controller to send back an accept signal to set it again. The ending of the go central signal returns the P counter to the preset mode and allows the loading of the word count of the next record. The process continues until the last word of the last record is transferred.

Q COUNTER

The Q counter (5LG0 pak) is reset to zero at the same time the P counter goes to a decrement mode. This is accomplished by the one-shot flip-flop not being set by the go central signal until time T1, 3. The Q counter increments once each clock time T3. The Q test circuit outputs a signal when Q equals 2. During an ECS write operation, the Q equals 2 signal indicates that two words have transferred from CMC to the coupler buffer. The Q equals 2 signal sets write block request controller flip-flop (CPLR 3.6) which allows the first request controller signal of the transfer operation to transfer to the ECS controller. After the Q equals 2 signal has once set the write block request controller flip-flop, it is not required again until after an end of transfer has taken place. However, the Q counter is continually reset for each record and incremented for each word. The Q equals 2 signal also resets the request controller flip-flop (CPLR 3.2) to allow the generation of the next request for the ECS controller.

The Q test circuit also generates a Q equals 7 signal that in combination with the fake read signal (CPLR 3.3) sets the go central flip-flop. This keeps the go central flip-flop set so that all zeros are written into CM for the remaining words of an ECS read operation if an ECS abort occurs.

P DELAY COUNTER

The P holding register (4LL7 pak) receives the selected P register bits that contain the word count of the next record. These bits are clocked into the holding register by the combination of the delayed go ECS, controller accept, and clock time T3 signals. The selected P register bits enter the P delay counter when the controller accept signal forces the counter into a preset mode. When the controller accept signal ends, the counter goes into a decrement mode. The counter is decremented each clock time T3 during an ECS write operation. When the counter reaches zero, the P delay test circuit outputs a signal to reset the write data flip-flop at the end of the record.

The write data flip-flop is set by the ECS write signal and the controller accept signal from the initial or previous record. The write data flip-flop output signal is delayed 500 nanoseconds (5KS0 pak) and ANDed with the 100-nanosecond square wave (4LH7 pak) to generate the load data signal. The load data signal is fanned out to gate the data from the coupler buffer (CPLR 3.5) into the data holding register. The data is then transmitted to the ECS controller. The buffer address circuit uses a complemented load data signal for synchronization of the address to the data (CPLR 3.6). The 500-nanosecond delay (5KS0 pak) affects the write data signal on the first word of the first record only in the transfer operation. The timing of the clock pulse to the reset time of the write data flip-flop is such that the 500-nanosecond delay flip-flop does not see the signal drop at the end of the record unless a gap occurs to delay the controller accept signal. A delay caused by a gap would still only delay the output from producing another load data signal for as long as the gap occurred.

DETAILED-PAK DIAGRAM DATA PATHS

WRITE DATA PATH

The CM read data to be written into ECS is received and enters the latch register (3KR0 pak) at time T1, 3. The 60 bits of data and a parity bit are accompanied by a data on line signal referred to as a go ECS in the CMC. The data on line signal is clocked by a 100-nanosecond square wave in the 3KF0 pak. The resultant output signal is a 50-nanosecond write strobe and a complemented write strobe once every 100 nanoseconds. The complemented write strobe provides the buffer address control (CPLR 3.6) with a timing pulse. The write strobe signal is fanned out (3FX0 pak) to gate the data and the parity bit into the data buffer 16-word random access memory (RAM) at the address designated by the buffer control address bits (5PKG or 1PTG pak).

The 16-word RAM buffer stores the data coming from the CMC that the ECS controller is not ready for. The first request for the ECS controller is blocked to give the CMC a headstart as the ECS controller is able to accept data as soon as the request is made. The 16-word RAM stores the first two words from CM according to the buffer address. The coupler generates a request for the ECS coupler only after the successful transfer of two words from the CMC. If the controller can do a coupler request at this time, it sends an accept signal to the coupler which starts the transfer of data. If the ECS controller cannot take the request and does not send an accept back to the coupler, the next word from CMC is stored in the next address of the 16 word RAM. The minimum number of words in the 16-word RAM before a transfer begins is two, while a maximum of 16 depends upon when the ECS controller sends back an accept signal. The initial blocked request for the ECS controller is due to the ability of the ECS controller to accept data sooner than the CMC can get the data out of central memory. The CMC therefore, operates the CM ahead of, and semi-independent of ECS. This allows the 16-word RAM buffer to store data and have data transferred out once every 100 nanoseconds which is as fast as the ECS controller can accept data from the coupler for writing in ECS. The buffer input address which is incremented by the complemented write strobe controls the data location within the 16-word RAM. The data in the 16-word RAM is transferred out on a first-in first-out basis by the buffer output address. The 100-nanosecond square wave, 50 nanoseconds of which is for the input address and the other 50 nanoseconds for the output address accomplishes address control. The ECS accept signal indicating the ECS controller is ready for the first or next word controls the incrementing of the output address. The data output from the 16-word RAM is in sequence and at the ECS transfer rate.

The buffer CM read data enters the data holding register (4AP0 pak) by the presence of the load data signal. The load data signal is the result of the ECS write signal and the controller accept signal that operates the P delay counter for an ECS write condition. In order to get a load data signal, the ECS controller must generate the initial accept or an accept due to a successful transfer of the previous word. The transmit data signal and clock time T25 gate the data to the transmitter. The transmit data signal is a result of the ECS write signal gated by the 100-nanosecond square wave.

The CM read data parity bit also passed through a 16-word RAM buffer and holding register in synchronization with the associated data word.

A section of the 16-word RAM parity buffer generates the go signal for the ECS controller that accompanies each data word by having that section full of logical ones. Therefore, every output word selected by an output address is accompanied by a go signal indicating that data is on the line to the ECS controller.

READ DATA PATH

The ECS read data to be written into CM is received and enters the data holding registers (4AP0 pak) unconditionally. No gating or clock signals are required to enter the data. The data transfers immediately to a latch register in the data transmitters (4KT0 pak) and is clocked at time T3 to the transmitter. The data parity bit accompanies the data through the coupler along a similar path.

If an ECS abort occurs during a read operation, the ECS controller transfers the remaining words of the transfer operation as all zeros. The coupler receives the ECS abort signal and generates the forced parity bit signal (1PPG, CPLR 3.7A and 3.5D pak) to accompany the remaining words made up of all zeros in the transfer operation.

If an error occurs in the ECS, the ECS controller sends the appropriate error signal to the coupler. The ECS controller parity error or ECS bank parity error signals enter the holding register on the 4AP0 pak. The holding register sends the error signals to the coupler error register (CPLR 3.7) and are eventually coded and transferred to the status and control registers for error identification.

DETAILED-PAK DIAGRAM BUFFER ADDRESS

BUFFER ADDRESS

The buffer address generates the I/O addresses for the buffer 16-word RAM in the CM read data path (CPLR 3.5). The generation of the buffer address requires the go ECS signal that is generated as a result of the coupler accept signal (CPLR 3.2). The go ECS signal is ANDed with the complemented output signal of the one-shot in a pulse forming combination. The go ECS signal makes the AND gate forming the leading edge of the pulse. The go ECS signal at the next clock time T1 sets the one-shot. This could be 100 nanoseconds after the arrival of the go ECS signal depending upon the relationship of the go ECS signal to time T1. When the one-shot sets, its signal breaks the AND gate forming the trailing edge of the pulse. This pulse provides an input signal to functional control input S2 of the I/O address registers (3LAW pak) that generate the buffer address. The output signal of the pulse forming AND gate is also complemented to break two AND gates. The complemented outputs of these AND gates provide input signals to input S1 of the I/O address registers. With the signals as described at this time, the I/O address registers are in a preset mode and load all zeros. When the one-shot sets at time T1, the signal from the pulse forming AND gate ends removing the input signals to S1 and S2. This forces the I/O address registers into a hold mode. The write strobe signal, which is generated each time a word arrives from CMC, controls the input signal to S2 of the input address. The write strobe forces the input address register into the increment mode for each word. The clock pulse increments the input address register at the transfer rate of once every 100 nanoseconds. The address is incremented each time a write strobe occurs until all 16 addresses have been used; then the address register returns to zero and starts all over again.

The output address register operates in the same manner as the input address register except that the load data signal forces the output address register into the increment mode. The load data signal is the result of the controller accept signal. The controller accept signal occurs once every 100 nanoseconds after the initial ECS request is accepted. The initial ECS request is not generated until at least two words have arrived from the CMC. The incrementing of the I/O address register continues each clock time there is a write strobe or load data for their respective registers until the transfer operation is complete. The next go ECS signal for the next transfer operation results in loading of all zeros into the I/O address registers to begin the process all over again.

The address bits from the input address register are complemented externally (5PKG pak) while the bits from the output address register are complemented internally (3LAO pak). The 100 nanosecond square wave signal (3KFO pak) gates the complemented I/O address bits so that every 50 nanoseconds either the input or the output address is fanned out (3FXO pak) to the data path buffer 16-word RAM (CPLR 3.5).

FORCE SET GO CENTRAL CONTROL

The CPU request for an ECS write operation requires the coupler to generate request signals for the ECS controller and the CMC. The ECS controller is capable of accepting data as soon as the request for use is accepted. Since it takes more time for the CMC to get data out of central memory and to the coupler, the force set go central circuit (4LL7 pak) generates a signal to initiate the CMC ahead of the ECS controller.

The go ECS signal that is a result of the coupler accepting the CPU request for a transfer operation is ANDed in a pulse forming combination with the output of the one-shot and ECS write signal. The pulse ends when the one-shot sets at time T1, which then sets the force set flip-flop. The flip-flop sends the force set go central signal to set the go central flip-flop (CPLR 3.3). The go central flip-flop causes the generation of the first bank initiate signal for the CMC. The coupler has not yet issued a request for the ECS controller to perform a transfer operation. The ECS controller accepting the request for a transfer operation is the normal control signal that sets the go central flip-flop during a transfer operation.

WRITE BLOCK REQUEST CONTROLLER CONTROL

The write block request controller circuit (4LL7 pak) generates the signal that prevents the coupler from generating a request for the ECS controller until at least two words from the CMC are in the coupler data path buffer.

The same pulse that sets the force set go central flip-flop resets the write block request controller flip-flop. The reset mode of the flip-flop generates a block request controller signal that prevents the coupler from sending the first request controller signal (CPLR 3.2) to the ECS controller. The coupler does send a request to the CMC at this time. When two words have transferred into the data path buffer from the CMC, the Q counter which is equal to the P counter (word count) sends the Q equals 2 signal. The Q equals 2 signal is complemented, ANDed with the ECS write signal, and the result complemented to set the write block request controller flip-flop. The setting of the write block request controller flip-flop removes the block request controller signal. This now allows the coupler to request the ECS controller as there are at least two words of data in the data path buffer and possibly more on the way.

During a read ECS transfer operation, a delay of the ECS request controller signal is not required. The absence of the ECS write signal sets the write block request controller flip-flop immediately as the Q equals 2 signal is not required. Since the flip-flop is set, a block request controller signal is not present, the coupler is allowed to generate the request for the ECS coupler.

DETAILED-PAK DIAGRAM

END TIME CONTROL, ERROR REGISTER, ERROR CODE GENERATOR AND PROCESSOR INTERFACE

END TIME CONTROL

The end time control circuits perform the termination functions of a successful or not so successful transfer operation for the coupler. The word count control circuit (CPLR 3.1) provides a Y equals 0 signal (word count) to the end time control (1PPG pak) when the Y register has decremented to zero indicating that all the words for the active transfer request are accounted for. In an ECS write operation, this means the coupler received a controller accept signal for the previous word from the ECS controller; whereas, in an ECS read operation only the word count must equal zero. When these conditions are present, the transfer operation is complete and the end time control can terminate the transfer operation.

The complemented Y equals 0 signal (1PPG pak) is recomplemented and sets the start end time flip-flop. The complemented output signal of the flip-flop forces the end time counter (3LA0 pak) into an increment mode. The end time counter operates in only two modes. The preset mode loads all zeros in the counter so that when the counter goes into an increment mode, it starts at zero and increases its count once each clock period (100 nanoseconds). The counter outputs end time bits 4, 8, and 16 for the end time decoder (1PPG pak). When bits 16 and 4 are set, 2000 nanoseconds have elapsed and when bits 16 and 8 are set, 2400 nanoseconds have elapsed. The end time decoder in the translator interprets the count and on the next clock pulse provides the enable end time signal 2100 nanoseconds (for a read ECS operation) or 2500 nanoseconds (for a write ECS operation) after the setting of the start end time flip-flop. Four other signals in the translator may set the end time flip-flop at time T1, 3.

The end time flip-flop output signal sets the end of transfer clear flip-flop on the next time T1, 3. The end of transfer clear flip-flop generates the end of transfer clear signal for all flip-flops on this pak, except the ECS abort flip-flop, and is fanned out to appropriate coupler circuits to indicate the end of the transfer operation. The ECS abort flip-flop is reset by coupler accept OR master clear.

The master clear signal serves the same purpose as the end of transfer clear signal. The resetting of the end time flip-flop ends the signal that sets the end of transfer clear flip-flop and ends the clear signal. The end time flip-flop output signal also makes an AND function in the translator if there has not been an exchange BI ECS signal. This results in the generation of an end of transfer signal if a controller abort or an error detected by the error register (5PVG pak) is not present. The end of transfer signal is transmitted to the CPU via a latch register and transmitter (4KT0 pak). The end time flip-flop also sends an ECS end of transfer signal to the CPU. The fact that the coupler generates these two signals indicates to the CMC and CPU that the transfer operation completed successfully.

If the ECS controller detects an error, it sends a controller abort signal to the coupler to set the ECS abort flip-flop via a receiver and latch register (CPLR 3.3). If this occurs during an ECS read operation, the AND function on the output of the ECS abort flip-flop is not made and the output of the ECS abort flip-flop is prevented from setting the end time flip-flop. However, the ECS abort flip-flop output signal is sent to the coupler transfer control circuit (CPLR 3.2) to generate a take read signal and cause a fill of all zeros for the remaining words of the

transfer operation to be written into central memory. The ECS abort flip-flop output signal is also sent to the error code generator and coded for transmission to the CPU. The ECS abort flip-flop output signal is also inverted and ANDed with the error signal. The resulting signal from this gate prevents the generation of the end of transfer signal sent by the end time flip-flop to the transmitter, and at the same time sends an error end of transfer signal to the transmitter for the CPU. If the transfer operation taking place is an ECS write operation when the controller sends an abort signal, the output of the ECS abort flip-flop sets the end time flip-flop to begin the process of ending the transfer operation without waiting for the word count to equal zero. The transfer operation ends immediately to prevent destroying other data already in ECS.

If the CPU sends an exchange BI ECS signal, it enters the coupler via a receiver and latch register (4KR0 pak) and sets the exchange terminate ECS flip-flop (1PPG pak). When the exchange terminate ECS flip-flop sets, the output signal immediately sets the end time flip-flop to generate an ECS end of transfer signal.

The transfer control circuit (CPLR 3.2) sends an end data transfer signal that immediately sets the end time flip-flop if the Y equals 0 signal (word count) or an error signal occurs within 550 nanoseconds of the coupler accept signal. The CPU sometimes sends the Y equals 0 condition along with the coupler request signal to generate a pass condition for ECS. An error signal at the time of the delayed coupler accept signal usually indicates that an address error exists and the end time flip-flop is set to prevent the transfer of data to an incorrect ECS location.

The CPU sends an address with bit 23 set as a request for a flag register operation. Bit 23 sets the flag register operation flip-flop (CPLR 3.3) that generates the flag register operation signal. This signal enters the end control circuit (1PPG pak) and partially makes three AND functions. One AND function is completed when the ECS controller sends an accept signal back upon receipt of the address with bit 23 set. The output signal of this AND function sets the end time flip-flop and generates an end of transfer signal. The end of transfer signal is generated after receipt of the ECS coupler accept signal for the address because the CPU and CMC are not required for a flag operation. However, the end of transfer signal to the CPU does indicate that the address containing bit 23 set successfully transferred.

If the ECS controller detects a parity error in the ECS address that has bit 23 set, it sends back a controller parity error signal. The signal enters the coupler via a receiver and holding register (CPLR 3.5) and passes through the error register (3AS0 pak) to complete the AND function (1PPG pak) with the flag register operation signal. The resulting signal from this AND function is the ECS flag register parity error signal for transmission (4KT0 pak) to the CMC. If the ECS controller sends back a controller abort signal, the function performed by the controller has had a negative result. The controller abort signal sets the ECS abort flip-flop (1PPG pak) that causes the generation of an error end of transfer signal to indicate the negative result to the CPU.

ERROR REGISTER AND CODE GENERATOR

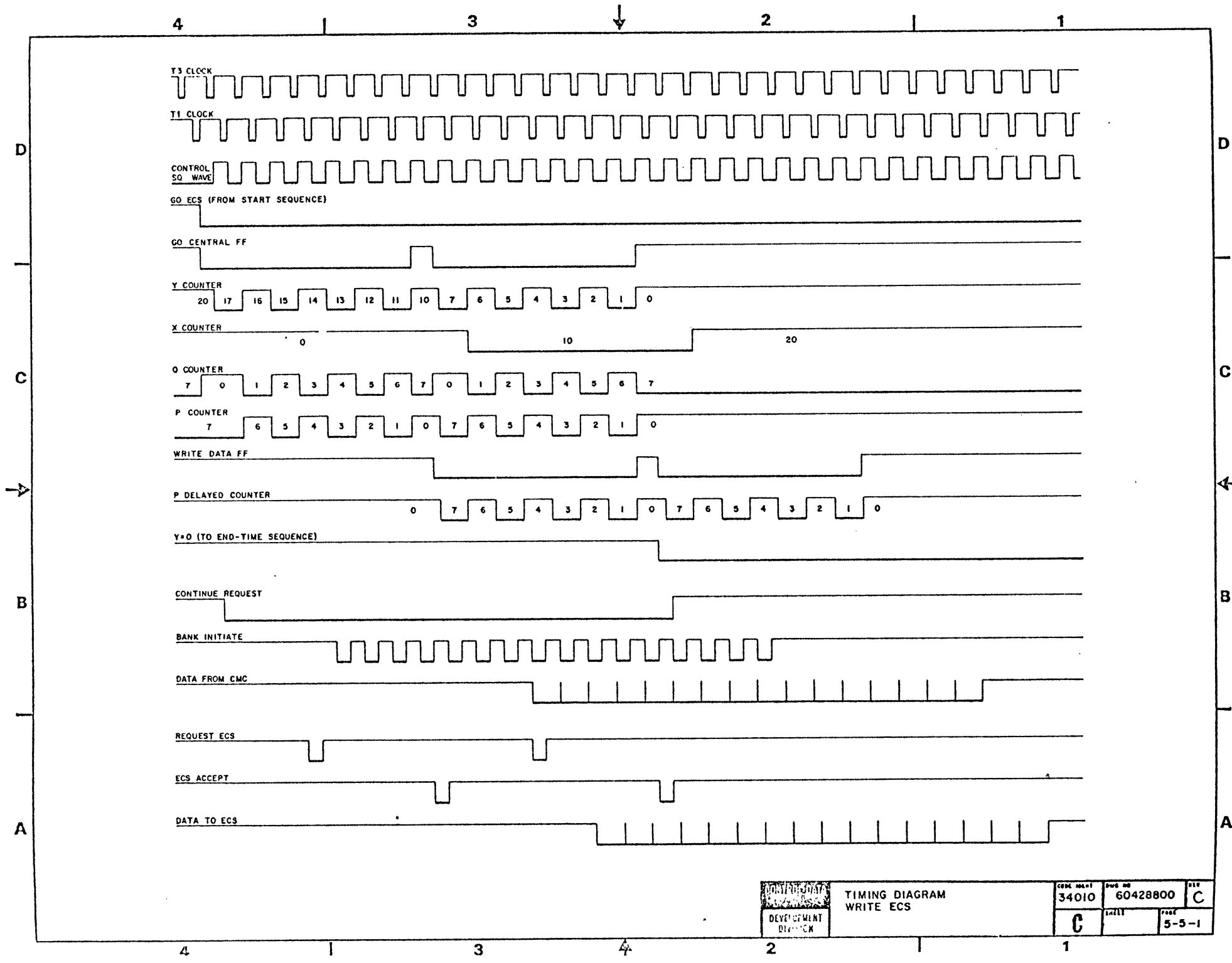
The error register (3AS0 pak) receives error signals that occur during an ECS operation from the various detection circuits. The errors detected in the CMC and CPU enter the error register via a receiver and latch register (3KR0 pak). The T3 and write strobe signal gates (4KQ0 pak) the CMC double error signal to the error register. The controller parity enhancement switch (4CP0) in the OFF position prevents the CMC input error signal from entering the error register. This switch is set in the ON position only when the ECS uses a controller that has the parity enhancement installed. The CMC would constantly indicate an input error if the controller did not have the parity enhancement circuit that sends a parity bit to the CMC and the switch was in the ON position. The errors detected in the ECS controller load into the error register via a receiver and holding register (CPLR 3.5). Any errors detected in the address and word count entering the coupler (CPLR 3.1) cause parity error signals that are gated to the error register. When an error signal sets the error register, the error test circuit outputs an error signal to the end time control circuits (1PPG pak) that causes an error end of transfer signal. The error register sends any error signals to the error code generator (5LK0 pak). The error register is reset by the coupler accept signal each time the coupler is requested for a transfer operation. This clears out all error signals in registers that were due to the previous request.

The error code generator (5LK0 pak) receives error signals from the error register and codes them for transmission via a 4KT0 pak to the CPU. When an error condition does exist, the ECS transfer error signal is always generated. The ECS transfer error signal requires the master clear or end of transfer clear signals as a gating signal to transfer to the latch register and transmitter. This causes the ECS transfer error signal to arrive at the status and control register after the end of transfer clear signal to indicate that an error occurred on the last transfer operation.

CPLR 3.7 TEST POINTS

Module	Location	Test Point	Description
1PPG	4D34	3 (T)	Controller Acpt
1PPG	4D34	5 (T)	ECS Write
1PPG	4D34	13 (T)	Controller Abort
1PPG	4D34	11 (F)	End Data Xfr
1PPG	4D34	1 (F)	ERR
1PPG	4D34	14 (F)	Start Clear
1PPG	4D34	4 (F)	Exchange BI ECS
1PPG	4D34	9 (T)	Y=0
1PPG	4D34	2 (T)	ECS Write • ECS Abort
1PPG	4D34	7 (T)	End Time Decoder + Flag Rgtr Opn • (ECS Abort + Controller Acpt)
1PPG	4D34	8 (T)	Clock Time T1, 3
1PPG	4D34	10 (T)	Enbl End Time
1PPG	4D34	6 (F)	ECS EOT
1PPG	4D34	12 (T)	Preset

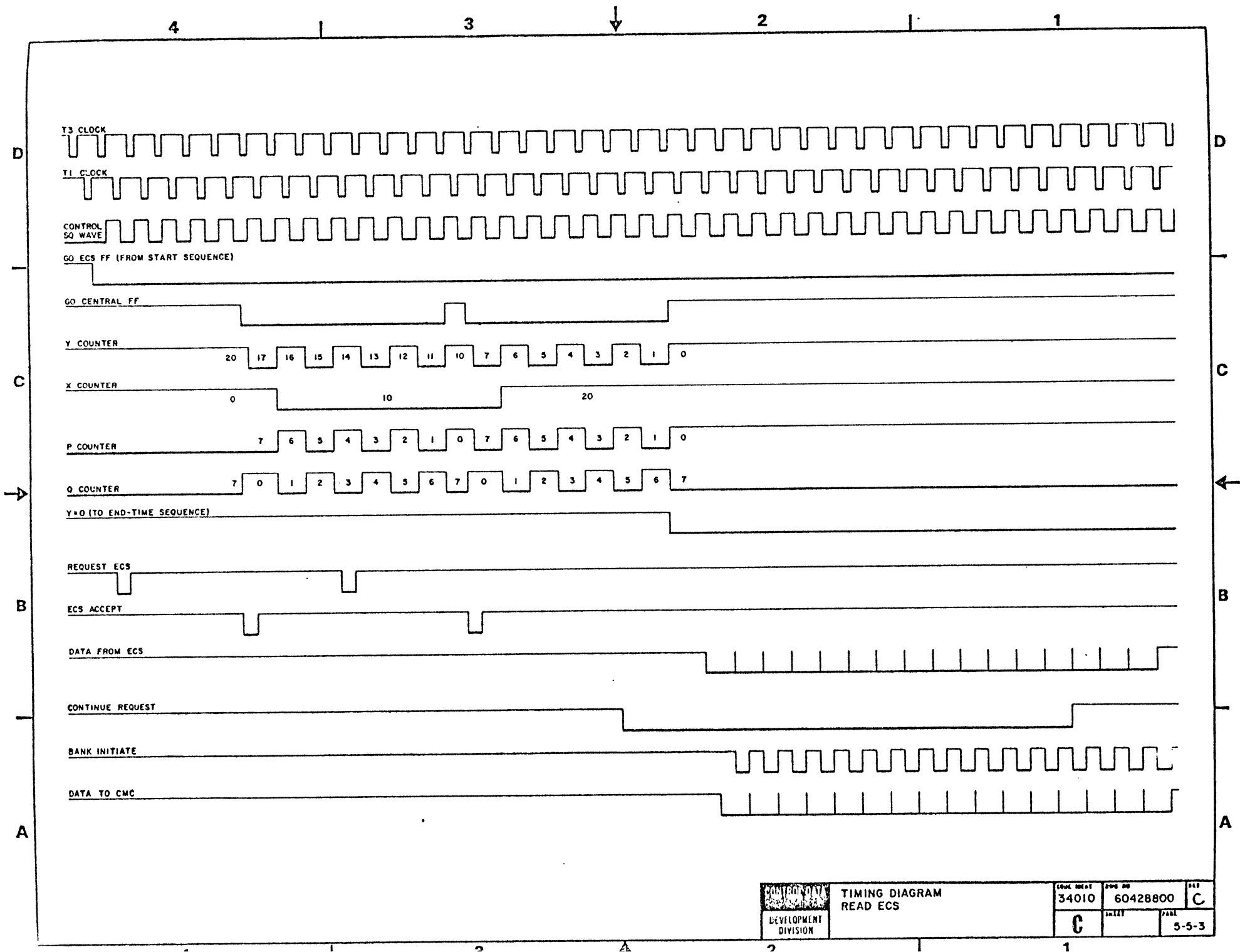
Module	Location	Test Point	Description
4KT0	4F15	3 (T)	ECS Transfer Error
4KT0	4F15	8 (T)	ECS End of Transfer
4KT0	4F15	2 (T)	Clock Time T1, 3
4KT0	4F20	10 (T)	ECS Flag PE
4KT0	4F20	7 (T)	End of Transfer
4KT0	4F20	6 (T)	Error End of Transfer
4KT0	4F20	2 (T)	Clock Time T1, 3
3CM0	4E36	1 (F)	MC + EOT Clear
3CM0	4E36	5 (F)	MC + EOT Clear
3KR0	4C15	7 (T)	Clock Time T1, 3
3KR0	4C15	13 (T)	CMC Double Error
3KR0	4C15	12 (T)	CMC Input Error
3KR0	4C15	8 (T)	CSU Address Parity Error
4KQ0	4E34	4 (T)	CMC Double Err
4KQ0	4E34	12 (T)	T3 • Write Strobe
5PVG	4C34	13 (F)	ECS Bank PE
5PVG	4C34	12 (F)	Controller PE
5PVG	4C34	11 (F)	CSU Adrs PE
5PVG	4C34	14 (F)	CMC Input Err
5PVG	4C34	9 (F)	CMC Double Err
5PVG	4C34	1 (T)	Error Clear
5PVG	4C34	2 (T)	Clk Adrs/Wd Count PE
5PVG	4C34	4 (F)	Clk Adrs/Wd Count PE
5PVG	4D33	1 (T)	Adrs/Wd Count PE
5LK0	4D33	2 (T)	CMC Input Err
5LK0	4D33	3 (T)	CSU Adrs PE
5LK0	4D33	4 (T)	CMC Double Err
4KT0	4F07	2 (T)	Clock Time T3
4KT0	4F07	8 (T)	ECS Transfer Error Code Bit 0
4KT0	4F07	5 (T)	ECS Transfer Error Code Bit 1
4KT0	4F07	3 (T)	ECS Transfer Error Code Bit 2



DEVELOPMENT DIVISION

TIMING DIAGRAM
WRITE ECS

FORM NO. 34010	REV. NO. 60428800	REV. C
C	DATE	PAGE 5-5-1



CONTROL DATA
DEVELOPMENT
DIVISION

TIMING DIAGRAM
READ ECS

LOCK NO	FIG NO	REV
34010	60428800	C
C	DATE	PAGE
		5-5-3

DDP

GENERAL DESCRIPTION

FUNCTIONAL DESCRIPTION

The DC145-B Distributive Data Path (DDP) is a piece of hardware which makes possible direct communication between the peripheral processor subsystem (PPS) and extended core storage (ECS) in CDC[®] CYBER 70 and 170 systems. The DDP contains one PPU interface (port) and is expandable up to three more identical interfaces. The additional interfaces are also referred to as DDP registers (DDPRs). The ports share a common interface with one access of an ECS controller. Figure 1-1 shows the DDP/system configuration.

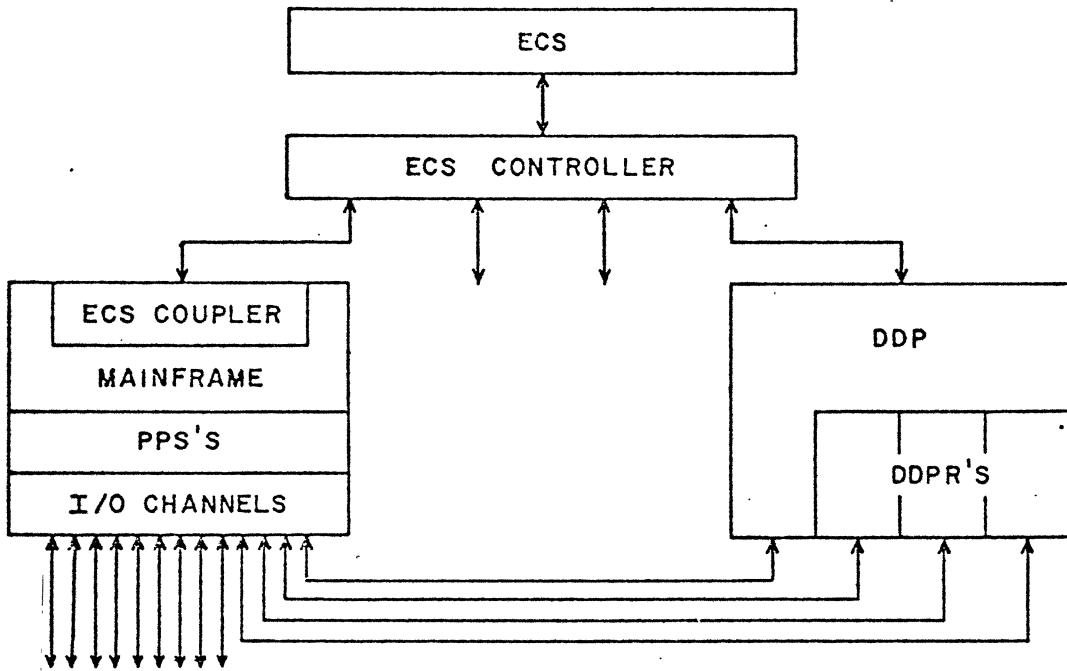


Figure 1-1. DDP/System Configuration

A PPS communicates with one port of a DC145-B through an I/O channel. Information transfers in words consisting of 12 data bits (plus odd parity on CDC CYBER 170 systems). Between the ECS controller and the DDP, information transfers in words of 60 data bits (plus odd parity on CDC CYBER 170 systems). The DDP assembles/disassembles 12/60-bit words during read and write operations. Only one port can reference ECS at any given time. A scanner monitors the ports and sequentially connects requesting ports to the ECS controller. The scanner allows eight 60-bit words (one ECS record) to transfer before moving on. If the reference is for more than one ECS record, the next transfer does not occur until the scanner has monitored the other ports. In addition to data transfer, the DDP provides flag register communications.

The DC145-B can transfer 2 million 12-bit words per second when it is connected to a CDC CYBER 170 I/O channel. To maintain this transfer rate, the DC145-B must be the only device on the channel. On a CDC CYBER 70 I/O channel, the DC145-B can transfer at a rate of 1 million 12-bit words per second. To maintain this rate, it must be the first or second device on a channel and it must be the last device on the channel. The DDP does not have pass-on capability. Information transfers between the DDP and the ECS controller at a maximum rate of 10 million 60-bit words per second.†

PROGRAMMING INFORMATION

Refer to the CDC CYBER 170 ECS Subsystem Hardware Reference Manual, listed in the preface, for detailed programming information.

WORD FORMAT

The DDP assembles 12-bit words to 60-bit words during write operations and disassembles 60-bit words during read operations. Figure 2-1 illustrates the word format.

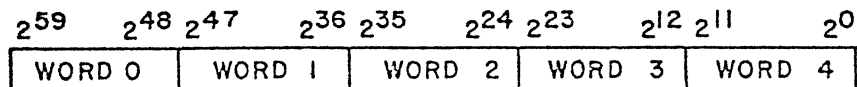


Figure 2-1. DDP Word Format

FUNCTION CODES

Twelve-bit function codes transmitted by a PPU on an I/O channel control the DDP. The upper three bits of the function code form the equipment select code (5g). The remaining nine bits designate which function the DDP performs. The PPU transmits the function codes on the channel, and the DDP responds by inactivating the I/O channel. The DDP responds to all function codes with the correct equipment select code and correct parity. When I/O channel parity checking is disabled (by a switch on the port status module), the DDP responds to all function codes with the proper equipment select code. Table 2-1 lists the DDP function codes.

TABLE 2-1. DC145-B DDP FUNCTION CODES

Function	Code
Block read ECS	5X01
Read ECS, one reference	5X41
Function flag register (bit 2 ²³ of the address register must be set.)	5X01
Write ECS	5Y02
Select status	5Z04
Master clear port	5010
Maintenance mode read	5Z21
Maintenance mode write	5022
Note:	
Normal Operation	X, Y, or Z = 0
Zero Channel Parity	X or Z = 1
Function ECS Controller To Send Zero Parity With Data	X = 2
Send Zero Parity to ECS Controller With Data	Y = 2
Send Zero Parity to ECS Controller With Address	X or Y = 4

STATUS RESPONSES

The PPU receives a 12-bit status word in response to a select status (5Z04) function. Bits 2⁰ through 2⁵ of the status word indicate conditions as shown in the following.

- Bit 2⁰ ECS abort. Indicates that the DDP has received an abort signal from the ECS controller.
- Bit 2¹ ECS accept. Indicates that the DDP has received an accept signal from the ECS controller.
- Bit 2² ECS parity error. Indicates that the DDP has received an ECS parity error signal from the ECS controller.
- Bit 2³ ECS write. Indicates that the DDP port is busy with a write to ECS. This status bit clears when the write operation terminates.
- Bit 2⁴ I/O channel parity error. Indicates that the DDP has received a 12-bit word with incorrect parity from the PPU.
- Bit 2⁵ ECS controller parity error. Indicates that the DDP has detected incorrect parity in data sent to or received from the ECS controller.

Various combinations of status bits indicate the following error conditions.

- 0020 Indicates that the DDP has detected a parity error in the address received from the PPU during a read ECS or write ECS operation.
- 0022 Indicates that the DDP has detected an error in the data received from the PPU during a write ECS operation.
- 0041 Indicates a parity error on the address sent to the ECS controller during a read ECS, write ECS, or function flag register operation.
- 0042 Indicates that the DDP has detected a parity error in the data transferred to/from the ECS controller during a write ECS or read ECS operation.
- 0046 Indicates that the DDP has detected an ECS parity error in the data received from the ECS controller during a read ECS operation.

The following bits in the status word are cleared by a master clear port function, power-on master clear, or the next nonstatus function after reading status.

- Bit 2⁰ ECS abort
- Bit 2¹ ECS accept
- Bit 2² ECS parity error
- Bit 2⁴ I/O channel parity error
- Bit 2⁵ ECS controller parity error

Bit 2¹ is also cleared by a request to ECS. Bit 2³ clears when the write ECS operation terminates.

THEORY OF OPERATION

Three subsections comprise the theory of operation section.

- Basic Theory
- Function Flow
- Detailed Theory

The Basic Theory subsection provides an overview of the basic read and write operations.

The Function Flow subsection describes the sequence of signals generated as each function executes.

The Detailed Theory subsection provides a chip-level description of the logic operation.

BASIC THEORY

This subsection provides an overview of the basic read and write operations.

To describe the basic read and write operations, it is convenient to divide the DDP into the following functional blocks.

A scanner which time shares the ports

A function decoder

A 24-bit address register and buffer

Read control logic

Write control logic

A 2-rank, 960-bit (60-bit parallel by 16-bit serial) data storage register and two buffers

BASIC READ OPERATION

Refer to Figure 4-1 during the following discussion.

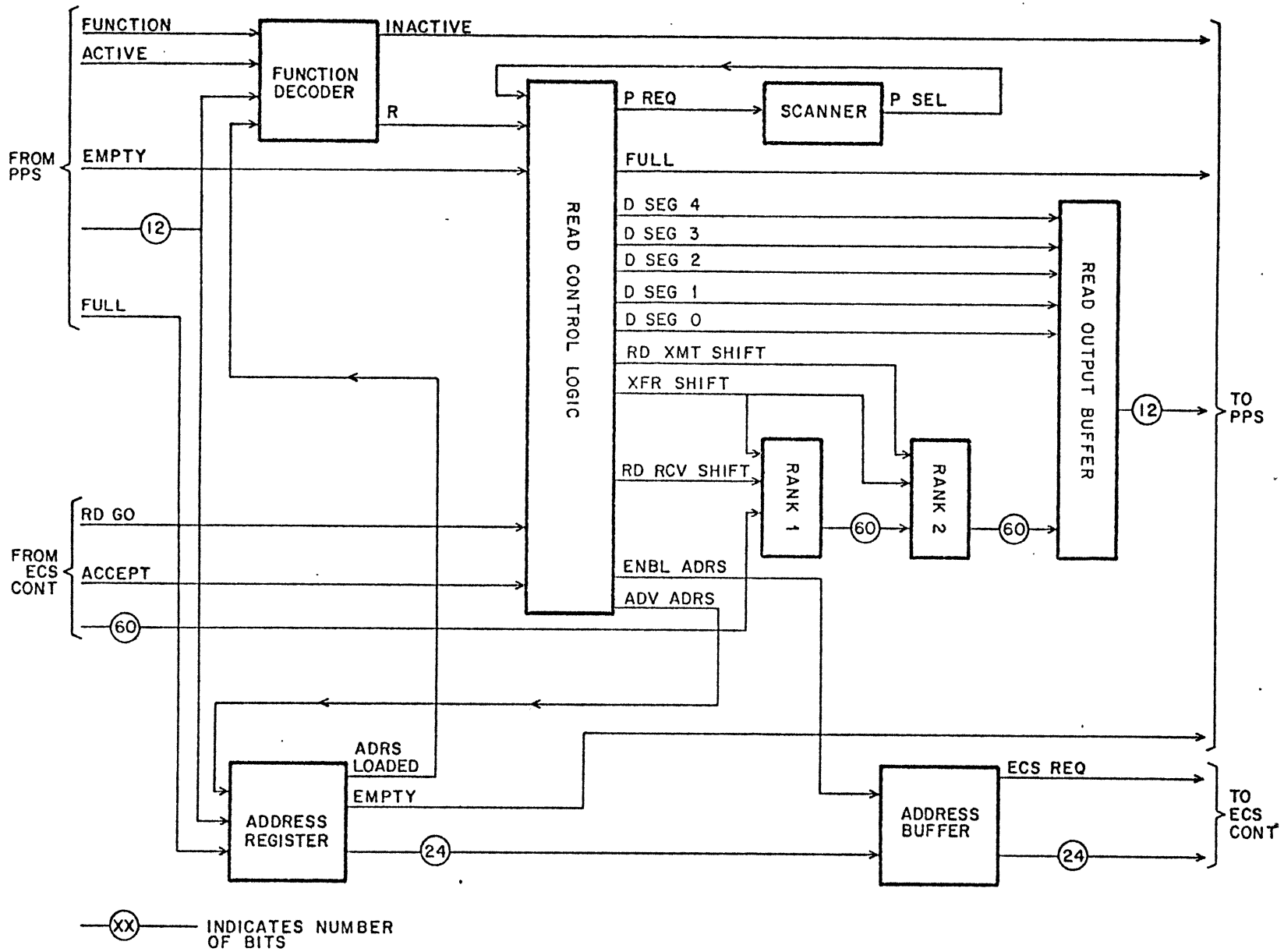


Figure 4-1. Basic Read Operation (One Port Shown)

To initiate a read operation, the PPS sends, in parallel, a 12-bit function code word and a FUNCTION signal to a DDP port via an I/O channel.† If the equipment select and function codes are correct, the port's function decoder partially enables for a read operation and inactivates the I/O channel. The PPS then activates the I/O channel and transmits two 12-bit data words which indicate the address from which the first ECS reference reads. The FULL signal accompanying each address word causes the address register to load the first address word into the upper 12 bits and the second address word into the lower 12 bits. An EMPTY signal is transmitted to the PPS after each address word is received. After the two address words are loaded, ADRS LOADED causes the function decoder to generate R (read).

R causes P/REQ (port request) to be generated. When the scanner detects a P REQ signal, it sends P SEL to the read control logic. The read control logic then generates ENBL ADRS which gates the 24-bit address word and an ECS REQ signal out to the ECS controller. If the ECS controller can transfer data, it sends an ACCEPT signal to the DDP and transmits eight 60-bit words. A READ GO pulse accompanies each 60-bit word from the ECS controller. Each READ GO pulse causes the read control logic to generate a RD RCV SHIFT pulse. These pulses shift the eight 60-bit words into rank 1 of the data storage register. The scanner then moves on to monitor the next port.

When rank 1 is full, the read control logic generates eight XFR SHIFT pulses. XFR SHIFT transfers the eight 60-bit data words from rank 1 into rank 2 of the data storage register. A 60-bit word is now at the output of the data storage register.

The read control logic then sequentially generates D SEG 0 through D SEG 4. These signals sequentially enable consecutive 12-bit segments of the 60-bit output buffer. As each 12-bit segment enables, the 12-bit segment transmits to the PPS via the I/O channel. A FULL pulse accompanies each 12-bit word to the PPS. The PPS responds with an EMPTY pulse for each FULL pulse it receives. When five consecutive 12-bit segments have enabled, a 60-bit word has been disassembled and transmitted to the PPS. A RD XMT SHIFT pulse from the read control logic then shifts the 60-bit words in rank 2 forward one stage. The sequence of D SEG 0 through D SEG 4 begins again, and a second 60-bit word disassembles and transmits to the PPS. This continues until eight 60-bit words transfer to the PPS. If the reference is for more than one ECS record, the scanner is reengaged on its next cycle, and a second ECS record transfers into rank 1. Data transfer continues in this way until the read operation is complete.

† For more information, refer to the CDC CYBER 170 Input/Output Specifications Manual, listed in the preface.

During a read operation, the read control logic generates ADV ADRS, which increments the address register for each 60-bit word received by the port.

BASIC WRITE OPERATION

Refer to Figure 4-2 during the following discussion.

To initiate a write operation, the PPS sends, in parallel, a 12-bit function code word and a FUNCTION signal to a DDP port via an I/O channel.† If the equipment select and function codes are correct, the port's function decoder partially enables for a write operation and inactivates the I/O channel. The PPS then activates the I/O channel and transmits two 12-bit data words which indicate the address into which the first ECS reference writes. The FULL signal accompanying each address word causes the address register to load the first address word into the upper 12 bits and the second address word into the lower 12 bits. An EMPTY signal is transmitted to the PPS after each address word is received. After the two address words are loaded, ADRS is generated which causes the function decoder to generate W (write).

W causes the write control logic to sequentially generate A SEG 0 through A SEG 4 as 12-bit words are received from the PPS. These signals sequentially load consecutive 12-bit input segments of rank 1 of the data storage register. When five consecutive 12-bit segments have been loaded, a 60-bit word is assembled at the input stage of the data storage register. The write control logic causes an EMPTY signal to transmit to the PPS after each 12-bit word enters the storage register. The A SEG 0 through A SEG 4 sequence repeats until eight 60-bit words have assembled in the data storage register. The write control logic then generates an XFR SHIFT pulse.

XFR SHIFT pulses transfer the eight 60-bit data words from rank 1 into rank 2 of the data storage register. After the data is in rank 2, the write control logic generates P REQ (port request). When a P REQ signal is detected by the scanner, it sends P SEL to the write control logic. The write control logic then generates ENBL ADRS which gates the 24-bit address word, STORE and ECS REQ out to the ECS controller. STORE indicates that the operation is a write. If the ECS controller can transfer data, it sends an ACCEPT signal to the DDP. The ACCEPT signal causes the write control logic to generate WR XMT SHIFT and P ENBL. WR XMT SHIFT shifts the 60-bit words out of rank 2 of the data storage register. P ENBL gates the 60-bit words to the ECS transmitter. A WR GO signal accompanies each 60-bit word to the ECS controller.

† For more information, refer to the CDC CYBER 170 Input/Output Specifications Manual, listed in the preface.

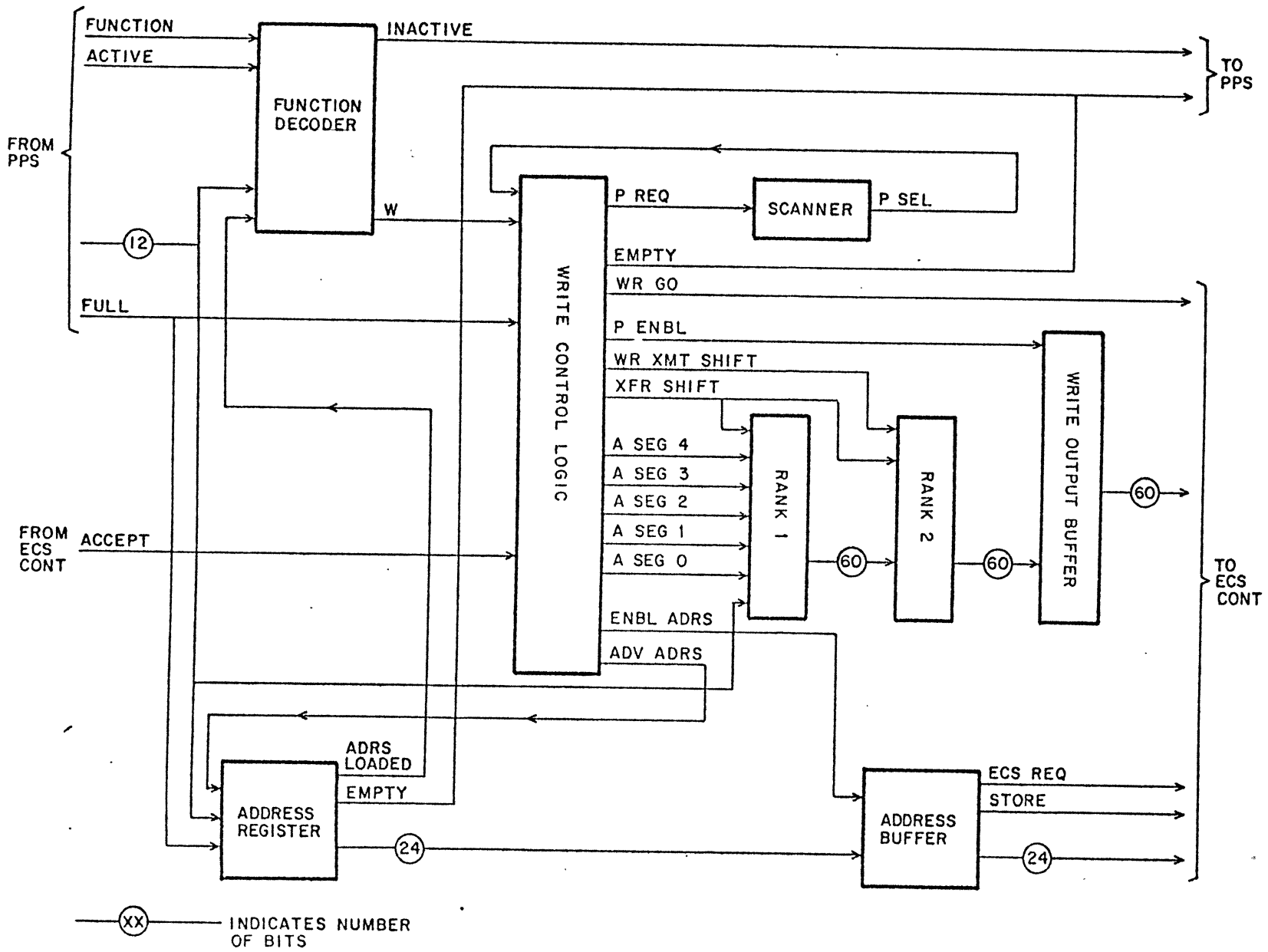


Figure 4-2. Basic Write Operation (One Port Shown)

After eight 60-bit words transfer to the ECS controller, the scanner moves on to monitor the next port. The scanner must be reengaged if the reference is for more than one ECS record. Data transfer continues in this way until the write operation is complete.

During a write operation, ADV ADRS increments the address register for each 60-bit word sent to the ECS controller.