60420300

CONTROL DATA CORPORATION

CDC® CYBER 170
MODELS 175, 740, 750, 760, 865, 875
FUNCTIONAL UNITS

THEORY OF OPERATION
DIAGRAMS

HARDWARE MAINTENANCE MANUAL

REVISION RECORD

REVISION	DESCRIPTION
01	Preliminary edition.
(12-74)	
02	Updated diagrams to reflect ECO 36000. This edition obsoletes previous edition.
(2-75)	
03	Updated diagrams to reflect ECOs 35738, 35539, 35693, and 36121. This edition obsoletes pre-
(5-75)	vious editions.
A	Manual released. This edition obsoletes all previous editions.
(7-75)	
B	No change to this manual (ECO 36403).
(9-75)	
C	Updated manual to reflect ECO 36429.
(9-75)	
D	Updated manual to reflect ECO 36183.
(9-75)	
E	Updated manual to reflect ECO 36724. (ECO PD1346 did not list this manual.)
(9-75)	
F	Updated manual to reflect ECO/FCO 36194.
(9-75)	
G	Updated manual to reflect ECO/FCO 36699.
10-75)	
H	Updated manual to reflect ECO 36185.
(11-75)	
J	Updated manual to reflect ECO 36849.
(12-75)	
K	Updated manual to reflect ECO/FCO 36197. Added secondary block and ESE diagrams. This
(2-76)	edition obsoletes all previous editions.
L	Manual revised; includes Field Change Order 37147. Pages vi, 5-1-4, 5-1-5 in volume 1, vi, viii,
(12-76)	5-11-11, 5-11-13, 5-11-29, 5-11-33, 5-11-34.8, 5-11-35, 5-11-53, 5-11-71, 5-11-147, 5-12-3, 5-12-17,
	5-12-25, 5-12-47, 5-12-49, 5-12-65, and 5-14-3 in volume 2 are revised.
M	Manual revised; includes Engineering Change Order 37731. Page 5-1-5 in volume 1 and pages 5-12-3,
(8-77)	5-12-5, 5-12-6.1, 5-12-6.3, 5-12-6.4, 5-12-7, 5-12-12.2, 5-12-13, and 5-14-1 in volume 2 are revised.
N	Manual revised; includes Field Change Order 37843 (ECO 37767). Pages vi, x, 5-1-5, 5-4-5, 5-5-1,
(8-77)	5-7-5, 5-7-15, 5-10-9, and 5-10-13 in volume 1 and pages vi, ix, 5-11-5, 5-11-13, 5-11-17, 5-11-19,
	5-11-29, 5-11-33, 5-11-35, 5-11-93, 5-11-153, 5-12-1, 5-12-3, 5-12-5, 5-12-7, 5-12-9, 5-12-10.2,
	5-12-11, 5-12-13, 5-12-15, 5-12-17, 5-12-25, 5-12-43, 5-12-63, 5-12-65, 5-12-69, 5-12-71, 5-12-78.5,
	Part 13 divider, 5-13-1, and 5-13-9 in volume 2 are revised. Page 5-12-73.1 is added in volume 2.

Publication No.
60420300

REVISION LETTERS I, O, Q, S, X AND Z ARE NOT USED.

Address comments concerning this manual to:
Control Data Corporation
Publications and Graphics Division
4201 North Lexington Avenue
St. Paul, Minnesota 55112

or use Comment Sheet in the back of this manual.

AA 4005 REV. 3/83

PRINTED IN U.S.A.

REVISION RECORD (CONT'D)

REVISION	DESCRIPTION
P	Manual revised; includes Engineering Change Order 37722. Removes volume 2 and adds model C
(10-77)	information. Front cover, title page and pages iv, v, vi, vii, x, 5-1-2, 5-1-4, and 5-1-5 are
	revised.
R	Manual revised; includes Engineering Change Order 38980. Pages iii, iv, v, vi, 5-1-4, 5-1-5,
(04-10-78)	5-2-15, 5-3-15, 5-4-23, 5-5-41, 5-6-11, 5-7-81, 5-8-45, 5-8-53, 5-9-11, 5-10-17, 5-10-19, and
	5-10-21 are revised. Page iv-a/iv-b is added. Pages 5-1-7, 5-2-9, 5-2-11, 5-2-13, 5-4-3, 5-4-9,
	5-4-11, 5-5-1, 5-5-31, 5-6-7, 5-7-17, 5-7-19, 5-7-25, 5-7-29, 5-7-31, 5-7-33, 5-7-35, 5-7-37,
	5-7-39, 5-7-41, 5-7-43, 5-7-49, 5-7-51, 5-7-55, 5-7-67, 5-7-69, 5-7-83, 5-8-23, 5-8-27, 5-8-29,
	5-8-43, and 5-10-15 are reprinted for purposes of clarity.
S	Manual revised; includes Engineering Change Order 39978. Front cover, title page and pages
(05-10-79)	iv-a/iv-b, v, vii, viii, ix, x, xi, 5-1-2, 5-1-9, 5-1-11, 5-1-13, 5-8-1, 5-8-3, 5-8-5, 5-8-7,
	5-8-19, 5-8-47, and 5-8-49 are revised. Pages 5-1-15 and 5-1-17 are added.
T	Engineering Change Order 39735. No change to this manual.
(07-05-79)	
U	Manual revised; includes Engineering Change Order 40383, publications change only. Pages ii-a/ii-b,
(07-6-79)	iv-a, and v are revised. Page iv-b is added.
V	Manual revised; includes Engineering Change Order 40489. Pages ii-a/ii-b, iv-a/iv-b, vii, xi,
(11-26-79)	5-1-1, 5-1-2, 5-1-4, 5-1-5, and 5-1-6 are revised.
W	Manual revised; includes Engineering Change Order 41276. Grid locators 52B2, 52B4, 52B6, 52B7, 52B8,
(03-16-81)	52B9, 52B10, 52B12, 52B13, 52E3/E4, and 52E7/E8 are revised.
Y	Manual revised to include Engineering Change Order 44495 which adds support for models 865 and 875.
(05-30-83)	This edition obsoletes the microfiche edition.
AA	Manual revised; includes Engineering Change Order 44802. Pages iii through iv-c and viii are
(08-05-83)	revised. Pages iv-d and iv-e are added.

Publication No.
60420300

MANUAL TO EQUIPMENT LEVEL CORRELATION SHEET

This manual reflects the equipment configurations listed below.

EXPLANATION: Locate the equipment type and series number, as shown on the equipment FCO log, in the list below. Immediately to the right of the series number is an FCO number. If that number and all of the numbers underneath it match all of the numbers on the equipment FCO log, then this manual accurately reflects the equipment.

EQUIPMENT TYPE	SERIES	WITH FCOs	COMMENTS
AA120	A01		Released
	A02	EC035091	
	A03	EC035467	
	A04	EC036623	
	A05	EC036634	
	A06	EC036637	
	A06	EC036403	
	A06	EC036429	
	A06	EC036183	
	A06	EC036724	
	A07	FC036579	
	A08	FC036699	
	A09	FC036194	
	A09	EC036185	
	A09	EC036849	
	A10	FCOPD1407	
	A11	FCOPD1377	
	A12	FCOPD1408	
	A13	FC036199	
	A14	FCOPD1511	
	A15	FCOPD1456	
	A16	FCOPD1392	
	A17	FCOPD1454	
	A18	FCOPD1597	
	A19	FCOPD1562	
	A20	FC036197	
	A21	FCOPD938	
	A21	Included	
	A22	FC036639	
	A23	FC036647	
	A24	FC036624	
	A25	FC036641	
	A26	FC036999	
	A27	FC036646	
	A28	FC036645	
	A29	FC036642	
	A30	FC036681	
	A31	FC036644	
	A32	FC036656	
	A33	FC036654	
	A34	FC036630	
	A35	FC037073	
	A36	FC037054	
	A37	FC036631	
	A38	FC036653	
	A39	FC036651	
	A40	FC036652	
	A41	FC036875	
	A42	FC036626	
	A43	FC036192	

EQUIPMENT TYPE	SERIES	WITH FCOs	COMMENTS
	A44	FC036854	
	A45	FC037362	
	A46	FC037530	
	A47	FC037147	
	A48	FC036643	
	A52	EC037731	
	A53	FC037843	
	A53	FC037949	
	A54	FC038031	
	A55	FC038236	
	A56	FC038135	
	A57	FC037840	
	A58	FC038252	
	A59	FC038308	
	A60	FC038171	
	A60	FC038022	
	A60	EC037722	
	A61	FC037813	
	A62	FC038338	
	A63	FC038386	
	A63	FC038712	
	A64	FC038418	
	A64	EC038980	
	A64	EC038856	
	A64	EC038858	
	A65	EC038897	
	A65	EC039044	
	A66	EC038584	
	A66	EC038670	
	A67	EC038764	
	A67	EC039041	
	A67	EC038781	
	A67	EC038871	
	A68	EC038941	
	A68	EC039411	
	A69	EC039368	
	A69	EC039319	
	A70	EC038867	
	A70	EC039978	
	A70	EC039735	
	A70	EC040383	
	A71	FC039396	
	A71	FC039678	
	A72	FC039800/ FC039778	
	A73	FC039590/ FC039773	
	A74	FC039732	
	A74	FC039726	
	A74	EC040489	
	A75	FC040748/ FC040843	
	A75	FC040479	
	A75	EC041276	
	B01	Included	
	B01	FC036639	
	B01	FC036647	
	B01	FC036624	
	B01	FC036641	
	B01	FC036999	
	B01	FC036646	
	B01	FC036645	
	B01	FC036642	
	B01	FC036681	
	B01	FC036644	
	B01	FC036656	
	B01	FC036654	

EQUIPMENT TYPE	SERIES	WITH FCOs	COMMENTS
	B01	FC036630	
	B01	FC037073	
	B01	FC037054	
	B01	FC036631	
	B01	FC036653	
	B01	FC036651	
	B01	FC036652	
	B01	FC036875	
	B01	FC036626	
	B01	FC036192	
	B01	FC036854	
	B02	FC037362	
	B02	FC037530	
	B02	FC037147	
	B02	FC036643	
	B05	EC037731	
	B06	FC037843	
	B06	FC037949	
	B07	FC038031	
	B08	FC038236	
	B09	FC038135	
	B10	FC037840	
	B11	FC038252	
	B11	FC038308	
	B12	FC038171	
	B13	FC038022	
	B13	EC037722	
	B14	FC037813	
	B15	FC038338	
	B16	FC038386	
	B17	FC038712	
	B18	FC038418	
	B18	EC038980	
	B18	EC038856	
	B18	EC038858	
	B18	EC038897	
	B19	EC039044	
	B20	EC038584	
	B20	EC038670	
	B21	EC038764	
	B22	EC039041	
	B22	EC038781	
	B22	EC038871	
	B23	EC038941	
	B24	EC039411	
	B25	EC039368	
	B25	EC039319	
	B26	EC038867	
	B26	EC039978	
	B26	EC039735	
	B26	EC040383	
	B27	FC039396	
	B27	FC039678	
	B28	FC039800/ FC039778	
	B29	FC039590/ FC039773	
	B29	FC039732	
	B30	FC039726	
	B30	EC040489	
	B31	FC040748/ FC040843	
	B31	FC040479	
	B31	EC041276	
	C02	FC038135	
	C02	FC037840	
	C02	FC038252	

EQUIPMENT TYPE	SERIES	WITH FCOs	COMMENTS
	C02	FC038308	
	C03	FC038171	
	C03	FC038022	
	C03	EC037722	
	C03	FC037813	
	C04	FC038338	
	C04	FC038386	
	C05	FC038712	
	C06	FC038418	
	C06	EC038980	
	C07	EC038856	
	C08	EC038858	
	C08	EC038897	
	C09	EC039044	
	C10	EC038584	
	C11	EC038670	
	C11	EC038764	
	C12	EC039041	
	C13	EC038781	
	C14	EC038871	
	C15	EC038941	
	C16	EC039411	
	C17	EC039368	
	C18	EC039319	
	C18	EC038867	
	C18	EC039978	
	C18	EC039735	
	C18	EC040383	
	C19	FC039396	
	C19	FC039678	
	C19	FC039800/ FC039778	
	C20	FC039590/ FC039773	
	C20	FC039732	
	C21	FC039726	
	C21	EC040489	
	C21	FC040748/ FC040843	
	C22	FC040479	
	C22	EC041276	
	D01	EC039978	
	D01	EC039735	
	D01	EC040383	
	D01	FC039396	
	D01	FC039678	
	D01	FC039800/ FC039778	
	D01	FC039590/ FC039773	
	D01	FC039732	
	D01	FC039726	
	D01	EC040489	
	D01	FC040748/ FC040843	
	D01	FC040479	
	D01	EC041276	
	D01		Released
	D02	FC039944	
	D03	FC040112	
	D04	FC040106	
	D05	FC039936/ FC040225	
	D06	FC040205	
	D07	FC040221	
	D08	FC040432	

EQUIPMENT TYPE	SERIES	WITH FCOs	COMMENTS
	D09	EC039947/ FC040364	
	D10	FC040379	
	D11	FC039945	
	D12	FC040607	
	D13	FC040658	
	D14	FC040030	
	D15	FC040583	
	D16	FC040589A	
	D17	FC040740	
	D18	FC040857	
	D19	FC040830	
	D20	FC040827	
	D21	FC040915	
	D22	FC040904	
	D23	FC041012	
	D24	FC041021	
	D25	FC041043	
	D26	EC041358	
	D27	FC041346	
	D28	EC041363	
	D29	FC041359	
	D30	FC041833	
	D31	EC041632	
	D32	EC041639	
	D33	FC041636	
	D34	EC041790	
	D35	FC042000	
	D36	EC041372	
	D37	FC042050	
	D38	FC041736	
	D39	FC042323	
	D40	FC042348	
	D41	FC042913	
	E01	EC044495	
AT364	A01	EC038980	
	A01	EC038856	
	A01	EC038858	
	A01	EC038897	
	A01	EC039044	
	A01	EC038584	
	A01	EC038670	
	A01	EC038764	
	A01	EC039041	
	A01	EC038781	
	A01	EC038871	
	A01	EC038941	
	A01	EC039411	
	A01	EC039368	
	A01	EC039319	
	A01	EC038867	
	A01	EC039978	
	A01	EC039735	
	A01	EC040383	
	A01	FC039396	
	A01	FC039678	
	A01	FC039800/ FC039778	
	A01	FC039590/ FC039773	
	A01	FC039732	
	A01	FC039726	
	A01	EC040489	
	A01	FC040748/ FC040843	
	A01	FC040479	

EQUIPMENT TYPE	SERIES	WITH FCOs	COMMENTS
	A01	EC041276	
	A01		Released
	A01	FC039944	
	A01	FC040112	
	A01	FC040106	
	A01	FC039936/ FC040225	
	A01	FC040205	
	A01	FC040221	
	A01	FC040432	
	A01	EC039947/ FC040364	
	A01	FC040379	
	A01	FC039945	
	A01	FC040607	
	A01	FC040658	
	A01	FC040030	
	A01	FC040583	
	A01	FC040589A	
	A01	FC040740	
	A01	FC040857	
	A01	FC040830	
	A01	FC040827	
	A01	FC040915	
	A01	FC040904	
	A01	FC041012	
	A01	FC041021	
	A01	FC041043	
	A01	EC041358	
	A01	FC041346	
	A01	EC041363	
	A01	FC041359	
	A01	FC041833	
	A01	EC041632	
	A01	EC041639	
	A01	FC041636	
	A01	EC041790	
	A01	FC042000	
	A01	EC041372	
	A01	FC042050	
	A01	FC041736	
	A01	FC042323	
	A01	FC042348	
	A01	FC042913	
	B01	EC039978	
	B01	EC039735	
	B01	EC040383	
	B01	FC039396	
	B01	FC039678	
	B01	FC039800/ FC039778	
	B01	FC039590/ FC039773	
	B01	FC039732	
	B01	FC039726	
	B01	EC040489	
	B01	FC040748/ FC040843	
	B01	FC040479	
	B01	EC041276	
	B01		Released
	B01	FC039944	
	B01	FC040112	
	B01	FC040106	
	B01	FC039936/ FC040225	
	B01	FC040205	

EQUIPMENT TYPE	SERIES	WITH FCOs	COMMENTS
	B01	FC040221	
	B01	FC040432	
	B01	EC039947/ FC040364	
	B01	FC040379	
	B01	FC039945	
	B01	FC040607	
	B01	FC040658	
	B01	FC040030	
	B01	FC040583	
	B01	FC040589A	
	B01	FC040740	
	B01	FC040857	
	B01	FC040830	
	B01	FC040827	
	B01	FC040915	
	B01	FC040904	
	B01	FC041012	
	B01	FC041021	
	B01	FC041043	
	B01	EC041358	
	B01	FC041346	
	B01	EC041363	
	B01	FC041359	
	B01	FC041833	
	B01	EC041632	
	B01	EC041639	
	B01	FC041636	
	B01	EC041790	
	B01	FC042000	
	B01	EC041372	
	B01	FC042050	
	B01	FC041736	
	B01	FC042323	
	B01	FC042348	
	B01	FC042913	
AT402	A01	EC040383	
	A01	EC040489	
	A01	EC041276	

PREFACE

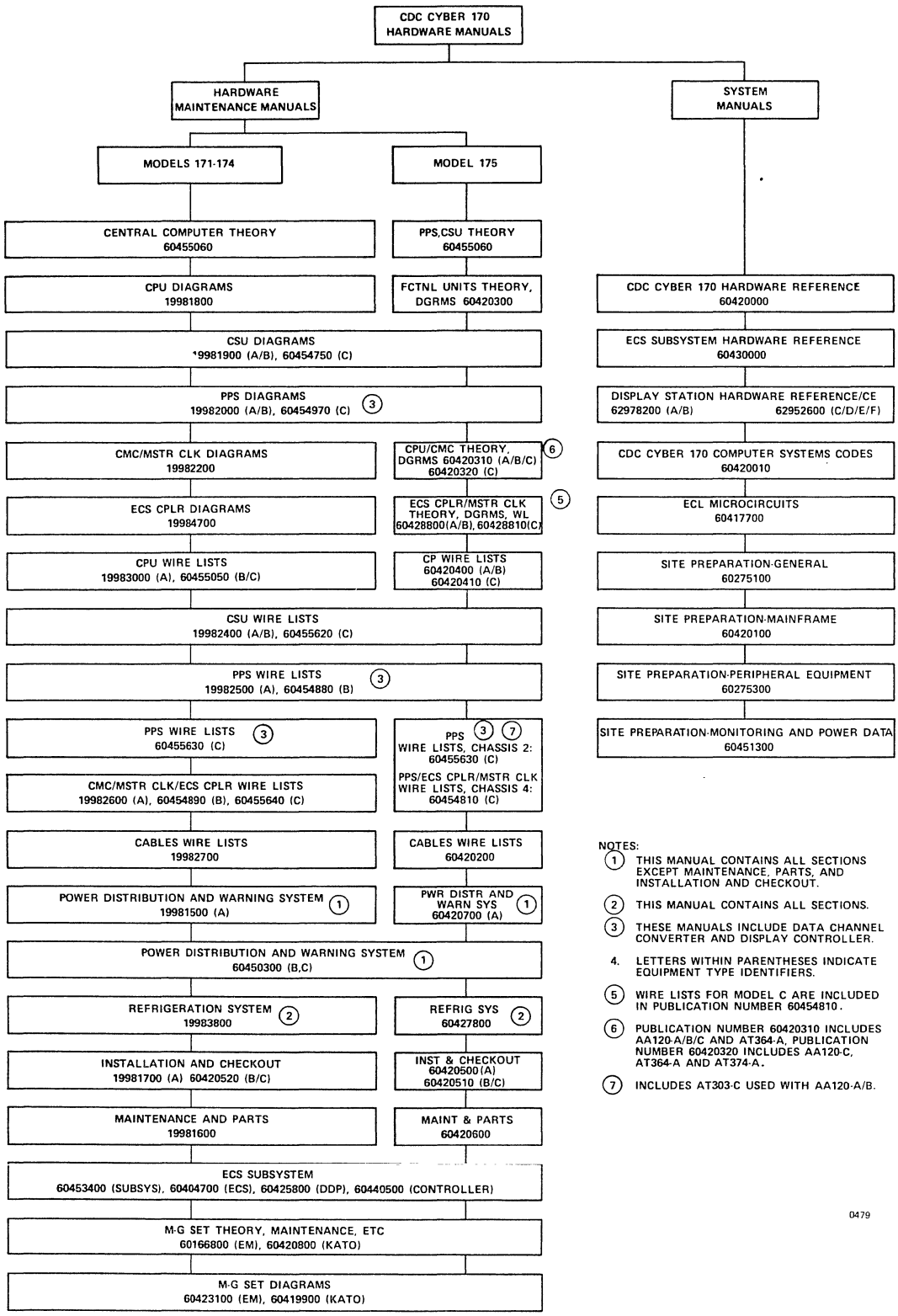
This manual contains theory of operation and diagrams for the functional unit portion of the CONTROL DATA® AA120-A/B/C/D/E Central Computer and the AA131-D Central Computer with an AT402-A Upgrade Option installed. Also included is information about the AT364-A/B Central Processor Enhancement which is used with the AA120-C/D Central Computer.

Models 740, 750, and 760 are defined as 7X0. Models 865 and 875 are defined as 8X5.

Comparable information on the central processing unit (CPU), central memory control (CMC), central storage unit (CSU), peripheral processor subsystem (PPS), and extended core storage (ECS) coupler is contained in other manuals. The system publication indexes on the following pages graphically list the related publications. Refer to the Literature and Distribution Services catalog for the latest revision of each manual.

This manual contains a manual to equipment level correlation sheet. The last line of this sheet indicates the latest equipment level (series code) that the manual covers. All manuals for the central computer indicate the latest series code even if the particular manual is not affected by the field change order.

SYSTEM PUBLICATION INDEX

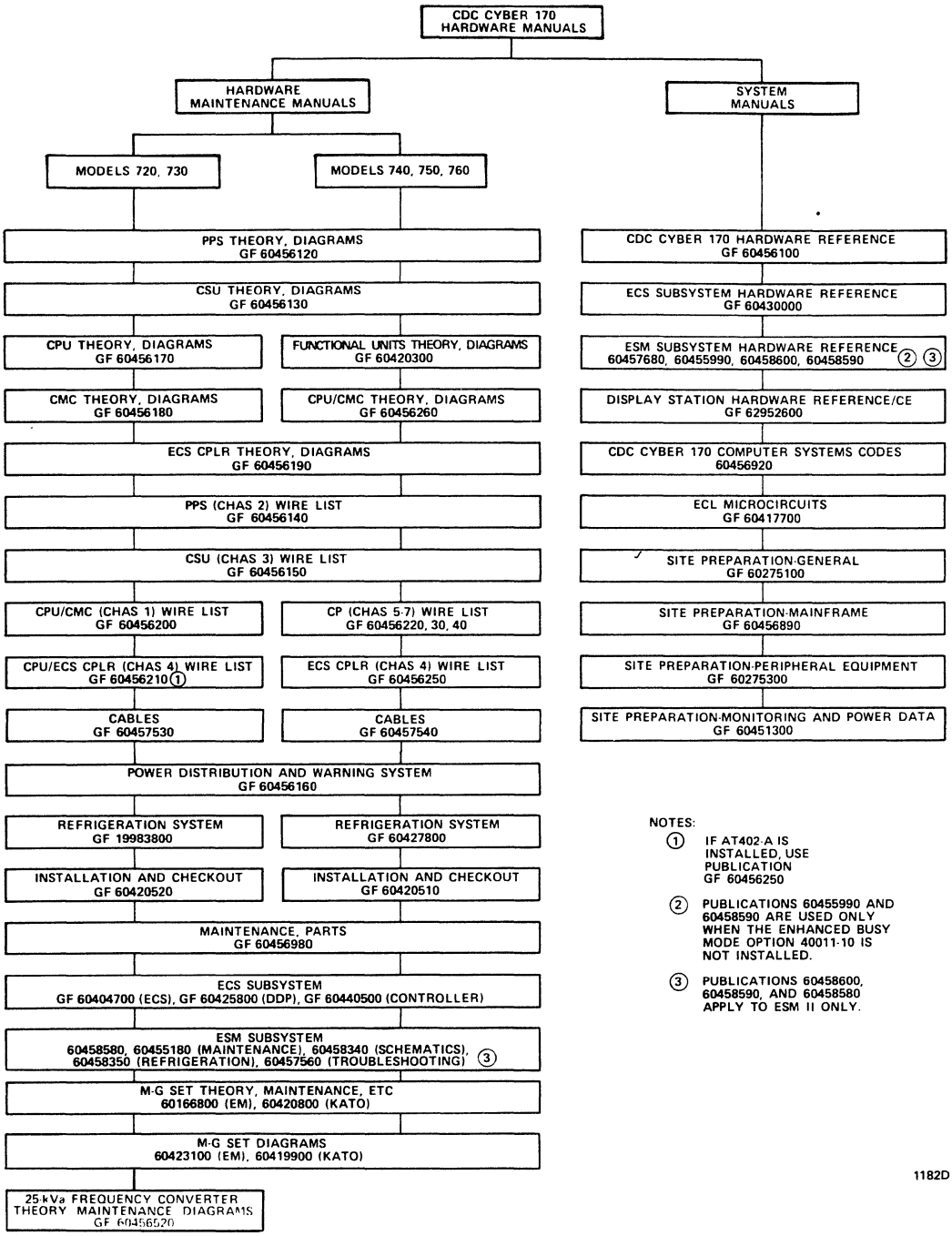


NOTES:

1. THIS MANUAL CONTAINS ALL SECTIONS EXCEPT MAINTENANCE, PARTS, AND INSTALLATION AND CHECKOUT.
2. THIS MANUAL CONTAINS ALL SECTIONS.
3. THESE MANUALS INCLUDE DATA CHANNEL CONVERTER AND DISPLAY CONTROLLER.
4. LETTERS WITHIN PARENTHESES INDICATE EQUIPMENT TYPE IDENTIFIERS.
5. WIRE LISTS FOR MODEL C ARE INCLUDED IN PUBLICATION NUMBER 60454810.
6. PUBLICATION NUMBER 60420310 INCLUDES AA120-A/B/C AND AT364-A, PUBLICATION NUMBER 60420320 INCLUDES AA120-C, AT364-A AND AT374-A.
7. INCLUDES AT303-C USED WITH AA120-A/B.

0479

SYSTEM PUBLICATION INDEX



NOTES:

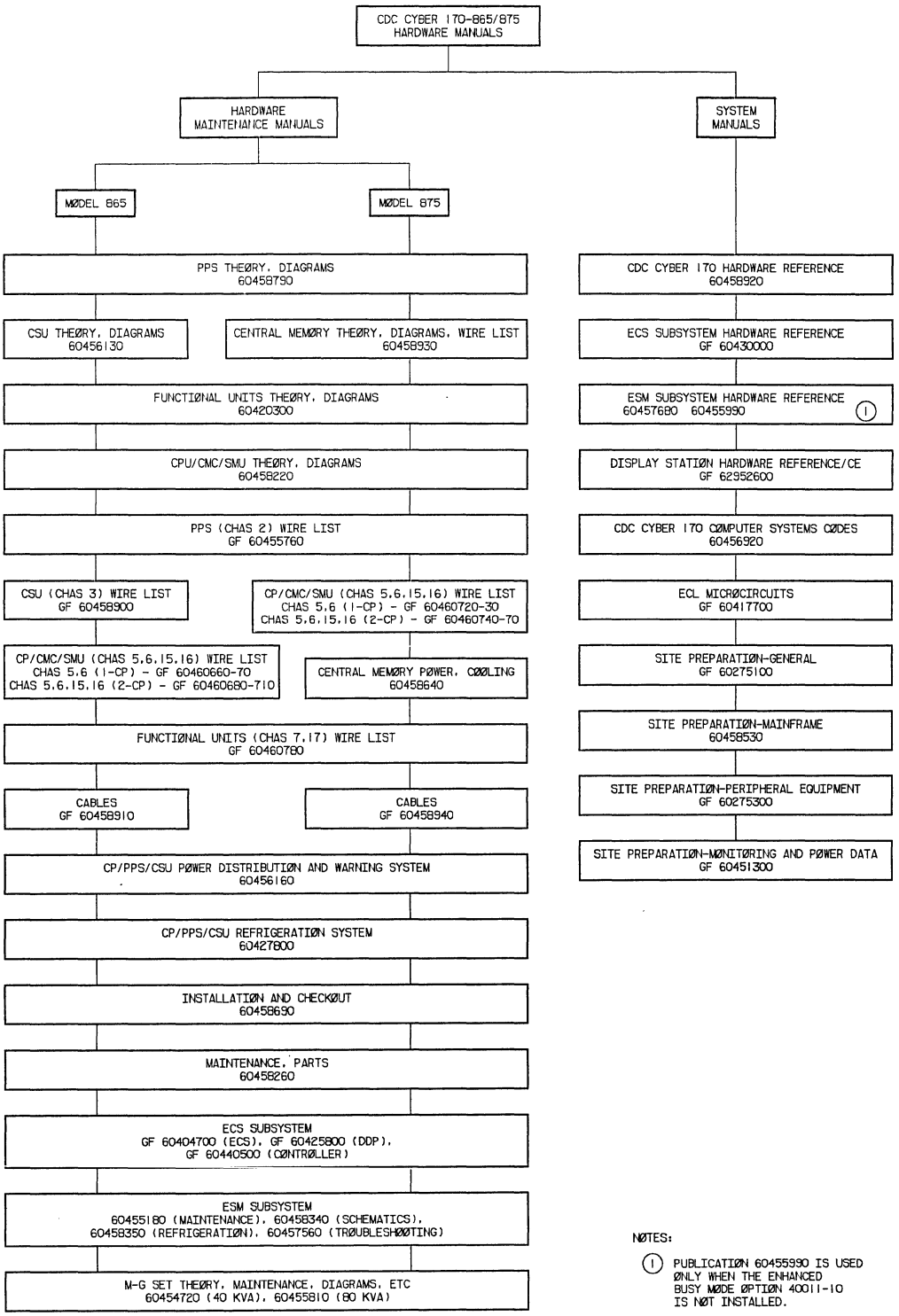
(1) IF AT402-A IS INSTALLED, USE PUBLICATION GF 60456250

(2) PUBLICATIONS 60455990 AND 60458590 ARE USED ONLY WHEN THE ENHANCED BUSY MODE OPTION 40011-10 IS NOT INSTALLED.

(3) PUBLICATIONS 60458600, 60458590, AND 60458580 APPLY TO ESM II ONLY.

1182D

SYSTEM PUBLICATIØN INDEX



NØTES:

(1) PUBLICATIØN 60455990 IS USED ØNLY WHEN THE ENHANCED BUSY MØDE ØPTIØN 40011-10 IS NØT INSTALLED.

10869 (06/22/85)

FIGURES

TABLE

SECTION 1

GENERAL DESCRIPTION

SECTION 2

OPERATION

(Information for sections 1 and 2 is contained in the CDC CYBER 170 Hardware Reference Manual.)

SECTION 3

INSTALLATION AND CHECKOUT

(Information for section 3 is contained in the CDC CYBER 170 Installation and Checkout Manual.)

SECTION 4

THEORY OF OPERATION

(Information for section 4 is combined with the diagrams in section 5 of this manual.)

C3

C4

SECTION 1

GENERAL DESCRIPTION

SECTION 2

OPERATION

(Information for sections 1 and 2 is contained in the CDC CYBER 170 Hardware Reference Manual, publication number 60420000.)

SECTION 3

INSTALLATION AND CHECKOUT

(Information for section 3 is contained in the CDC CYBER 170 Model 175 Installation and Checkout Manual, publication number 60420500.)

SECTION 4

THEORY OF OPERATION

(Information for section 4 is combined with the diagrams in section 5 of this manual.)

SECTION 5

DIAGRAMS

PART 1

INTRODUCTION

GLOSSARY

The glossary is a list of terms, mnemonics, and abbreviations used on the diagrams.

MODULE-TO-DIAGRAMS CROSS-REFERENCES

GENERAL

Table 5-1 provides a list of functional unit module types along with the following information on each type.

Quantity	Number of modules of this type located on the mainframe
Location	All physical locations at which this module type can be found
Diagram	Secondary block and detailed-modules diagrams on which this module type is depicted.

USES

Table 5-1 can be used to locate a particular module logic diagram in the manual without knowing the functional entity to which it pertains. Another use is to determine the availability and the location of a substitute module during maintenance. The table also helps to locate all modules of the same type without having to scan all module placement diagrams.

KEY TO SYMBOLS (Figures 5-1 and 5-2)

The number of unfamiliar symbols on the block and detailed-modules diagrams has been kept to a minimum. The symbology has been chosen because it simplifies or clarifies and is therefore essential to the use and understanding of the diagrams. Time spent familiarizing oneself with these conventions is not wasted. Note particularly that the AND and OR symbols define functions, not gates. In some cases, hardware constraints caused the use of AND gates to perform OR functions. Therefore, the block and detailed-modules diagrams depict the OR function, not the AND hardware.

SYSTEM BLOCK AND INTERFACE DIAGRAMS

The system block diagrams (figures 5-3, 5-4, 5-5) provide a comprehensive view of the central computer. In addition to aiding one's understanding of the system, it relates physical to electrical data (for example, multiply functional unit located on chassis 7) and defines the boundaries of the central processor diagrams in the various parts of this section.

The CP interface diagrams (figures 5-6, 5-7) depict all data and control paths between the functional entities. Also, this diagram includes all paths between the CP and the other portions of the central computer which are covered in other manuals.

The status and control register interface diagram (figure 5-8) includes all status/control paths in the central computer.

DIAGRAMS

Within each functional entity, the diagrams are arranged in the following order: primary, secondary, detailed-modules, and module logic. The diagrams are intended to ease the trainee gradually but quickly into the central processor or to allow the initiated to choose from four levels of drawings for maintenance assistance or memory jogging. The primary block diagram shows the basic relationship between the various logical components (registers, adders, and so on) comprising the functional entity. The next, more detailed level is the secondary block diagram. It depicts the physical building blocks (modules) and shows the data and control paths between them. The third and most explicit level is the detailed-modules diagram. This level is similar to the secondary level except that it also features a simplified block diagram of each unique module. From this level, the user must proceed to the module logic diagrams only for test point and connector information. However, when it is necessary to do so, the user arrives at the module logic diagram already familiar with its contents. Supplemental timing and troubleshooting information are provided at the end of each part of this section.

A test point chart is provided with each detailed-modules diagram. These charts list test points associated with signals and registers. Translator test points are not listed. An asterisk adjacent to the module type indicates that additional test points are available but not listed. Unless otherwise specified, all test points display the complement of the data indicated in the charts.

Maintenance personnel may find it helpful to reposition the pages in this section so the diagrams are not separated by theory of operation pages. Although the present arrangement is best for training purposes, contiguous diagrams may be easier to use during maintenance.

MODULE PLACEMENT DIAGRAMS

The module placement diagrams for all CP chassis (5, 6, and 7) are included in the CPU/CMC manual.

TABLE 5-1. MODULE-TO-DIAGRAMS CROSS-REFERENCES (Cont'd)

Module Type	Quantity	Location	Diagram	Module Type	Quantity	Location	Diagram
4KA7	3	6B06-08	POP CT 2.0, 3.0				
4KB7	1	6C06	POP CT 2.0, 3.0				
4KC7	5	6B01-05	LG ADD 2.0, 3.0				
4KD7	5	6C01-05	LG ADD 2.0, 3.0				
4KE7	4	6K09-12	INCR 2.0, 3.0				
4KF7	1	6K13	INCR 2.0, 3.0				
4KG7	2	6K14-15	INCR 2.0, 3.0				
†1KGH	2	6K14-15	INCR 2.0, 3.0				
4KH7	1	6K16	INCR 2.0, 3.0				
4K17	2	6L14-15	INCR 2.0, 3.0				
†1KIH	2	5K01-02	INCR 2.0, 3.0				
3KJ7	1	6L16	INCR 2.0, 3.0				
†1KJH	1	5K03	INCR 2.0, 3.0				
4KK7	7						
		6L12	INCR 2.0, 3.0;				
		6C07-11	BOOL 2.0, 3.0				
4KL7	3	6C12-14	BOOL 2.0, 3.0				
4KM7	1	6C15	BOOL 2.0, 3.0				
4KN7	1	6C16	BOOL 2.0, 3.0				
3KO7	1	6E10	SHIFT 2.0, 3.0				
4KP7	10	6A07-16	SHIFT 2.0, 3.0				
4KQ7	4	6B09-12	SHIFT 2.0, 3.0				
4KR7	2	7P02	DIV 2.0, 3.0				
		7F16	MULT 2.0, 3.5				
4KS7	1	7G16	MULT 2.0, 3.5				
4KU7	1	7H15	MULT 2.0, 3.5				
3KV7	1	7H14	MULT 2.0, 3.5				
4KW7	4	6B13-16	SHIFT 2.0, 3.0				

†Model 865/875 only.

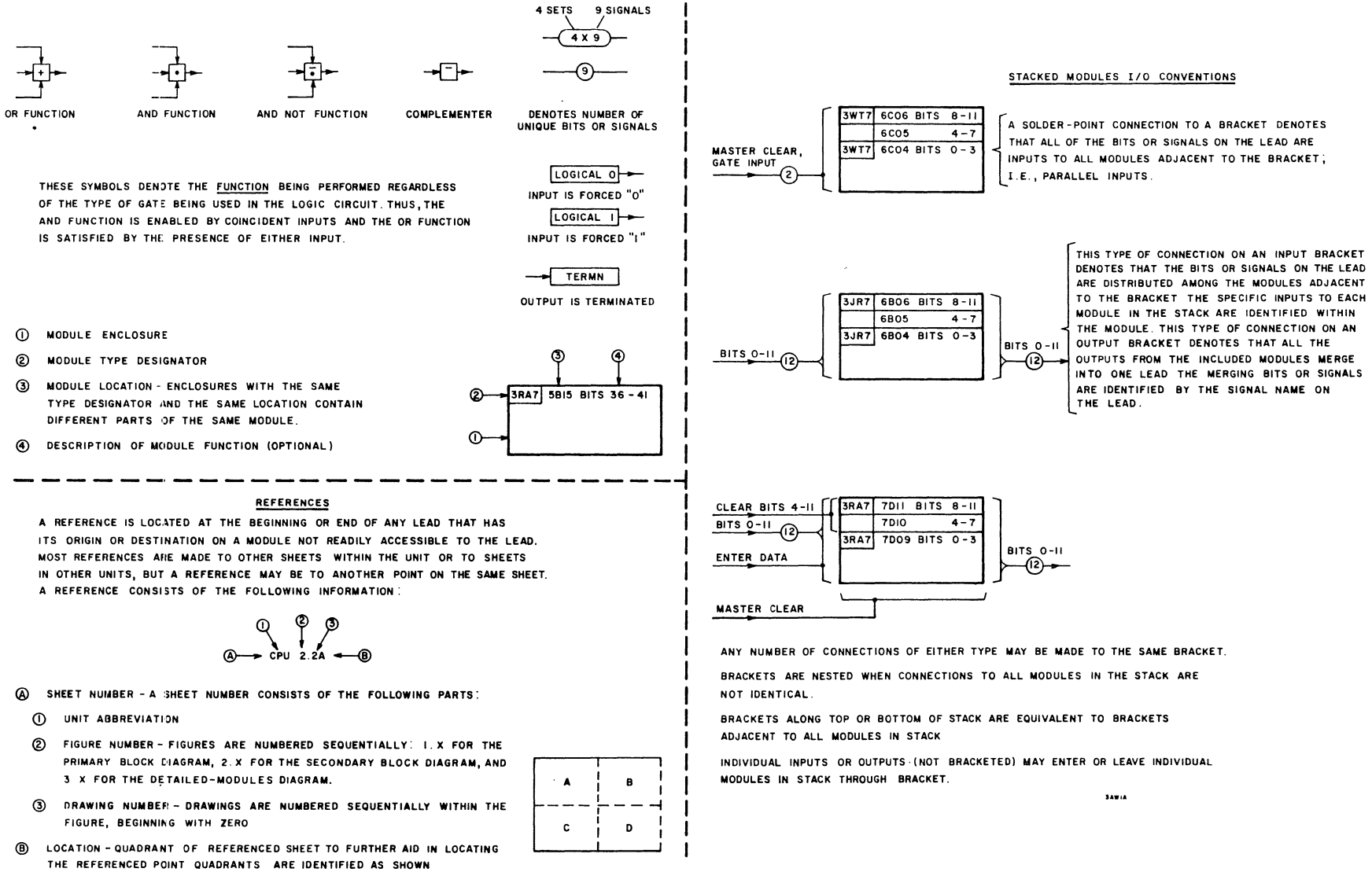


Figure 5-1. Key to Diagram Symbols

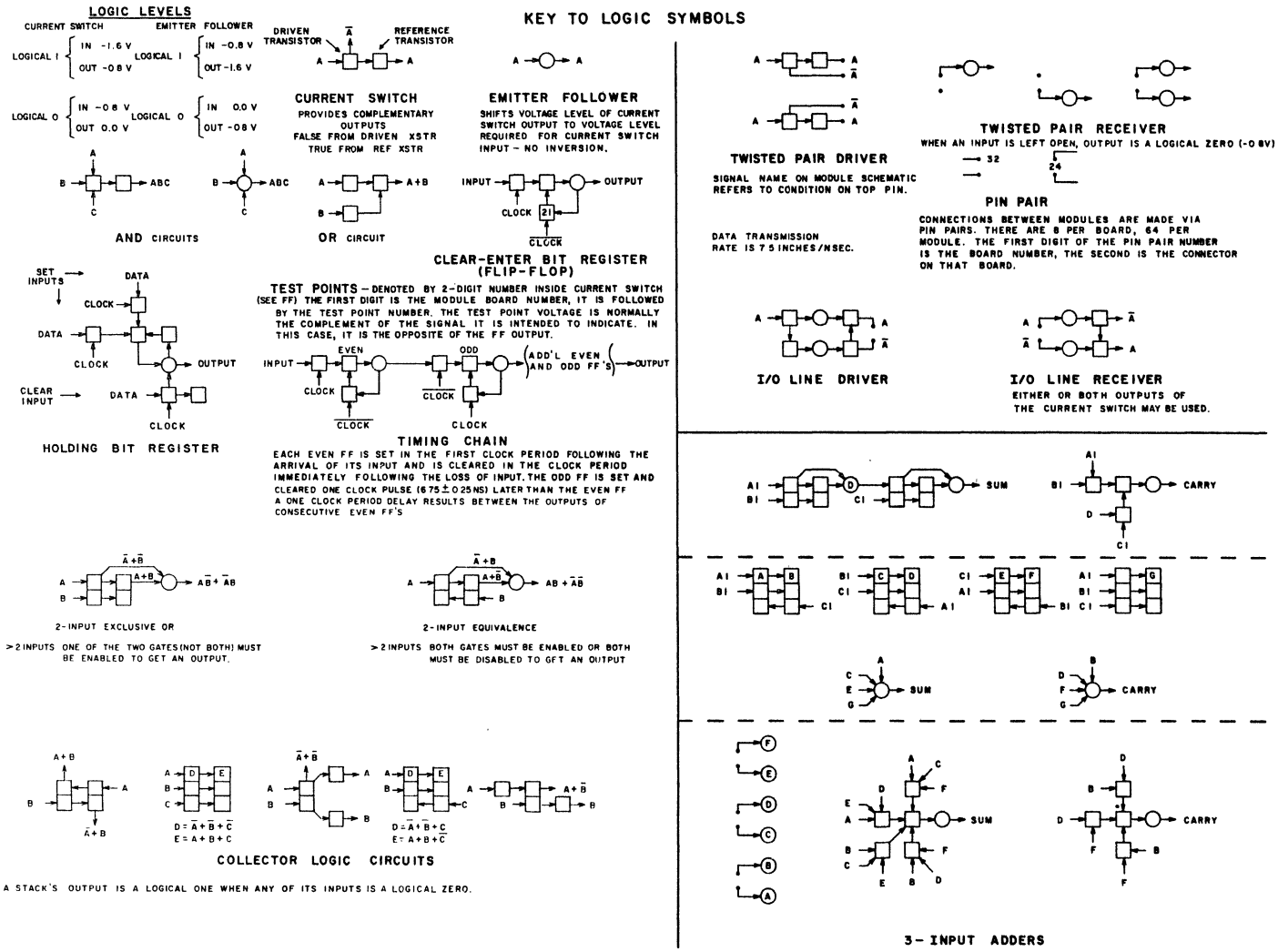


Figure 5-2. Key to Logic Symbols

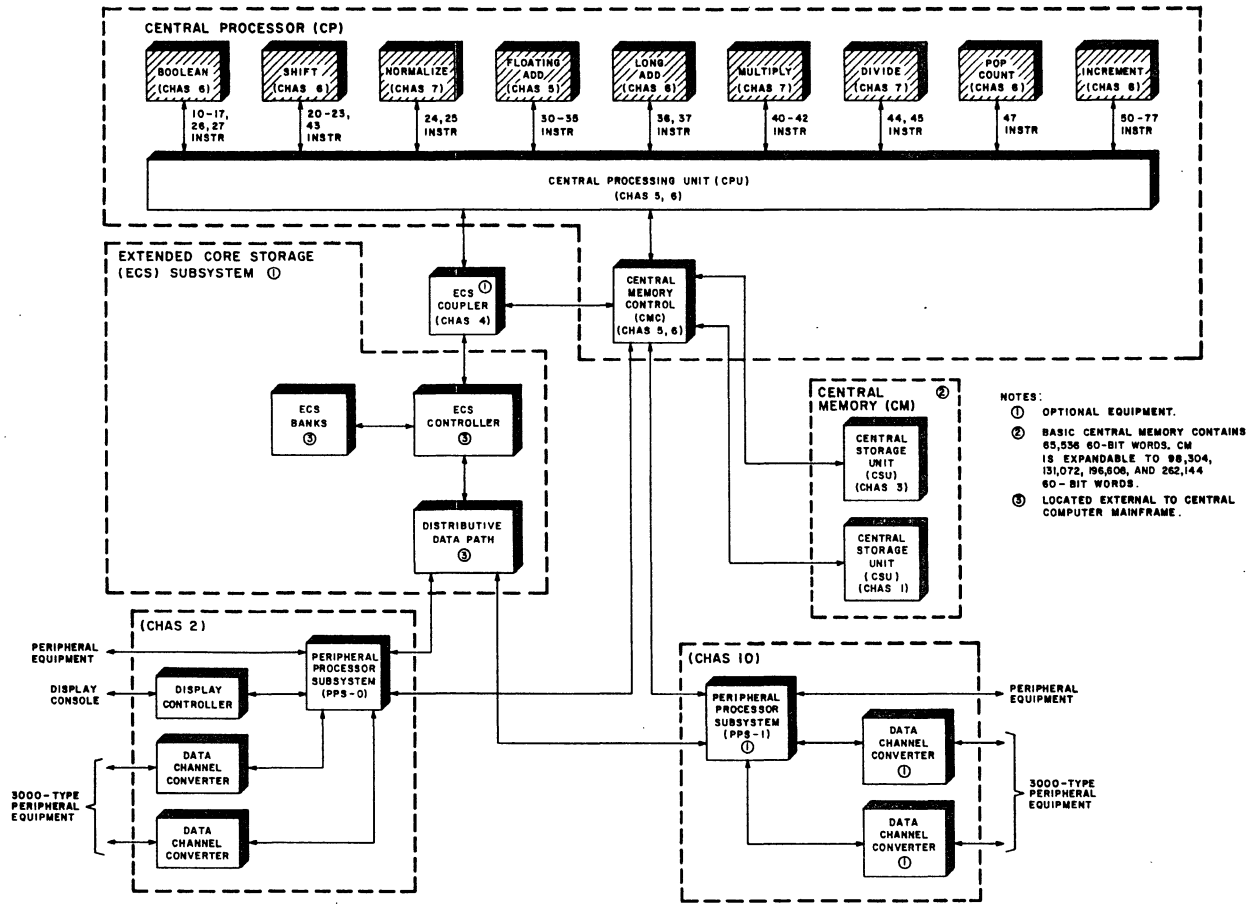
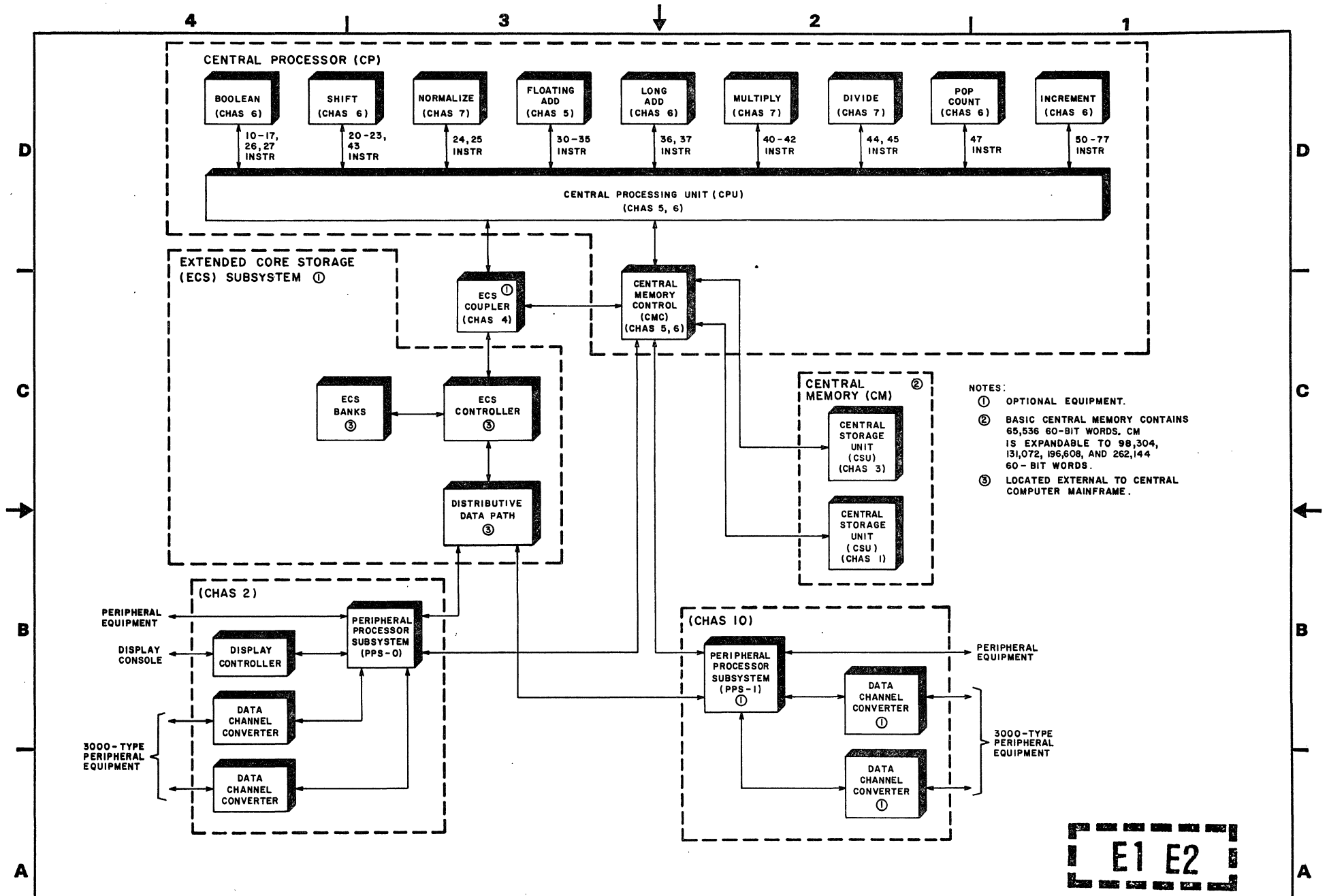
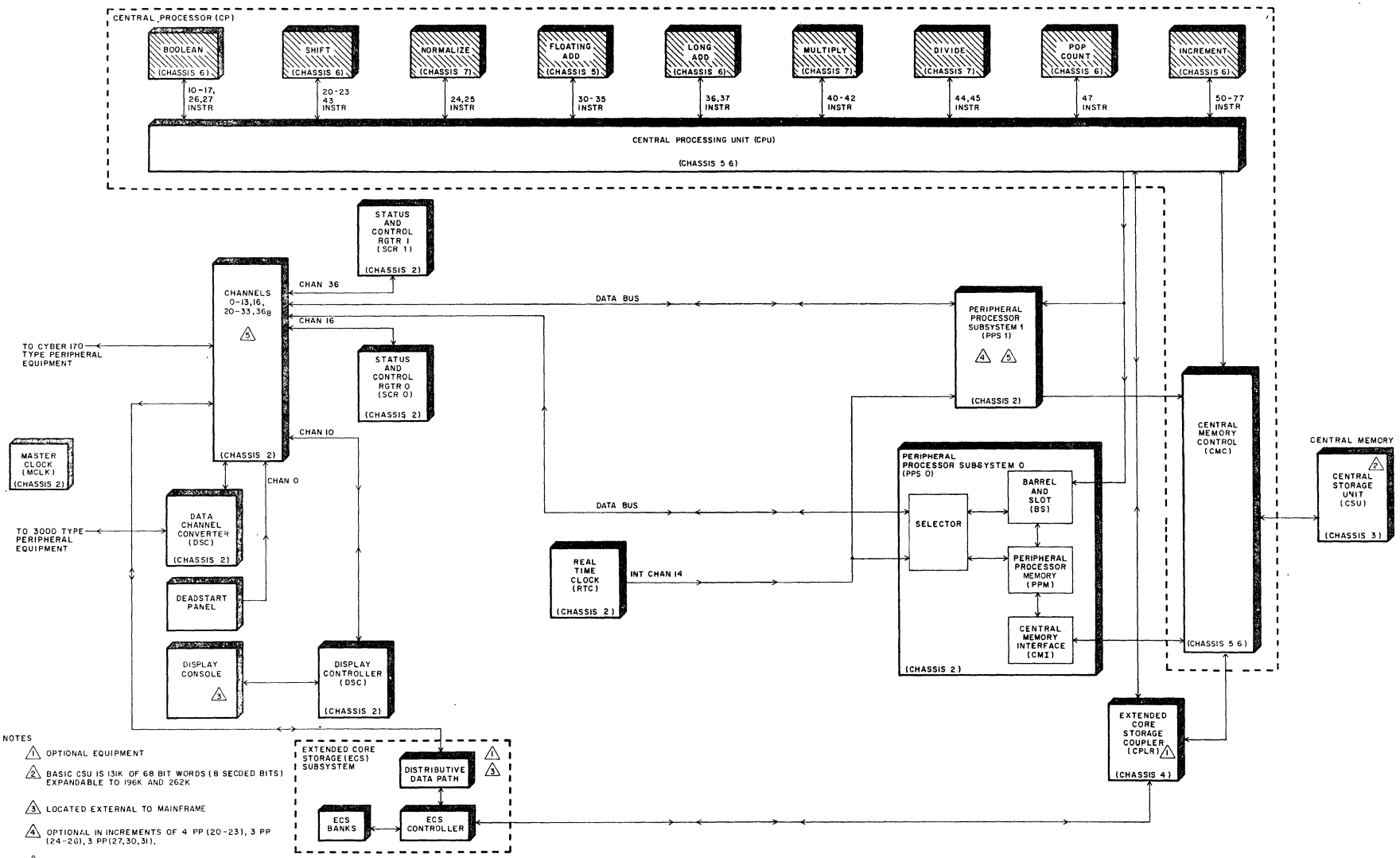


Figure 5-3. System Block Diagram, Model 175

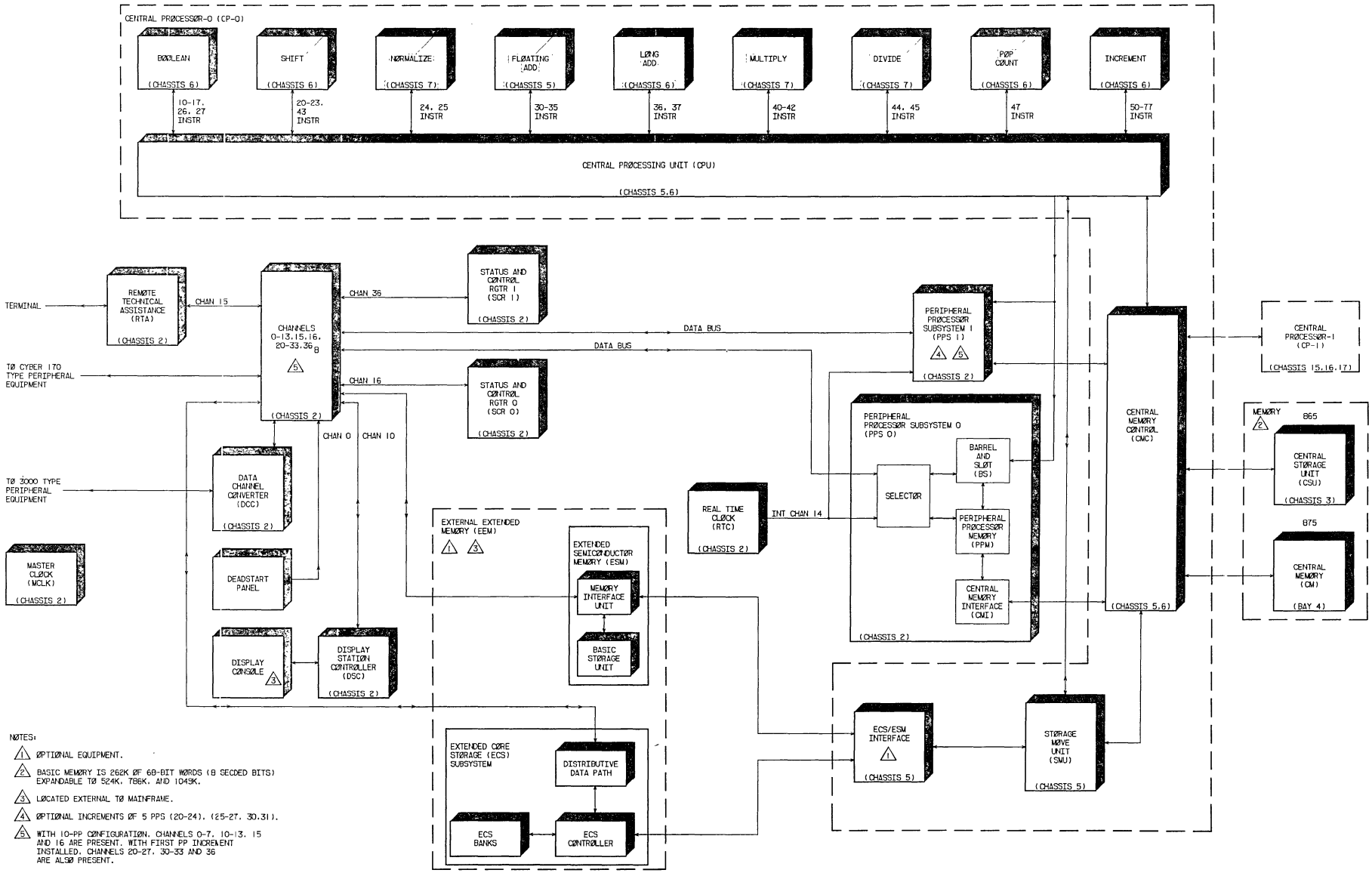


CONTROL DATA DEVELOPMENT DIVISION	SYSTEM BLOCK DIAGRAM CENTRAL COMPUTER MODEL 175	CODE IDENT 34010	DWG NO 60420300	REV S
		C	SHEET	PAGE 5-1-9



NOTES

1. OPTIONAL EQUIPMENT
2. BASIC CSU IS 131K OF 68 BIT WORDS (8 SECDED BITS) EXPANDABLE TO 196K AND 262K
3. LOCATED EXTERNAL TO MAINFRAME
4. OPTIONAL IN INCREMENTS OF 4 PP (20-23), 3 PP (24-26), 3 PP (27,30,31).
5. WITH 10-PP CONFIGURATION, CHANNELS 0-7, 10-13 AND 16 ARE PRESENT. WITH FIRST PP INCREMENT INSTALLED, CHANNELS 20-27, 30-33, AND 36 ARE ALSO PRESENT.



NØTES:

1. ØPTIØNAL EQUIPMENT.
2. BASIC MEMØRY IS 262K ØF 60-BIT WØRDS (8 SECDED BITS) EXPANDABLE TØ 524K, 786K, AND 1049K.
3. LØCATED EXTERNAL TØ MAINFRAME.
4. ØPTIØNAL INCREMENTS ØF 5 PPS (20-24), (25-27, 30,31).
5. WITH IO-PP CØNFIGURATIØN, CHANNELS 0-7, 10-13, 15 AND 16 ARE PRESENT. WITH FIRST PP INCREMENT INSTALLED, CHANNELS 20-27, 30-33 AND 36 ARE ALSØ PRESENT.

MØDELS 865, 875 SYSTEM BLØCK DIAGRAM

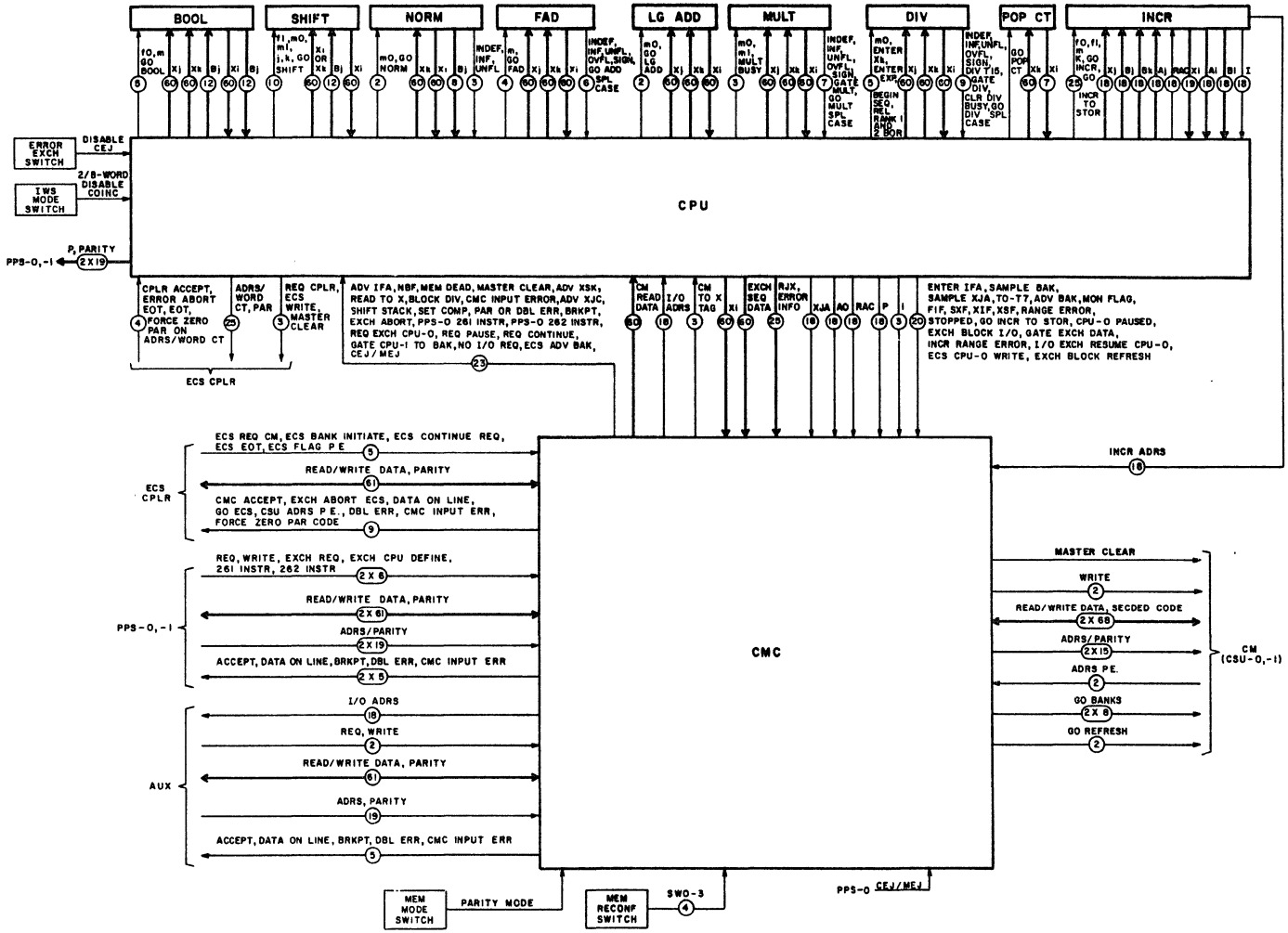


Figure 5-6. Interface Diagram, Central Processor, Model 175

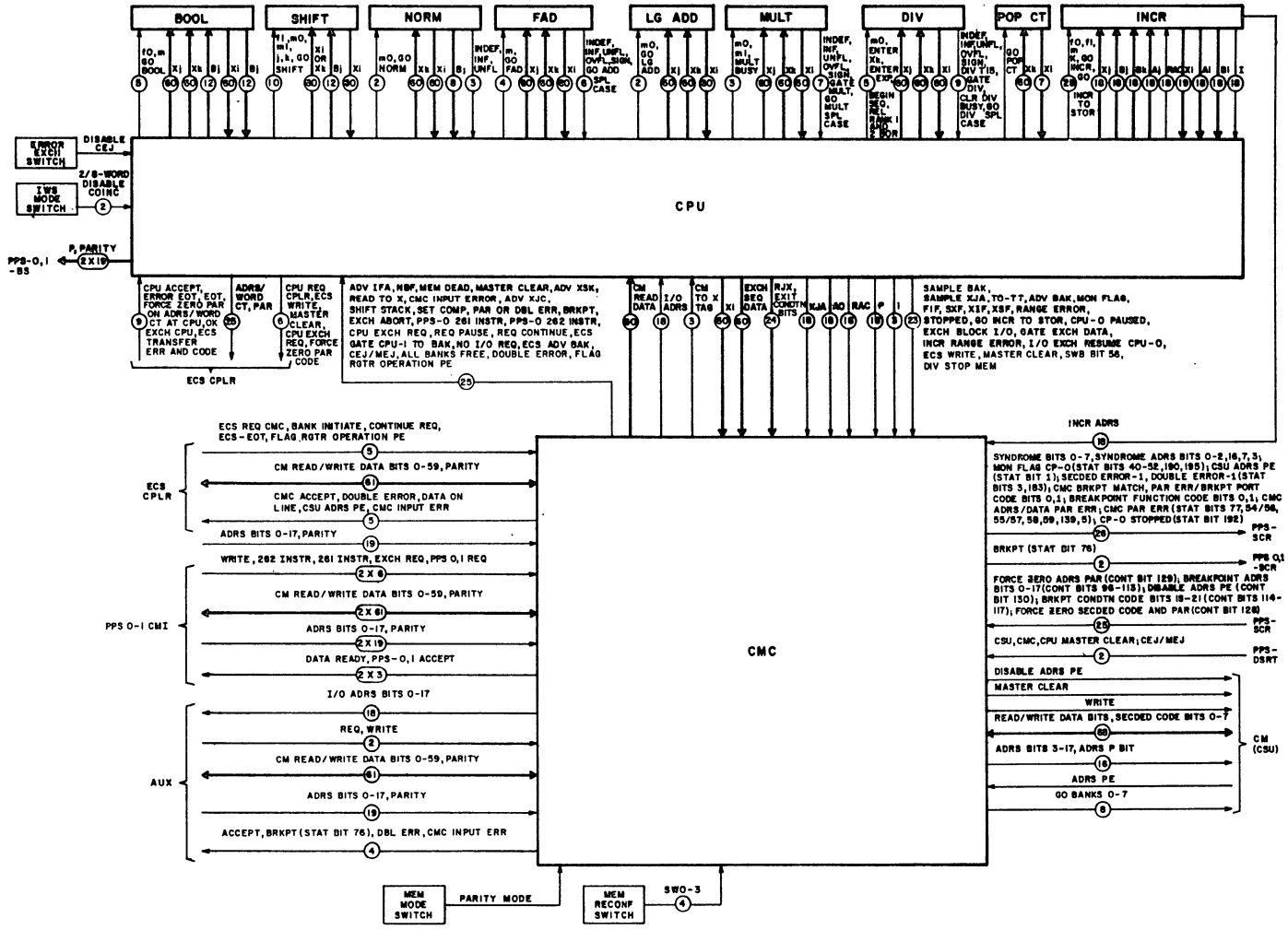


Figure 5-7. Interface Diagram, Central Processor, Models 740, 750, 760

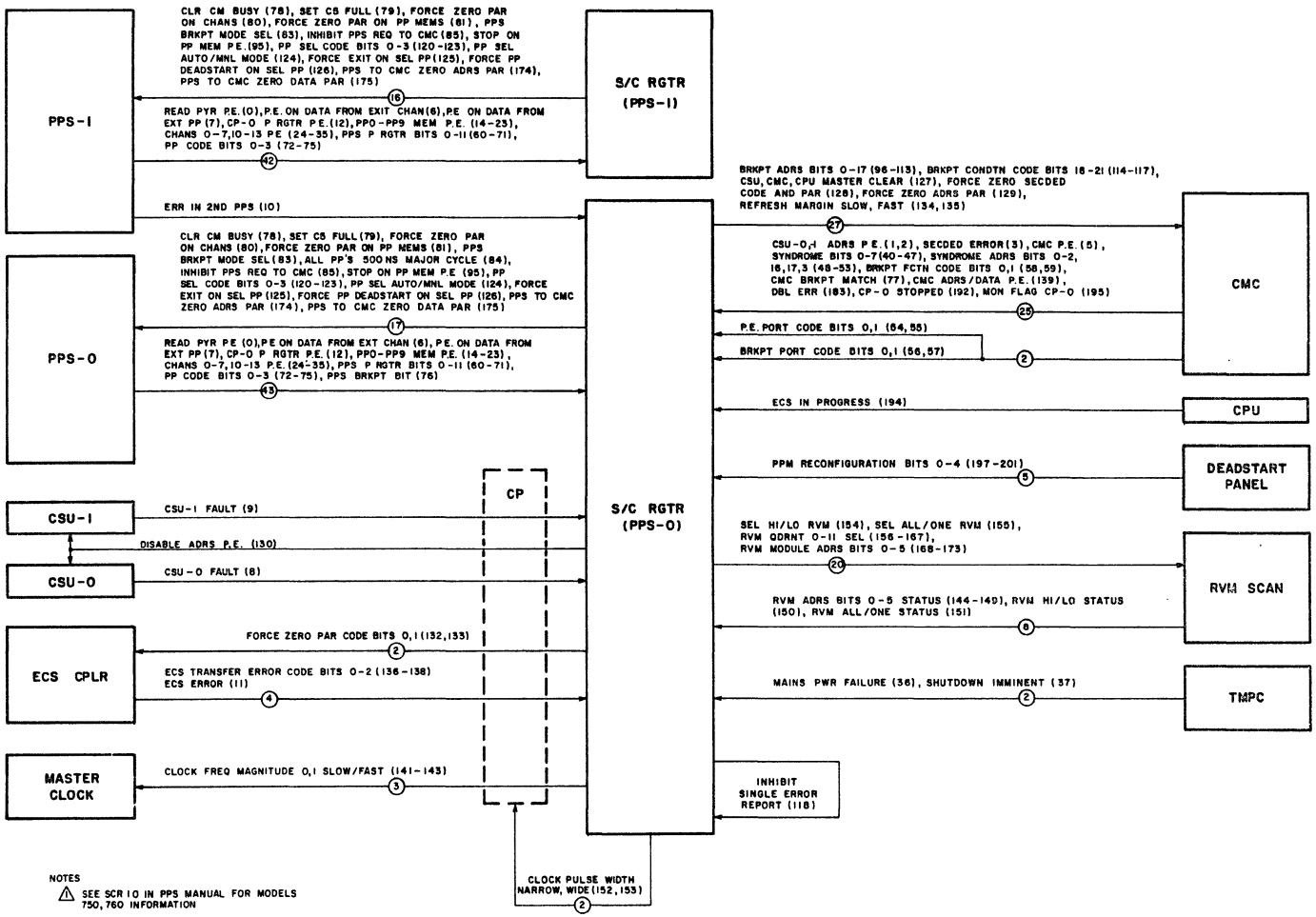
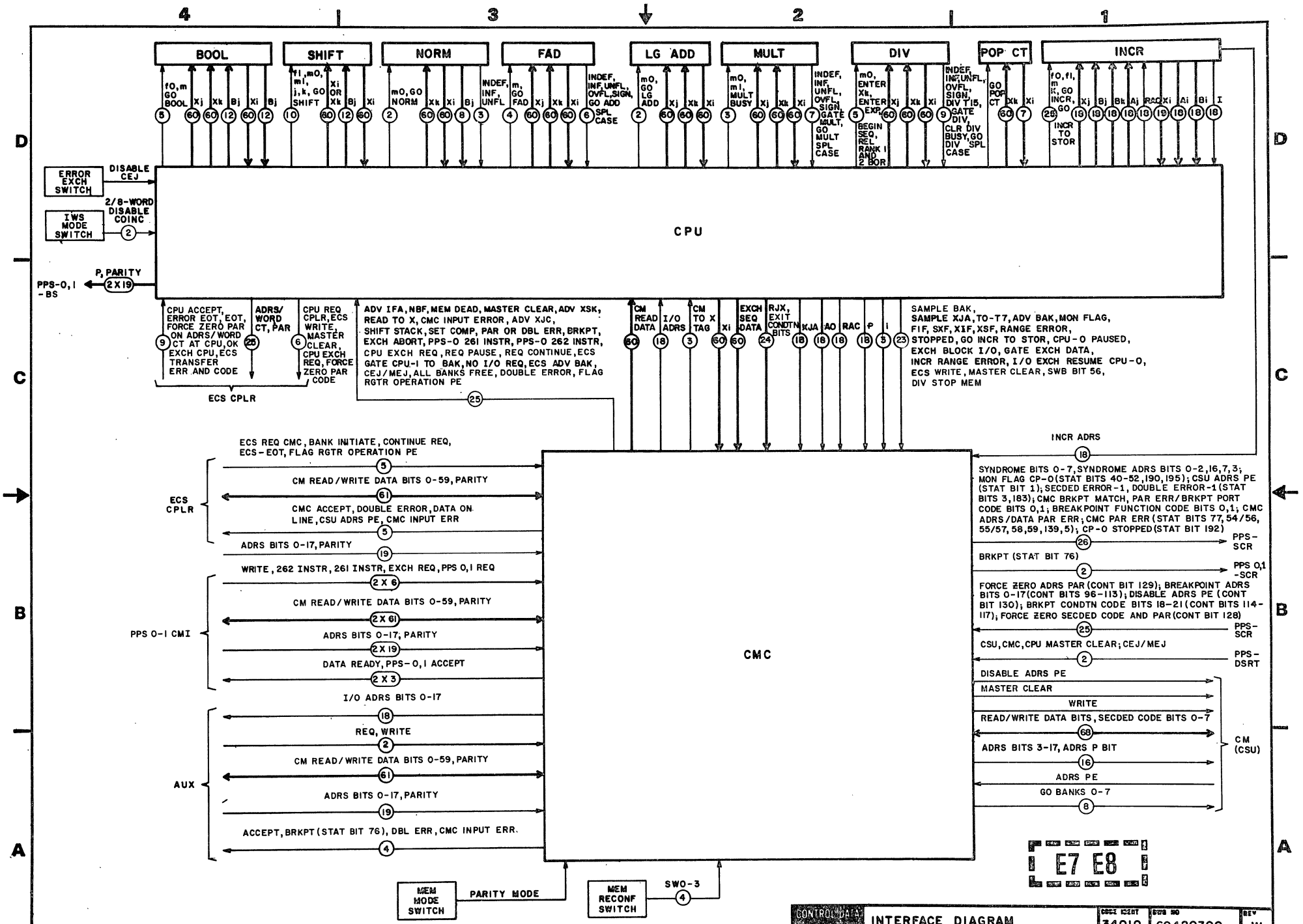
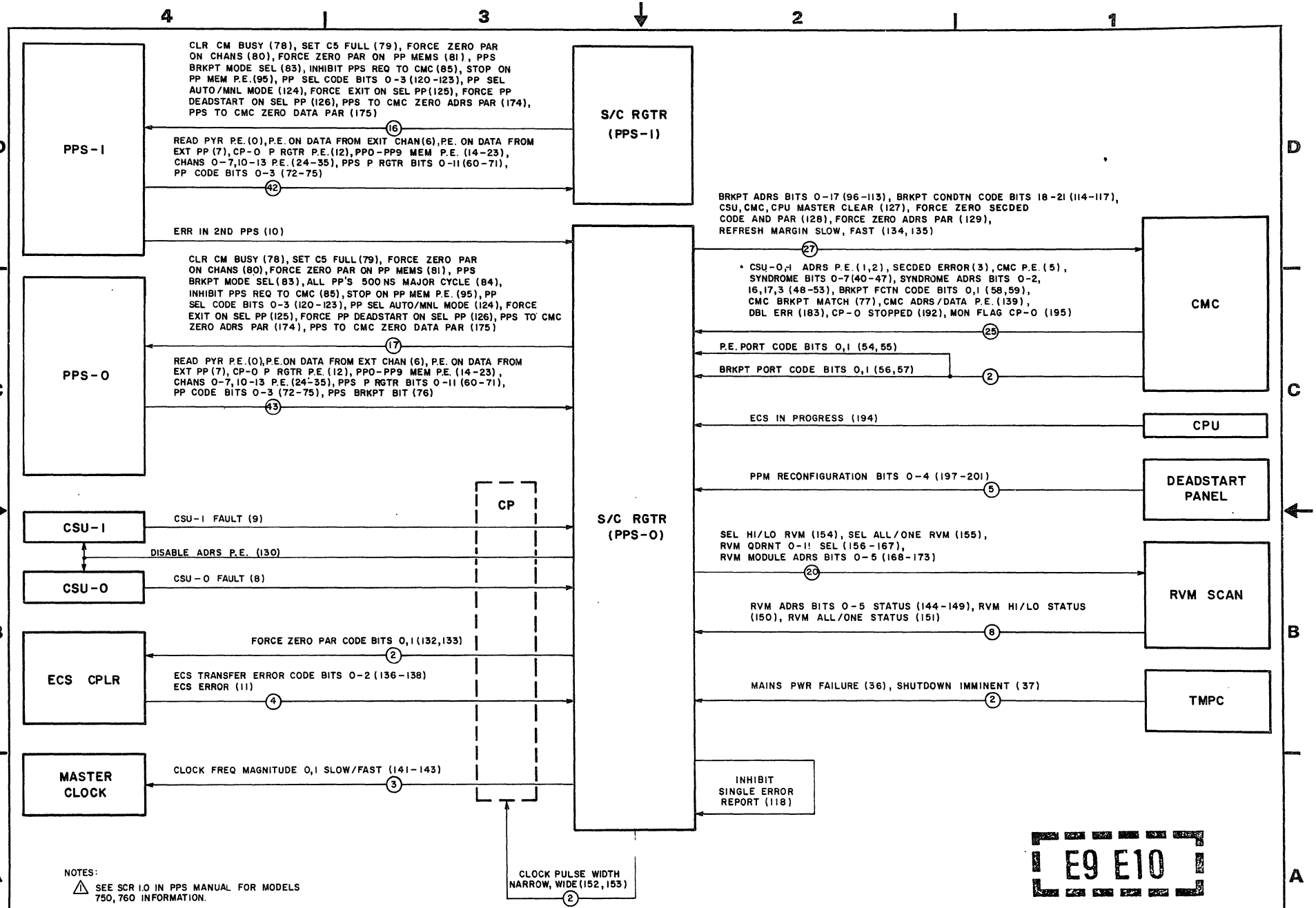


Figure 5-8. Interface Diagram, Status/Control Register, Model 175

PART 2

BOOLEAN UNIT


BOOL
SHIFT
NORM
FAD
LG ADD
MULT
DIV
POP CT
INCR
CPU
ERROR EXCH SWITCH
DISABLE CEJ
2/8-WORD DISABLE COINC
IWS MODE SWITCH
P, PARITY
PPS-0,1 -BS
CPU ACCEPT, ERROR EOT, EOT, FORCE ZERO PAR ON ADRS/WORD CT AT CPU, OK EXCH CPU, ECS TRANSFER ERR AND CODE
ADRS/ WORD CT, PAR
CPU REQ CPLR, ECS WRITE, MASTER CLEAR, CPU EXCH REQ, FORCE ZERO PAR CODE
ECS CPLR
ADV IFA, NBF, MEM DEAD, MASTER CLEAR, ADV XSK, READ TO X, CMC INPUT ERROR, ADV XJC, SHIFT STACK, SET COMP, PAR OR DBL ERR, BRKPT, EXCH ABORT, PPS-0 261 INSTR, PPS-0 262 INSTR, CPU EXCH REQ, REQ PAUSE, REQ CONTINUE, ECS GATE CPU-1 TO BAK, NO I/O REQ, ECS ADV BAK, CEJ/MEJ, ALL BANKS FREE, DOUBLE ERROR, FLAG RGTR OPERATION PE
CM READ DATA
I/O ADRS
CM TO X TAG
EXCH SEQ DATA
RJX, EXIT CONDTN BITS
SAMPLE BAK, SAMPLE XJA, TO-T7, ADV BAK, MON FLAG, FIF, SXF, XIF, XSF, RANGE ERROR, STOPPED, GO INCR TO STOR, CPU-0 PAUSED, EXCH BLOCK I/O, GATE EXCH DATA, INCR RANGE ERROR, I/O EXCH RESUME CPU-0, ECS WRITE, MASTER CLEAR, SWB BIT 56, DIV STOP MEM
ECS REQ CMC, BANK INITIATE, CONTINUE REQ, ECS-EOT, FLAG RGTR OPERATION PE
ECS CPLR
CM READ/WRITE DATA BITS 0-59, PARITY
CMC ACCEPT, DOUBLE ERROR, DATA ON LINE, CSU ADRS PE, CMC INPUT ERR
ADRS BITS 0-17, PARITY
PPS 0-1 CMI
WRITE, 262 INSTR, 261 INSTR, EXCH REQ, PPS 0,1 REQ
CM READ/WRITE DATA BITS 0-59, PARITY
ADRS BITS 0-17, PARITY
DATA READY, PPS-0,1 ACCEPT
I/O ADRS BITS 0-17
AUX
REQ, WRITE
CM READ/WRITE DATA BITS 0-59, PARITY
ADRS BITS 0-17, PARITY
ACCEPT, BRKPT (STAT BIT 76), DBL ERR, CMC INPUT ERR.
CMC
INCR ADRS
SYNDROME BITS 0-7, SYNDROME ADRS BITS 0-2, 16, 7, 3; MON FLAG CP-0 (STAT BITS 40-52, 190, 195); CSU ADRS PE (STAT BIT 1); SECDED ERROR-1, DOUBLE ERROR-1 (STAT BITS 3, 183); CMC BRKPT MATCH, PAR ERR/BRKPT PORT CODE BITS 0,1; BREAKPOINT FUNCTION CODE BITS 0,1; CMC ADRS/DATA PAR ERR; CMC PAR ERR (STAT BITS 77, 54/56, 55/57, 58, 59, 139, 5); CP-0 STOPPED (STAT BIT 192)
PPS-SCR
BRKPT (STAT BIT 76)
PPS 0,1 -SCR
FORCE ZERO ADRS PAR (CONT BIT 129); BREAKPOINT ADRS BITS 0-17 (CONT BITS 96-113); DISABLE ADRS PE (CONT BIT 130); BRKPT CONDTN CODE BITS 18-21 (CONT BITS 114-117); FORCE ZERO SECDED CODE AND PAR (CONT BIT 128)
PPS-SCR
CSU, CMC, CPU MASTER CLEAR; CEJ/MEJ
PPS-DSRT
DISABLE ADRS PE
MASTER CLEAR
WRITE
READ/WRITE DATA BITS, SECDED CODE BITS 0-7
ADRS BITS 3-17, ADRS P BIT
ADRS PE
GO BANKS 0-7
CM (CSU)
MEM MODE SWITCH
PARITY MODE
MEM RECONF SWITCH
SW0-3
E7 E8
CONTROL DATA
DEVELOPMENT DIVISION
INTERFACE DIAGRAM
CENTRAL PROCESSOR
MODELS 740, 750, 760
CODE IDENT 34010
DOC NO 60420300
REV W
SHEET C
PAGE 5-1-15


PPS-1
PPS-0
CSU-1
CSU-0
ECS CPLR
MASTER CLOCK
CP
S/C RGTR (PPS-1)
S/C RGTR (PPS-0)
CMC
CPU
DEADSTART PANEL
RVM SCAN
TMPC
INHIBIT SINGLE ERROR REPORT (118)
CLR CM BUSY (78), SET C5 FULL (79), FORCE ZERO PAR ON CHANS (80), FORCE ZERO PAR ON PP MEMS (81), PPS BRKPT MODE SEL (83), INHIBIT PPS REQ TO CMC (85), STOP ON PP MEM P.E.(95), PP SEL CODE BITS 0-3 (120-123), PP SEL AUTO/MNL MODE (124), FORCE EXIT ON SEL PP (125), FORCE PP DEADSTART ON SEL PP (126), PPS TO CMC ZERO ADRS PAR (174), PPS TO CMC ZERO DATA PAR (175)
16
READ PYR P.E.(0), P.E. ON DATA FROM EXIT CHAN(6), P.E. ON DATA FROM EXT PP (7), CP-0 P RGTR P.E.(12), PP0-PP9 MEM P.E. (14-23), CHANS 0-7, 10-13 P.E.(24-35), PPS P RGTR BITS 0-11 (60-71), PP CODE BITS 0-3 (72-75)
42
ERR IN 2ND PPS (10)
CLR CM BUSY (78), SET C5 FULL (79), FORCE ZERO PAR ON CHANS (80), FORCE ZERO PAR ON PP MEMS (81), PPS BRKPT MODE SEL (83), ALL PP'S 500 NS MAJOR CYCLE (84), INHIBIT PPS REQ TO CMC (85), STOP ON PP MEM P.E. (95), PP SEL CODE BITS 0-3 (120-123), PP SEL AUTO/MNL MODE (124), FORCE EXIT ON SEL PP (125), FORCE PP DEADSTART ON SEL PP (126), PPS TO CMC ZERO ADRS PAR (174), PPS TO CMC ZERO DATA PAR (175)
17
READ PYR P.E.(0), P.E. ON DATA FROM EXT CHAN (6), P.E. ON DATA FROM EXT PP (7), CP-0 P RGTR P.E. (12), PP0-PP9 MEM P.E. (14-23), CHANS 0-7, 10-13 P.E.(24-35), PPS P RGTR BITS 0-11 (60-71), PP CODE BITS 0-3 (72-75), PPS BRKPT BIT (76)
43
CSU-1 FAULT (9)
DISABLE ADRS P.E. (130)
CSU-0 FAULT (8)
FORCE ZERO PAR CODE BITS 0,1 (132,133)
2
ECS TRANSFER ERROR CODE BITS 0-2 (136-138)
ECS ERROR (11)
4
CLOCK FREQ MAGNITUDE 0,1 SLOW/FAST (141-143)
3
CLOCK PULSE WIDTH NARROW, WIDE (152,153)
2
BRKPT ADRS BITS 0-17 (96-113), BRKPT CONDTN CODE BITS 18-21 (114-117), CSU, CMC, CPU MASTER CLEAR (127), FORCE ZERO SECDED CODE AND PAR (128), FORCE ZERO ADRS PAR (129), REFRESH MARGIN SLOW, FAST (134, 135)
27
* CSU-0,1 ADRS P.E. (1,2), SECDED ERROR (3), CMC P.E. (5), SYNDROME BITS 0-7 (40-47), SYNDROME ADRS BITS 0-2, 16, 17, 3 (48-53), BRKPT FCTN CODE BITS 0,1 (58,59), CMC BRKPT MATCH (77), CMC ADRS/DATA P.E. (139), DBL ERR (183), CP-0 STOPPED (192), MON FLAG CP-0 (195)
25
P.E. PORT CODE BITS 0,1 (54,55)
BRKPT PORT CODE BITS 0,1 (56,57)
2
ECS IN PROGRESS (194)
PPM RECONFIGURATION BITS 0-4 (197-201)
5
SEL HI/LO RVM (154), SEL ALL/ONE RVM (155), RVM QDRNT 0-1! SEL (156-167), RVM MODULE ADRS BITS 0-5 (168-173)
20
RVM ADRS BITS 0-5 STATUS (144-149), RVM HI/LO STATUS (150), RVM ALL/ONE STATUS (151)
8
MAINS PWR FAILURE (36), SHUTDOWN IMMINENT (37)
2
E9 E10
NOTES:
⚠ SEE SCR 1.0 IN PPS MANUAL FOR MODELS 750, 760 INFORMATION.
CONTROL DATA
DEVELOPMENT DIVISION
INTERFACE DIAGRAM
STATUS/CONTROL REGISTER
MODEL 175 ⚠
CODE IDENT 34010
DWG NO 60420300
REV S
SIZE C
SHEET
PAGE 5-1-17

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 Xj and Xk to Xi
12	Logical sum of Xj and Xk to Xi
13	Logical difference of Xj and Xk to Xi
15	Logical product of Xj and $\overline{Xk}$ to Xi
16	Logical sum of Xj and $\overline{Xk}$ to Xi
17	Logical difference of Xj and $\overline{Xk}$ to Xi

The instructions providing transmissive operations are:

10	Transmit Xj to Xi
14	Transmit $\overline{Xk}$ to Xi
26	Unpack Xk to Bj and Xi
27	Pack Xk and Bj to Xi

INPUT REGISTERS

Three data input registers exist for the Bj, Xj, and Xk operands. These registers are cleared and entered with new data each clock period. The contents of the Bj, Xj, and Xk 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 • Xk bit 59
Gate Xk upper:	12 + 13 + 14 + 16 + 17
Comp Xk:	14 + 15 + 17 + 17
Gate Xj:	10 + 12 + 13 + 16 + 17
Gate LP:	11 + 12 + 15 + 16
Gate B:	27
Comp B:	27 • Xk bit 59
Gate Xk 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 Xj and Xk registers (or its complement) and places the product in the Xi register. The operation is controlled by the gate LP signal. Bits in Xi are set to one when the corresponding bits in Xj and Xk (or its complement) are one as in the following examples.

Xj = 0101	Xj = 0101
Xk = 1100	Xk = 0011
Xi = 0100	Xi = 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 = 1100	Xk = 0011
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 = 1100	Xk = 0011
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.

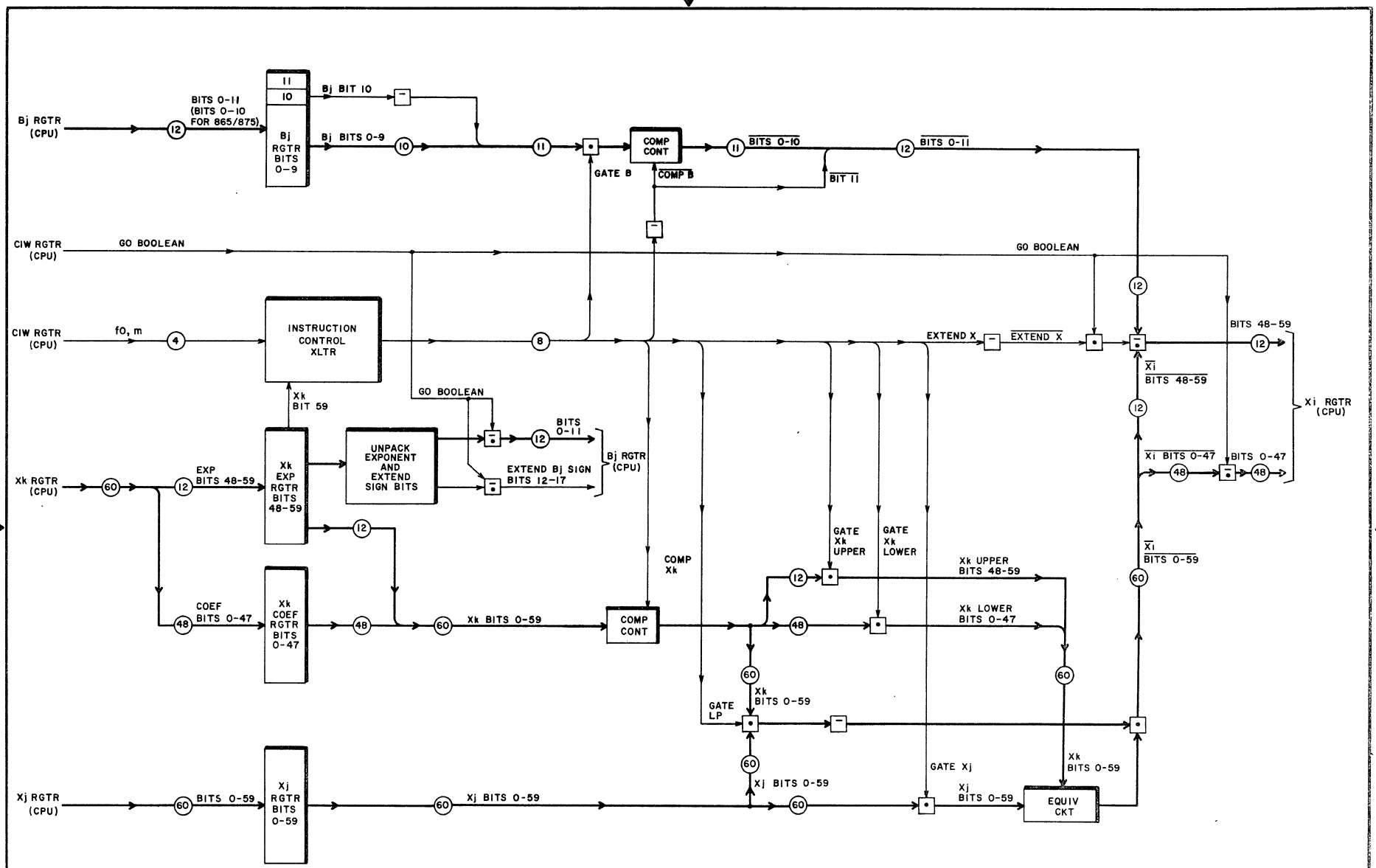
UNPACK

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

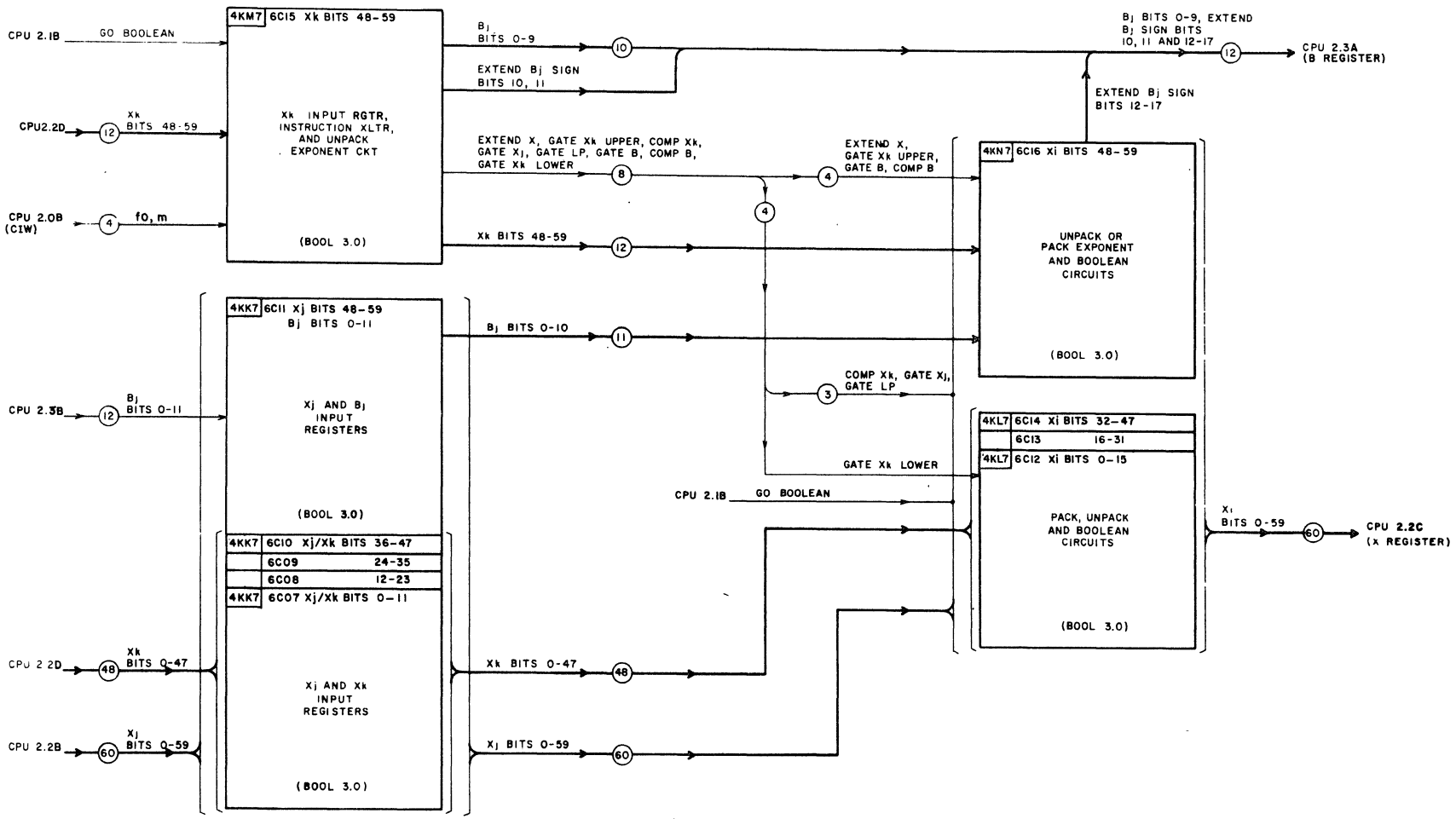
The boolean unit packs a floating-point number in Xi. The coefficient of the number is obtained from Xk and the exponent from Bj. The boolean unit adds bias to the exponent before merging it with the Xk operand. If Xk is positive, the packed exponent occupying positions 48 through 58 of Xi is obtained 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 Xj TO Xi
11 LOGICAL PRODUCT Xj AND Xk TO Xi
12 LOGICAL SUM Xj AND Xk TO Xi
13 LOGICAL DIFFERENCE Xj AND Xk TO Xi
14 TRANSMIT $\overline{Xk}$ TO Xi
15 LOGICAL PRODUCT Xj AND $\overline{Xk}$ TO Xi
16 LOGICAL SUM Xj AND $\overline{Xk}$ TO Xi
17 LOGICAL DIFFERENCE Xj AND $\overline{Xk}$ TO Xi
26 UNPACK Xk TO Bj TO Xi
27 PACK Xk AND Bj TO Xi

CONTROL DATA DEVELOPMENT DIVISION	PRIMARY BLOCK DIAGRAM BOOLEAN UNIT	CODE IDENT 34010	DWG NO 60420300	REV Y
		C	SHEET BOOL 1.0	PAGE 5-2-1



B11 B12

CONTROL DATA DEVELOPMENT DIVISION

SECONDARY BLOCK DIAGRAM
BOOLEAN UNIT

CODE IDENT	DWG NO	REV
34010	60420300	K
C	SHEET BOOL 2.0	PAGE 5-2-3

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.

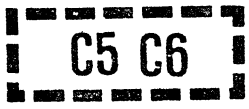
For example, if Xj equals 0101 and Xk equals 1100, the following logical operations take place.

1. The first AND circuit forms the logical product of Xj and Xk.

   Xj = 0101
   Xk = 1100
   LP = 0100

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:

   Xj = 0101
   Xk = 1100
   EQ = 0110



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 ($\overline{Bj\ sign}$) is determined by the logical difference of Xk bits 58 and 59. $\overline{Bj\ sign}$ 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, $\overline{extend\ X}$ 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.

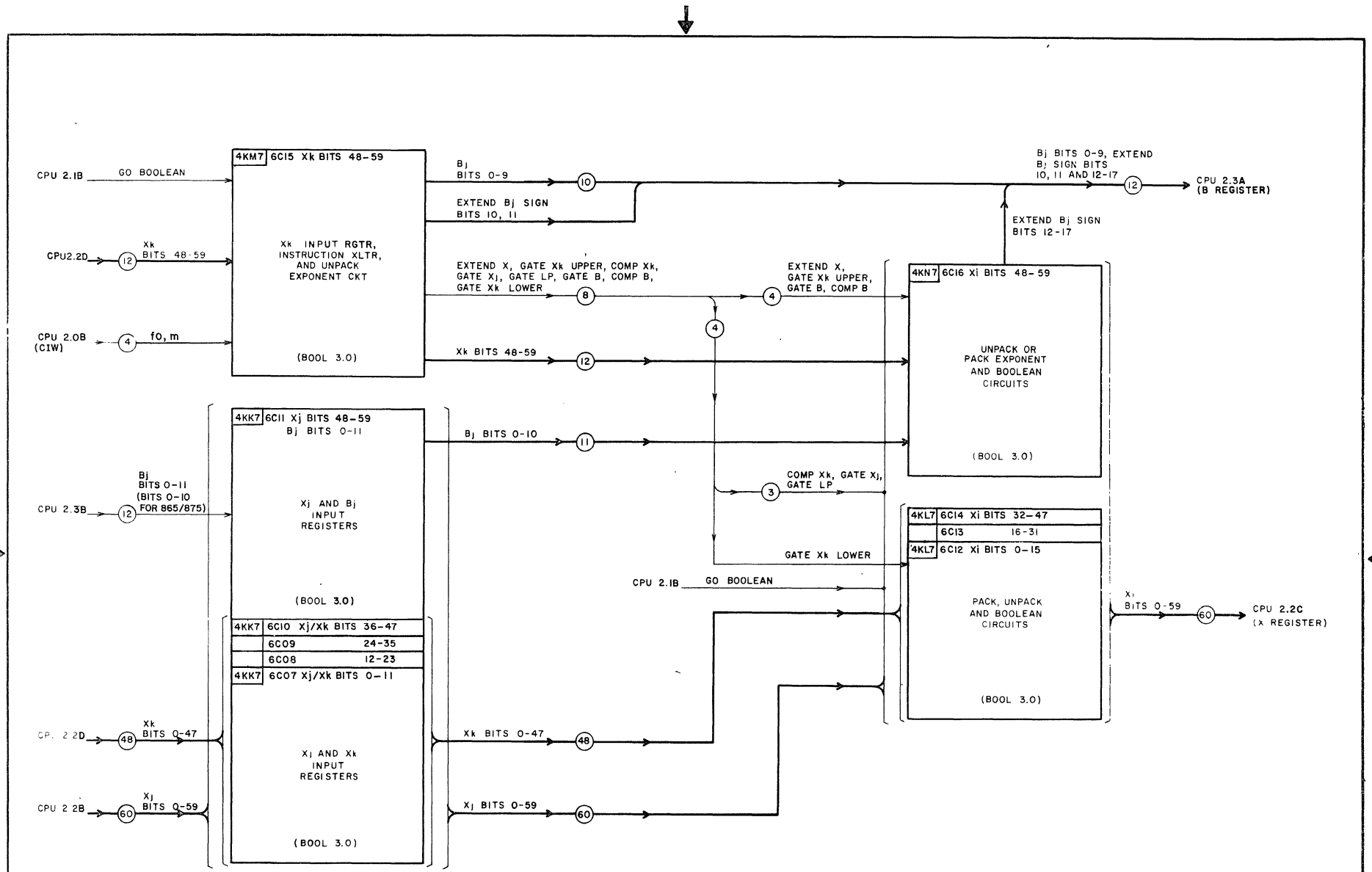
BOOL 3.0 TEST POINTS

Module	Location	Test Point	Description
4KK7	6C07-11	01	Xj rgtr bit N
		04	N+1
		03	N+2
		06	N+3
		11	N+4
		14	N+5
		12	N+6
		16	N+7
		21	N+8
		24	N+9
		23	N+10
		26	N+11
		46	Xk or Bj rgtr bit N
		43	N+1
		45	N+2
		41	N+3
		56	N+4
		53	N+5
		54	N+6
		51	N+7
		66	N+8
		63	N+9
		65	N+10
		61	N+11
		33, 34	25-nanosecond clock
4KL7*	6C12-14	11, 12	Go boolean
		21, 22	Comp Xk
		45, 46	Gate Xj
		31, 32	Gate Xk lower
		55, 56	Gate LP
4KM7	6C15	01	Xk rgtr bit 48
		02	49
		05	50
		06	51
		11	52
		12	53
		15	54
		16	55
		21	56
		22	57
		25	58
		26, 43, 44	59
		62	f0 holding rgtr
		63	m0
		65	m1
		61	m2
		52	Gate Xk lower
		53	Gate Xk upper
		56	Extend X
		55	Comp B
		51	Comp Xk
		54	Gate B
		74	25-nanosecond clock

Module	Location	Test Point	Description
4KN7*	6C16	71, 72	Gate B
		74, 75	Comp B
		15, 16	Comp Xk
		11, 12	Gate LP
		05, 06	Gate Xk upper
		01, 02	Gate Xj
		53	Extend Bj sign bits 12 through 17

Module	Location	Test Point	Description

C3 C4



CONTROL DATA DEVELOPMENT DIVISION	SECONDARY BLOCK DIAGRAM BOOLEAN UNIT	CODE IDENT 34010	DWG NO 60420300	REV Y
		C	SHEET BOOL 2.0	PAGE 5-2-3

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.

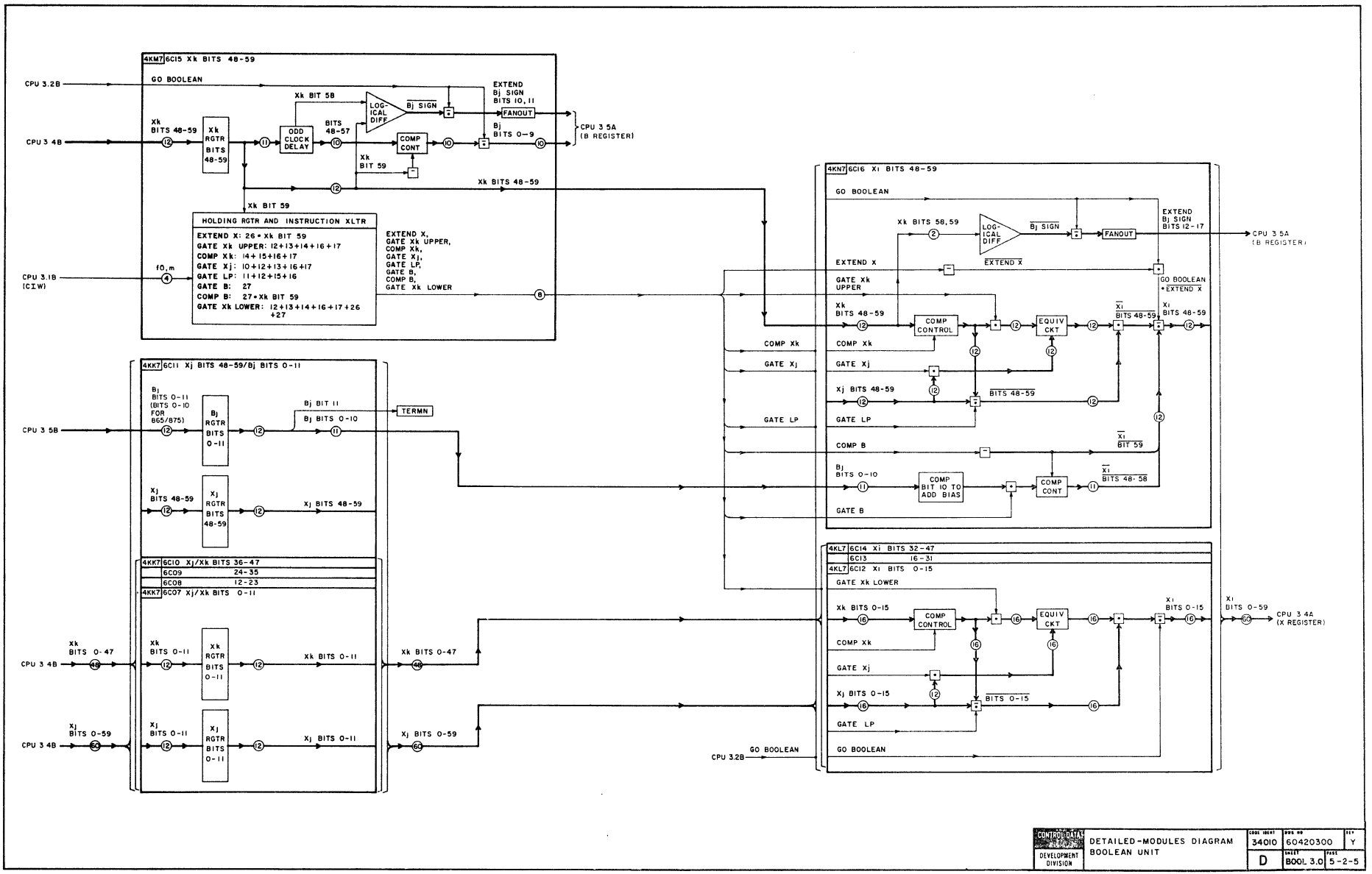
For example, if Xj equals 0101 and Xk equals 1100, the following logical operations take place.

1. The first AND circuit forms the logical product of Xj and Xk.

   Xj = 0101
   Xk = 1100
   LP = 0100

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:

   Xj = 0101
   Xk = 1100
   EQ = 0110



CONTROL DATA DEVELOPMENT DIVISION	DETAILED-MODULES DIAGRAM BOOLEAN UNIT	CODE IDENT 34010	DOC NO 60420300	REV Y
		D	SHEET BOOL 3.0	PAGE 5-2-5

PART 3

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 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 (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.

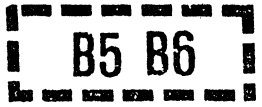
The shift unit does not actually shift data in a left circular mode when Bj is negative; it simulates the shift. A three-bit left circular shift correction is performed and the shift is made in a right circular mode. The shift count is complemented because the Bj register contains a negative number.

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 thee 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.

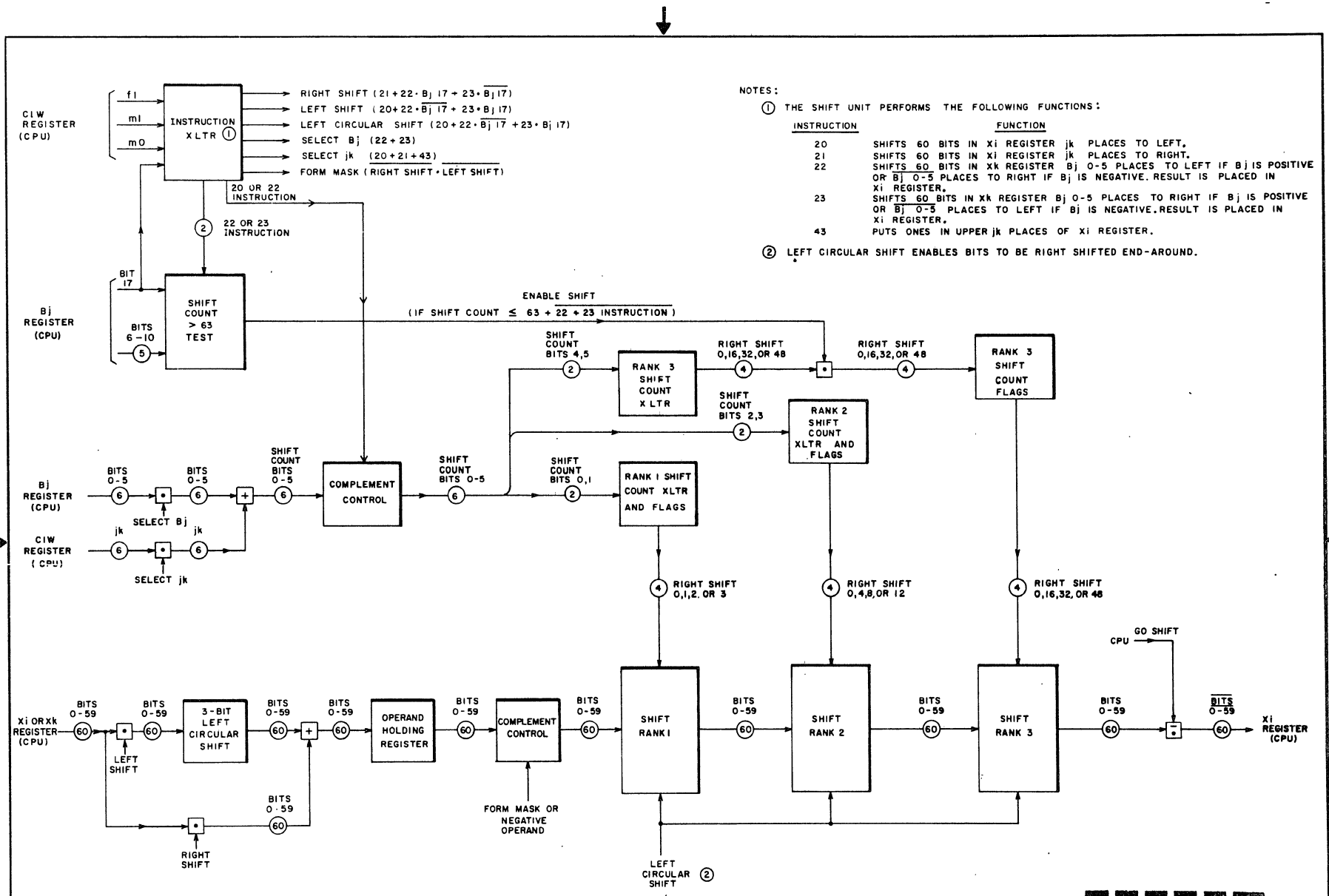
SHIFT RANKS

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:

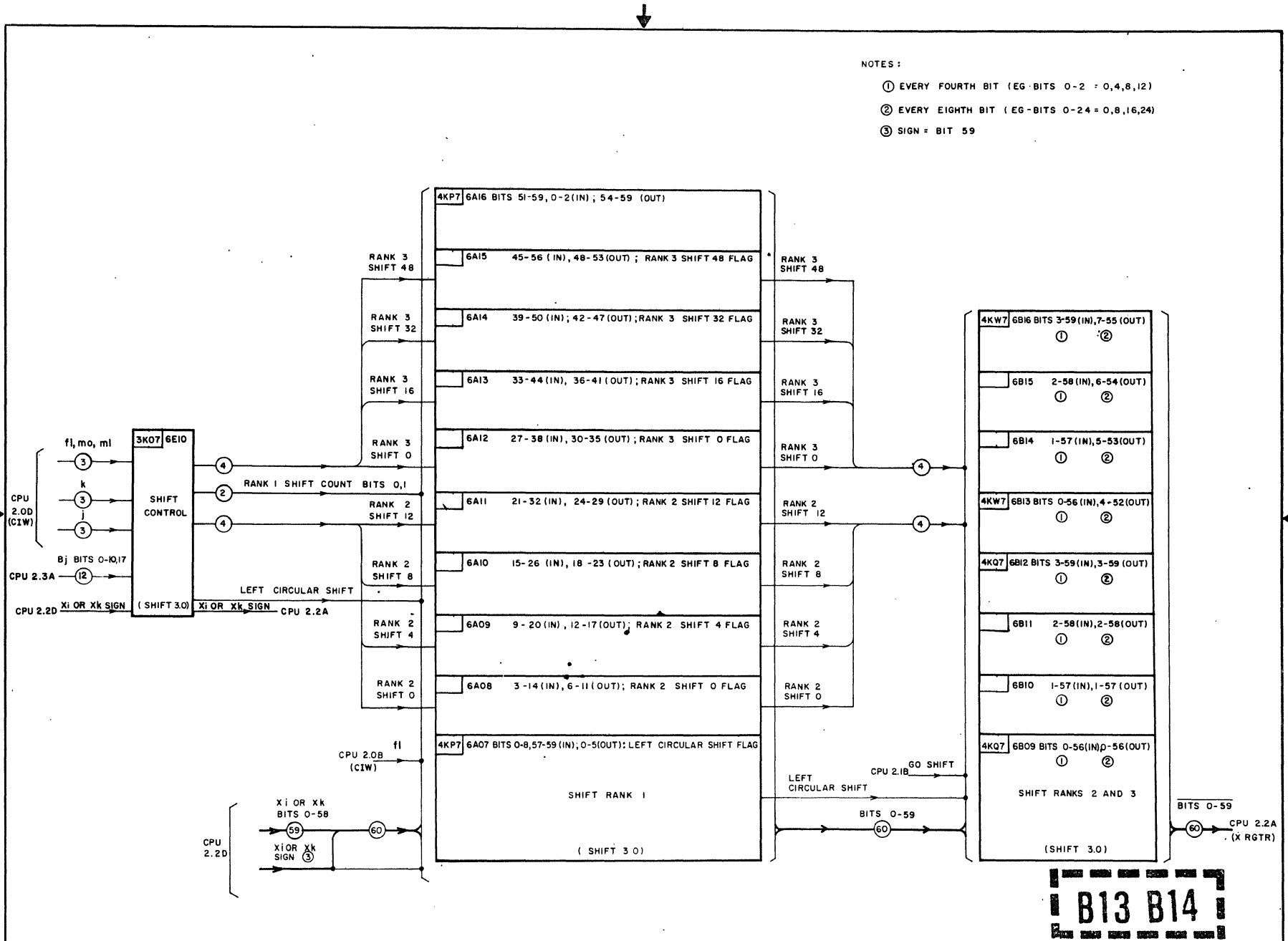
(1) THE SHIFT UNIT PERFORMS THE FOLLOWING FUNCTIONS:

INSTRUCTION	FUNCTION
20	SHIFTS 60 BITS IN Xi REGISTER jk PLACES TO LEFT.
21	SHIFTS 60 BITS IN Xi REGISTER jk PLACES TO RIGHT.
22	SHIFTS 60 BITS IN Xk REGISTER Bj 0-5 PLACES TO LEFT IF Bj IS POSITIVE OR $\overline{Bj\ 0-5}$ PLACES TO RIGHT IF Bj IS NEGATIVE. RESULT IS PLACED IN Xi REGISTER.
23	SHIFTS 60 BITS IN Xk REGISTER Bj 0-5 PLACES TO RIGHT IF Bj IS POSITIVE OR $\overline{Bj\ 0-5}$ PLACES TO LEFT IF Bj IS NEGATIVE. RESULT IS PLACED IN Xi REGISTER.
43	PUTS ONES IN UPPER jk PLACES OF Xi REGISTER.

(2) LEFT CIRCULAR SHIFT ENABLES BITS TO BE RIGHT SHIFTED END-AROUND.

B3 B4

CONTROL DATA DEVELOPMENT DIVISION	PRIMARY BLOCK DIAGRAM SHIFT UNIT	CODE IDENT 34010	DWG NO 60420300	REV K
		C	SHEET SHIFT 1.0	PAGE 5-3-1


NOTES :
① EVERY FOURTH BIT (EG BITS 0-2 = 0,4,8,12)
② EVERY EIGHTH BIT (EG - BITS 0-24 = 0,8,16,24)
③ SIGN = BIT 59
4KP7 6A16 BITS 51-59, 0-2(IN) ; 54-59 (OUT)
6A15 45-56 (IN), 48-53(OUT) ; RANK 3 SHIFT 48 FLAG
6A14 39-50 (IN) ; 42-47(OUT) ; RANK 3 SHIFT 32 FLAG
6A13 33-44 (IN), 36-41 (OUT) ; RANK 3 SHIFT 16 FLAG
6A12 27-38 (IN), 30-35 (OUT) ; RANK 3 SHIFT 0 FLAG
6A11 21-32 (IN), 24-29 (OUT) ; RANK 2 SHIFT 12 FLAG
6A10 15-26 (IN), 18-23 (OUT) ; RANK 2 SHIFT 8 FLAG
6A09 9-20 (IN), 12-17 (OUT) ; RANK 2 SHIFT 4 FLAG
6A08 3-14 (IN), 6-11 (OUT) ; RANK 2 SHIFT 0 FLAG
4KP7 6A07 BITS 0-8, 57-59 (IN) ; 0-5(OUT) : LEFT CIRCULAR SHIFT FLAG
SHIFT RANK 1
(SHIFT 3.0)
RANK 3 SHIFT 48
RANK 3 SHIFT 32
RANK 3 SHIFT 16
RANK 3 SHIFT 0
RANK 2 SHIFT 12
RANK 2 SHIFT 8
RANK 2 SHIFT 4
RANK 2 SHIFT 0
fl, mo, ml
k
j
CPU 2.0D (CIW)
3K07 6E10
SHIFT CONTROL
(SHIFT 3.0)
RANK 1 SHIFT COUNT BITS 0,1
Bj BITS 0-10,17
CPU 2.3A
CPU 2.2D
Xi OR Xk SIGN
LEFT CIRCULAR SHIFT
CPU 2.2A
CPU 2.0B (CIW)
fl
Xi OR Xk BITS 0-58
Xi OR Xk SIGN
CPU 2.2D
4KW7 6B16 BITS 3-59 (IN), 7-55 (OUT)
6B15 2-58 (IN), 6-54 (OUT)
6B14 1-57 (IN), 5-53 (OUT)
4KW7 6B13 BITS 0-56 (IN), 4-52 (OUT)
4KQ7 6B12 BITS 3-59 (IN), 3-59 (OUT)
6B11 2-58 (IN), 2-58 (OUT)
6B10 1-57 (IN), 1-57 (OUT)
4KQ7 6B09 BITS 0-56 (IN) 0-56 (OUT)
SHIFT RANKS 2 AND 3
(SHIFT 3.0)
CPU 2.1B GO SHIFT
LEFT CIRCULAR SHIFT
BITS 0-59
BITS 0-59
CPU 2.2A (X RGTR)
B13 B14
CONTROL DATA
DEVELOPMENT DIVISION
SECONDARY BLOCK DIAGRAM
SHIFT UNIT
CODE IDENT 34010
DWG NO 60420300
REV K
SIZE C
SHEET SHIFT 2.0
PAGE 5-3-3

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 resutling 60-bit word into the Xi register. However, 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 slected 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

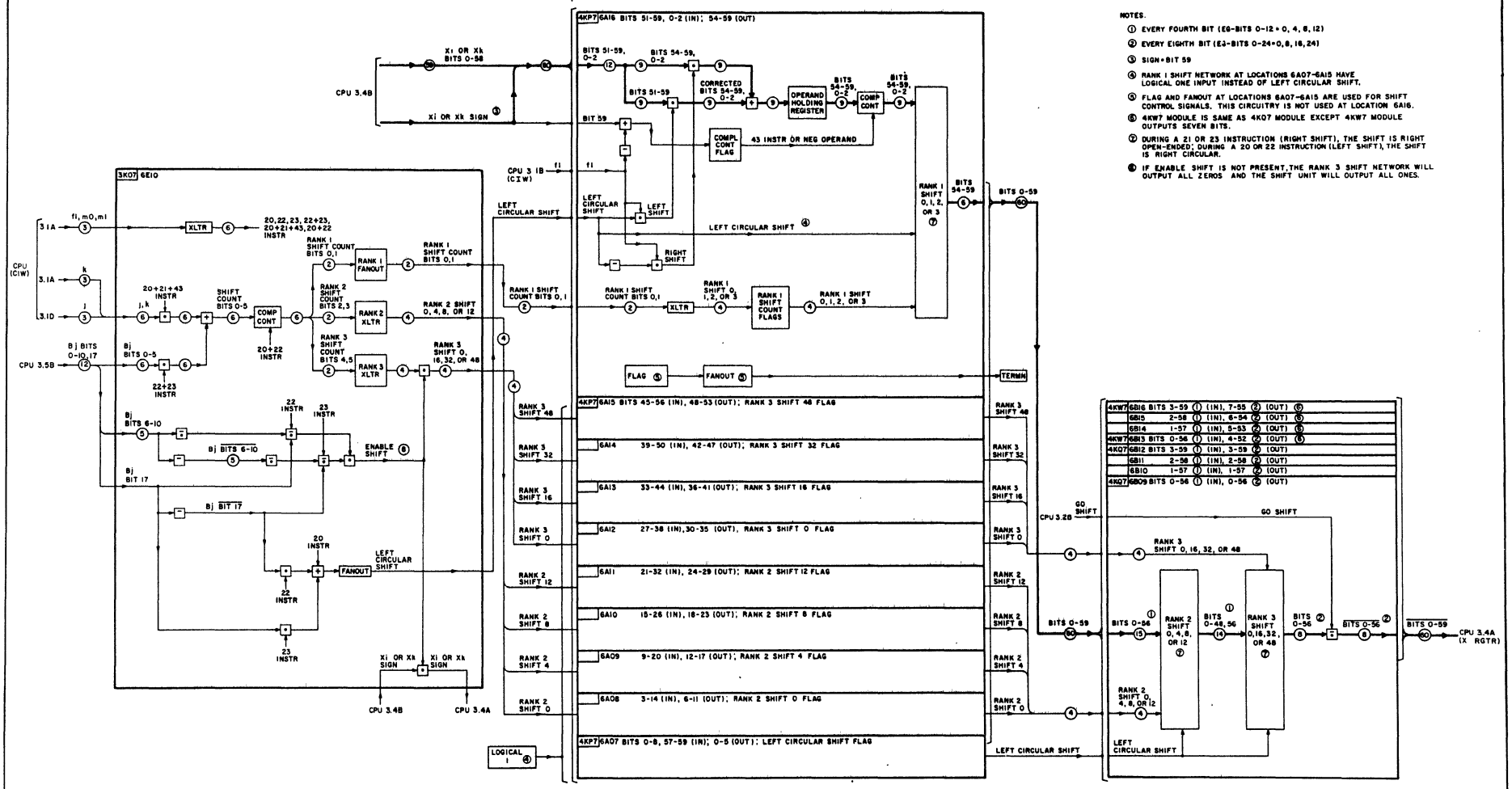
The third rank of the shift unit (4KQ7 and 4KW7 modules) outputs the shifted operand (complemented) to the X registers whenever a go shift signal from the 4LE7 module is present. If this signal is not present, all ones are sent to the X registers. If the original operand was positive, the shifted operand 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 recomplemented.

C9 C10

SHIFT 3.0 TEST POINTS

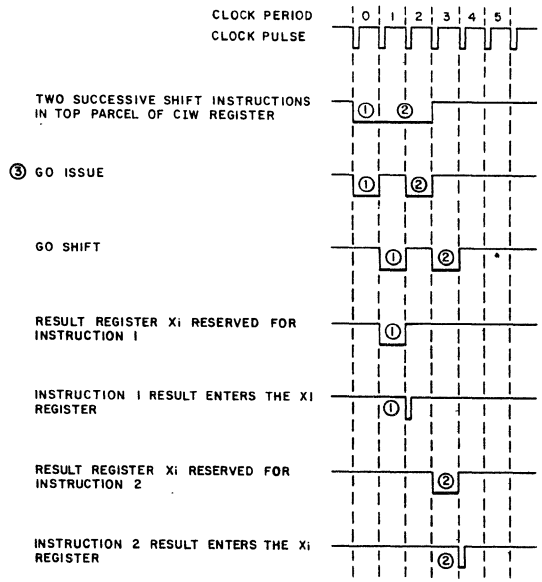
Module	Location	Test Point	Description
3KO7*	6E10	54	20 instruction
↓	↓	53	22 ↓
↓	↓	55	23 ↓
4KP7*	6A07-16	16	Operand holding rgtr bit N
↓	↓	11	Operand holding rgtr bit N+1
↓	↓	24	Operand holding rgtr bit N+2
↓	↓	23	Operand holding rgtr bit N+3
↓	↓	34	Operand holding rgtr bit N+4
↓	↓	33	Operand holding rgtr bit N+5
↓	↓	43	Operand holding rgtr bit N+6
↓	↓	45	Operand holding rgtr bit N+7
↓	↓	54	Operand holding rgtr bit N+8
↓	↓	01, 06	Compl cont flag
↓	↓	74	Rank 1 shift 0 flag
↓	↓	73	↓ 1
↓	↓	76	↓ 2
↓	↓	75	↓ 3
↓	↓	41, 46	Shift cont flag
↓	↓	71, 72	25-nanosecond clock
4KQ7*	6B09-12	71	Left circular shift
↓	↓	73	Go shift
↓	↓	02	Rank 2 shift 0
↓	↓	06	↓ 4
↓	↓	05	↓ 8
↓	↓	01	↓ 12
↓	↓	74	Rank 3 shift 0
↓	↓	75	↓ 16
↓	↓	76	↓ 32
↓	↓	72	↓ 48
4KW7*	6B13-16	71	Left circular shift
↓	↓	73	Go shift
↓	↓	03	Rank 2 shift 0
↓	↓	06	↓ 4
↓	↓	05	↓ 8
↓	↓	01	↓ 12
↓	↓	74	Rank 3 shift 0
↓	↓	75	↓ 16
↓	↓	76	↓ 32
↓	↓	72	↓ 48

C3 C4



C1 C2

CONTROL DATA DEVELOPMENT DIVISION	DETAILED-MODULES DIAGRAM SHIFT UNIT	CODE IDENT 34010	DRG NO 60420300	REV K
		D	SHEET SHIFT 3.0	PAGE 5-3-5



NOTES:

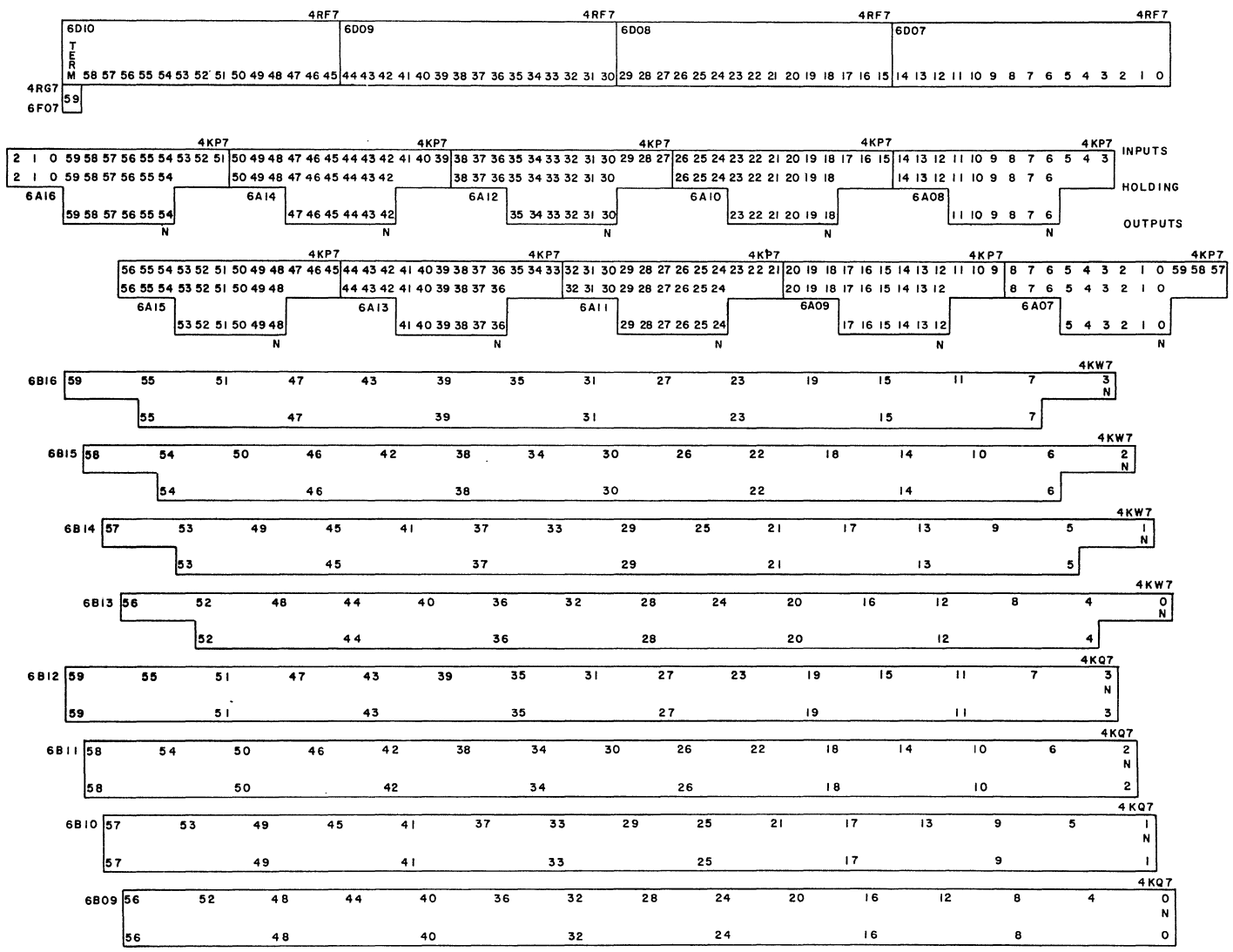
① 431jk (ASSUMING NO ISSUE CONFLICTS EXIST)

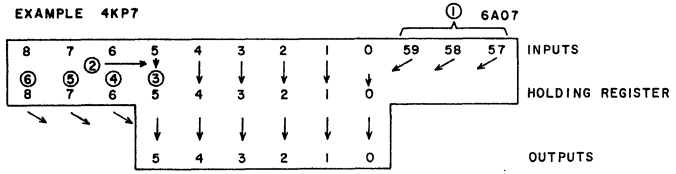
② 23ijl (OPERAND REGISTER CONFLICTS WITH INSTRUCTION 1 DESTINATION REGISTER)

③ ENABLE ISSUE IS DELAYED 4 CLOCK PERIODS (THEREFORE, GO ISSUE IS DELAYED A MINIMUM OF 4 CLOCK PERIODS) IN THE AA120-C. THIS DELAY IS REMOVED BY INSTALLING OPTION AT364-A.

D11 D12

CONTROL DATA DEVELOPMENT DIVISION	TIMING DIAGRAM SHIFT INSTRUCTION	CODE IDENT 34010	DWG NO 60420300	REV R
		C	SHEET	PAGE 5-3-15





NOTES:

① LS3 TO HOLDING REGISTER ON LEFT SHIFT BIT 57 TO 0, BIT 58 TO 1, BIT 59 TO 2, ETC.
② BITS 0-8 TO HOLDING REGISTER BITS 0-8 ON RIGHT SHIFT.
③ RS0 HOLDING RGTR BITS 0-5 TO OUTPUT BITS 0-5.
④ RS1 HOLDING RGTR BITS 1-6 TO OUTPUT BITS 0-5.
⑤ RS2 HOLDING RGTR BITS 2-7 TO OUTPUT BITS 0-5.
⑥ RS3 HOLDING RGTR BITS 3-8 TO OUTPUT BITS 0-5.

D13 D14

CONTROL DATA DEVELOPMENT DIVISION

TROUBLESHOOTING CHART
SHIFT UNIT

CODE IDENT	DWG NO	REV
34010	60420300	K
SIZE C	SHEET	PAGE 5-3-17

PART 4

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.

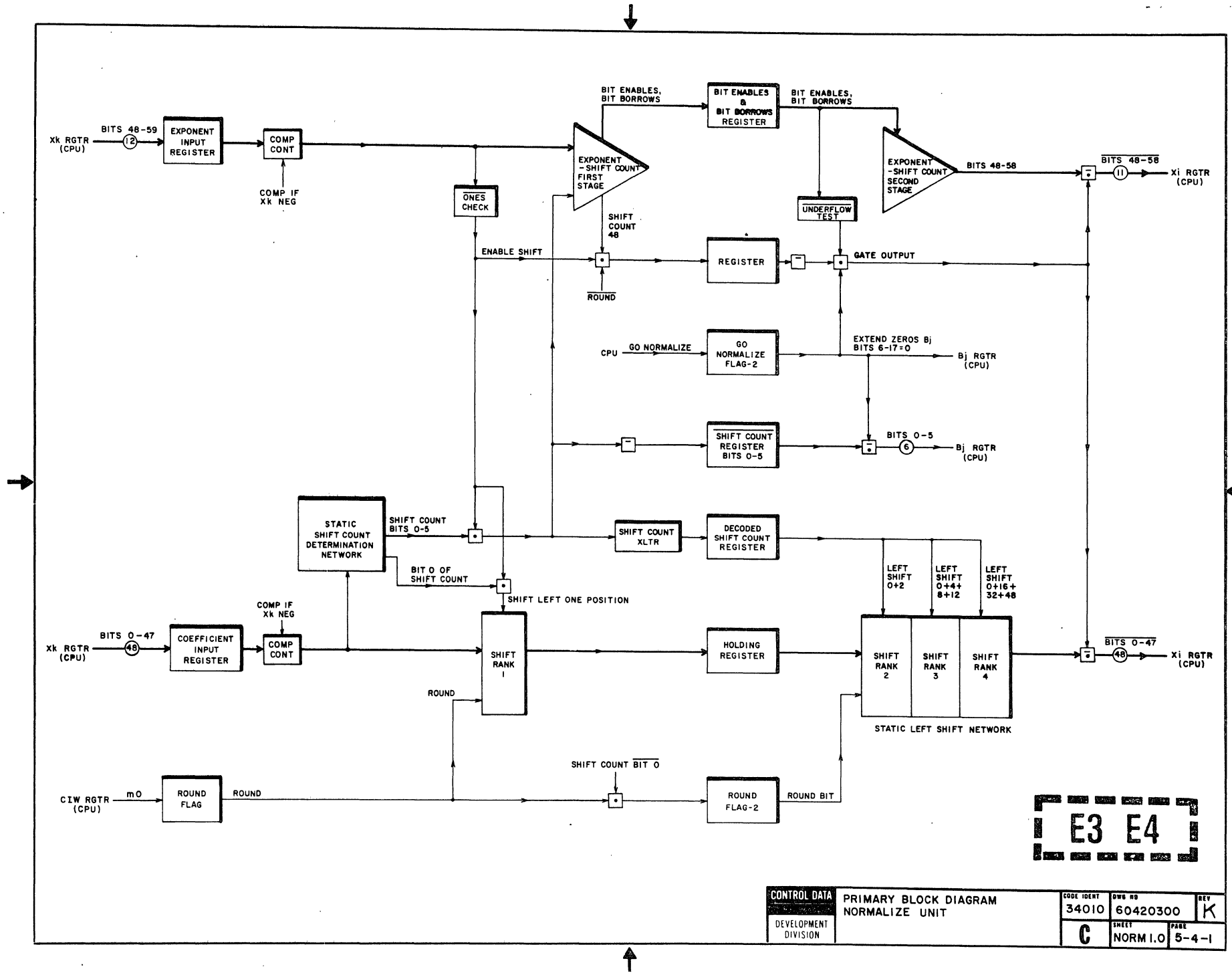
The shift count determination network also sends bit zero of the binary shift count to shift rank 1 where it left-shifts the coefficient one position if bit 0 is a one. During the third clock period, shift ranks 2, 3, and 4 complete the shifting of the coefficient, and the second stage of the exponent 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.

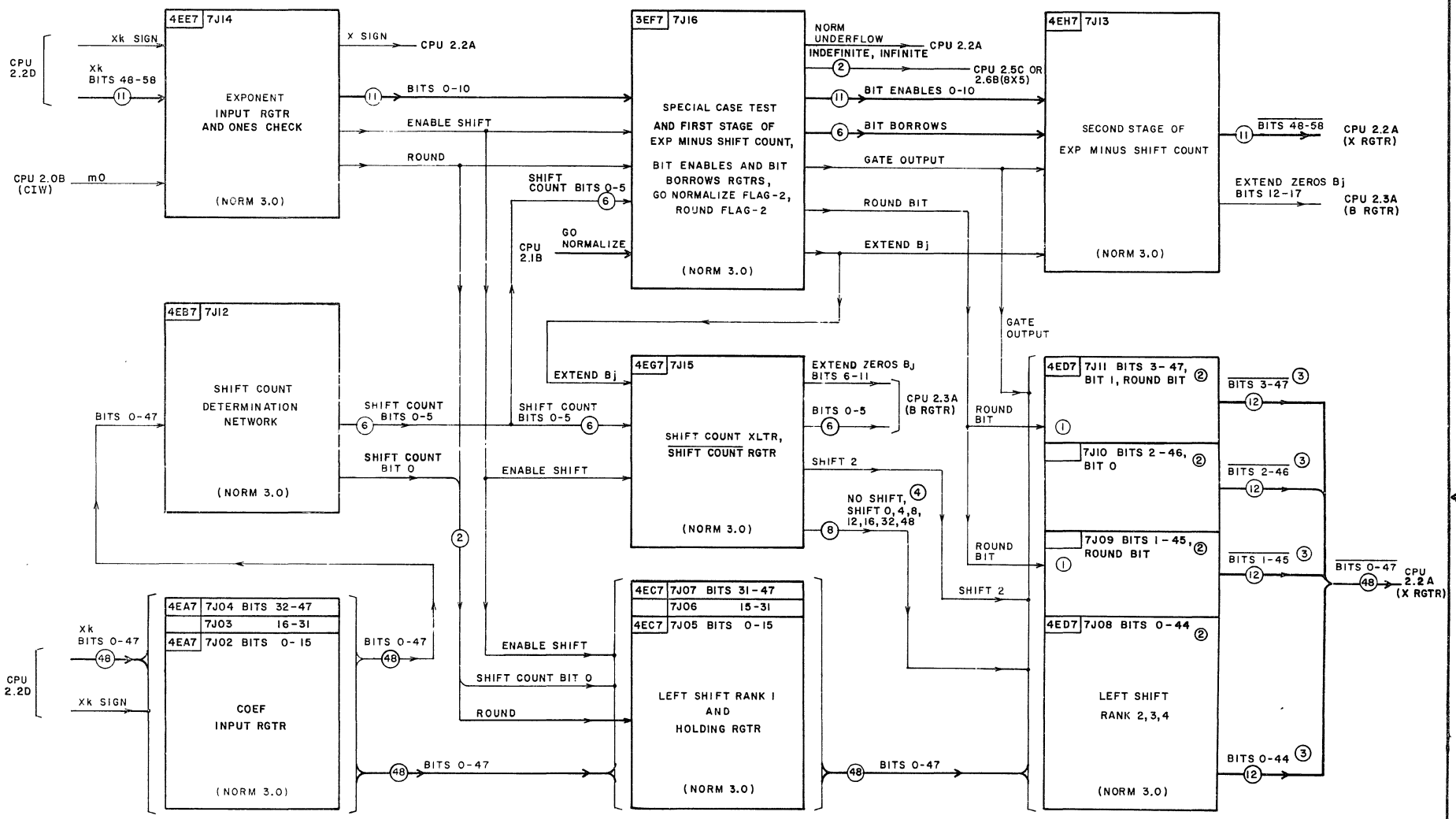
The subtraction of the six-bit shift count from the exponent may result in an underflow of the floating-point exponent range. This situation causes all zeros to enter the destination X register. The shift count delivered to the B register is not affected by this situation.



CONTROL DATA DEVELOPMENT DIVISION

PRIMARY BLOCK DIAGRAM
NORMALIZE UNIT

CODE IDENT	DWG NO	REV
34010	60420300	K
C	SHEET NORM 1.0	PAGE 5-4-1



NOTES:

① THE ROUND BIT IS USED AS ONE BIT POSITION LESS THAN 0.

② EVERY SECOND BIT (EG. 0-12 = 0, 2, 4, 6, 8, 10, 12)

③ EVERY FOURTH BIT (EG. 0-12 = 0, 4, 8, 12).

④ NO SHIFT IS USED IN RANK 4 AND MEANS NO SHIFT 16 + 32 + 48.

CONTROL DATA DEVELOPMENT DIVISION	SECONDARY BLOCK DIAGRAM NORMALIZE	CODE IDENT 34010	DWG NO 60420300	REV Y
		C	SHEET NORM 2.0	PAGE 5-4-3

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 $\overline{\text{round}}$ 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).

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 corfficient 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

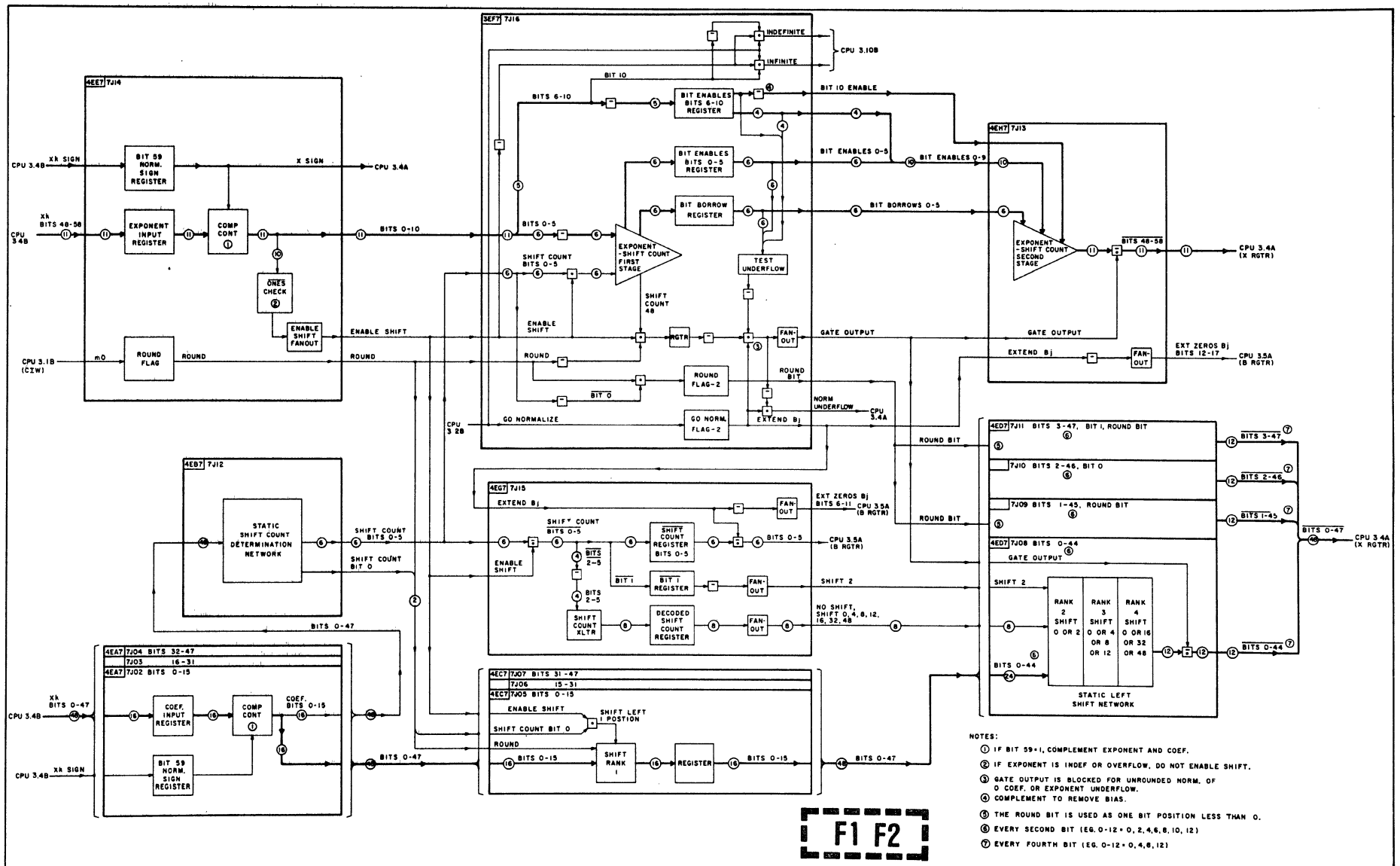
Coefficient bits 0 through 47 enter these modules during the third clock period. Shift ranks 2, 3, and 4 complete the left shifts needed to normalize the coefficient. Gate output gates coefficient bits 0 through 47 to the destination X register. The result is complemented as it is sent to the X registers. If the result is positive, it is recomplemented by the X register input control as it enters the destination X register.

F7 F8

NORM 3.0 TEST POINTS

Module	Location	Test Point	Description
4EA7	7J02-04	41, 42, 45, 46	Bit 59 norm sign rgtr
		01	Coef input rgtr bit N
		03	N+1
		05	N+2
		06	N+3
		11	N+4
		12	N+5
		15	N+6
		16	N+7
		21	N+8
		23	N+9
		25	N+10
		26	N+11
		31	N+12
		32	N+13
		35	N+14
		36	N+15
		52	Coef bit N
		53	N+1
		55	N+2
		56	N+3
		51	N+4
		54	N+5
		64	N+6
		66	N+7
		62	N+8
		61	N+9
		63	N+10
		65	N+11
		71, 72	N+12
		73, 74	N+13
		75	N+14
		76	N+15
		43, 44	25-nanosecond clock
4EB7*	7J12	73	Shift count bit 5
4EC7*	7J05-07	06	Shift rank 1 rgtr bit N
		01	Shift rank 1 rgtr bit N+1
		16	Shift rank 1 rgtr bit N+2
		11	Shift rank 1 rgtr bit N+3
		26	Shift rank 1 rgtr bit N+4
		21	Shift rank 1 rgtr bit N+5
		36	Shift rank 1 rgtr bit N+6
		31	Shift rank 1 rgtr bit N+7
4ED7*	7J08-11	44	Shift rank 1 rgtr bit N+8
		43	Shift rank 1 rgtr bit N+9
		54	Shift rank 1 rgtr bit N+10
		53	Shift rank 1 rgtr bit N+11
		64	Shift rank 1 rgtr bit N+12
		63	Shift rank 1 rgtr bit N+13
		72	Shift rank 1 rgtr bit N+14
		76	Shift rank 1 rgtr bit N+15
		71	25-nanosecond clock
		75	Gate output
		72	No shift
		04	Shift 0
		03	4
		06	8
		05	12
		63	16
		52	32
		55	48
4EE7*	7J14	51, 54, 55, 62	Bit 59 norm sign rgtr
		02	Exponent input rgtr bit 48
		04	Exponent input rgtr bit 49
		12	Exponent input rgtr bit 50
		14	Exponent input rgtr bit 51
		22	Exponent input rgtr bit 52
		24	Exponent input rgtr bit 53
		32	Exponent input rgtr bit 54
		34	Exponent input rgtr bit 55
		45	Exponent input rgtr bit 56
		43	Exponent input rgtr bit 57
		52	Exponent input rgtr bit 58
		65	Round flag
3EF7*	7J16	76	Enable shift
		03	Bit 0 enable
		04	1
		13	2
		14	3
		23	4
		24	5
		35	6
		32	7
		42	8
		45	9
		55	10
		01	Bit 0 borrow
		06	1
		11	2
		16	3
		21	4
		26	5
		52	Extend Bj
		72	25-nanosecond clock
4EG7*	7J15	56	Enable shift
		62	Shift count bit 0
		63	1
		65	2
		75	3
		73	4
		72	5
		32	No shift
		12	Shift 0
		02, 05	Shift 2
		15	Shift 4
		22	8
		25	12
		35	16
		42	32
		45	48
		52, 54	25-nanosecond clock
4EH7*	7J13	41, 42	Gate output
		72	Extend Bj

F3 F4



NOTES:

① IF BIT 59=1, COMPLEMENT EXPONENT AND COEF.

② IF EXPONENT IS INDEF OR OVERFLOW, DO NOT ENABLE SHIFT.

③ GATE OUTPUT IS BLOCKED FOR UNROUNDED NORM. OF 0 COEF. OR EXPONENT UNDERFLOW.

④ COMPLEMENT TO REMOVE BIAS.

⑤ THE ROUND BIT IS USED AS ONE BIT POSITION LESS THAN 0.

⑥ EVERY SECOND BIT (EG. 0-12 = 0, 2, 4, 6, 8, 10, 12)

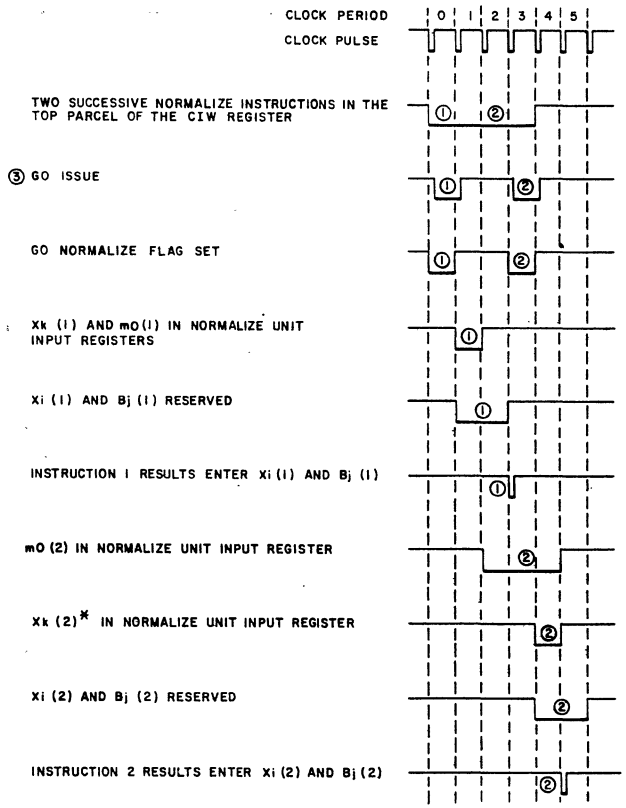
⑦ EVERY FOURTH BIT (EG. 0-12 = 0, 4, 8, 12)

F1 F2

CONTROL DATA DEVELOPMENT DIVISION

DETAILED - MODULES DIAGRAM
NORMALIZE UNIT

CODE IDENT	DWG NO	REV
34010	60420300	N
SIZE D	SHEET NORM 3.0	PAGE 5-4-5



NOTES:

① INSTRUCTION 1 - ASSUME NO ISSUE CONFLICTS

② INSTRUCTION 2 - ASSUME OPERAND REGISTER Xk (2) CONFLICTS WITH INSTRUCTION 1 DESTINATION REGISTER Xi (1). THAT IS, k (2) = i (1). THUS, ISSUE IS DELAYED TWO CLOCK PERIODS UNTIL THE Xi (1) RESERVATION IS CLEARED.

* THE Xk (2) QUANTITY IS THE INSTRUCTION 1 RESULT THAT IS ENTERED INTO Xi (1).

③ ENABLE ISSUE IS DELAYED 4 CLOCK PERIODS (THEREFORE, GO ISSUE IS DELAYED A MINIMUM OF 4 CLOCK PERIODS) IN THE AA120-C. THIS DELAY IS REMOVED BY INSTALLING OPTION AT364-A.

H9 H10

CONTROL DATA DEVELOPMENT DIVISION	TIMING DIAGRAM NORMALIZE INSTRUCTION	CODE IDENT 34010	DWG NO 60420300	REV R
		C	SHEET	PAGE 5-4-23

PART 5

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 m1 input flip-flop on top and the borrow register for the first stage of Xk minus Xj exponent on the bottom. The second column of registers has the m1 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 Xi 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 Xk 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 Xj 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.

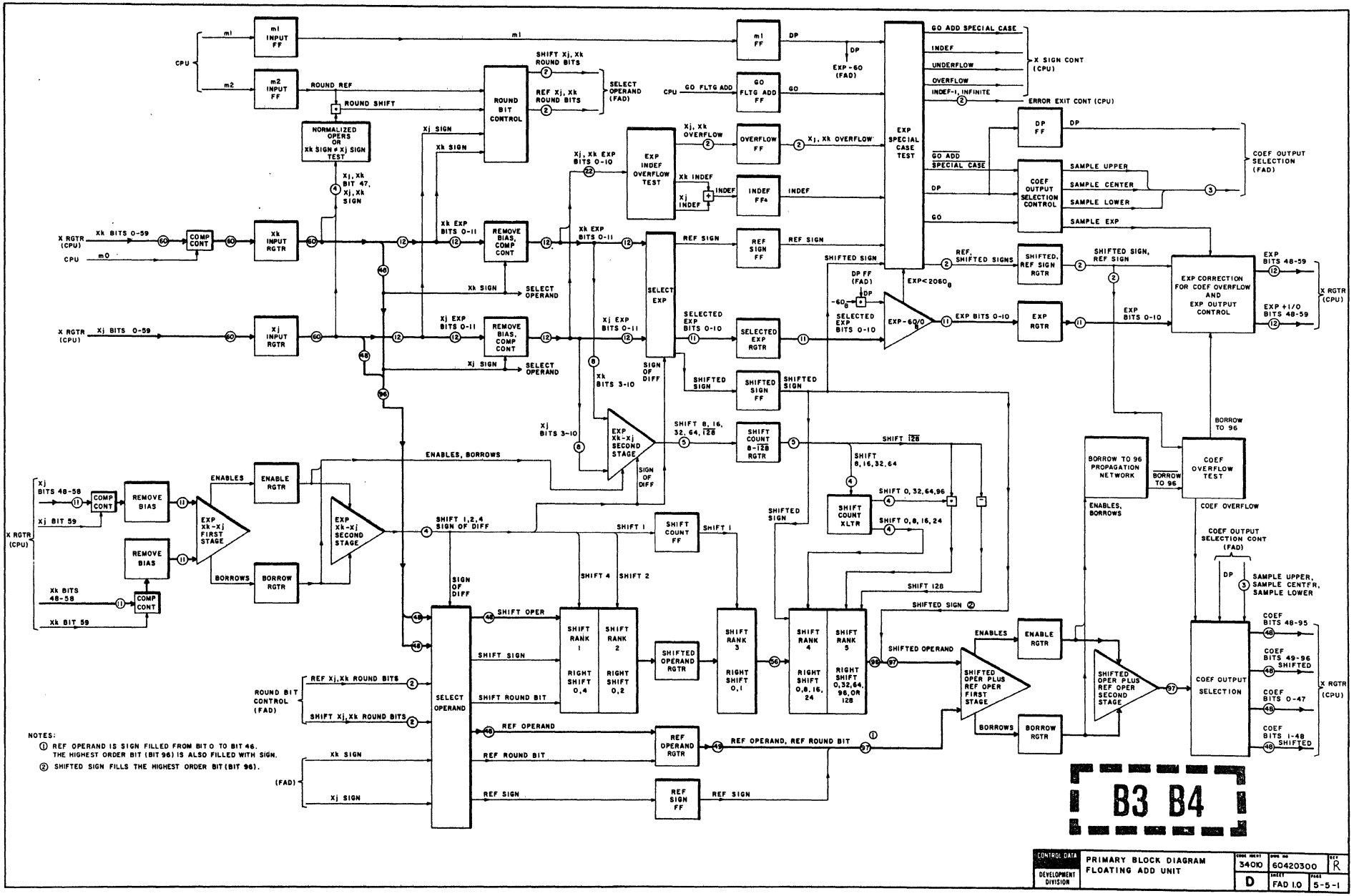
After the amount of the difference between the two exponents is determined, the smaller exponent is discarded, under control of sign of difference. The larger one is selected to continue through the unit to become the exponent of the result.

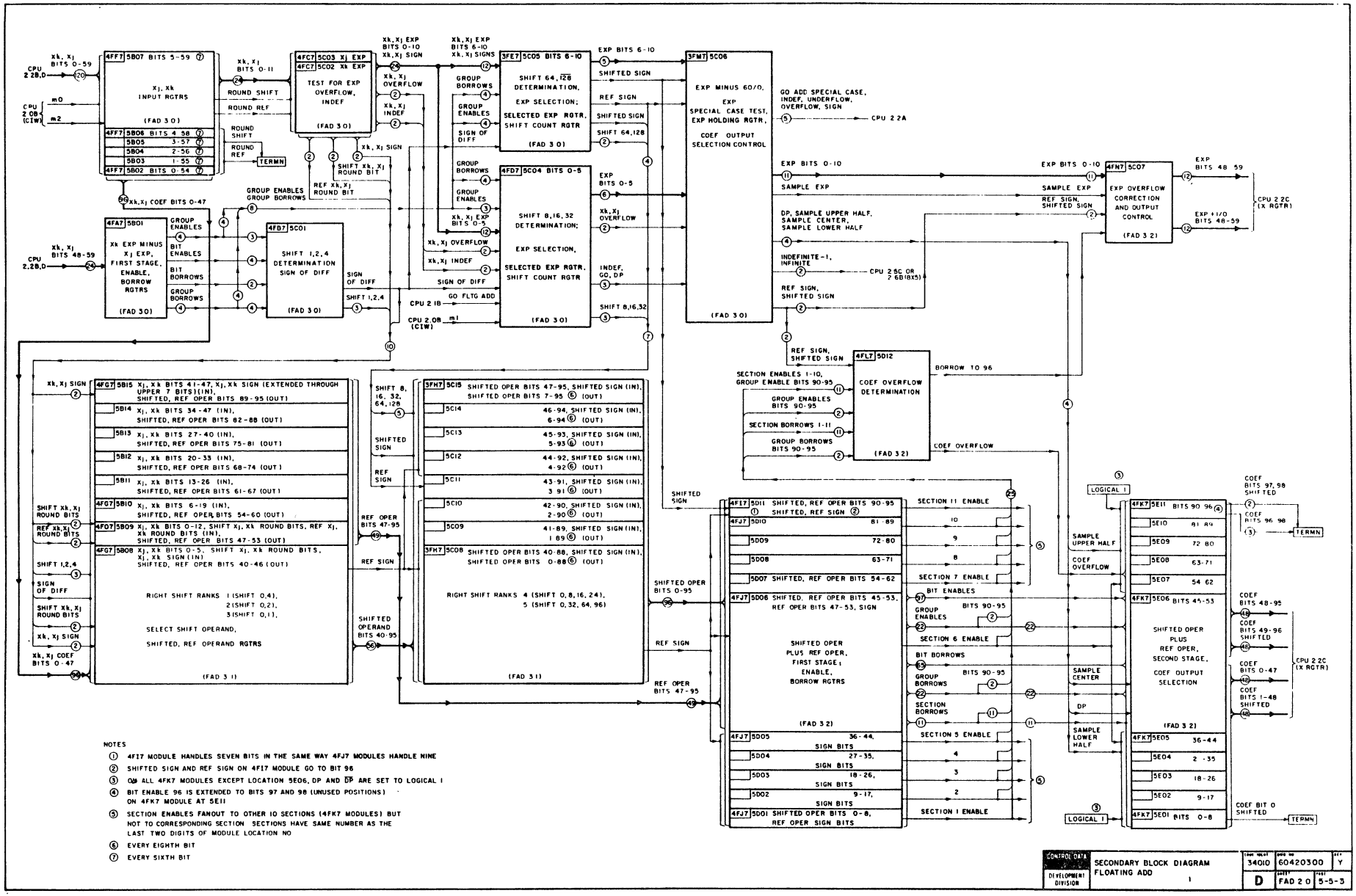
When the command is a 34 or 35 (round) instruction, m designator bit 2 causes the unit to add a round bit to one or both operands. If both operands are normalized or if the signs of the two operands are different, the round bit is added to both operands. If the above conditions are not met, the round 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.


CPU
m1
m2
m1 INPUT FF
m2 INPUT FF
ROUND REF
ROUND SHIFT
NORMALIZED OPERS OR Xk SIGN ≠ Xj SIGN TEST
ROUND BIT CONTROL
SHIFT Xj, Xk ROUND BITS
REF Xj, Xk ROUND BITS
SELECT OPERAND (FAD)
Xj SIGN
Xk SIGN
Xj, Xk BIT 47, Xj, Xk SIGN
X RGTR (CPU)
Xk BITS 0-59
COMP CONT
Xk INPUT RGTR
mO
Xk EXP BITS 0-11
REMOVE BIAS, COMP CONT
SELECT OPERAND
Xj BITS 0-59
Xj INPUT RGTR
Xj EXP BITS 0-11
Xj, Xk EXP BITS 0-10
EXP INDEF OVERFLOW TEST
Xj, Xk OVERFLOW
Xk INDEF
Xj INDEF
OVERFLOW FF
INDEF FF
Xj, Xk OVERFLOW
INDEF
SELECT EXP
REF SIGN
REF SIGN FF
SHIFTED SIGN
SELECTED EXP BITS 0-10
SELECTED EXP RGTR
SIGN OF DIFF
Xk BITS 3-10
Xj BITS 3-10
EXP Xk-Xj SECOND STAGE
SHIFT 8, 16, 32, 64, 128
SHIFTED SIGN FF
SHIFT COUNT 8-128 RGTR
ENABLES, BORROWS
Xj BITS 48-58
Xj BIT 59
Xk BITS 48-58
Xk BIT 59
REMOVE BIAS
EXP Xk-Xj FIRST STAGE
ENABLES
ENABLE RGTR
BORROWS
BORROW RGTR
SHIFT 1, 2, 4 SIGN OF DIFF
SHIFT COUNT FF
SHIFT 1
SHIFT 4
SHIFT 2
SHIFT OPER
SHIFT SIGN
SHIFT ROUND BIT
REF OPERAND
REF ROUND BIT
REF SIGN
SELECT OPERAND
ROUND BIT CONTROL (FAD)
REF Xj,Xk ROUND BITS
SHIFT Xj,Xk ROUND BITS
Xk SIGN
Xj SIGN
(FAD)
SHIFT RANK 1 RIGHT SHIFT 0,4
SHIFT RANK 2 RIGHT SHIFT 0,2
SHIFT RANK 3 RIGHT SHIFT 0,1
SHIFT RANK 4 RIGHT SHIFT 0,8,16,24
SHIFT RANK 5 RIGHT SHIFT 0,32,64,96,OR 128
SHIFTED OPERAND RGTR
REF OPERAND RGTR
REF SIGN FF
REF OPERAND, REF ROUND BIT
SHIFTED SIGN
SHIFT 128
SHIFT COUNT XLTR
SHIFT 0,32,64,96
SHIFT 0,8,16,24
SHIFTED OPERAND
SHIFTED OPER PLUS REF OPER FIRST STAGE
SHIFTED OPER PLUS REF OPER SECOND STAGE
ENABLES
ENABLE RGTR
BORROWS
BORROW RGTR
ENABLES, BORROWS
BORROW TO 96 PROPAGATION NETWORK
BORROW TO 96
COEF OVERFLOW TEST
COEF OVERFLOW
COEF OUTPUT SELECTION CONT (FAD)
DP
SAMPLE UPPER, SAMPLE CENTER, SAMPLE LOWER
COEF OUTPUT SELECTION
COEF BITS 48-95
COEF BITS 49-96 SHIFTED
COEF BITS 0-47
COEF BITS 1-48 SHIFTED
X RGTR (CPU)
m1 FF
DP
Exp-60 (FAD)
GO FLTG ADD
GO FLTG ADD FF
GO
EXP SPECIAL CASE TEST
GO ADD SPECIAL CASE
INDEF
UNDERFLOW
OVERFLOW
INDEF-1, INFINITE
X SIGN CONT (CPU)
ERROR EXIT CONT (CPU)
DP FF
COEF OUTPUT SELECTION CONTROL
SAMPLE UPPER
SAMPLE CENTER
SAMPLE LOWER
SAMPLE EXP
COEF OUTPUT SELECTION (FAD)
REF, SHIFTED SIGNS
SHIFTED, REF SIGN RGTR
SHIFTED SIGN, REF SIGN
EXP CORRECTION FOR COEF OVERFLOW AND EXP OUTPUT CONTROL
EXP BITS 48-59
EXP +1/0 BITS 48-59
EXP BITS 0-10
EXP RGTR
DP FF (FAD)
-60₈
EXP-60/0
EXP<2060₈
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).
B3 B4
CONTROL DATA DEVELOPMENT DIVISION
PRIMARY BLOCK DIAGRAM
FLOATING ADD UNIT
CODE IDENT 34010
DWG NO 60420300
REV R
SHEET D
PAGE FAD 1.0
5-5-1



NOTES

1. 4F17 MODULE HANDLES SEVEN BITS IN THE SAME WAY 4FJ7 MODULES HANDLE NINE
2. SHIFTED SIGN AND REF SIGN ON 4F17 MODULE GO TO BIT 96
3. ON ALL 4FK7 MODULES EXCEPT LOCATION 5E06, DP AND $\overline{DP}$ ARE SET TO LOGICAL 1
4. BIT ENABLE 96 IS EXTENDED TO BITS 97 AND 98 (UNUSED POSITIONS) ON 4FK7 MODULE AT 5E11
5. SECTION ENABLES FANOUT TO OTHER 10 SECTIONS (4FK7 MODULES) BUT NOT TO CORRESPONDING SECTION. SECTIONS HAVE SAME NUMBER AS THE LAST TWO DIGITS OF MODULE LOCATION NO
6. EVERY EIGHTH BIT
7. EVERY SIXTH BIT

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.

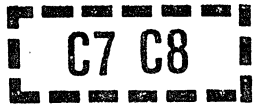
Sign of difference gates the sign of the operand with the smaller exponent into the shifted sign flip-flop and the sign of the operand with the larger exponent into the reference sign flip-flop. These registers and flip-flops hold the selected exponent, shifted signs, and reference signs for use during 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 postive 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.

C9 C10

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

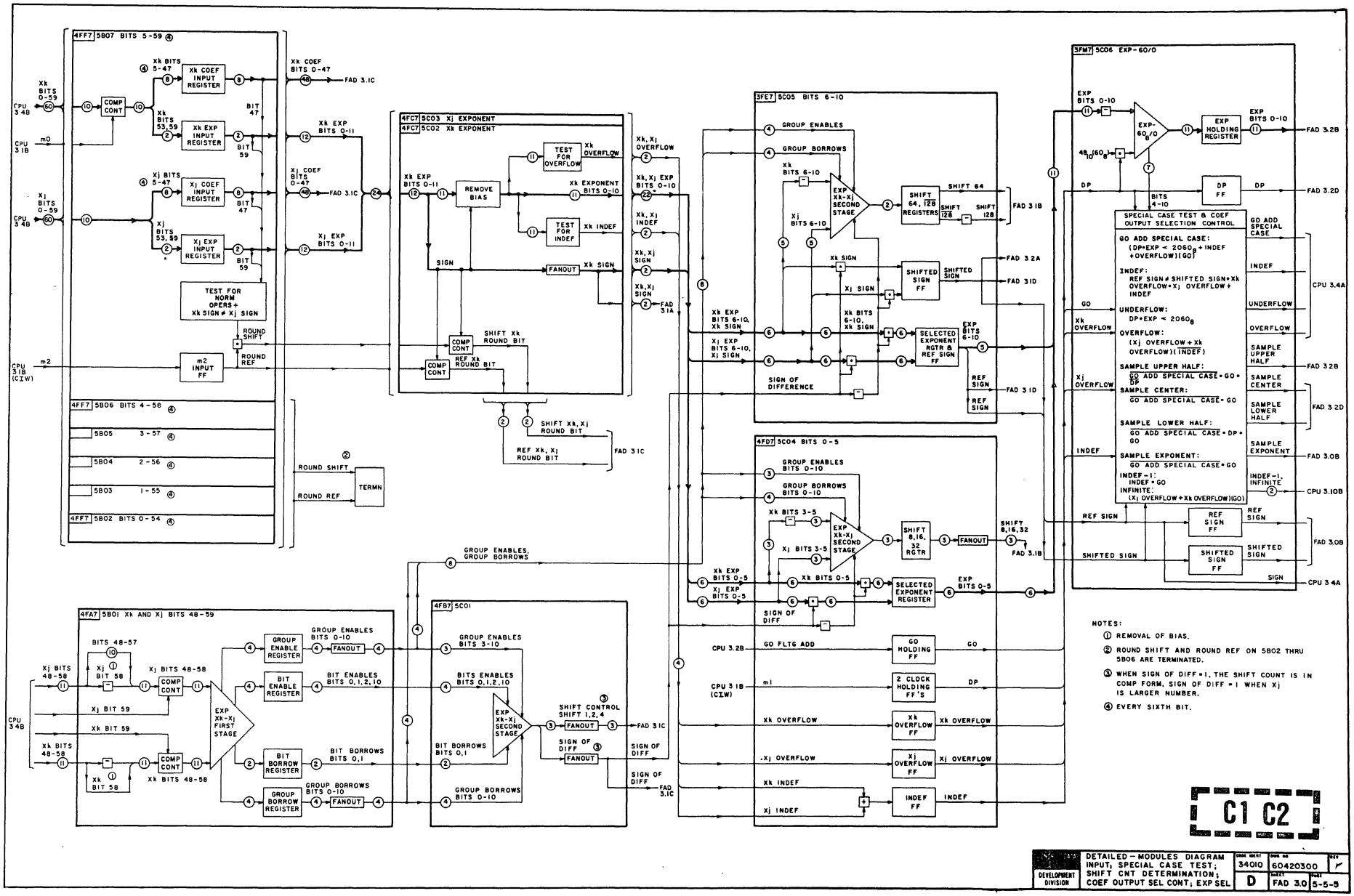
The exponent for the result coefficient is transmitted to the destination X register during the fourth clock period. Control for enabling the exponent output takes place on the 3FM7 module. If there are no special cases and go floating-add is received from the CPU, the sample exponent signal is sent to the 4FL7 module where it enables the output of the exponent.

FAD 3.0 TEST POINTS

Module	Location	Test Point	Description
4FA7*	5B01	74	Bit 0 enable
		73	1
		71	2
		66	10
		35	Group enable bits 0-2
		43	3-5
		53	6-8
		65	9-10
		72	Bit 0 borrow
		76	1
		36	Group borrow bits 0-2
		42	3-5
		52	6-8
		61	9-10
		75	25-nanosecond clock
4FB7*	5C01		(No signal/rgtr TPs)
4FC7*	5C02-03	25, 42	Sign
		51, 62	
4FD7*	5C04	76	Indefinite
		71	Xk overflow
		73	Xj overflow
		06	Exponent bit 0
		01	1
		11	2
		16	3
		26	4
		21	5
		64-66	Double precision
		63	Go
		32, 33	Shift 8
		41, 46	16
		51, 56	32
		61	25-nanosecond clock
3FE7*	5C05	41, 73	Shifted sign
		25	Ref sign
		05	Exponent bit 6
		06	7
		11	8
		16	9
		23	10
		31, 36	Shift 64
		51, 55	128
		75, 76	25-nanosecond clock
4FF7*	5B02-07	76	m0
		72	m2
		73	Round ref
		02	Xk coef bit N
		01	N+6
		22	N+12
		21	N+18
		35	Xk coef bit N+24
		36	N+30
		55	N+36
		56	N+42
		65	Xk exp bit N
		63	N+6
		04	Xj coef bit N
		06	N+6
		24	N+12
		26	N+18
		31	N+24
		33	N+30
		53	N+36
		51	N+42
		61	Xj exp bit N
		62	N+6
		71	25-nanosecond clock
3FM7*	5C06	56	Go
		66	Shifted sign
		62	Ref sign
		33	Exponent bit 0 enable
		05	1
		04	2
		03	3
		25	6
		24	7
		23	8
		35	10
		32	Exponent bit 0
		06	1
		01	2
		02	3
		11	4
		12	5
		26	6
		21	7
		22	8
		36	9
		31	10
		75, 76	25-nanosecond clock

Module	Location	Test Point	Description

C3 C4


4FF7 5B07 BITS 5-59
Xk BITS 0-59
Xk BITS 5-47
Xk COEF INPUT REGISTER
Xk BITS 53,59
Xk EXP INPUT REGISTER
BIT 59
BIT 47
Xk COEF BITS 0-47
FAD 3.1C
Xk EXP BITS 0-11
Xj BITS 0-59
Xj BITS 5-47
Xj COEF INPUT REGISTER
Xj BITS 53,59
Xj EXP INPUT REGISTER
Xj COEF BITS 0-47
Xj EXP BITS 0-11
TEST FOR NORM OPERS + Xk SIGN ≠ Xj SIGN
ROUND SHIFT
ROUND REF
m2 INPUT FF
CPU 3.4B
CPU 3.1B
CPU 3.1B (CIW)
m0
m2
4FF7 5B06 BITS 4-58
5B05 3-57
5B04 2-56
5B03 1-55
4FF7 5B02 BITS 0-54
4FC7 5C03 Xj EXPONENT
4FC7 5C02 Xk EXPONENT
REMOVE BIAS
TEST FOR OVERFLOW
Xk OVERFLOW
Xk EXPONENT BITS 0-10
TEST FOR INDEF
Xk INDEF
FANOUT
Xk SIGN
SIGN
Xk,Xj OVERFLOW
Xk,Xj EXP BITS 0-10
Xk,Xj INDEF
Xk,Xj SIGN
FAD 3.1A
COMP CONT
SHIFT Xk ROUND BIT
REF Xk ROUND BIT
SHIFT Xk,Xj ROUND BIT
REF Xk,Xj ROUND BIT
FAD 3.1C
TERMN
3FE7 5C05 BITS 6-10
GROUP ENABLES
GROUP BORROWS
EXP Xk-Xj SECOND STAGE
SHIFT 64, 128 REGISTERS
SHIFT 64
SHIFT 128
FAD 3.1B
Xk BITS 6-10
Xj BITS 6-10
Xk SIGN
Xj SIGN
SHIFTED SIGN FF
SHIFTED SIGN
FAD 3.2A
FAD 3.1D
Xk EXP BITS 6-10, Xk SIGN
Xj EXP BITS 6-10, Xj SIGN
SELECTED EXPONENT RGTR & REF SIGN FF
EXP BITS 6-10
SIGN OF DIFFERENCE
REF SIGN
4FD7 5C04 BITS 0-5
GROUP ENABLES BITS 0-10
GROUP BORROWS BITS 0-10
Xk BITS 3-5
Xj BITS 3-5
SHIFT 8,16, 32 RGTR
FANOUT
SHIFT 8,16,32
FAD 3.1B
Xk EXP BITS 0-5
Xj EXP BITS 0-5
Xk BITS 0-5
SELECTED EXPONENT REGISTER
EXP BITS 0-5
SIGN OF DIFF
4FA7 5B01 Xk AND Xj BITS 48-59
BITS 48-57
Xj BIT 58
Xj BITS 48-58
Xj BIT 59
Xk BIT 58
Xk BITS 48-58
Xk BIT 59
COMP CONT
EXP Xk-Xj FIRST STAGE
GROUP ENABLE REGISTER
GROUP ENABLES BITS 0-10
BIT ENABLE REGISTER
BIT ENABLES BITS 0,1,2,10
BIT BORROW REGISTER
BIT BORROWS BITS 0,1
GROUP BORROW REGISTER
GROUP BORROWS BITS 0-10
4FB7 5C01
GROUP ENABLES BITS 3-10
BITS ENABLES BITS 0,1,2,10
EXP Xk-Xj SECOND STAGE
SHIFT CONTROL SHIFT 1,2,4
SIGN OF DIFF
FAD 3.1C
FAD 3.1C
3FM7 5C06 EXP-60/0
EXP BITS 0-10
EXP-60/0
EXP HOLDING REGISTER
DP
DP FF
BITS 4-10
FAD 3.2B
FAD 3.2D
SPECIAL CASE TEST & COEF OUTPUT SELECTION CONTROL
GO ADD SPECIAL CASE:
(DP+EXP < 2060₈ + INDEF + OVERFLOW)(GO)
INDEF:
REF SIGN ≠ SHIFTED SIGN + Xk OVERFLOW + Xj OVERFLOW + INDEF
UNDERFLOW:
DP+EXP < 2060₈
OVERFLOW:
(Xj OVERFLOW + Xk OVERFLOW)(INDEF)
SAMPLE UPPER HALF:
GO ADD SPECIAL CASE • GO • DP
SAMPLE CENTER:
GO ADD SPECIAL CASE • GO
SAMPLE LOWER HALF:
GO ADD SPECIAL CASE • DP • GO
SAMPLE EXPONENT:
GO ADD SPECIAL CASE • GO
INDEF-1:
INDEF • GO
INFINITE:
(Xj OVERFLOW + Xk OVERFLOW)(GO)
GO ADD SPECIAL CASE
INDEF
UNDERFLOW
OVERFLOW
SAMPLE UPPER HALF
SAMPLE CENTER
SAMPLE LOWER HALF
SAMPLE EXPONENT
INDEF-1, INFINITE
CPU 3.4A
FAD 3.2B
FAD 3.2D
FAD 3.0B
CPU 3.10B
REF SIGN FF
SHIFTED SIGN FF
SIGN
FAD 3.0B
CPU 3.4A
GO
Xk OVERFLOW
Xj OVERFLOW
INDEF
REF SIGN
SHIFTED SIGN
CPU 3.2B
GO FLTG ADD
GO HOLDING FF
CPU 3.1B (CIW)
m1
2 CLOCK HOLDING FF'S
Xk OVERFLOW FF
Xj OVERFLOW FF
Xk INDEF
Xj INDEF
INDEF FF
NOTES:
① REMOVAL OF BIAS.
② ROUND SHIFT AND ROUND REF ON 5B02 THRU 5B06 ARE TERMINATED.
③ WHEN SIGN OF DIFF = 1, THE SHIFT COUNT IS IN COMP FORM, SIGN OF DIFF = 1 WHEN Xj IS LARGER NUMBER.
④ EVERY SIXTH BIT.
C1 C2
DEVELOPMENT DIVISION
DETAILED – MODULES DIAGRAM
INPUT, SPECIAL CASE TEST; SHIFT CNT DETERMINATION; COEF OUTPUT SEL CONT; EXP SEL
CODE IDENT 34010
DWG NO 60420300
REV
SIZE D
SCALE FAD 3.0
SHEET 5-5-9

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 Xj is the larger exponent. Sign of difference also gates the proper round bits. If the Xk coefficient becomes the shift operand, the shift Xk round bit is selected for the shift operand. If the Xk coefficient becomes the reference operand, the reference Xk 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 $\overline{\text{shift 128}}$ and go to shift rank 5. Shift 128 goes directly to shift rank 5.

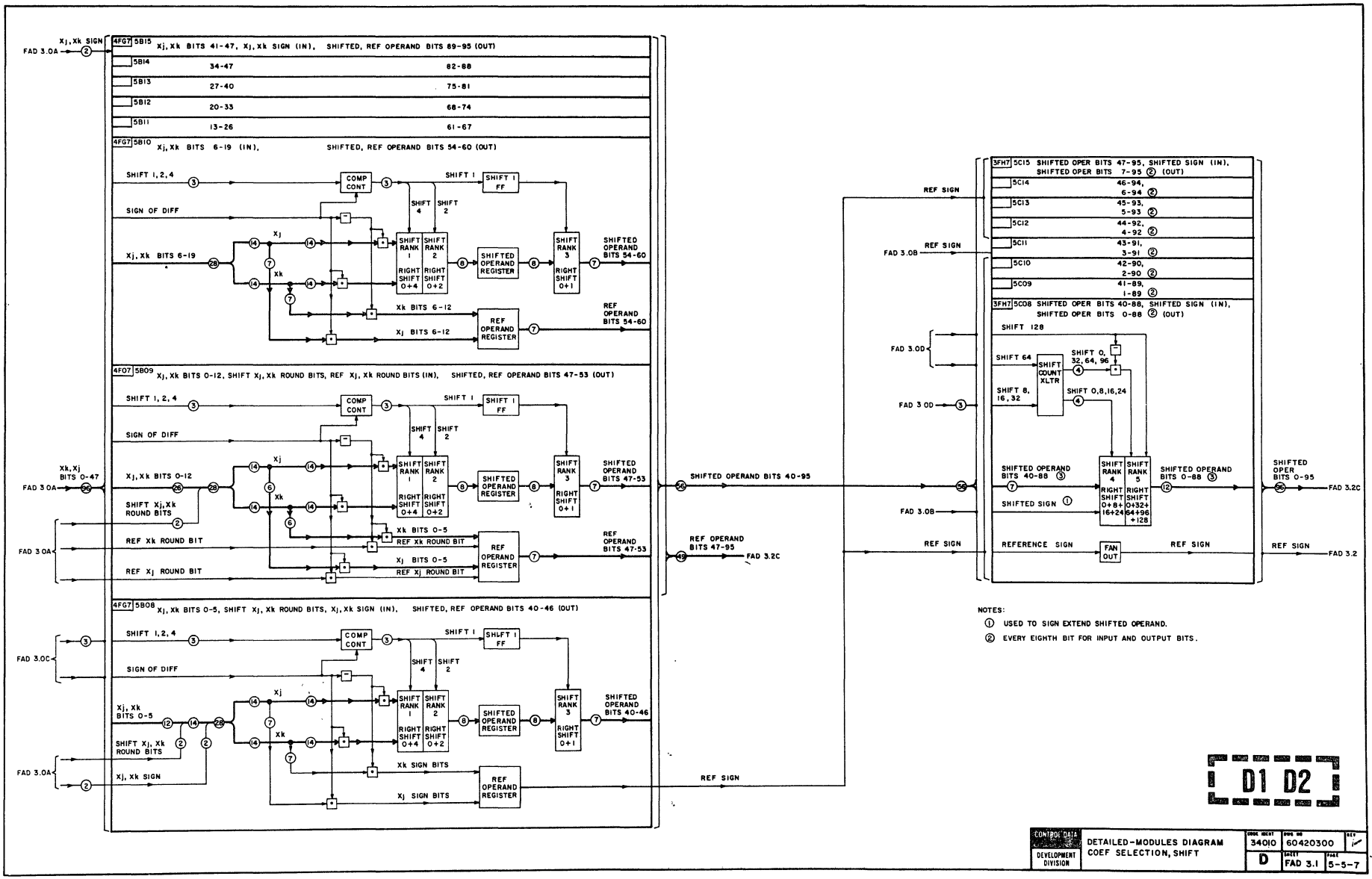
The output of shift rank 5 is shifted operand bits 0 through 95. The 3FH7 module sends this shifted operand to the 4FJ7 and 4FI7 modules for the first stage of addition.

D7 D8

FAD 3.1 TEST POINTS

Module	Location	Test Point	Description	Module	Location	Test Point	Description	Module	Location	Test Point	Description
4FG7*	5(B08, B10-15)	53	Ref operand bit N								
		52	N+1								
		51	N+2								
		56	N+3								
		55	N+4								
		54	N+5								
		15	Shifted operand rgtr bit N								
		25	Shifted operand rgtr bit N+1								
		35	Shifted operand rgtr bit N+2								
		42	Shifted operand rgtr bit N+3								
		16	Shifted operand rgtr bit N+4								
		26	Shifted operand rgtr bit N+5								
		36	Shifted operand rgtr bit N+6								
		41	Shifted operand rgtr bit N+7								
3FH7*	5C08-15	32, 33 35, 36	Shifted sign								
4FO7*	5B09	53	Ref operand bit 47								
		52	48								
		51	49								
		56	50								
		55	51								
		54	52								

D3 D4


Xj, Xk SIGN
FAD 3.0A
4FG7 5B15 Xj, Xk BITS 41-47, Xj, Xk SIGN (IN), SHIFTED, REF OPERAND BITS 89-95 (OUT)
5B14 34-47 82-88
5B13 27-40 75-81
5B12 20-33 68-74
5B11 13-26 61-67
4FG7 5B10 Xj, Xk BITS 6-19 (IN), SHIFTED, REF OPERAND BITS 54-60 (OUT)
SHIFT 1, 2, 4
COMP CONT
SHIFT 1
SHIFT 1 FF
SIGN OF DIFF
SHIFT 4
SHIFT 2
Xj
Xk
Xj, Xk BITS 6-19
SHIFT RANK 1
SHIFT RANK 2
RIGHT SHIFT 0+4
RIGHT SHIFT 0+2
SHIFTED OPERAND REGISTER
SHIFT RANK 3
RIGHT SHIFT 0+1
SHIFTED OPERAND BITS 54-60
Xk BITS 6-12
Xj BITS 6-12
REF OPERAND REGISTER
REF OPERAND BITS 54-60
4FO7 5B09 Xj, Xk BITS 0-12, SHIFT Xj, Xk ROUND BITS, REF Xj, Xk ROUND BITS (IN), SHIFTED, REF OPERAND BITS 47-53 (OUT)
Xk, Xj BITS 0-47
FAD 3.0A
Xj, Xk BITS 0-12
SHIFT Xj, Xk ROUND BITS
FAD 3.0A
REF Xk ROUND BIT
REF Xj ROUND BIT
Xk BITS 0-5
REF Xk ROUND BIT
Xj BITS 0-5
REF Xj ROUND BIT
SHIFTED OPERAND BITS 47-53
REF OPERAND BITS 47-53
4FG7 5B08 Xj, Xk BITS 0-5, SHIFT Xj, Xk ROUND BITS, Xj, Xk SIGN (IN), SHIFTED, REF OPERAND BITS 40-46 (OUT)
FAD 3.0C
Xj, Xk BITS 0-5
SHIFT Xj, Xk ROUND BITS
FAD 3.0A
Xj, Xk SIGN
Xk SIGN BITS
Xj SIGN BITS
SHIFTED OPERAND BITS 40-46
REF SIGN
SHIFTED OPERAND BITS 40-95
REF OPERAND BITS 47-95
FAD 3.2C
REF SIGN
FAD 3.0B
FAD 3.0D
FAD 3.0D
FAD 3.0B
REF SIGN
3FH7 5C15 SHIFTED OPER BITS 47-95, SHIFTED SIGN (IN), SHIFTED OPER BITS 7-95 (2) (OUT)
5C14 46-94, 6-94 (2)
5C13 45-93, 5-93 (2)
5C12 44-92, 4-92 (2)
5C11 43-91, 3-91 (2)
5C10 42-90, 2-90 (2)
5C09 41-89, 1-89 (2)
3FH7 5C08 SHIFTED OPER BITS 40-88, SHIFTED SIGN (IN), SHIFTED OPER BITS 0-88 (2) (OUT)
SHIFT 128
SHIFT 64
SHIFT 8, 16, 32
SHIFT COUNT XLTR
SHIFT 0, 32, 64, 96
SHIFT 0, 8, 16, 24
SHIFTED OPERAND BITS 40-88 (3)
SHIFTED SIGN (1)
SHIFT RANK 4
SHIFT RANK 5
RIGHT SHIFT 0+8+16+24
RIGHT SHIFT 0+32+64+96+128
SHIFTED OPERAND BITS 0-88 (3)
REFERENCE SIGN
FAN OUT
REF SIGN
SHIFTED OPER BITS 0-95
REF SIGN
FAD 3.2C
REF SIGN
FAD 3.2
NOTES:
(1) USED TO SIGN EXTEND SHIFTED OPERAND.
(2) EVERY EIGHTH BIT FOR INPUT AND OUTPUT BITS.
D1 D2
CONTROL DATA DEVELOPMENT DIVISION
DETAILED-MODULES DIAGRAM
COEF SELECTION, SHIFT
DRWG SIZE D
DRWG NO 60420300
REV I
SHEET FAD 3.1
PAGE 5-5-7

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 upperor 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	$\overline{\text{Borrow to 96}}$	$\overline{\text{Borrow to 96}}$
+ sign of the result	- sign of the result	+ sign of the result
Output in true form	Output in ones complement form	Output in 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
$\overline{\text{Borrow to 96}}$	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.

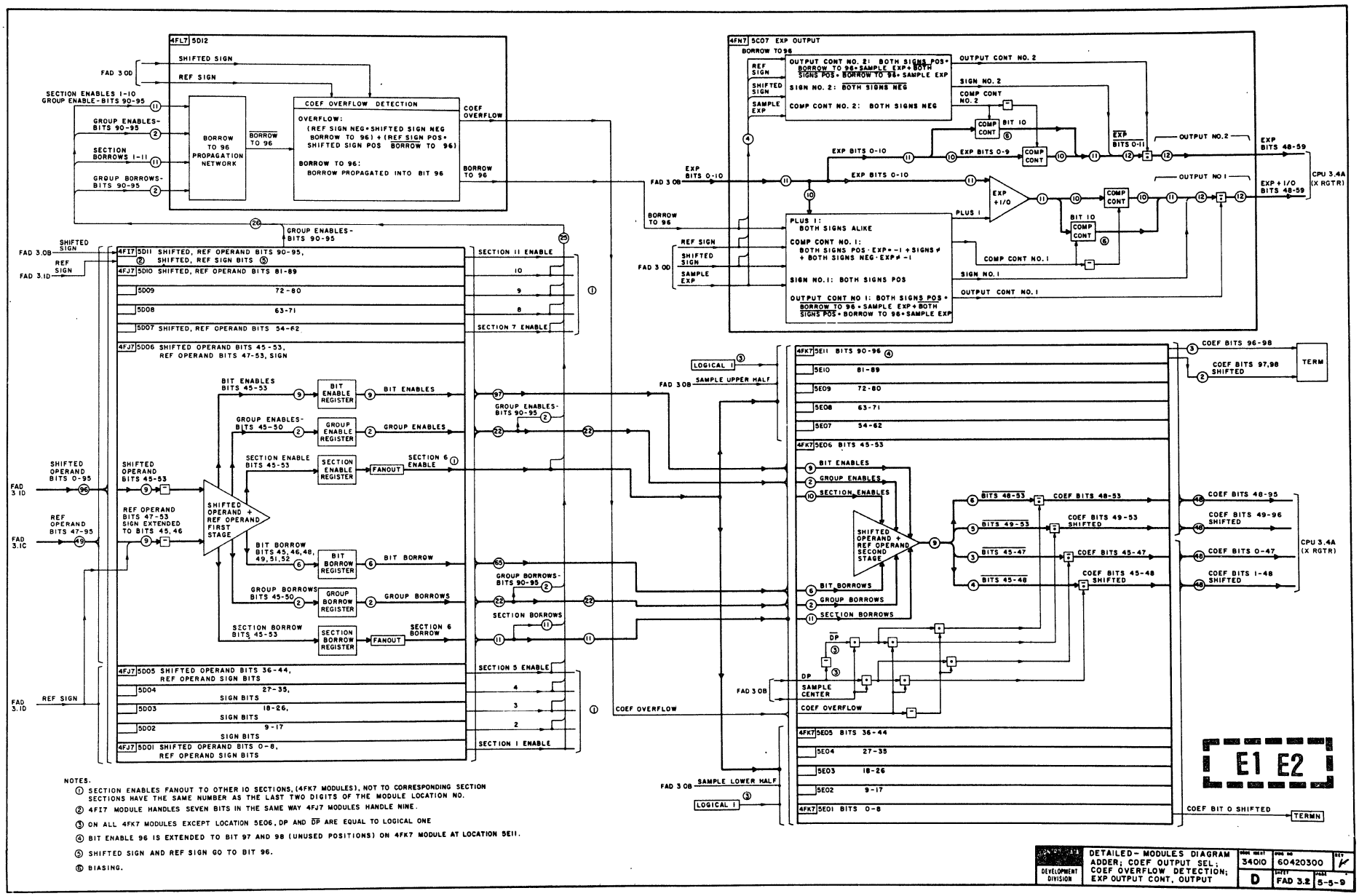
FAD 3.2 TEST POINTS

Module	Location	Test Point	Description
4FI7*	5D11	34	Bit 90 enable
		36	91
		56	92
		45	93
		46	94
		52	95
		25	96-98
		74	Bit 90 borrow
		75	91
		76	93
		73	94
		24	96, 97
		66	Group enable bits 90-92
		61	93-95
		43	90-95
		35	Group borrow bits 90-92
		71	Group borrow bits 93-95
		42, 63	Section enable bits 90-96
		31, 53	Section borrow bits 90-96
		22	25-nanosecond clock
4FJ7*	5D01-10	32	Bit N enable
		33	N+1
		45	N+2
		44	N+3
		65	N+4
		66	N+5
		55	N+6
		54	N+7
		56	N+8
		71	Bit N borrow
		74	N+1
		72	N+3
		75	N+4
		73	N+6
		76	N+7
		62	Group enable bits N-N+2
		61	Group enable bits N+3-N+5
		31	Group borrow bits N-N+2
		46	Group borrow bits N+3-N+5
		43, 63	Section enable bits N-N+8
		34, 53	Section borrow bits N-N+8

Module	Location	Test Point	Description
4FK7*	5E01-11		(No signal/rgtr TPs)
4FL7*	5D12		(No signal/rgtr TPs)
4FN7*	5C07		(No signal/rgtr TPs)

Module	Location	Test Point	Description

E3 E4



NOTES.

① SECTION ENABLES FANOUT TO OTHER 10 SECTIONS, (4FK7 MODULES), NOT TO CORRESPONDING SECTION SECTIONS HAVE THE SAME NUMBER AS THE LAST TWO DIGITS OF THE MODULE LOCATION NO.

② 4FI7 MODULE HANDLES SEVEN BITS IN THE SAME WAY 4FJ7 MODULES HANDLE NINE.

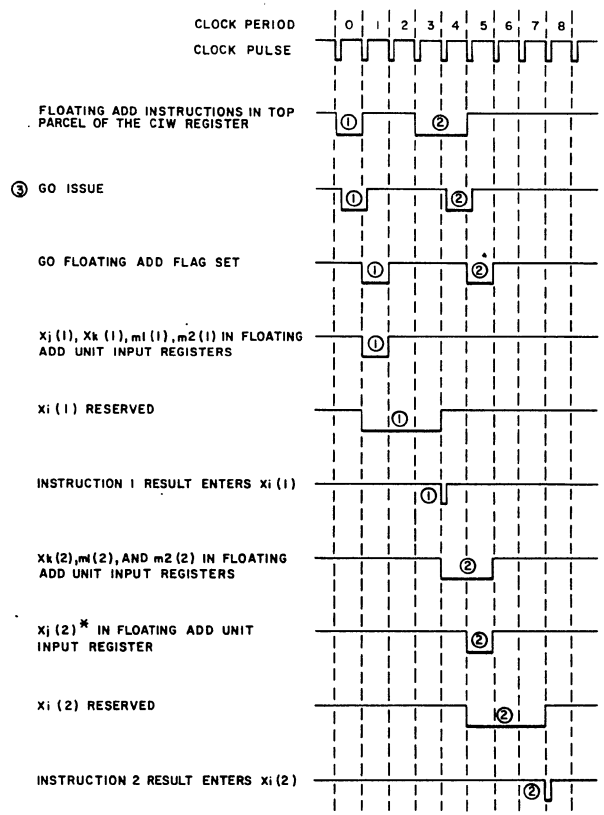
③ ON ALL 4FK7 MODULES EXCEPT LOCATION 5E06, DP AND $\overline{DP}$ ARE EQUAL TO LOGICAL ONE

④ BIT ENABLE 96 IS EXTENDED TO BIT 97 AND 98 (UNUSED POSITIONS) ON 4FK7 MODULE AT LOCATION 5E11.

⑤ SHIFTED SIGN AND REF SIGN GO TO BIT 96.

⑥ BIASING.

DEVELOPMENT DIVISION	DETAILED- MODULES DIAGRAM ADDER; COEF OUTPUT SEL; COEF OVERFLOW DETECTION; EXP OUTPUT CONT, OUTPUT	34010	60420300	K
		D	FAD 3.2	5-5-9



NOTES:

① INSTRUCTION 1 - ASSUME NO ISSUE CONFLICTS.

② INSTRUCTION 2 - ASSUME OPERAND REGISTER Xj(2) CONFLICTS WITH INSTRUCTION 1 DESTINATION REGISTER Xi(1). THAT IS, j(2) = i(1) THUS, ISSUE DELAYED ONE CLOCK PERIOD UNTIL Xi(1) RESERVATION IS CLEARED.

* THE Xj(2) QUANTITY IS THE INSTRUCTION 1 RESULT THAT IS ENTERED INTO Xi(1)

③ ENABLE ISSUE IS DELAYED 4 CLOCK PERIODS (THEREFORE, GO ISSUE IS DELAYED A MINIMUM OF 4 CLOCK PERIODS) IN THE AA120-C. THIS DELAY IS REMOVED BY INSTALLING OPTION AT364-A.

C7 C8

CONTROL DATA DEVELOPMENT DIVISION	TIMING DIAGRAM FLOATING ADD INSTRUCTION	CODE IDENT 34010	DWG NO 60420300	REV R
		C	SHEET	PAGE 5-5-41

PART 6

LONG ADD UNIT

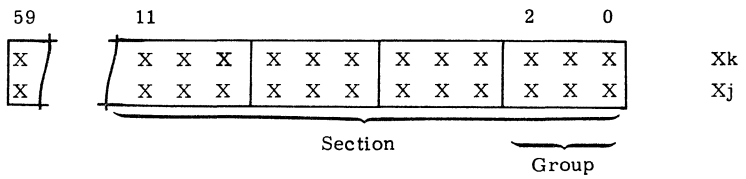
LONG ADD UNIT

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 i 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.

$$\text{Bit enables} = \begin{matrix}0\\1\\1\end{matrix} \quad \text{or} \quad \begin{matrix}1\\0\\1\end{matrix}$$

$$\text{Bit borrow generates} = \begin{matrix}1\\1\\0\end{matrix}$$

Group borrow generates = borrow-out of three-bit group

Example:

$$\begin{matrix}110\\010\\000\end{matrix} \quad \text{or} \quad \begin{matrix}101\\110\\011\end{matrix}$$

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

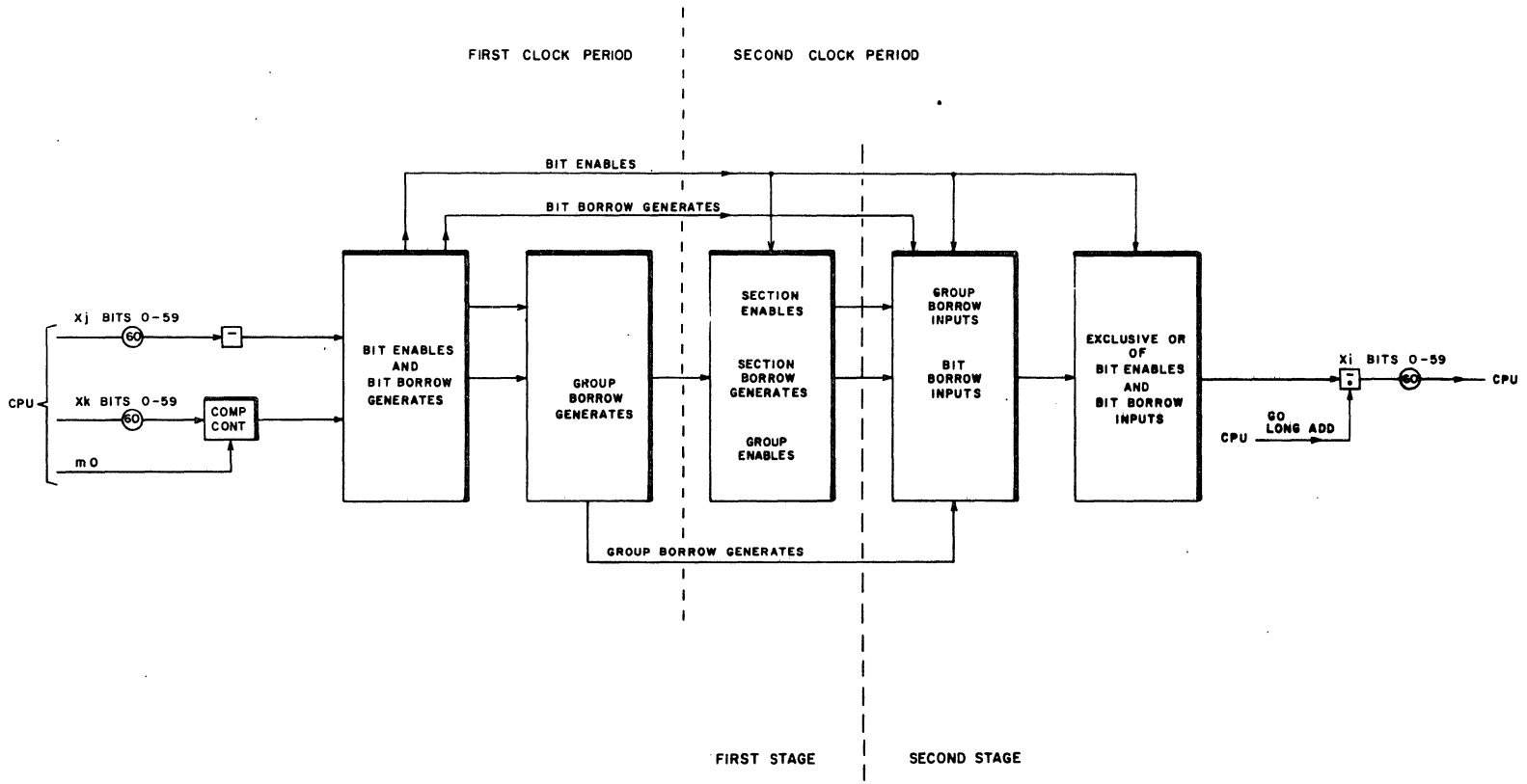
Bit borrow inputs = borrow into bit

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.

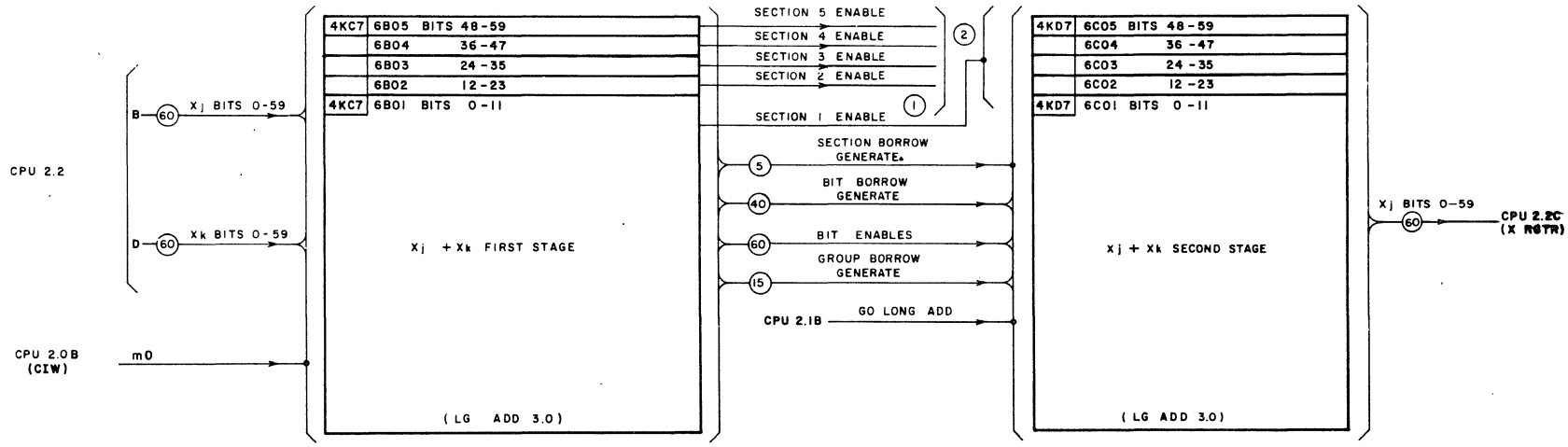
36 Instruction Example:

4470 → Comp → 3307 →	011	011	000	111		
1532 → Comp → 6245 →	110	010	100	101		
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)	



D3 D4

CONTROL DATA DEVELOPMENT DIVISION	PRIMARY BLOCK DIAGRAM LONG ADD UNIT	CODE IDENT 34010	DWG NO 60420300	REV K
		C	SHEET LG ADD 10	PAGE 5-6-1



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

D11 D12

CONTROL DATA DEVELOPMENT DIVISION	SECONDARY BLOCK DIAGRAM LONG ADD	CODE IDENT 34010	DWG NO 60420300	REV K
		C	SHEET LG ADD 2.0	PAGE 5-6-3

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.

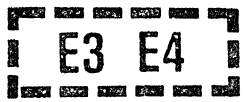
During the first clock period, the 4KC7 modules complement both operands. If the instruction is integer difference (37), m bit 0 is a one and the 4KC7 modules recomplement the Xk operand. After complementing, the operands enter the first stage of the adder.

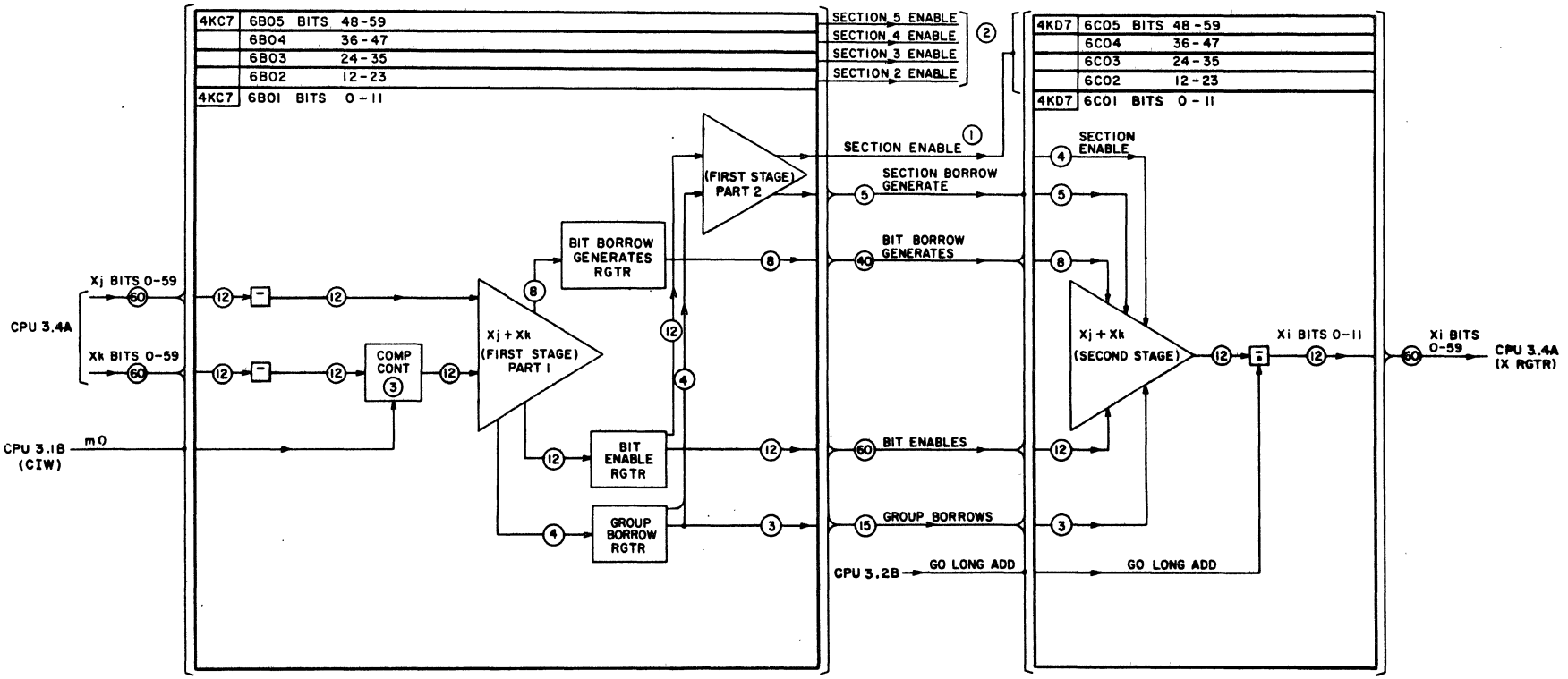
The first stage of the adder is divided into two parts. The first part forms the bit enables, bit borrow generates, and group borrow generates. Registers hold them for use during the second clock period. The second part of the first stage of addition takes place during the second clock period. It forms 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.

LG ADD 2.0 TEST POINTS

Module	Location	Test Point	Description	Module	Location	Test Point	Description	Module	Location	Test Point	Description
4KC7*	6B01-05	35	Bit N enable								
		33	N+1								
		31	N+2								
		42	N+3								
		44	N+4								
		46	N+5								
		52	N+6								
		54	N+7								
		56	N+8								
		65	N+9								
		64	N+10								
		61	N+11								
		36	Bit N borrow								
		34	N+1								
		41	N+3								
		43	N+4								
		51	N+6								
		53	N+7								
		66	N+9								
		63	N+10								
		32	Group N borrow								
		45	N+1								
		55	N+2								
		72	Section enable								
		71	25-nanosecond clock								
4KD7*	6C01-05	71, 72	Go long add								





NOTE:

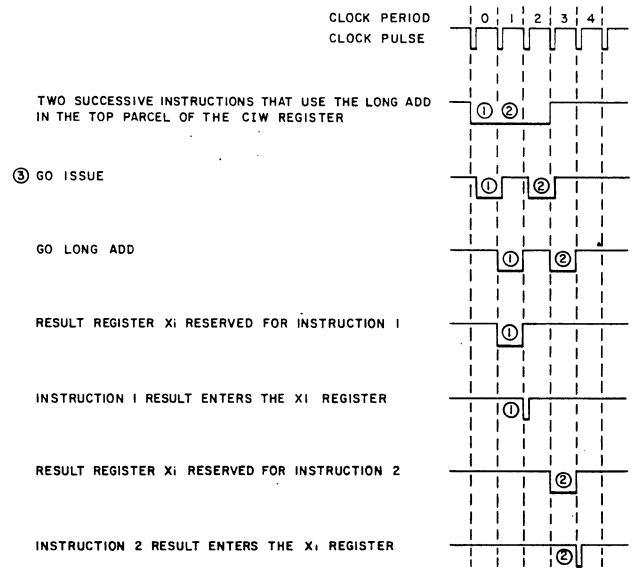
(1) SECTION 1 ENABLE GOES TO 6C02, 6C03, 6C04, 6C05

(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

E1 E2

CONTROL DATA	DETAILED-MODULES DIAGRAM	CODE IDENT	DWG NO	REV
DEVELOPMENT DIVISION	LONG ADD UNIT	34010	60420300	K
		C	SHEET LG AD03,0	PAGE 5-6-5



NOTES:

① 36 ijk (ASSUMING NO ISSUE CONFLICTS EXIST).

② 37 ijl (OPERAND REGISTER CONFLICTS WITH INSTRUCTION 1 DESTINATION REGISTER).

③ ENABLE ISSUE IS DELAYED 4 CLOCK PERIODS (THEREFORE, GO ISSUE IS DELAYED A MINIMUM OF 4 CLOCK PERIODS) IN THE AA120-C. THIS DELAY IS REMOVED BY INSTALLING OPTION AT364-A.

F5 F6

CONTROL DATA DEVELOPMENT DIVISION	TIMING DIAGRAM LONG ADD INSTRUCTION	CODE IDENT 34010	DWG NO 60420300	REV R
		C	SHEET	PAGE 5-6-11

PART 7

MULTIPLY UNIT

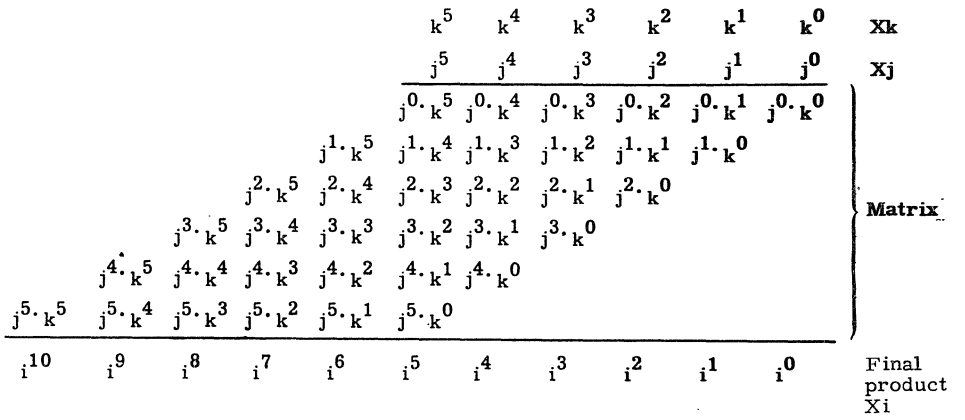
The floating-multiply unit executes the following three instructions.

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.



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.

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 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	
					$j^0 \cdot k^5$	$j^0 \cdot k^4$	$j^0 \cdot k^3$	$j^0 \cdot k^2$	$j^0 \cdot k^1$	$j^0 \cdot k^0$	
				$j^1 \cdot k^5$	$j^1 \cdot k^4$	$j^1 \cdot k^3$	$j^1 \cdot k^2$	$j^1 \cdot k^1$	$j^1 \cdot k^0$		
			$j^2 \cdot k^5$	$j^2 \cdot k^4$	$j^2 \cdot k^3$	$j^2 \cdot k^2$	$j^2 \cdot k^1$	$j^2 \cdot k^0$			
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			
			$j^3 \cdot k^5$	$j^3 \cdot k^4$	$j^3 \cdot k^3$	$j^3 \cdot k^2$	$j^3 \cdot k^1$	$j^3 \cdot k^0$			
		$j^4 \cdot k^5$	$j^4 \cdot k^4$	$j^4 \cdot k^3$	$j^4 \cdot k^2$	$j^4 \cdot k^1$	$j^4 \cdot k^0$				
	$j^5 \cdot k^5$	$j^5 \cdot k^4$	$j^5 \cdot k^3$	$j^5 \cdot k^2$	$j^5 \cdot k^1$	$j^5 \cdot k^0$					
	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.

Bit	10	9	8	7	6	5	4	3	2	1	0
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.

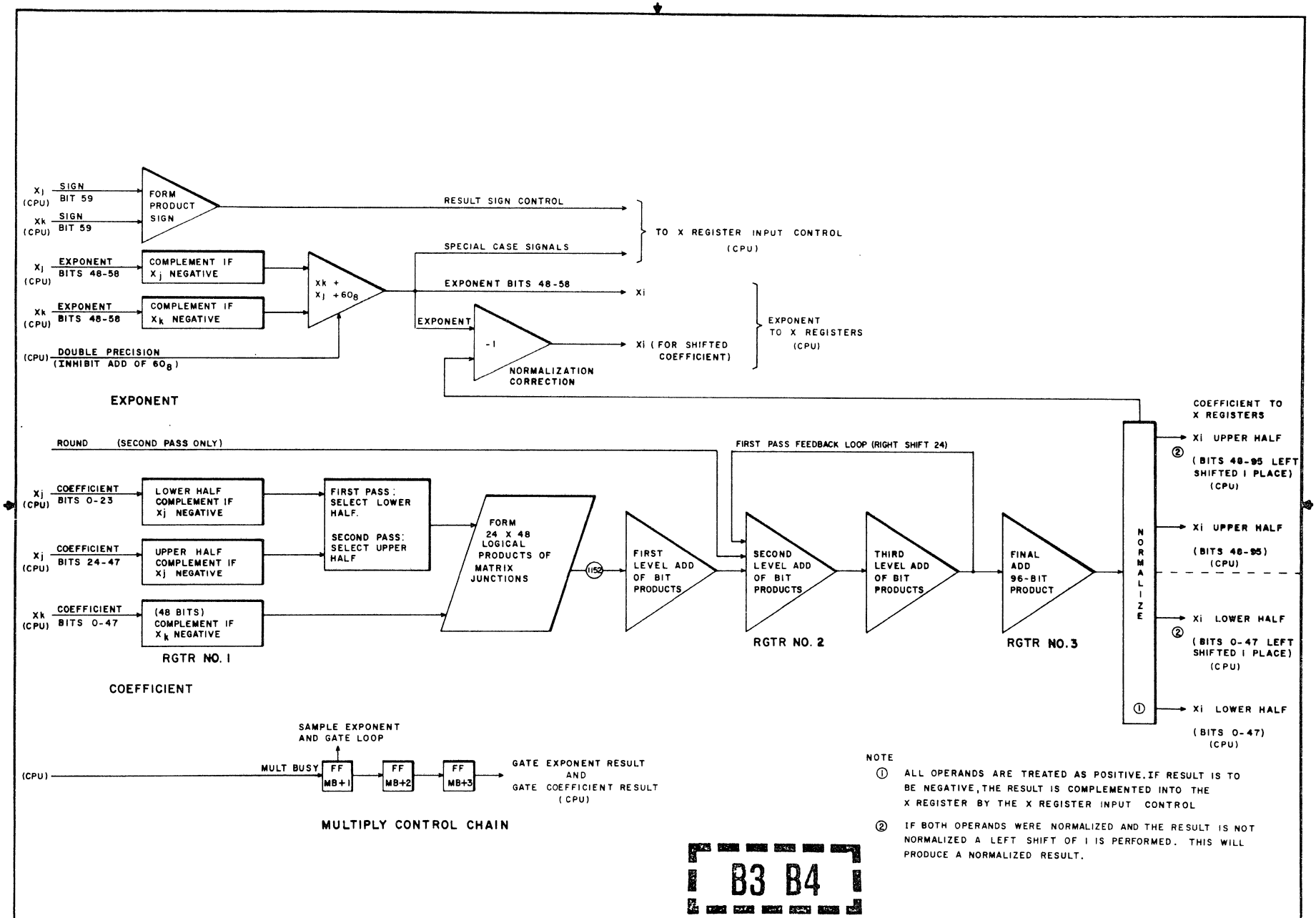
MINUS ONE NETWORK

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.



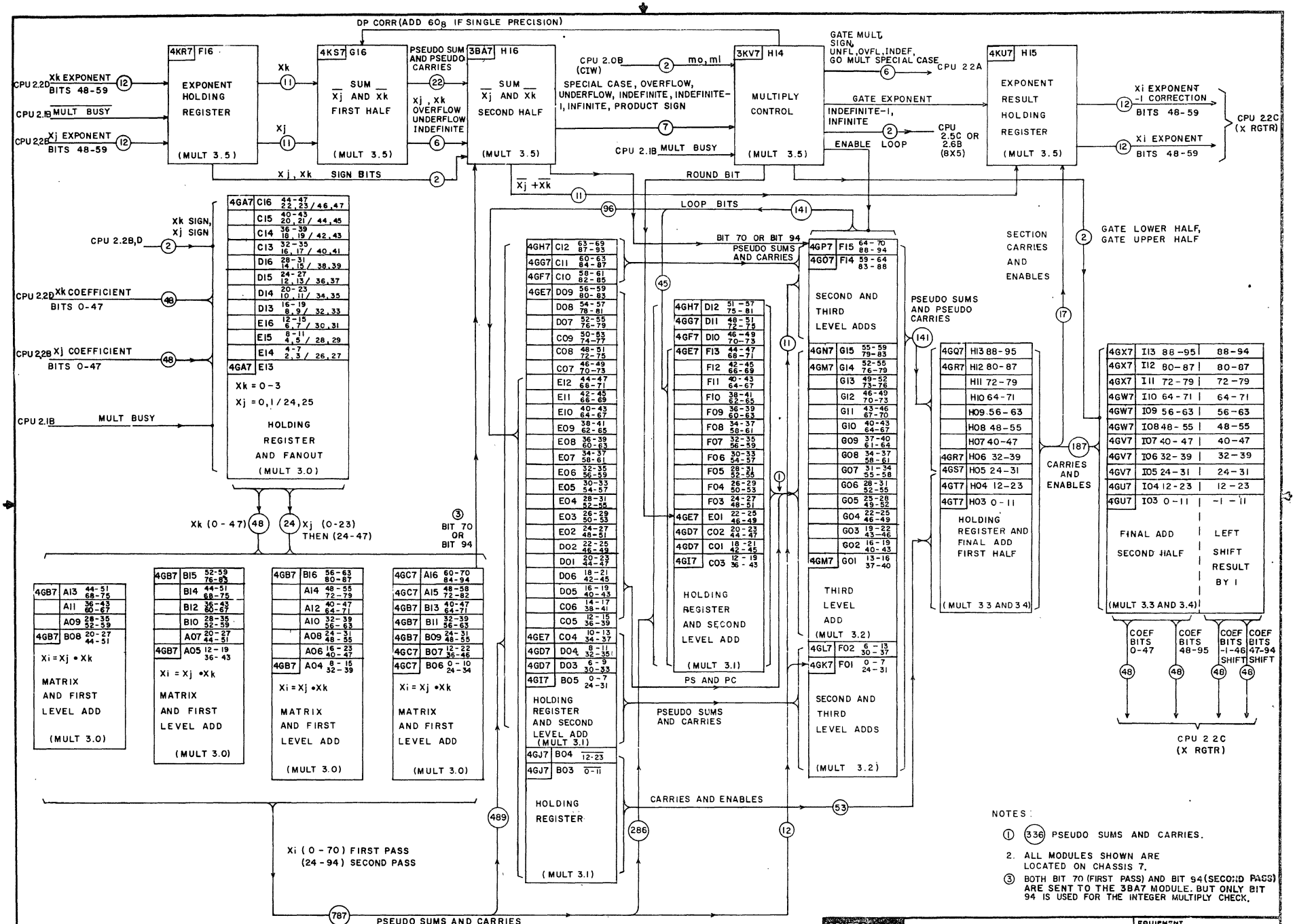
NOTE

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.

B3 B4

CONTROL DATA DEVELOPMENT DIVISION	PRIMARY BLOCK DIAGRAM MULTIPLY UNIT	EQUIPMENT
		SIZE C · DRAWING NO 60420300 · REV K
		SHEET MULT 1.0 · PAGE 5-7-1


DP CORR (ADD 60B IF SINGLE PRECISION)
4KR7 FI6
EXPONENT HOLDING REGISTER
(MULT 3.5)
CPU 2.2D Xk EXPONENT BITS 48-59
CPU 2.1B MULT BUSY
CPU 2.2E Xj EXPONENT BITS 48-59
Xk
Xj
4KS7 G16
SUM Xj AND Xk FIRST HALF
(MULT 3.5)
PSEUDO SUM AND PSEUDO CARRIES
Xj, Xk OVERFLOW UNDERFLOW INDEFINITE
3BA7 H16
SUM Xj AND Xk SECOND HALF
(MULT 3.5)
CPU 2.0B (CIW)
mo, ml
SPECIAL CASE, OVERFLOW, UNDERFLOW, INDEFINITE, INDEFINITE-I, INFINITE, PRODUCT SIGN
3KV7 H14
MULTIPLY CONTROL
(MULT 3.5)
CPU 2.1B MULT BUSY
GATE MULT, SIGN, UNFL, OVFL, INDEF, GO MULT SPECIAL CASE
CPU 2.2A
GATE EXPONENT
INDEFINITE-I, INFINITE
ENABLE LOOP
CPU 2.5C OR 2.6B (8X5)
4KU7 H15
EXPONENT RESULT HOLDING REGISTER
(MULT 3.5)
Xi EXPONENT -I CORRECTION BITS 48-59
Xi EXPONENT BITS 48-59
CPU 2.2C (X RGTR)
Xj, Xk SIGN BITS
ROUND BIT
Xj + Xk
LOOP BITS
BIT 70 OR BIT 94 PSEUDO SUMS AND CARRIES
SECTION CARRIES AND ENABLES
GATE LOWER HALF, GATE UPPER HALF
Xk SIGN, Xj SIGN
CPU 2.2B,D
CPU 2.2D Xk COEFFICIENT BITS 0-47
CPU 2.2B Xj COEFFICIENT BITS 0-47
CPU 2.1B MULT BUSY
4GA7 CI6 44-47 22,23/46,47
CI5 40-43 20,21/44,45
CI4 36-39 18,19/42,43
CI3 32-35 16,17/40,41
DI6 28-31 14,15/38,39
DI5 24-27 12,13/36,37
DI4 20-23 10,11/34,35
DI3 16-19 8,9/32,33
EI6 12-15 6,7/30,31
EI5 8-11 4,5/28,29
EI4 4-7 2,3/26,27
4GA7 EI3
Xk = 0-3
Xj = 0,1/24,25
HOLDING REGISTER AND FANOUT
(MULT 3.0)
Xk (0-47) 48
24 Xj (0-23) THEN (24-47)
BIT 70 OR BIT 94
4GH7 CI2 63-69 87-93
4GG7 CII 60-63 84-87
4GF7 CI0 58-61 82-85
4GE7 D09 56-59 80-83
D08 54-57 78-81
D07 52-55 76-79
C09 50-53 74-77
C08 48-51 72-75
C07 46-49 70-73
E12 44-47 66-71
EII 42-45 66-69
EI0 40-43 64-67
E09 38-41 62-65
E08 36-39 60-63
E07 34-37 58-61
E06 32-35 56-59
E05 30-33 54-57
E04 28-31 52-55
E03 26-29 50-53
E02 24-27 48-51
D02 22-25 46-49
DOI 20-23 44-47
D06 18-21 42-45
D05 16-19 40-43
C06 14-17 38-41
C05 12-15 36-39
4GE7 C04 10-13 34-37
4GD7 D04 8-11 32-35
4GD7 D03 6-9 30-33
4GI7 B05 0-7 24-31
HOLDING REGISTER AND SECOND LEVEL ADD (MULT 3.1)
4GJ7 B04 12-23
4GJ7 B03 0-11
HOLDING REGISTER
(MULT 3.1)
4GH7 DI2 51-57 75-81
4GG7 DII 48-51 72-75
4GF7 DI0 46-49 70-73
4GE7 FI3 44-47 68-71
FI2 42-45 66-69
FII 40-43 64-67
FI0 38-41 62-65
F09 36-39 60-63
F08 34-37 58-61
F07 32-35 56-59
F06 30-33 54-57
F05 28-31 52-55
F04 26-29 50-53
F03 24-27 48-51
4GE7 EOI 22-25 46-49
4GD7 C02 20-23 44-47
4GD7 COI 18-21 42-45
4GI7 C03 12-19 36-43
HOLDING REGISTER AND SECOND LEVEL ADD
(MULT 3.1)
PS AND PC
PSEUDO SUMS AND CARRIES
4GP7 FI5 64-70 88-94
4GQ7 FI4 59-64 83-88
SECOND AND THIRD LEVEL ADDS
4GN7 GI5 55-59 79-83
4GM7 GI4 52-55 76-79
GI3 49-52 73-76
GI2 46-49 70-73
GII 43-46 67-70
GI0 40-43 64-67
G09 37-40 61-64
G08 34-37 58-61
G07 31-34 55-58
G06 28-31 52-55
G05 25-28 49-52
G04 22-25 46-49
G03 19-22 43-46
G02 16-19 40-43
4GM7 GOI 13-16 37-40
THIRD LEVEL ADD
(MULT 3.2)
4GL7 FO2 6-13 30-37
4GK7 FOI 0-7 24-31
SECOND AND THIRD LEVEL ADDS
(MULT 3.2)
PSEUDO SUMS AND PSEUDO CARRIES
CARRIES AND ENABLES
4GQ7 HI3 88-95
4GR7 HI2 80-87
HII 72-79
HI0 64-71
H09 56-63
H08 48-55
H07 40-47
4GR7 H06 32-39
4GS7 H05 24-31
4GT7 H04 12-23
4GT7 H03 0-11
HOLDING REGISTER AND FINAL ADD FIRST HALF
(MULT 3.3 AND 3.4)
4GX7 II3 88-95 88-94
4GX7 II2 80-87 80-87
4GX7 III 72-79 72-79
4GW7 II0 64-71 64-71
4GW7 I09 56-63 56-63
4GW7 I08 48-55 48-55
4GV7 I07 40-47 40-47
4GV7 I06 32-39 32-39
4GV7 I05 24-31 24-31
4GU7 I04 12-23 12-23
4GU7 I03 0-11 -1-11
FINAL ADD SECOND HALF
LEFT SHIFT RESULT BY 1
(MULT 3.3 AND 3.4)
COEF BITS 0-47
COEF BITS 48-95
COEF BITS -1-46 SHIFT
COEF BITS 47-94 SHIFT
CPU 2.2C (X RGTR)
4GB7 AI3 44-51 68-75
AII 36-43 60-67
A09 28-35 52-59
4GB7 B08 20-27 44-51
Xi = Xj • Xk
MATRIX AND FIRST LEVEL ADD
(MULT 3.0)
4GB7 BI5 52-59 76-83
BI4 44-51 68-75
BI2 36-43 60-67
BIO 28-35 52-59
4GB7 A07 20-27 44-51
4GB7 A05 12-19 36-43
Xi = Xj • Xk
MATRIX AND FIRST LEVEL ADD
(MULT 3.0)
4GB7 BI6 56-63 80-87
AI4 48-55 72-79
AI2 40-47 64-71
AIO 32-39 56-63
A08 24-31 48-55
A06 16-23 40-47
4GB7 A04 8-15 32-39
Xi = Xj • Xk
MATRIX AND FIRST LEVEL ADD
(MULT 3.0)
4GC7 AI6 60-70 84-94
4GC7 AI5 48-58 72-82
4GB7 BI3 40-47 64-71
4GB7 BII 32-39 56-63
4GB7 B09 24-31 48-55
4GC7 B07 12-23 36-46
4GC7 B06 0-10 24-34
Xi = Xj • Xk
MATRIX AND FIRST LEVEL ADD
(MULT 3.0)
Xi (0-70) FIRST PASS
(24-94) SECOND PASS
PSEUDO SUMS AND CARRIES
CARRIES AND ENABLES
NOTES:
1. (336) PSEUDO SUMS AND CARRIES.
2. ALL MODULES SHOWN ARE LOCATED ON CHASSIS 7.
3. BOTH BIT 70 (FIRST PASS) AND BIT 94 (SECOND PASS) ARE SENT TO THE 3BA7 MODULE. BUT ONLY BIT 94 IS USED FOR THE INTEGER MULTIPLY CHECK.
CONTROL DATA
DEVELOPMENT DIVISION
SECONDARY BLOCK DIAGRAM
MULTIPLY UNIT
EQUIPMENT
SIZE C
DRAWING NO 60420300
REV Y
SHEET MULT 2.0
PAGE 5-7-3

LOGICAL PRODUCT OF MATRIX JUNCTIONS AND FIRST LEVEL ADD

INPUT REGISTERS

All instructions performed in the floating-multiply unit require 5 clock periods for execution. Data moves from the operating registers to the multiply coefficient input registers on the 4GA7 modules in the same clock period in which a floating-multiply instruction issues from the CIW register. The input registers are cleared and new data entered whenever the multiply busy flag clears. When the multiply busy flag sets, the data in the input registers is held over into the following clock period. This data is held in the input register for a total of 2 clock periods.

The multiply busy flag serves the purpose of a go multiply flag as well as blocking further entry to the unit in the clock period following issue. The multiply busy flag serves as the basic timing control that gates data into the registers of the unit at the proper time.

As the Xj and Xk coefficients enter the input registers, they are complemented, if negative, so that the multiply coefficient hardware deals only with positive numbers. The 12 4GA7 modules hold 48 Xk bits and 48 Xj bits.

During the first clock period, bits 0 through 47 of the Xk coefficient and bits 0 through 23 of the Xj coefficient enter the input registers of the 4GA7 modules. The 4GB7 and 4GC7 modules use these bits during the second clock period for forming the first half of the product. During the first clock period, bits 24 through 47 of the Xj coefficient enter bit holding registers and are held there until gated into the input registers by the multiply busy signal in the second clock period. Xj bits 0 through 23 are discarded at this time. The 4GB7 and 4GC7 modules use Xk bits 0 through 47 and Xj bits 24 through 47 during the third clock period for forming the second half of the product.

MATRIX AND FIRST LEVEL ADD

The bit-by-bit product of Xk and Xj is formed on the 24 4GB7 and 4GC7 modules during the second and third clock periods. This product is formed in two passes through a 24-bit-by-48-bit multiply matrix.

On the first pass (second clock period), the product of Xk and the lower half of the Xj coefficient (bits 0 through 23) is formed. During the second pass (third clock period), the product of Xk and the upper half of Xj (bits 24 through 47) is formed. Each pass through the matrix produces 1152 binary products, which must be added in the proper groupings to form a combined sum.

Several bits are produced that represent the same bit position of the product. (Refer to diagram for 4GC7 module located at B06.) For example, Xi product bit 9 is produced from the following combinations of operand bits.

Xk bit 2 . Xj bit 7 = Xi bit 9 Xk bit 3 . Xj bit 6 = Xi bit 9	First level add = pseudo sum bit 9 pseudo carry bit 10
Xk bit 6 . Xj bit 3 = Xi bit 9 Xk bit 7 . Xj bit 2 = Xi bit 9	First level add = pseudo sum bit 9 pseudo carry bit 10

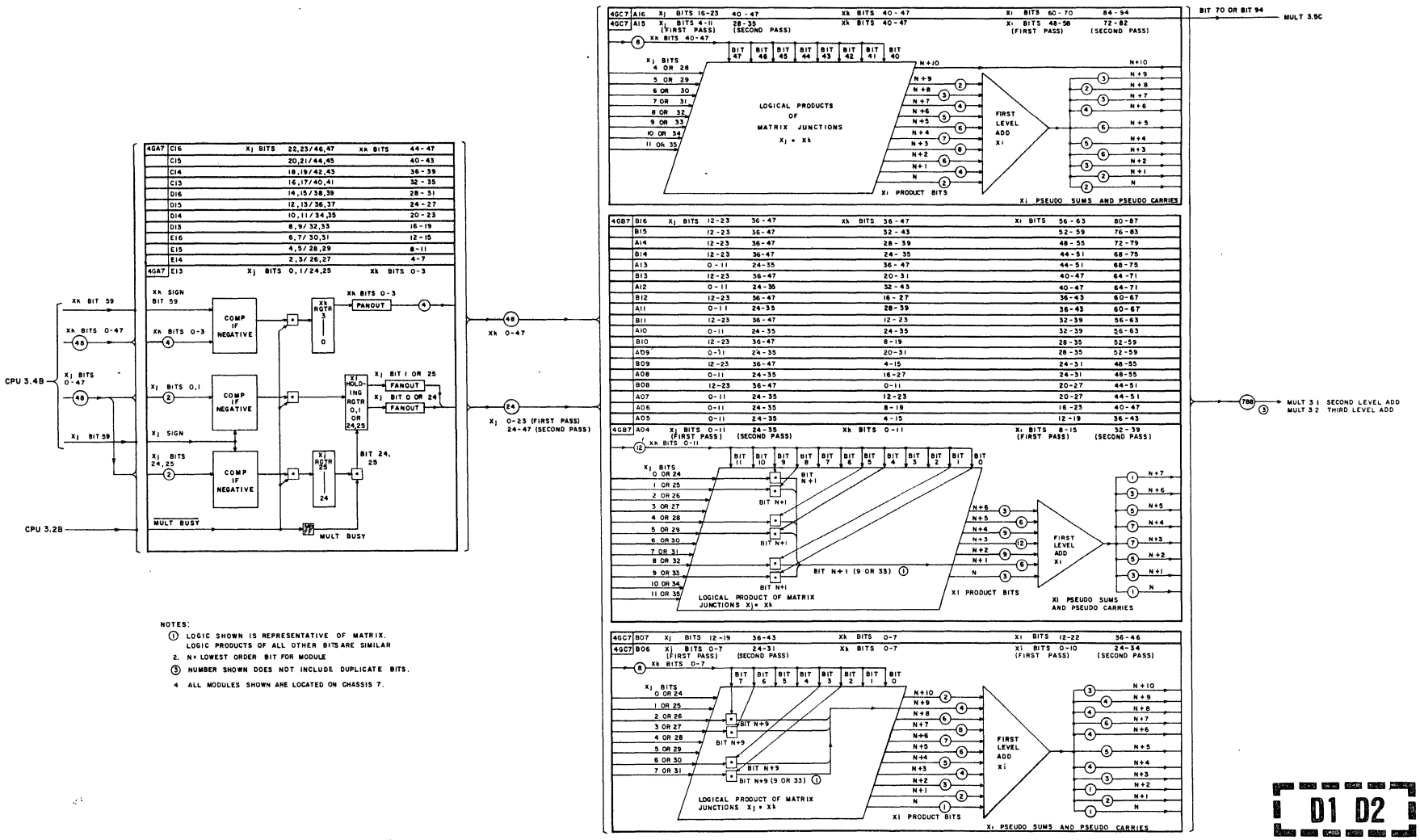
The first level add sums these product bits in groups of two or three bits to produce pseudo sum and carry bits. For example, the first level add of the top two Xi bits 9 produces a pseudo sum bit 9 and a pseudo carry bit 10. Similarly, the sum of the lower two Xi bits 9 produces a pseudo sum bit 9 and a carry bit 10. The first level add of the six Xi bits 8 produces two pseudo carry bits 9. Thus, four bits are output by the 4GC7 module for Xi bit 9: two pseduo sum bits and two carry bits.

The results of the first level add (788 bits) are sent to the 4GD7, 4GE7, 4GF7, 4GG7, 4GH7, 4GI7, 4GK7, 4GL7, 4GO7, and 4GP7 modules for a second level add. The result of the add of Xk bit 47 and Xj bit 47 (bit 94) is sent to the 3BA7 module where it is used in the integer multiply check. If bit 94 is a zero and both operands have underflow exponents, an integer multiply is performed.

MULT 3.0 TEST POINTS

Module	Location	Test Point	Description	Module	Location	Test Point	Description	Module	Location	Test Point	Description
4GA7*	7(C-E)	34	Mult Busy								
↓	13-16	04	Xk Bit N								
	↓	03	↓ N+1								
		24	↓ N+2								
		23	↓ N+3								
		41,42	Xj Bit N or N+24								
		61,62	↓ N+1 or N+25								
		54	↓ N+24								
		53	↓ N+25								
4GB7	7(A04-14, B08-16)	-	(No Signal/Rgtr TP's)								
4GC7*	7(A15-16, B06-07)	-	(No Signal/Rgtr TP's)								

D3 D4



NOTES:

1. LOGIC SHOWN IS REPRESENTATIVE OF MATRIX. LOGIC PRODUCTS OF ALL OTHER BITS ARE SIMILAR
2. N = LOWEST ORDER BIT FOR MODULE
3. NUMBER SHOWN DOES NOT INCLUDE DUPLICATE BITS.
4. ALL MODULES SHOWN ARE LOCATED ON CHASSIS 7.

D1 D2

CONTROL DATA DEVELOPMENT DIVISION	DETAILED – MODULES DIAGRAM – LOGICAL PRODUCT OF MATRIX JUNCTIONS AND FIRST LEVEL ADD	CODE IDENT 34010	DWG NO 60420300	REV N
		SIZE D	SHEET MULT 3.0	PAGE 5-7-5

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.

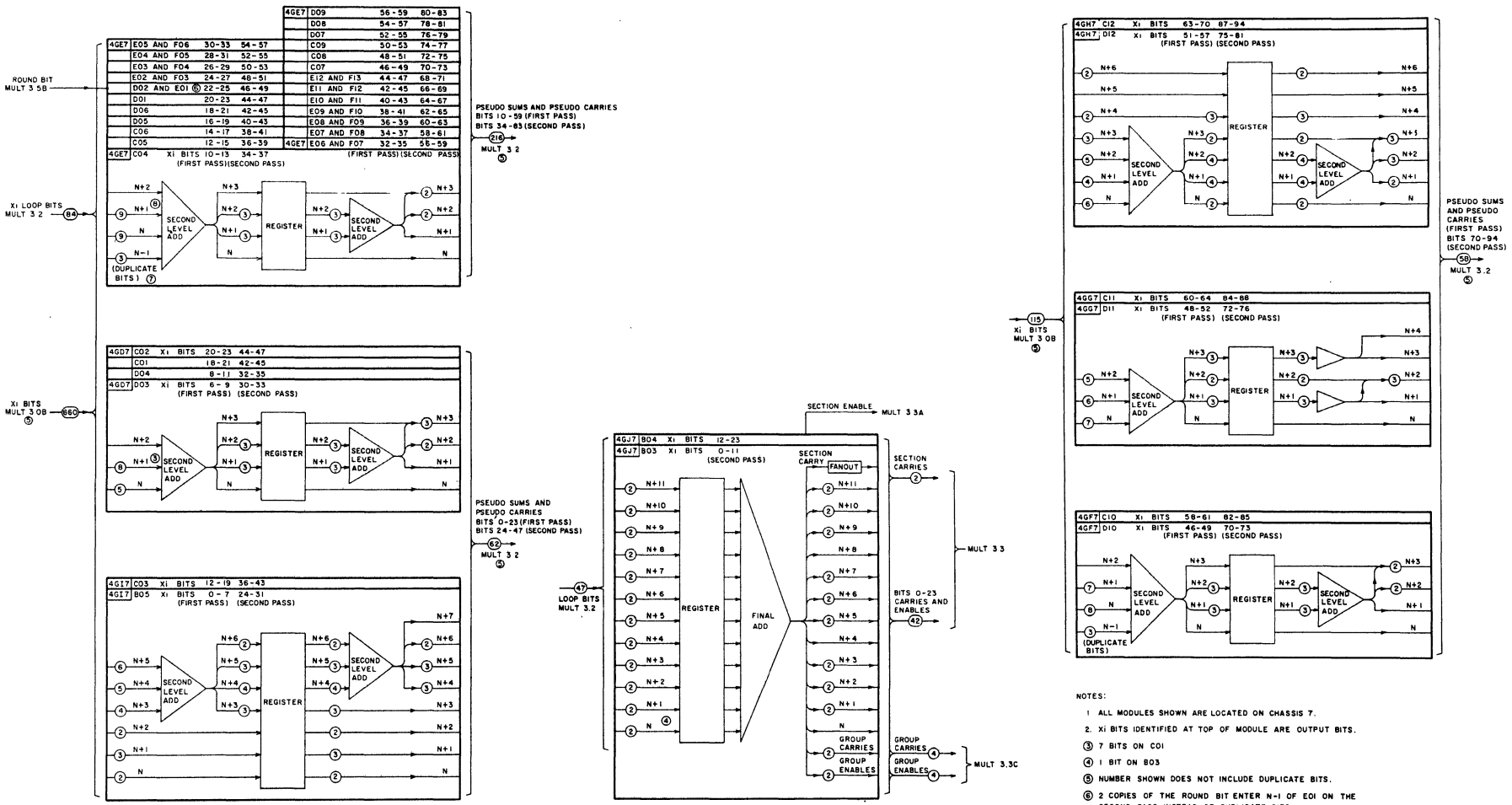
ROUNDING

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.

MULT 3.1 TEST POINTS

Module	Location	Test Point	Description
4GD7*	7(C01-02, D03-04)	25	Second level add bit N
		34	N+1
		22	N+1
		45	N+1
		32	N+2
		46	N+2
		55	N+2
		56	N+3
		01	25-nanosecond clock
4GE7*	7(C04-09, D01-02, D05-09, E01-12, F03-13)	24	Second level add bit N
		23	N+1
		34	N+1
		43	N+1
		33	N+2
		44	N+2
		53	N+2
		54	N+3
		11	25-nanosecond clock
4GF7*	7(C-D)10	24	Second level add bit N
		23	N+1
		45	N+1
		32	N+1
		46	N+2
		31	N+2
		55	N+2
		56	N+3
		11	25-nanosecond clock
4GG7*	7(C-D)11	31	Second level add bit N
		32	N+1
		45	N+1
		42	N+1
		46	N+2
		56	N+2
		52	N+3
		54	N+3
		55	N+3
		21	25-nanosecond clock
4GH7*	7(C-D)12	54	Second level add bit N
		06	N
		26	N+1
		21	N+1
		45	N+1
		05	N+1
		55	N+2
		51	N+2
		42	N+2
		44	N+2
		56	N+3
		25	N+3
		24	N+4
		32	N+4
		35	N+4
4GI7*	7(B05, C03)	03	Second level add bit N
		05	N
		01	N+1
		02	N+1
		12	N+1
		14	N+2
		15	N+2
		21	N+3
		26	N+3
		32	N+3
		24	N+4
		34	N+4
		43	N+4
		45	N+4
		64	N+5
		62	N+5
		66	N+5
		76	N+6
		73	N+6
		71, 72	25-nanosecond clock
4GJ7*	7B03-04	05	Final add bit N
		06	N
		04	N+1
		03	N+1
		15	N+2
		16	N+2
		14	N+3
		13	N+3
		25	N+4
		26	N+4
		24	N+5
		23	N+5
		35	N+6
		36	N+6
		34	N+7
		33	N+7
		42	N+8
		41	N+8
		43	N+9
		44	N+9
		52	N+10
		51	N+10
		53	N+11
		54	N+11
		75, 76	25-nanosecond clock
		46	Second level add bit N+5
		31	N+6
		34	N+6
		11, 12	25-nanosecond clock

D11 D12



NOTES:

1. ALL MODULES SHOWN ARE LOCATED ON CHASSIS 7.
2. Xi BITS IDENTIFIED AT TOP OF MODULE ARE OUTPUT BITS.
3. 7 BITS ON C01
4. 1 BIT ON B03
5. NUMBER SHOWN DOES NOT INCLUDE DUPLICATE BITS.
6. 2 COPIES OF THE ROUND BIT ENTER N-1 OF E01 ON THE SECOND PASS INSTEAD OF DUPLICATE BITS.
7. NO DUPLICATE BITS ENTER C04 AND E01.
8. 8 BITS ON E01

D9 D10

CONTROL DATA DEVELOPMENT DIVISION	DETAILED — MODULES DIAGRAM SECOND LEVEL ADD	CODE IDENT 34010	DWG NO 60420300	REV /
		SIZE D	SHEET MULT 3.1	PAGE 5-7-7

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

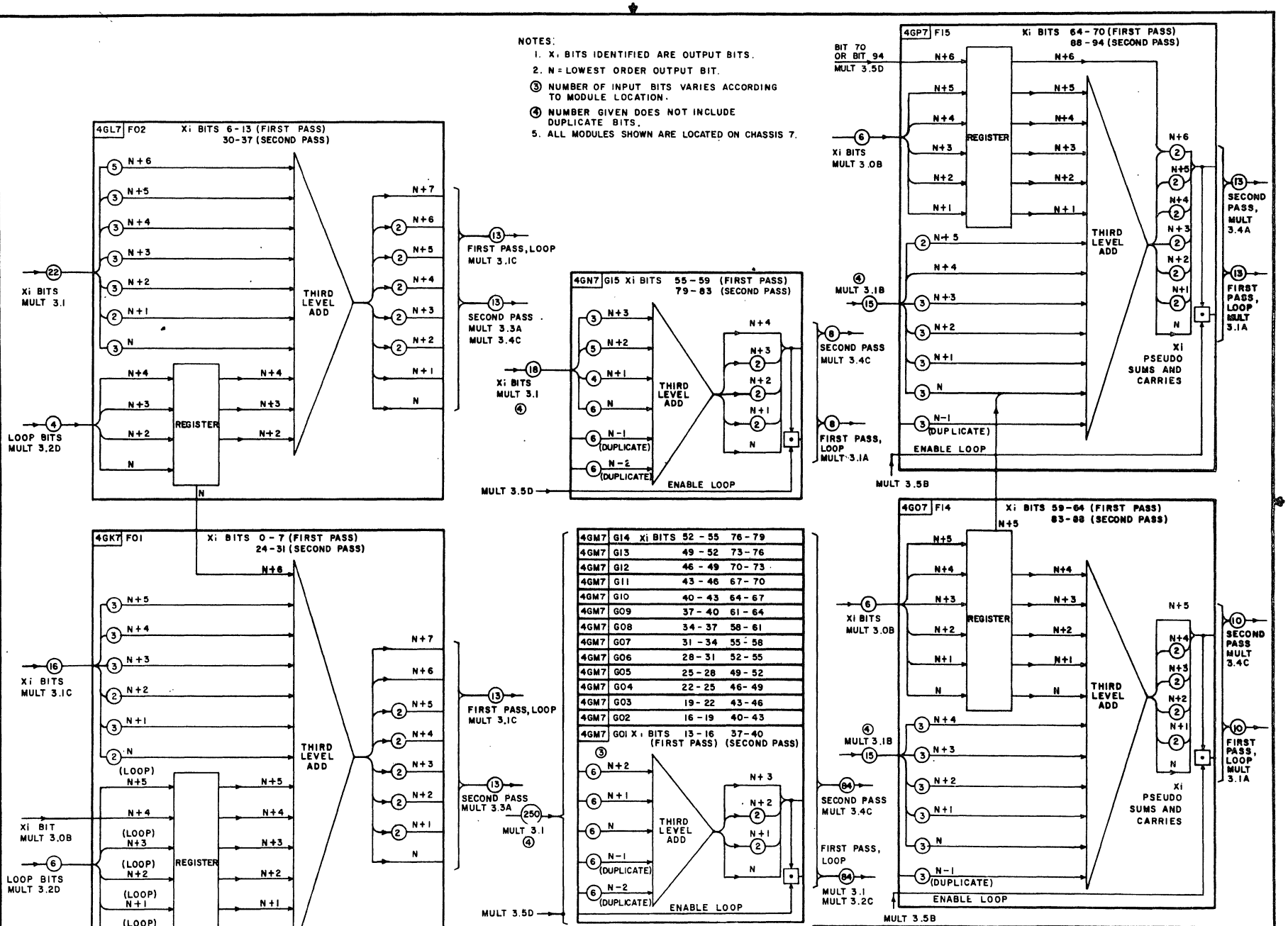
The second pass through the third level add takes place during the fourth clock period. At this time, the third level add further sums the entire product, the results of both passes through the matrix, until only two bits of data remain in each bit position of the upper part of the double-precision sum (bits 24 through 95). These 141 pseudo sum and carry bits are delivered to the 4GS7, 4GR7, and 4GQ7 modules for the final add.

MULT 3.2 TEST POINTS

Module	Location	Test Point	Description	Module	Location	Test Point	Description	Module	Location	Test Point	Description
4GK7*	7F01	02	Third level add input bit N								
		04	Third level add input bit N+1								
		06	Third level add input bit N+2								
		01	Third level add input bit N+3								
		03	Third level add input bit N+4								
		05	Third level add input bit N+5								
		11	25-nanosecond clock								
4GL7*	7F02	02	Third level add input bit N								
		06	Third level add input bit N+2								
		01	Third level add input bit N+3								
		05	Third level add input bit N+4								
		03	25-nanosecond clock								
4GM7*	7G01-14	01	Enable loop								
4GN7*	7G15	01	Enable loop								
4GO7*	7F14	01, 02	Enable loop								
		16	Third level add input bit N								
		13	Third level add input bit N+1								
		12	Third level add input bit N+2								
		26	Third level add input bit N+3								
		23	Third level add input bit N+4								
		22	Third level add input bit N+5								
		03	25-nanosecond clock								
4GP7*	7F15	11, 13	Enable loop								
		05	Third level add input bit N+1								
		02	Third level add input bit N+2								
		04	Third level add input bit N+3								
		01	Third level add input bit N+4								
		14	Third level add input bit N+5								
		16	Third level add input bit N+6								
		12	25-nanosecond clock								

E3 E4

NOTES:
1. Xi BITS IDENTIFIED ARE OUTPUT BITS.
2. N = LOWEST ORDER OUTPUT BIT.
3. NUMBER OF INPUT BITS VARIES ACCORDING TO MODULE LOCATION.
4. NUMBER GIVEN DOES NOT INCLUDE DUPLICATE BITS.
5. ALL MODULES SHOWN ARE LOCATED ON CHASSIS 7.



E1 E2

CONTROL DATA	DETAILED - MODULES DIAGRAM	EQUIPMENT
DEVELOPMENT DIVISION	THIRD LEVEL ADD	SIZE C · DRAWING NO. 60420300 · REV. K
		SHEET MULT 3.2 · PAGE 5-7-9

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.

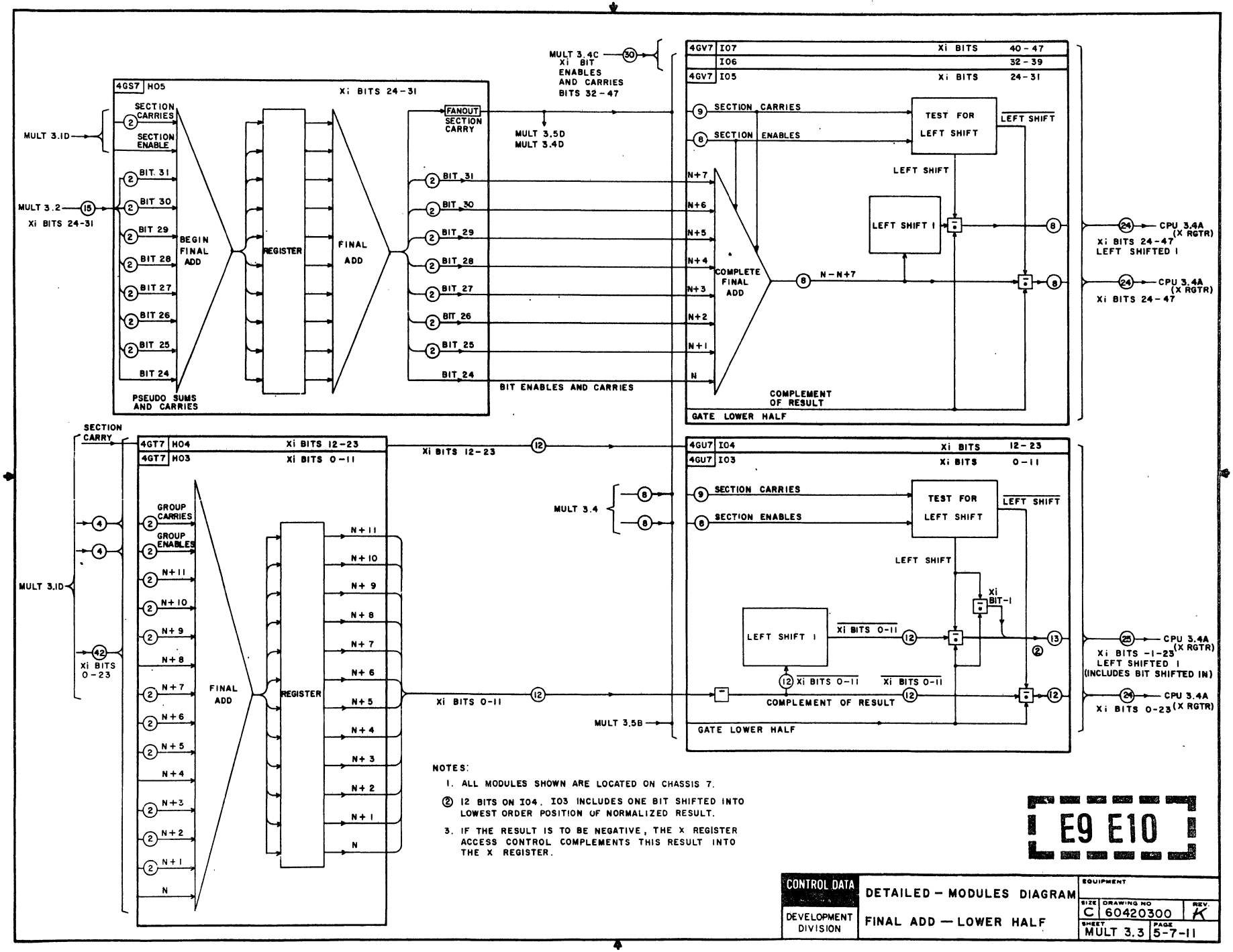
The 4GS7 module generates carries for bits 25 through 31 during the fourth clock period by performing an AND of the pseudo sum and carry bits input for each bit position. The enable and carry bits for bits 24 through 31 of the final product go to a 4GV7 module.

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.

MULT 3.3 TEST POINTS

Module	Location	Test Point	Description
4GS7*	7H05	06	Final add bit 25 carry
↓	↓	45, 46	26
		05	27
		56	28
		16	29
		66	30
		15	31
		75	32
		22	Final add bit 24 enable
		23	25
		26	26
		24	27
		32	28
		33	29
		35	30
		34	31
		25	Final add bits 26 and 27 enable
		31, 65	Final add bits 28 and 29 enable
		36, 76	Final add bits 30 and 31 enable
		54	25-nanosecond clock
4GT7*	7H03-04	01	Xi bit 0 or 12
↓	↓	03	1 or 13
		05	2 or 14
		11	3 or 15
		13	4 or 16
		15	5 or 17
		56	6 or 18
		54	7 or 19
		52	8 or 20
		66	9 or 21
		64	10 or 22
		62	11 or 23
		71	25-nanosecond clock
4GU7*	7I03-04	33-36	Gate lower half
4GV7*	7I05-07	02, 05	Gate lower half
↓	↓	15	Bit N enable

E11 E12



NOTES:

1. ALL MODULES SHOWN ARE LOCATED ON CHASSIS 7.
2. 12 BITS ON I04, I03 INCLUDES ONE BIT SHIFTED INTO LOWEST ORDER POSITION OF NORMALIZED RESULT.
3. IF THE RESULT IS TO BE NEGATIVE, THE X REGISTER ACCESS CONTROL COMPLEMENTS THIS RESULT INTO THE X REGISTER.

E9 E10

CONTROL DATA
DEVELOPMENT DIVISION

DETAILED – MODULES DIAGRAM

FINAL ADD — LOWER HALF

EQUIPMENT

SIZE C | DRAWING NO 60420300 | REV K

SHEET MULT 3.3 | PAGE 5-7-11

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 (CPU) before it enters the destination X register.

LEFT SHIFT NETWORK

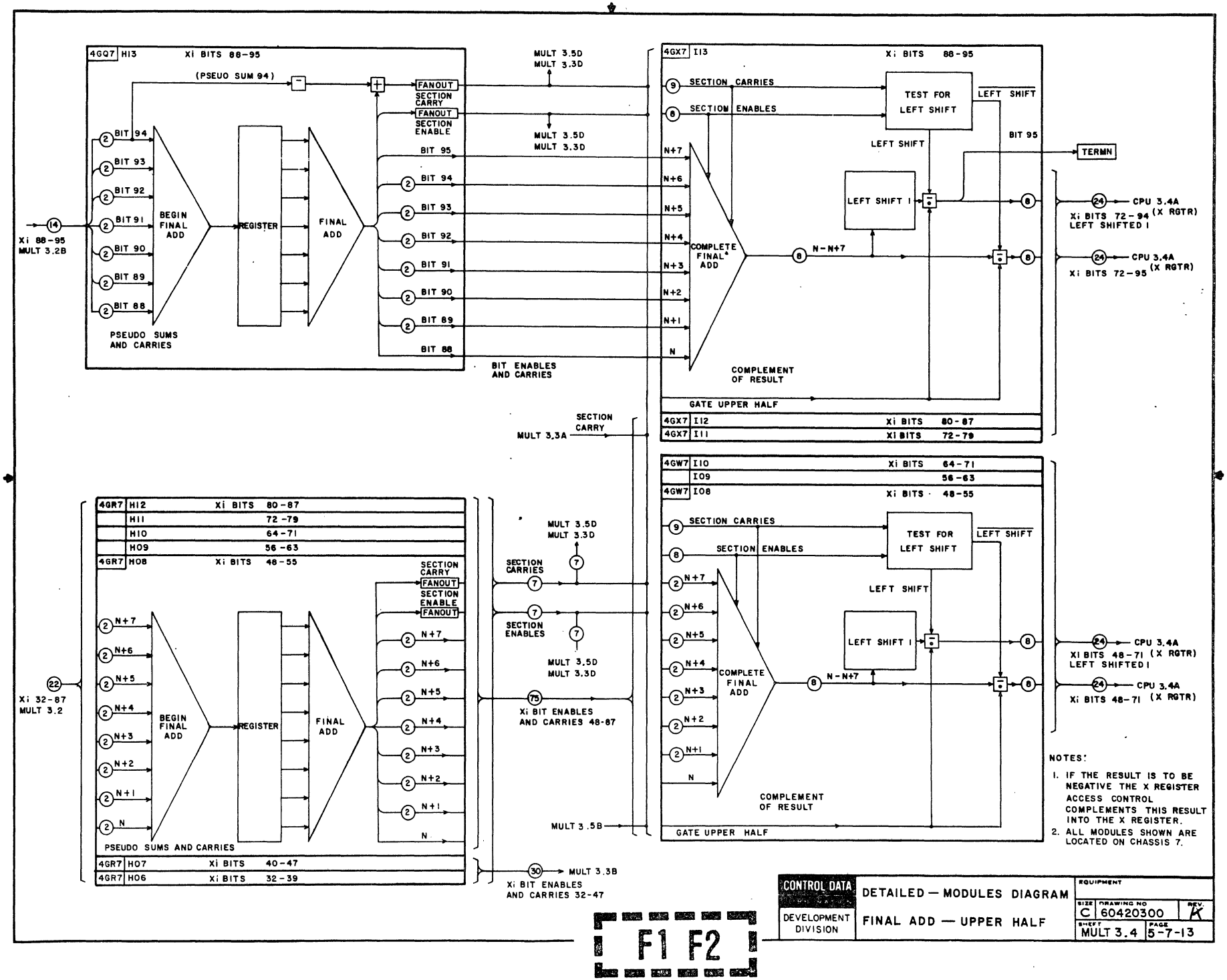
Before gating the final result, the 4GU7 and 4GV7 modules determine whether to left-shift the result by one to normalize it. If bit 95 is a one, shifting is not performed. Bit 95 is a one when the multiply process has generated a carry to it or it is forced to a one when one or both of the multiply source 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.

MULT 3.4 TEST POINTS

Module	Location	Test Point	Description
4GQ7*	7H13	06	Final add bit 89 carry
		43	90
		05	91
		56	92
		16	93
		66	94
		15, 75	95
		22	Final add bit 88 enable
		23	89
		26	90
		24	91
		32	92
		33	93
		36, 35, 76	94
		21	Final add bits 88 and 89 enable
		25, 55	Final add bits 90 and 91 enable
		31, 65	Final add bits 92 and 93 enable
		54	25-nanosecond clock
4GR7*	7N06-12	06	Final add bit N+1 carry
		42, 43	Final add bit N+2 carry
		05	Final add bit N+3 carry
		56	Final add bit N+4 carry
		16	Final add bit N+5 carry
		66	Final add bit N+6 carry
		15	Final add bit N+7 carry
		71	Final add bit N+8 carry
		22	Final add bit N enable
		23	N+1
		26	N+2
		24	N+3
		32	N+4
		33	N+5
		36	N+6
		34	N+7

Module	Location	Test Point	Description
		21	Final add bit N and N+1 enable
		25, 55	Final add bit N+2 and N+3 enable
		31, 65	Final add bit N+4 and N+5 enable
		36, 76	Final add bit N+6 and N+7 enable
		54	25-nanosecond clock
4GW7*	7I08-10	04, 05	Gate upper half
4GX7*	7I11-13	04, 05	Gate upper half

Module	Location	Test Point	Description


4GQ7 HI3 Xi BITS 88-95
(PSEUO SUM 94)
FANOUT SECTION CARRY
FANOUT SECTION ENABLE
BIT 94
BIT 93
BIT 92
BIT 91
BIT 90
BIT 89
BIT 88
BEGIN FINAL ADD
REGISTER
FINAL ADD
PSEUDO SUMS AND CARRIES
Xi 88-95 MULT 3.2B
MULT 3.5D MULT 3.3D
BIT ENABLES AND CARRIES
4GX7 II3 Xi BITS 88-95
SECTION CARRIES
SECTION ENABLES
TEST FOR LEFT SHIFT
LEFT SHIFT
BIT 95
TERMN
LEFT SHIFT 1
COMPLETE FINAL ADD
N-N+7
COMPLEMENT OF RESULT
GATE UPPER HALF
CPU 3.4A Xi BITS 72-94 (X RGTR) LEFT SHIFTED 1
CPU 3.4A Xi BITS 72-95 (X RGTR)
4GX7 II2 Xi BITS 80-87
4GX7 II1 Xi BITS 72-79
MULT 3.3A
SECTION CARRY
4GR7 HI2 Xi BITS 80-87
HII 72-79
HI0 64-71
HO9 56-63
4GR7 HO8 Xi BITS 48-55
Xi 32-87 MULT 3.2
MULT 3.5B
Xi BIT ENABLES AND CARRIES 48-87
4GR7 HO7 Xi BITS 40-47
4GR7 HO6 Xi BITS 32-39
MULT 3.3B
Xi BIT ENABLES AND CARRIES 32-47
4GW7 II0 Xi BITS 64-71
II09 56-63
4GW7 IO8 Xi BITS 48-55
CPU 3.4A Xi BITS 48-71 (X RGTR) LEFT SHIFTED 1
CPU 3.4A Xi BITS 48-71 (X RGTR)
NOTES:
1. IF THE RESULT IS TO BE NEGATIVE THE X REGISTER ACCESS CONTROL COMPLEMENTS THIS RESULT INTO THE X REGISTER.
2. ALL MODULES SHOWN ARE LOCATED ON CHASSIS 7.
CONTROL DATA DEVELOPMENT DIVISION
DETAILED — MODULES DIAGRAM
FINAL ADD — UPPER HALF
EQUIPMENT
SIZE C DRAWING NO 60420300 REV. A
SHEET MULT 3.4 PAGE 5-7-13
F1 F2

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 precison 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.

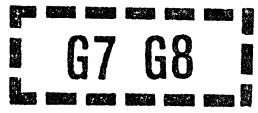
The unpacked exponent of the result (sensed prior to reducing it by one when a left-shift is performed) is less than -1776 (octal).

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-precison 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.

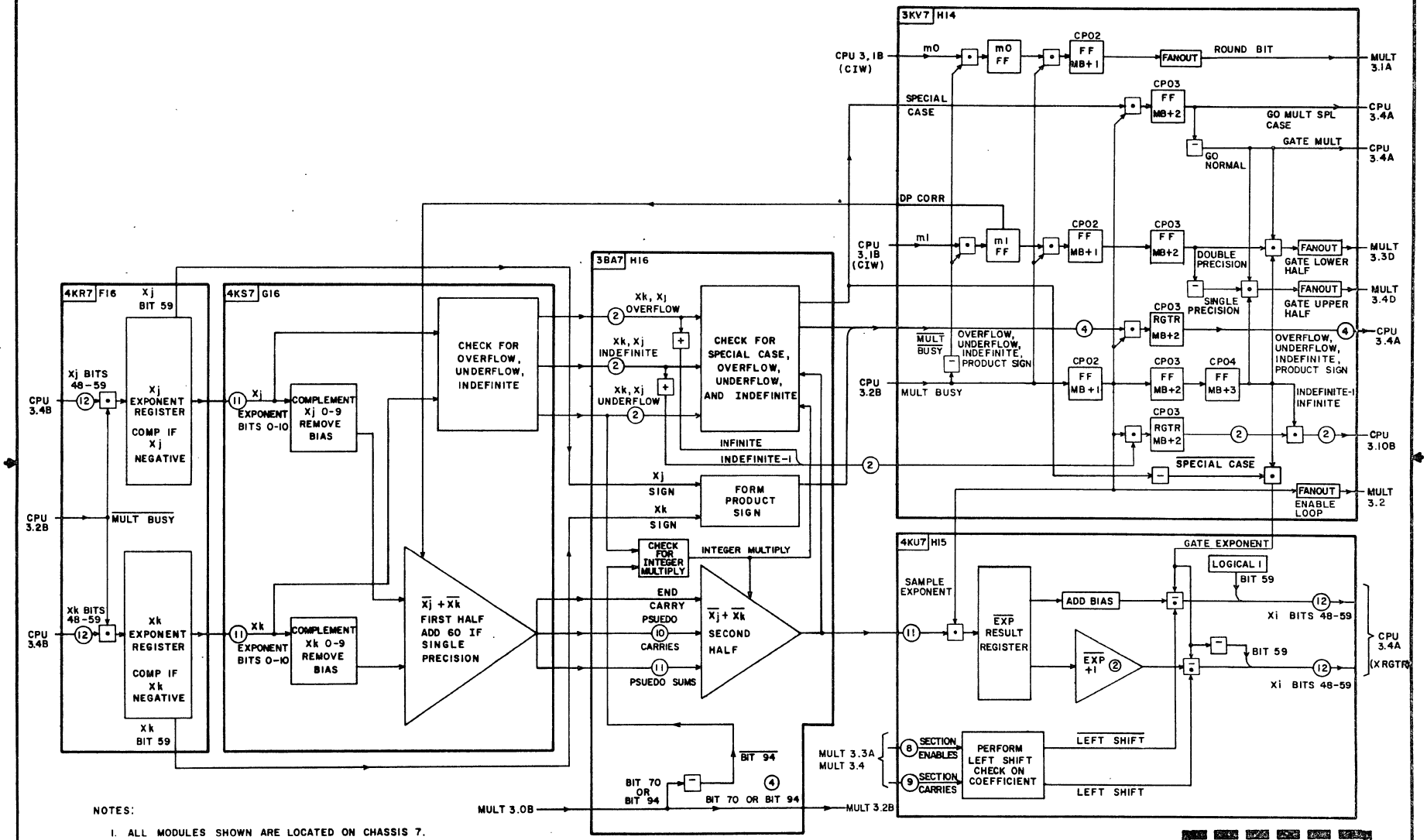
MULT 3.5 TEST POINTS

Module	Location	Test Point	Description
3BA7*	7H16		(No signal/rgtr TPs)
4KR7*	7F16	33, 34	Mult busy
		42	Xk exponent rgtr bit 48
		45	49
		43	50
		44	51
		52	52
		54	53
		55	54
		56	55
		62	56
		64	57
		65	58
		66	59
		01	Xj exponent rgtr bit 48
		03	49
		05	50
		06	51
		15	52
		13	53
		12	54
		11	55
		22	56
		25	57
		23	58
		26	59
		73, 74	25-nanosecond clock
4KS7*	7G16		(No signal/rgtr TPs)
		21	Gate exponent
4KU7*	7H15	25	Sample exponent
		02	Exponent bit 0
		03	1
		06	2
		05	3
		14	4
		16	5
		11	6
		12	7
		24	8
		22	9
		23	10
		26	25-nanosecond clock
3KV7*	7H14	35	m0 FF
		36	m0 FF (MB+1)
		33	m1 FF
		34	m1 FF (MB+1)
		32	m1 FF (MB+2)
		13	Special case FF
		12	Go mult special case
		25	Overflow FF
		22	Underflow FF
		24	Indefinite FF
		21	Product sign FF
		11	Gate mult
		02	Mult busy FF (MB+1)
		03	Mult busy FF (MB+2)
		05	Mult busy FF (MB+3)
		64	Gate upper half
		74	Gate lower half
		54, 44	Enable loop
		15, 16	25-nanosecond clock

Module	Location	Test Point	Description

Module	Location	Test Point	Description

G3 G4



NOTES:

1. ALL MODULES SHOWN ARE LOCATED ON CHASSIS 7.
2. THE NET RESULT IS TO SUBTRACT ONE FROM THE EXPONENT.
3. AT THE X REGISTER THE $\overline{\text{GATE EXP}}$ SIGNAL IS BIT 59 FOR BOTH EXP AND EXP-1.
4. BOTH BIT 70 (FIRST PASS) AND BIT 94 (SECOND PASS) ARE SENT TO THE 3BA7 MODULE, BUT ONLY BIT 94 IS USED FOR THE INTEGER MULTIPLY CHECK.

G1 G2

CONTROL DATA
DEVELOPMENT DIVISION

DETAILED — MODULES DIAGRAM
EXPONENT AND CONTROL

EQUIPMENT

SIZE	DRAWING NO.	REV.
C	60420300	N

SHEET	PAGE
MULT 3.5	5-7-15

The 3BA7 module forms the second half of the exponent add and examines the complement of the result for overflow or underflow of the floating-point range. The 3BA7 module also determines whether an integer multiply should be performed.

The overflow and underflow tests are made primarily by examining bits 10, 11, and 12 of the result. Since the result is in complement form at this point of the add operation, bit 12 (term H12) set indicates a positive exponent result, and term 7H set indicates a negative exponent result. Bit 11 may be set for a positive result, but should never be set when the result is negative. The value of bit 10 depends upon whether the exponent result is negative or postive.

OVERFLOW

When bits 10 and 11 of a positive result are set (terms H10 and H11), the result is $\leq +$ 1777. Term 7I indicates the complement of this condition or that the result is $> +$ 1777. Therefore, an overflow condition (term 7I set) occurs when the result is $> +$ 1777. The overflow signal is sent to the 3KV7 module if an indefinite condition does not already exist.

UNDERFLOW

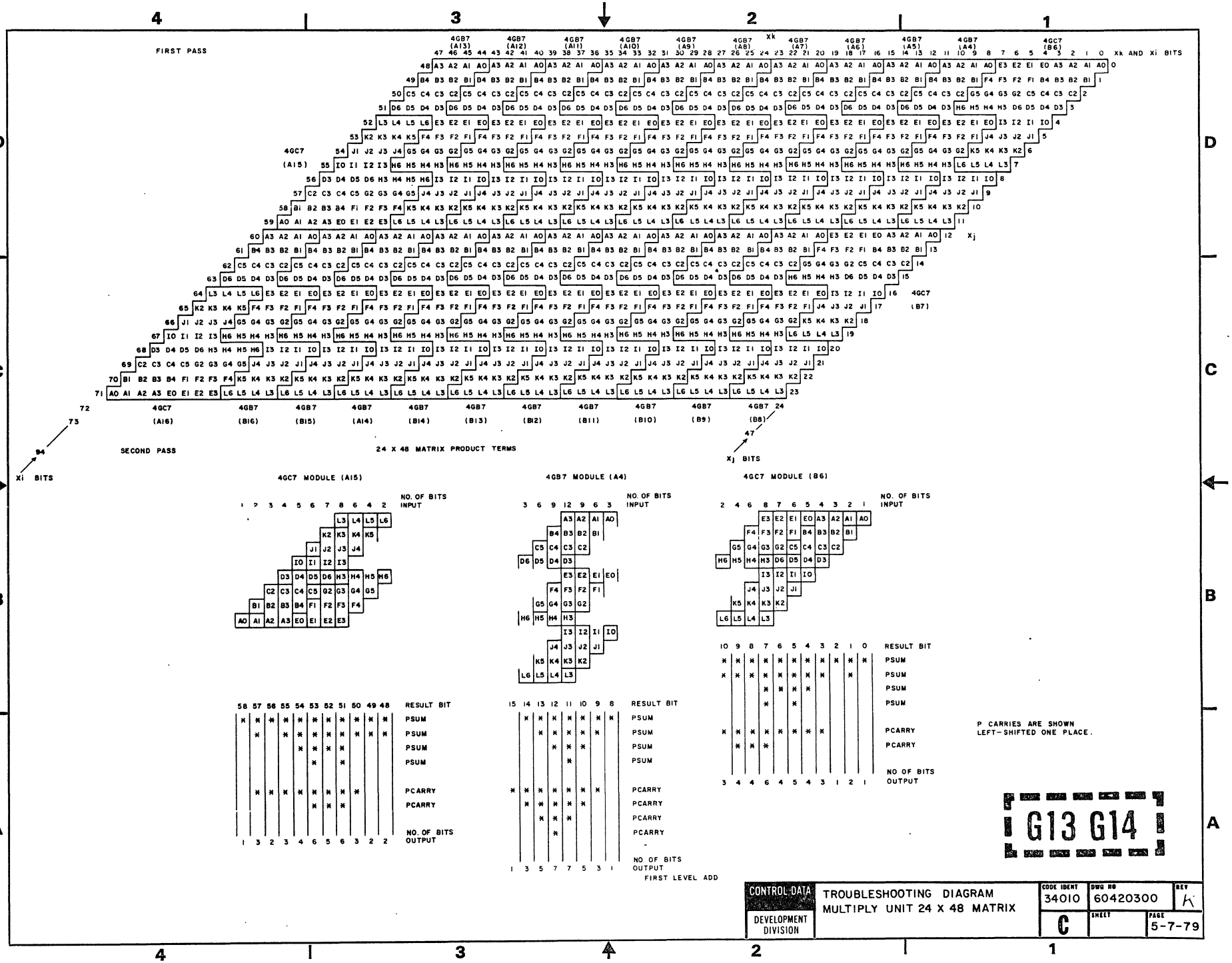
When the result is negative (term 7H set), two possible overflow conditions are examined. Bit 10 (term H10) set indicates that overflow has occurred because the exponent result is $\leq -$ 2000. Terms JA, JB, and JC set indicate that the result is equal to - 1777. If either of these conditions exists, an underflow condition (term UF) is detected. The underflow signal is sent to the 3KV7 module if an indefinite condition does not already exist.

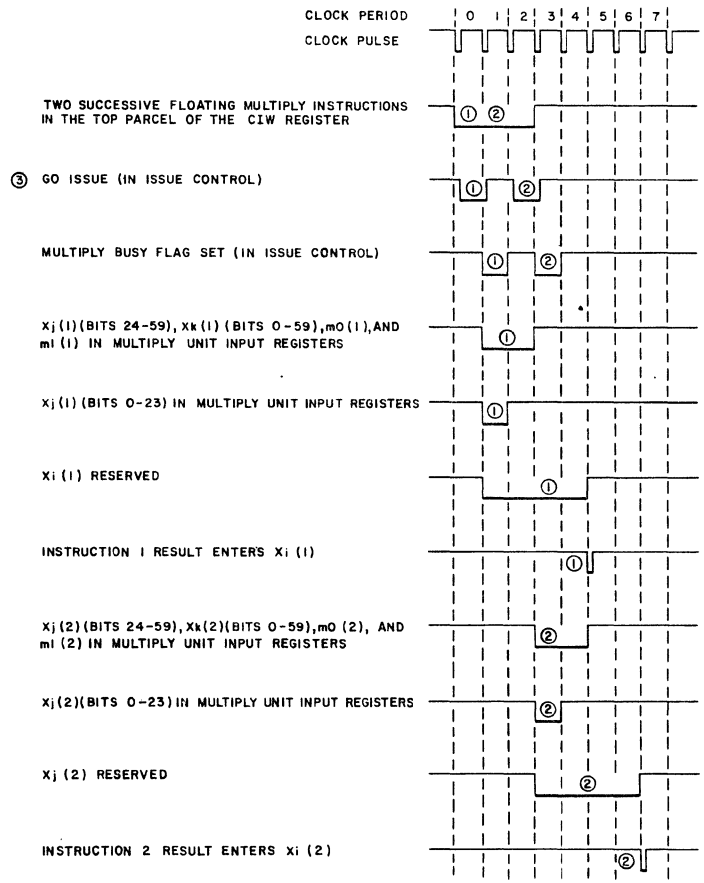
INTEGER MULTIPLY

When both operands have underflow exponents (terms KU and JU are ones) and the operand coefficients are not both normalized (term BI is one), an integer multiply is performed. However, if one of the operand coefficients is normalized, an undetected overflow result normally occurs.

When an integer multiply is performed, exponent adder output bits 0, 4, and 5 are forced to one by term IM, and the bias bit (bit 10) is forced to a zero. This ensures that all zero bits are sent to the Xi register for the exponent portion of the integer result.

During an integer multiply operation, an underflow condition exists, but the special case condition is blocked by term NIM. Blocking special case to multiply control (3KV7 module) enables the multiply unit to output to the Xi register even though the underflow condition exists.


FIRST PASS
SECOND PASS
24 X 48 MATRIX PRODUCT TERMS
Xk AND Xi BITS
Xj
Xi BITS
Xj BITS
4GC7 MODULE (A15)
4GB7 MODULE (A4)
4GC7 MODULE (B6)
NO. OF BITS INPUT
RESULT BIT
PSUM
PCARRY
NO. OF BITS OUTPUT
FIRST LEVEL ADD
P CARRIES ARE SHOWN LEFT-SHIFTED ONE PLACE.
G13 G14
CONTROL DATA
DEVELOPMENT DIVISION
TROUBLESHOOTING DIAGRAM
MULTIPLY UNIT 24 X 48 MATRIX
CODE IDENT 34010
DWG NO 60420300
REV K
SHEET C
PAGE 5-7-79



NOTES:

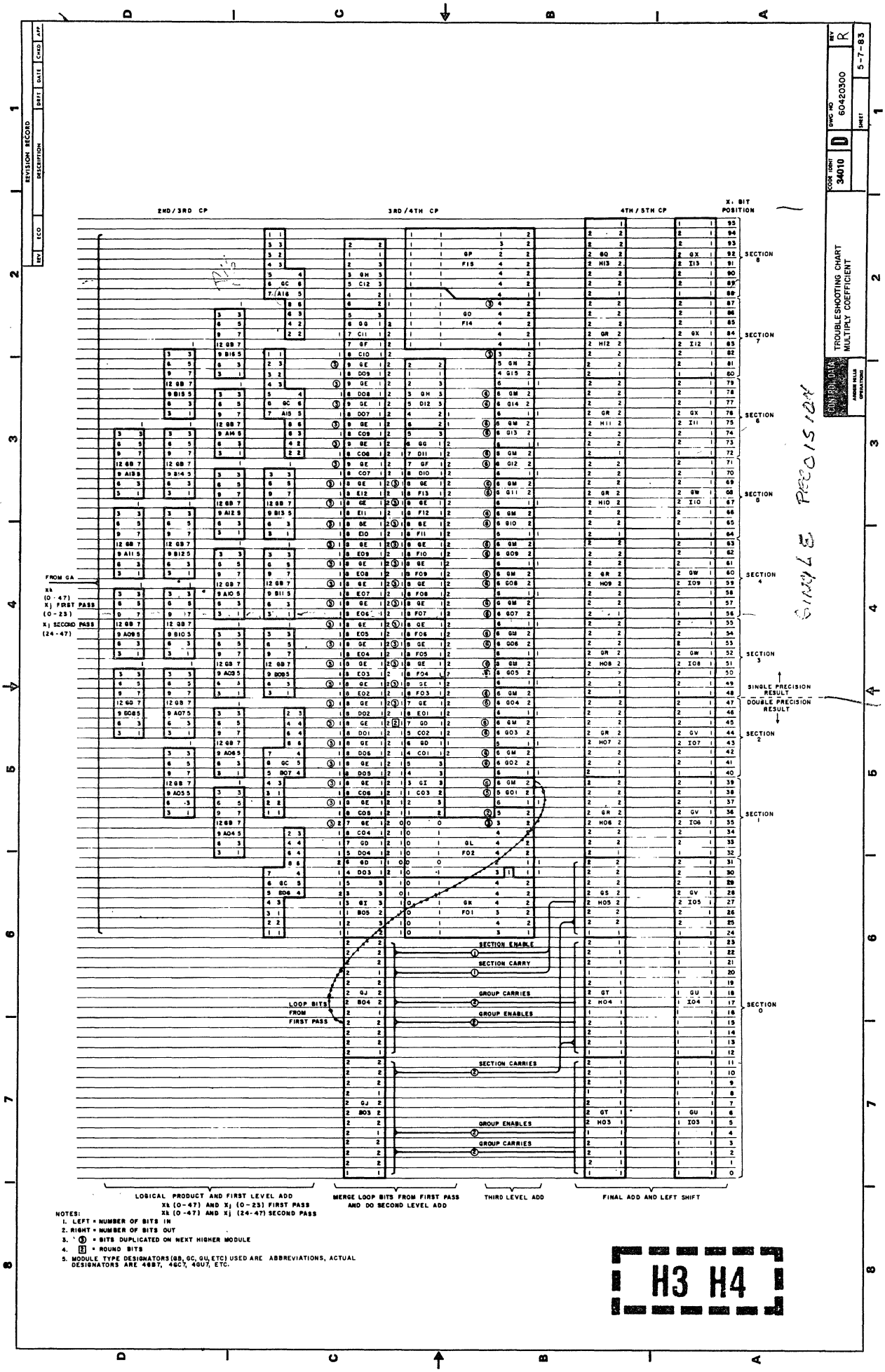
① INSTRUCTION 1 - ASSUME NO ISSUE CONFLICTS.

② INSTRUCTION 2 - ASSUME THAT ONLY ISSUE CONFLICT IS THAT THE MULTIPLY UNIT IS BUSY. THUS, ISSUE IS DELAYED ONE CLOCK PERIOD.

③ ENABLE ISSUE IS DELATED 4 CLOCK PERIODS (THEREFORE, GO ISSUE IS DELAYED A MINIMUM OF 4 CLOCK PERIODS) IN THE AA120-C. THIS DELAY IS REMOVED BY INSTALLING OPTION AT364-A.

H1 H2

CONTROL DATA DEVELOPMENT DIVISION	TIMING DIAGRAM MULTIPLY INSTRUCTION	CODE IDENT 34010	DWG NO 60420300	REV R
		C	SHEET	PAGE 5-7-81



NOTES:
1. LEFT = NUMBER OF BITS IN
2. RIGHT = NUMBER OF BITS OUT
3. (1) = BITS DUPLICATED ON NEXT HIGHER MODULE
4. [2] = ROUND BITS
5. MODULE TYPE DESIGNATORS (GB, GC, GU, ETC) USED ARE ABBREVIATIONS, ACTUAL DESIGNATORS ARE 46B7, 46C7, 40U7, ETC.

REVISION RECORD

REV	ECO	DESCRIPTION	SHT	DATE	CHKD	APP

CODE IDENT 34010 | SIZE D | DWG NO 60420300 | REV R | SHEET | 5-7-83

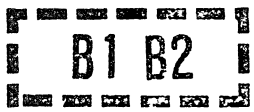
CONTROL DATA ARBOR HILLS OPERATIONS

TROUBLESHOOTING CHART
MULTIPLY COEFFICIENT

H3 H4

PART 8

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 Xj by the divisor Xk and send the quotient to Xi. 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 Xj and Xk. 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 Xj and Xk 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 Xj to Xk 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 Xj and the divisor Xk are normalized, the quotient Xi 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 Xj to Xk 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: Xi = Xj/Xk
(Xj) = 2057 4400000000000015
(Xk) = 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:

(Xi) = 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: Xi = Xj/Xk
(Xj) = 2016 7000000000000000
(Xk) = 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:

(Xi) = 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:

```
                      15 0000 0000 0000 000
                      14 0000 0000 0000 000
                       1 0000 0000 0000 0000
                         6000 0000 0000 0000
```

Dividend (Xj) = 4400 0000 0000 0015

Divisor (Xk) = 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
```

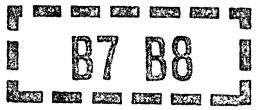
The dividend enters the remainder register.

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 ≤ 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

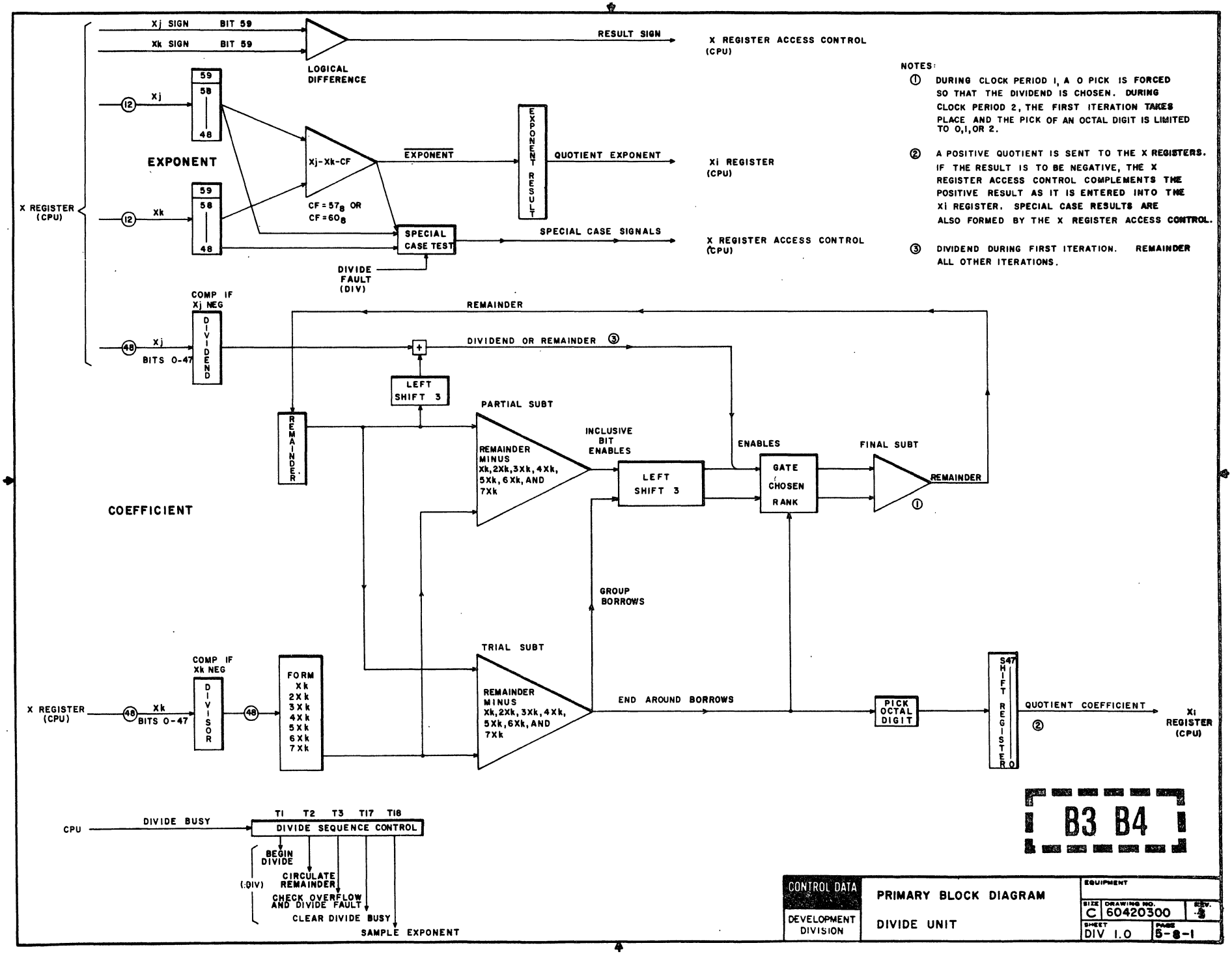
16. Compare RMDR versus Xk through 7Xk: $2Xk \leq RMDR < 3Xk$

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: $Xk \leq RMDR < 2Xk$

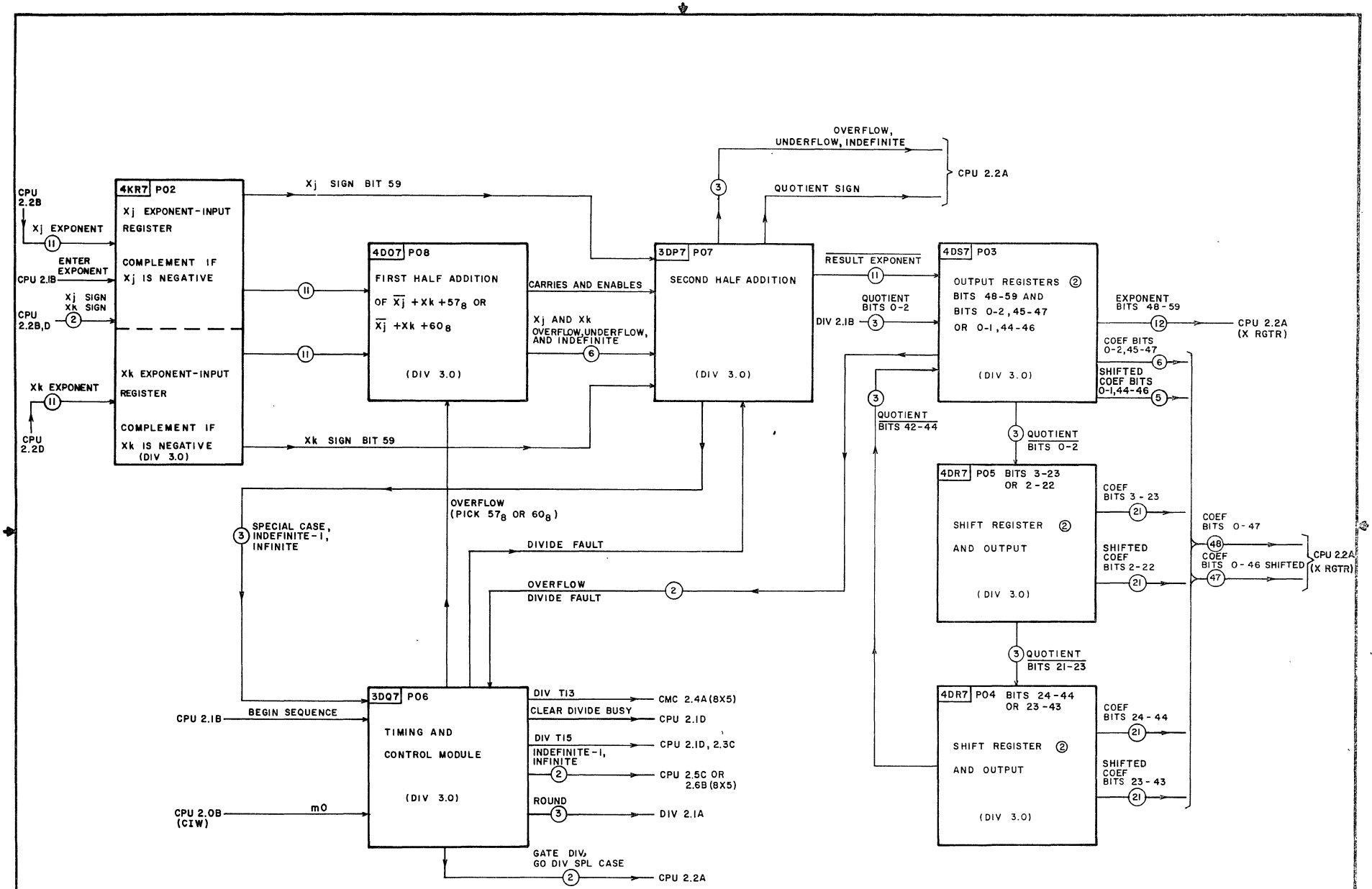
Enter octal digit 1: QUOT = 6000 0000 0000 0021
Discard remainder and transmit quotient to Xi

B9 B10



B3 B4

CONTROL DATA DEVELOPMENT DIVISION	PRIMARY BLOCK DIAGRAM DIVIDE UNIT	EQUIPMENT	
		SIZE C	DRAWING NO. 60420300
		SHEET DIV 1.0	PAGE 5-8-1



NOTES:

1. ALL MODULES SHOWN ARE LOCATED ON CHASSIS 7.

② OUTPUT AND SHIFT RECISTERS HOLD THE COMPLEMENT OF THE RESULT.

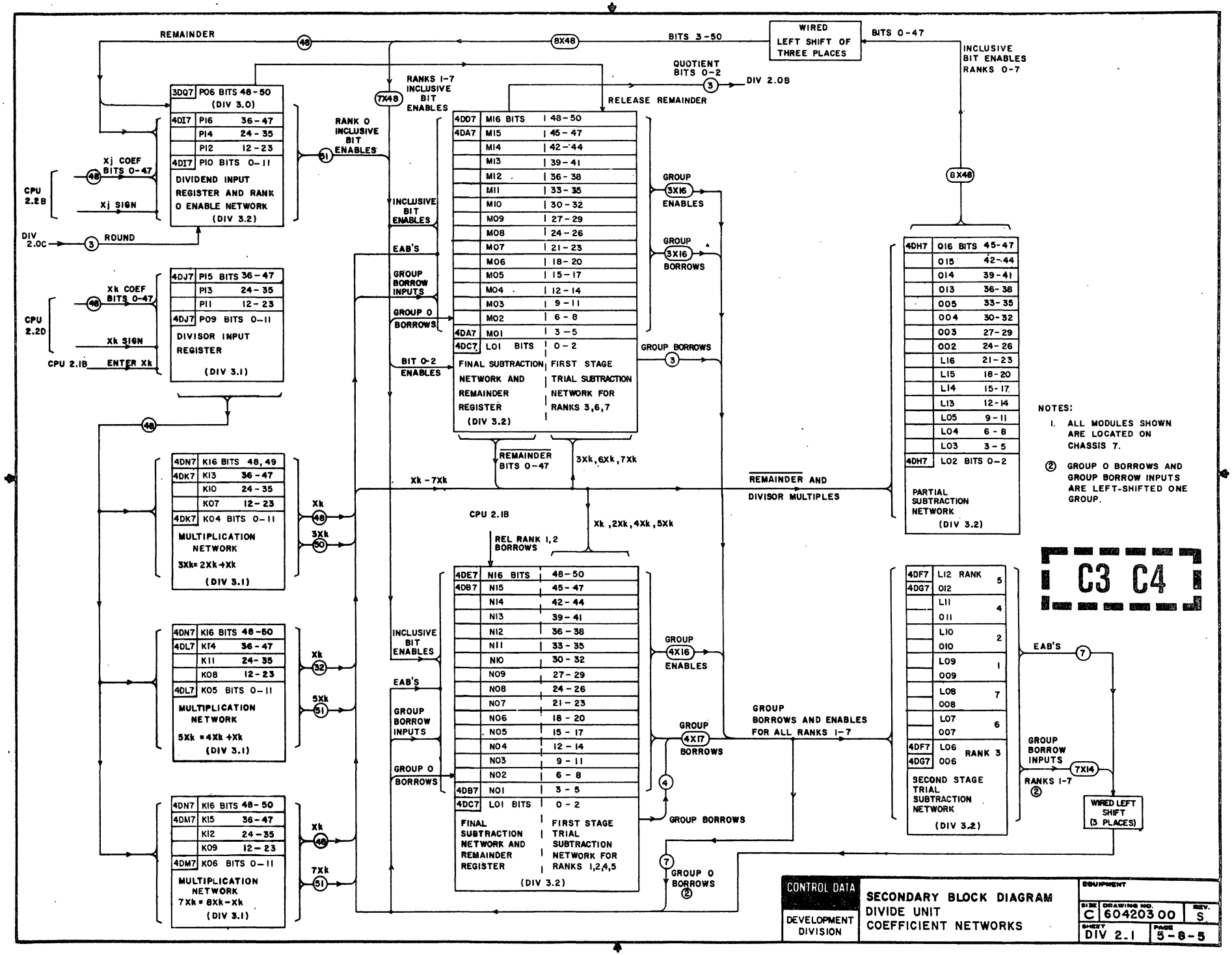
CONTROL DATA
DEVELOPMENT DIVISION

SECONDARY BLOCK DIAGRAM
DIVIDE UNIT
EXPONENT AND CONTROL NETWORKS

EQUIPMENT

SIZE C | DRAWING NO 60420500 | REV Y

SHEET DIV 2.0 | PAGE 5-8-3



CONTROL DATA	SECONDARY BLOCK DIAGRAM	EQUIPMENT	
DEVELOPMENT DIVISION	DIVIDE UNIT	SIZE C	DRAWING NO. 60420300
	COEFFICIENT NETWORKS	REV. S	
		SHEET DIV 2.1	PAGE 5-8-5

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 ($\overline{\text{exponent}}$) is complemented on output by the 4DS7 module to obtain the quotient exponent. In effect, the exponent network forms minus (-Xj + Xk + CF) instead of (Xj - Xk - 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.

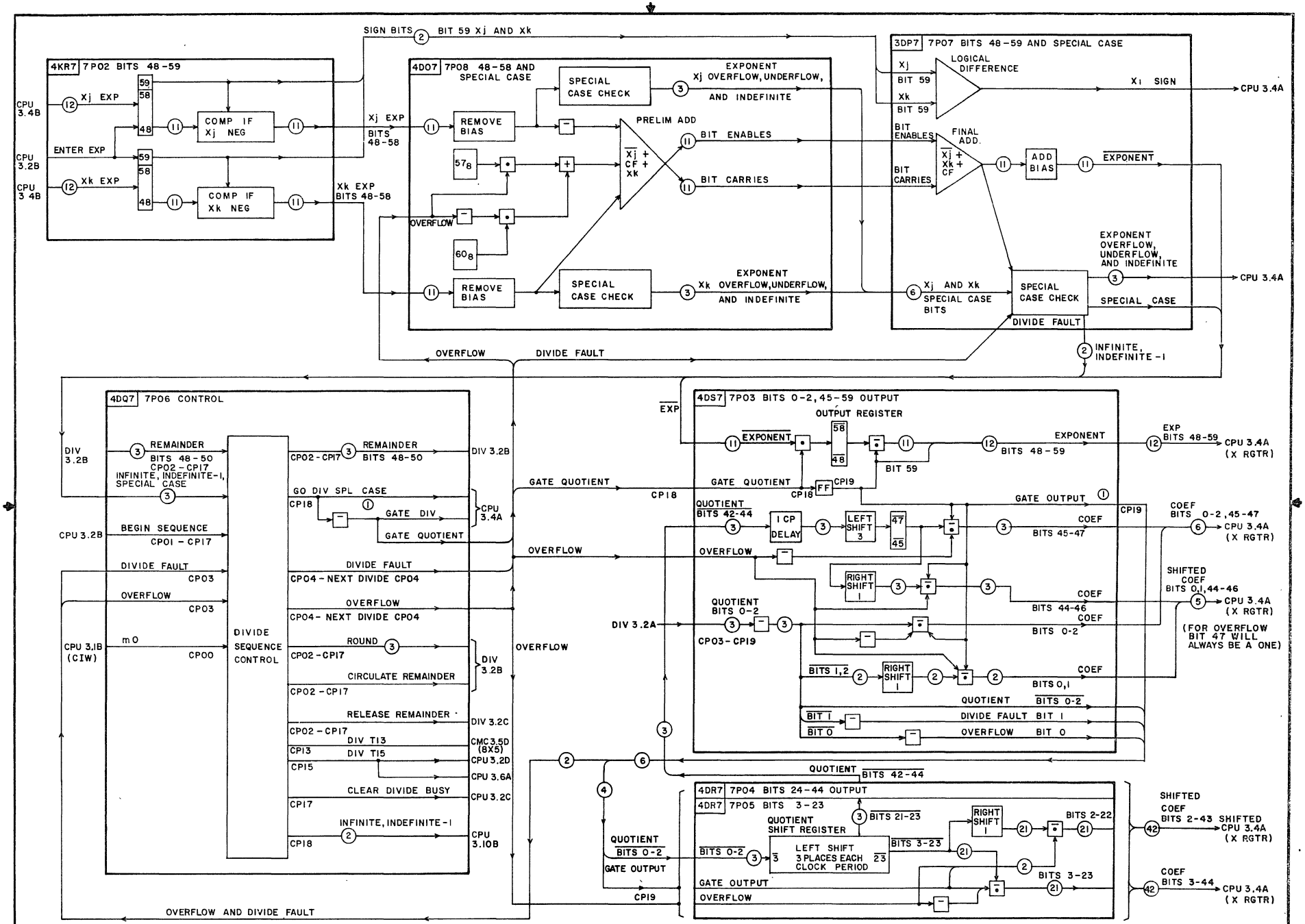
DIV 3.0 TEST POINTS

Module	Location	Test Point	Description
4DO7*	7P08		(No signal/rgtr TPs)
3DP7*	7P07		(No signal/rgtr TPs)
4DQ7*	7P06	41	Quotient overflow
		42	Divide fault
		06	Release remainder
		65	Clear div busy (true)
		71, 73	Circulate remainder
		72, 74	25-nanosecond clock
4DR7*	7P04-05	02	Result shift rgtr bit 3 or 24
		04	Result shift rgtr bit 4 or 25
		06	Result shift rgtr bit 5 or 26
		12	Result shift rgtr bit 6 or 27
		14	Result shift rgtr bit 7 or 28
		16	Result shift rgtr bit 8 or 29
		22	Result shift rgtr bit 9 or 30
		24	Result shift rgtr bit 10 or 31
		26	Result shift rgtr bit 11 or 32
		32	Result shift rgtr bit 12 or 33
		34	Result shift rgtr bit 13 or 34
		36	Result shift rgtr bit 14 or 35
		41	Result shift rgtr bit 15 or 36
		43	Result shift rgtr bit 16 or 37
		45	Result shift rgtr bit 17 or 38
		51	Result shift rgtr bit 18 or 39
		53	Result shift rgtr bit 19 or 40
		55	Result shift rgtr bit 20 or 41
		61	Result shift rgtr bit 21 or 42
		63	Result shift rgtr bit 22 or 43
		65	Result shift rgtr bit 23 or 44
		73, 74	25-nanosecond clock

Module	Location	Test Point	Description
4DS7*	7P03	26, 73	Gate quotient
		41	Pick number (quotient) bit 0
		42	Pick number (quotient) bit 1
		43	Pick number (quotient) bit 2
		36	Quotient bit 45
		34	46
		35	47
			Exponent bit 0
			1
			2
			3
			4
			5
			6
			7
			8
			9
			10
			25-nanosecond clock
4KR7	7P02	33, 34	Enter exponent
		42	Xk exponent bit 48
		45	49
		43	50
		44	51
		52	52
		54	53
		55	54
		56	55
		62	56
		64	57
		65	58
		66	59
		01	Xj exponent bit 48
		03	49
		05	50
		06	51
		15	52
		13	53
		12	54
		11	55
		22	56
		25	57
		23	58
		26	59
		73, 74	25-nanosecond clock

Module	Location	Test Point	Description

D3 D4



NOTES:

① IN CASE OF SPECIAL EXIT, GATE QUOTIENT AND GATE OUTPUT WILL NOT BE GENERATED, X REGISTER ACCESS CONTROL WILL GENERATE SPECIAL CASE RESULT IN Xi.

CONTROL DATA
DEVELOPMENT DIVISION

DETAILED – MODULES DIAGRAM
EXPONENT, OUTPUT, AND CONTROL

EQUIPMENT
SIZE C | DRAWING NO. 60420300 | REV. Y
SHEET DIV 3.0 | PAGE 5-8-7

The divisor (Xk) 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 Xk control signal is a one. When the divide instruction issues in CP00, enter Xk 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.

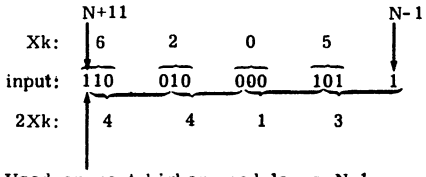
The divide unit uses the coefficient magnitude of the operands in executing the divide instructions. The coefficient is complemented (4DJ7 module) if it is negative. If bit 5 is a one, the operand is negative.

The Xk coefficient magnitude is sent to the multiplication network. The 4DK7, 4DL7, 4DM7, and 4DN7 modules form the seven multiples of the divisor coefficient from Xk to 7Xk.

3Xk ADDER

A network on the 4DK7 and 4DN7 modules creates a 50-bit quantity 3Xk and shifts this quantity to form 6Xk. Bits 0 through 47 of 3Xk 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 Xk from the divisor input registers on the 4DJ7 modules. These bits are used as both Xk and 2Xk. 2Xk is formed by left-shifting Xk one bit position. The 13 Xk bits received are bits N-1 through N+11. Of these bits, N-1 through N+10 are used as 2Xk 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 3Xk by adding bits 0 through 47 of 2Xk to bits 0 through 47 of Xk. 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 Xk bit 47, which left shifted one place is 2Xk bit 48. The sum of this bit and the section 4 carry bit produces bits 48 and 49 of 3Xk.

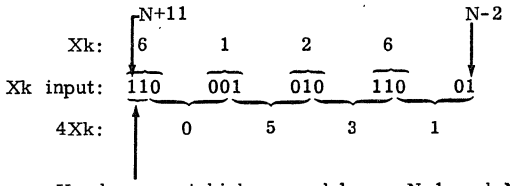
The other three section carries are fed back into the final add network on the 4DK7 modules to produce 3Xk bits 0 through 47.

6Xk NETWORK

6Xk is formed from 3Xk 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 grop. Therefore, each module needs three bits of 3Xk and three bits of 6Xk. The multiplication network sends four 3Xk bits to each module. These bits are labeled bits N-1 through N+2. Bits N-1 through N+1 of 3Xk are used by the 4DA7 and 4DH7 modules as bits N through N+2 of 6Xk. This is equivalent to left-shifting one bit or multiplying by two.

5Xk ADDER

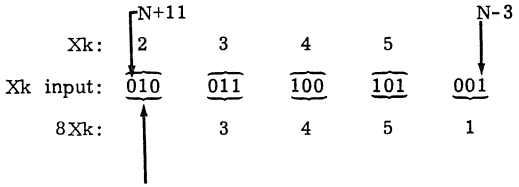
The network that forms 5Xk is on the 4DL7 and 4DN7 modules. By left-shifting Xk two places, 4Xk is formed, 5Xk is formed by adding Xk to this quantity. Thus, 5Xk is a 51-bit quantity. The 4DL7 modules generate bits 0 through 47; the 4DN7 module generates bits 48 through 50. Each 4DL7 module receives 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.

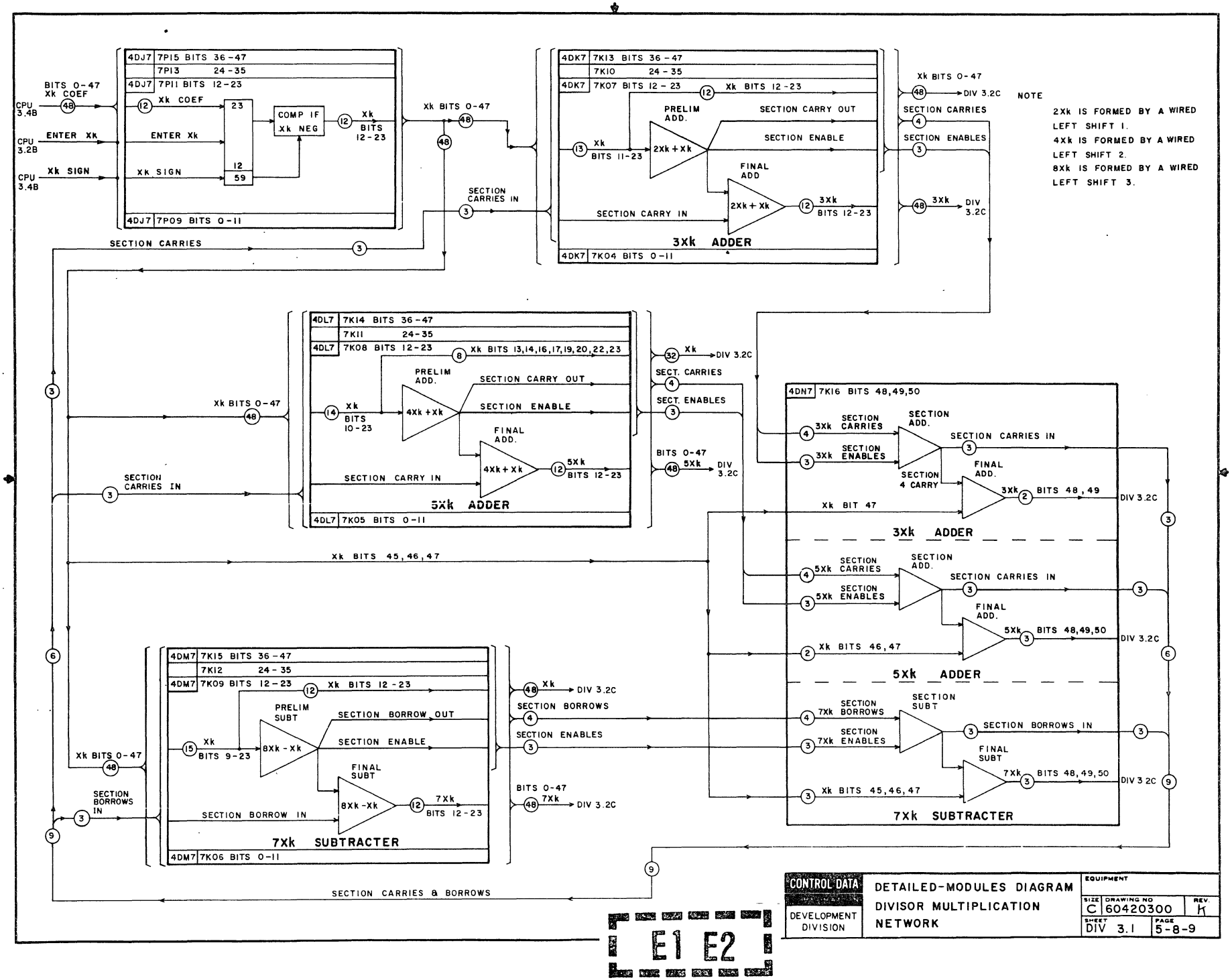
Xk, 2Xk, and 4Xk NETWORKS

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.

DIV 3.1 TEST POINTS

Module	Location	Test Point	Description	Module	Location	Test Point	Description	Module	Location	Test Point	Description
4DJ7*	7P09, 11, 13, 15	72, 73	Enter Xk								
		05	Xk coef bit 0, 12, 24, or 36								
		03	Xk coef bit 1								
		14	2								
		13	3								
		23	4								
		24	5								
		34	6								
		33	7								
		45	8								
		42	9								
		52	10								
		55	11								
		62, 65	Xk sign								
		74, 75	25-nanosecond clock								
4DK7*	7K04, 07, 10, 13	71	Section enable								
4DL7*	7K05, 08, 11, 14	71	Section enable								
4DM7*	7K06, 09 12, 15	71	Section enable								
4DN7*	7K16		(No signal/rgtr TPs)								

E3 E4



CONTROL DATA DEVELOPMENT DIVISION	DETAILED-MODULES DIAGRAM DIVISOR MULTIPLICATION NETWORK	EQUIPMENT	
		SIZE C	DRAWING NO 60420300 REV. K
		SHEET DIV 3.1	PAGE 5-8-9

E1 E2

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 barrow (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 $\overline{\text{remainder}}$ bits and the multiple bits to generate inclusive bit enables. Each 4DH7 module adds the $\overline{\text{remainder}}$ 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 $\overline{\text{remainder}}$ (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

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.

DIV 3.2 TEST POINTS

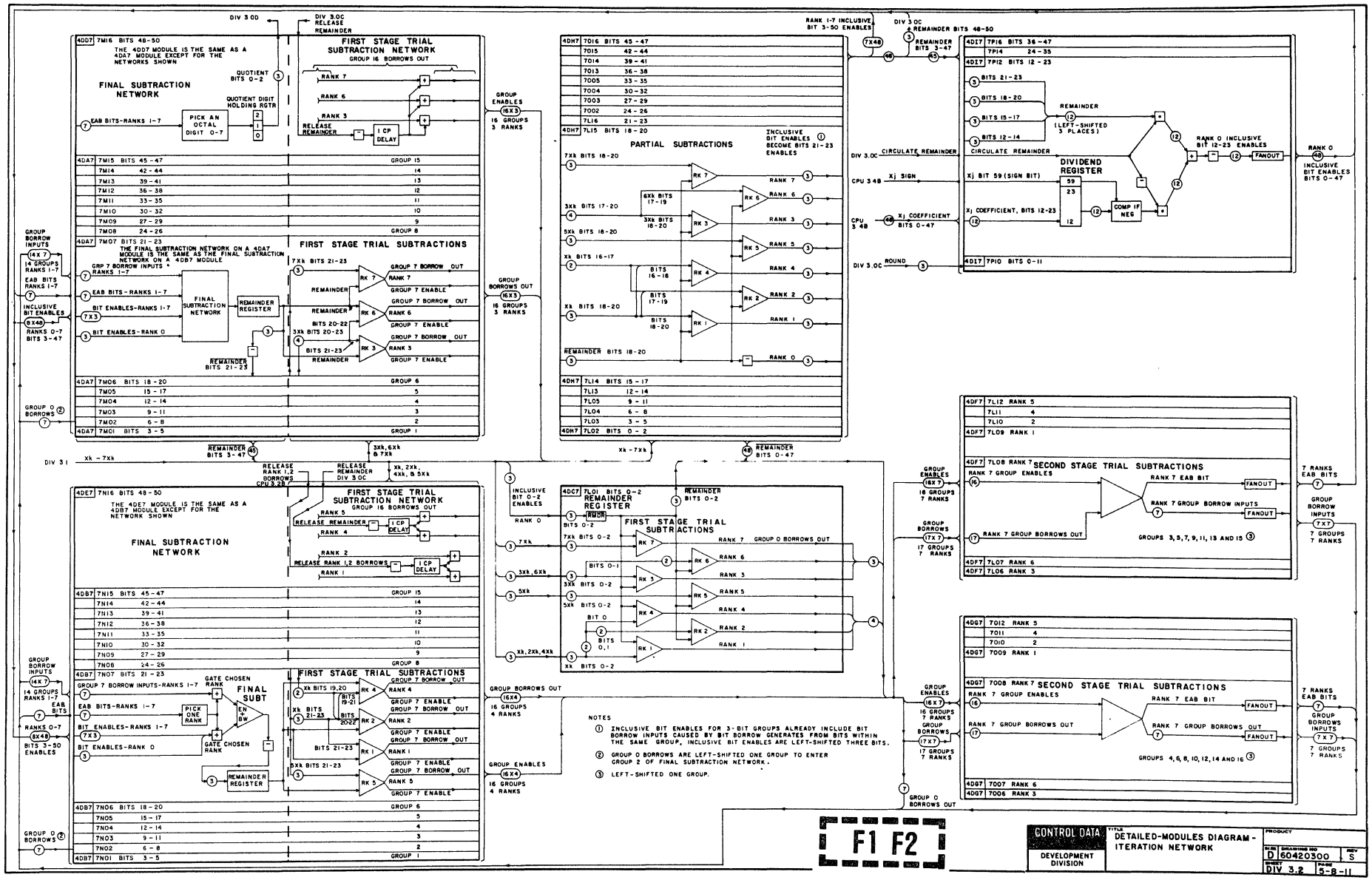
Module	Location	Test Point	Description
4DA7*	7M01-15	33	Remainder bit N
		32	N+1
		34	N+2
		04	Rank 3 enable
		03	6
		11	7
		35	25-nanosecond clock
4DB7*	7N01-15	33	Remainder bit N
		32	N+1
		34	N+2
		04	Rank 1 enable
		03	2
		14	4
		13	5
		35	25-nanosecond clock
4DC7*	7L01	63, 74	Remainder bit 0
		65, 73	1
		62, 72	2
		12	Rank 4 borrow
		71	25-nanosecond clock
4DD7*	7M16	13	Quotient bit 0
		12	1
		11	2
		04	Rank 3 enable
		05	6
		14	7
		31	25-nanosecond clock
4DE7*	7N16	05	Rank 1 enable
		06	2
		16	4
		15	5
		01	Rank 1 borrow
		35	25-nanosecond clock
4DF7*	7L06-12	02	Group 4 enable
		11	8
		21	12
		36	16
		56, 46	Group 16 borrow
4DG7*	7O06-12	01	Group 4 enable
		11	8
		21	12
		31	16
		56, 46	Group 16 borrow
		75	Group 4 borrow input

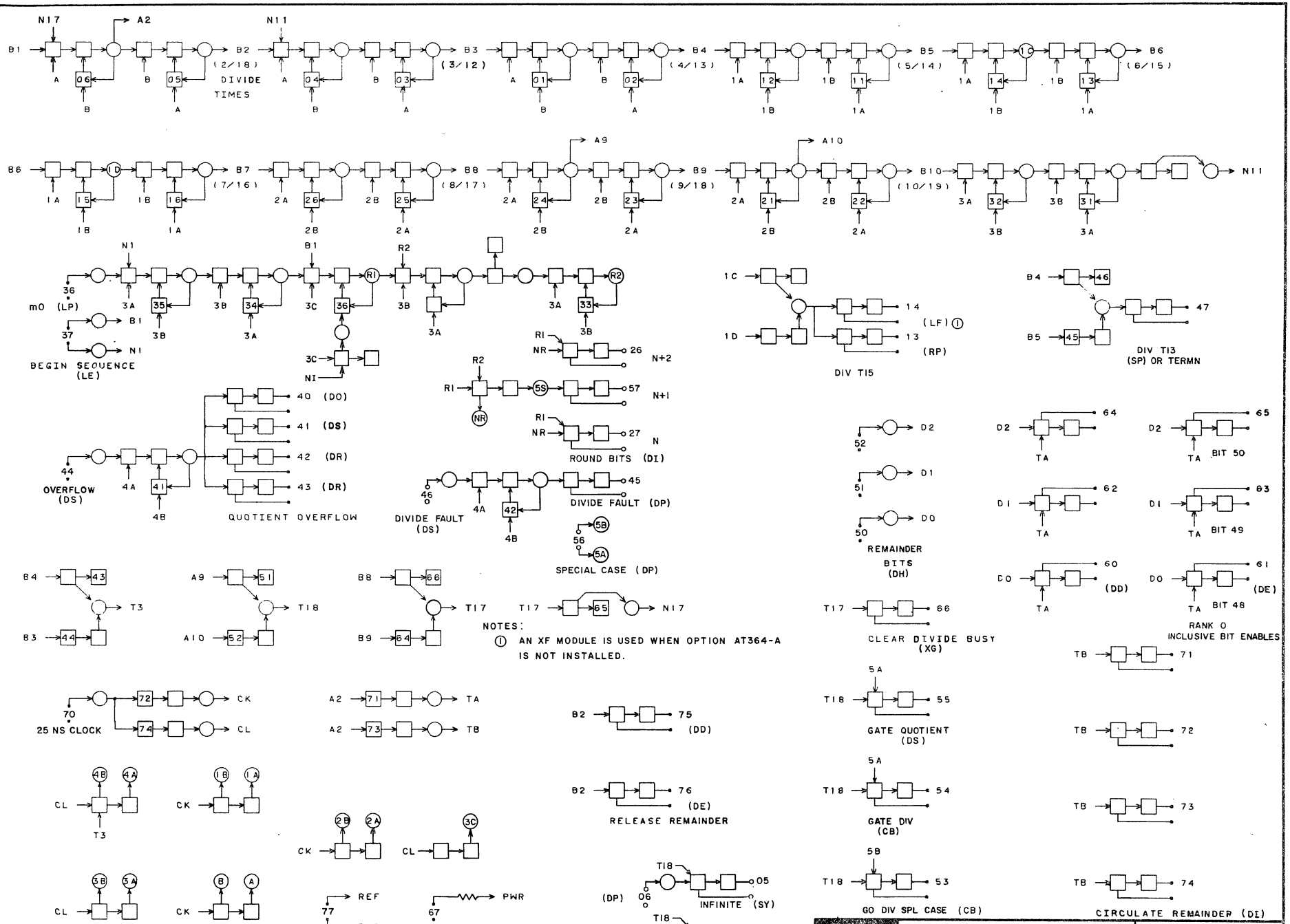
Module	Location	Test Point	Description

Module	Location	Test Point	Description

Module	Location	Test Point	Description

F3 F4


FINAL SUBTRACTION NETWORK
FIRST STAGE TRIAL SUBTRACTION NETWORK
FIRST STAGE TRIAL SUBTRACTIONS
PARTIAL SUBTRACTIONS
DIVIDEND REGISTER
REMAINDER REGISTER
SECOND STAGE TRIAL SUBTRACTIONS
CONTROL DATA
DEVELOPMENT DIVISION
DETAILED-MODULES DIAGRAM - ITERATION NETWORK
D 60420300
DIV 3.2
5-8-11



NOTES:

(I) AN XF MODULE IS USED WHEN OPTION AT364-A IS NOT INSTALLED.

CONTROL DATA DEVELOPMENT DIVISION	4DQ7 MODULE - DIV TIMING AND CONTROL NETWORKS	EQUIPMENT	SIZE C	DRAWING NO. 60420300	REV. Y	SHEET	PAGE 5-8-45

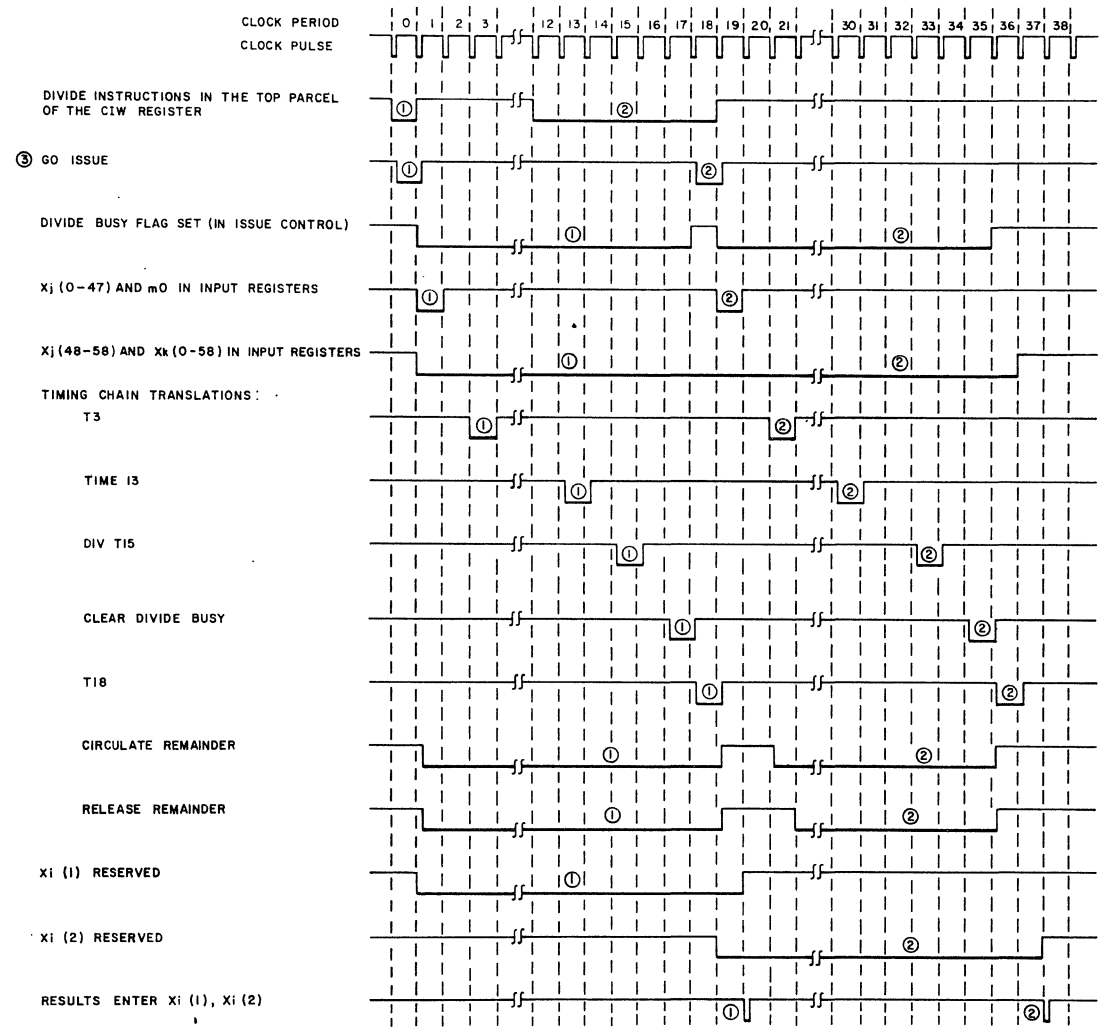
7P06

PART 10A

INCREMENT UNIT

BLOCK DIAGRAMS

(Models 175, 740, 750, 760)



NOTES:

① INSTRUCTION 1 - ASSUME NO ISSUE CONFLICTS.

② INSTRUCTION 2 - ASSUME ONLY ISSUE CONFLICT IS THE PREVIOUS DIVIDE INSTRUCTION.

③ ENABLE ISSUE IS DELAYED 4 CLOCK PERIODS (THEREFORE, GO ISSUE IS DELAYED A MINIMUM OF 4 CLOCK PERIODS) IN THE AAI20-C. THIS DELAY IS REMOVED BY INSTALLING OPTION AT364-A.

E11 E12

CONTROL DATA DEVELOPMENT DIVISION	TIMING DIAGRAM DIVIDE INSTRUCTION	CODE IDENT 34010	DWG NO 60420300	REV R
		C	SHEET	PAGE 5-8-53

PART 9

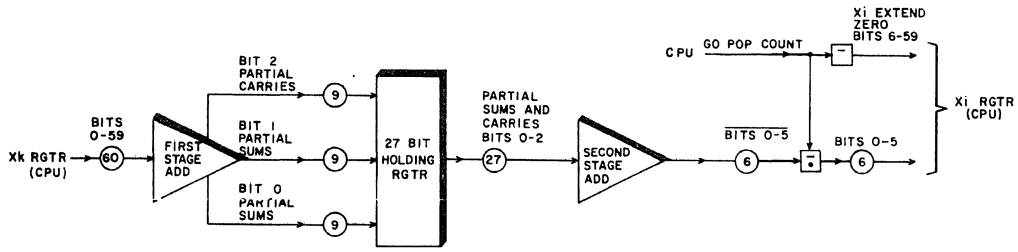
POPULATION COUNT UNIT

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.

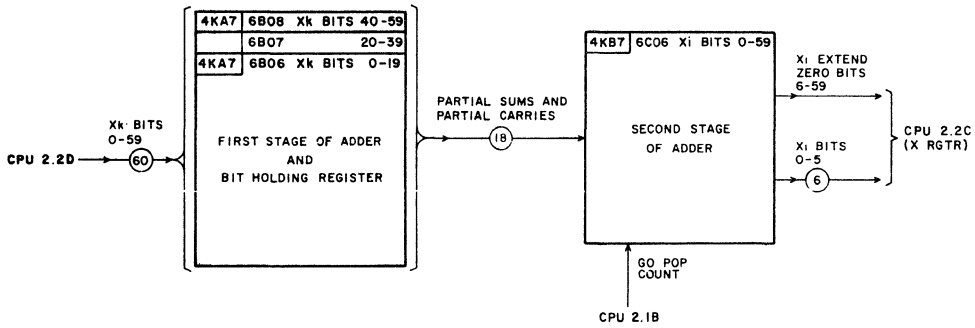
The population count instruction requires 2 clock periods for execution. Data moves from the Xk register to the population count unit in the same clock period in which the instruction issues from the CIW register. Data moves from the population count unit to the Xi register during the following clock 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.



B3 B4

CONTROL DATA	PRIMARY BLOCK DIAGRAM	CODE IDENT	DWG NO	REV
DEVELOPMENT DIVISION	POP. COUNT UNIT	34010	60420300	A
		C	SHEET POP CT 1.0	PAGE 5-9-1



B9 B10

CONTROL DATA DEVELOPMENT DIVISION	SECONDARY BLOCK DIAGRAM POP. COUNT UNIT	CODE IDENT 34010	DWG NO 60420300	REV H
		SIZE C	SHEET POP CT2.0	PAGE 5-9-3

POPULATION COUNT UNIT

The population count unit counts the number of one bits in the Xk register and stores the results in the lower order six bits of the Xi register. The population count unit can be considered to add a column of 60 bits (bits 0 through 59 of the operand). Instead of attempting to sum all 60 bits in the column at one time, the population count unit divides the operand into three six-bit sections and six seven-bit sections. During the first clock period of operation, the first stage of the adder generates two partial sum bits and a partial carry bit for each of these nine sections. The resulting 27 bits are held in a holding register during the second clock period while the second stage of the adder combines the results of all nine sections into one six-bit result.

When go pop count is received from the CPU, this six-bit result is sent to the X register as bits 0 through 5. Go pop count is also complemented and fanned out to extend zero bits in Xi bits 6 through 59.

FIRST-STAGE ADD

The first-stage add is located on three 4KA7 modules. Each module handles 20 bits of the operand. These 20 bits are divided into three sections. Sections 1 and 2 are 7-bit sections that handle the lower order 14 bits of the 20 bits on the module. Section 3 is a six-bit section that handles the higher order six bits on the module.

The first-stage add network for each section consists of four adders. Initially, six bits in each section enter two three-bit adders (X adders) and are partially added to form two partial sum bits (sum 0) and two partial carry bits (carry 1). (The numbers represent the position each partial sum or partial carry bit has relative to the final result.)

In sections 1 and 2, the two sum 0 bits are added to the seventh bit in the section to produce another partial sum (sum 0) and a partial carry bit (carry 1). Section 3 merely totals the two sum 0 bits. The resulting sum 0 bit in each section enters the bit holding register.

There are now three carry 1 bits in each section. They enter the last adder in the stage (B adder) and are summed to form partial sum bit (sum 1) and a partial carry bit (carry 2). The sum 1 and the carry 2 bits from each stage also enter the bit holding register. There are now three sum 0 bits, three sum 1 bits, and three carry 2 bits in the bit holding register on each module.

SECOND-STAGE ADD

The second-stage add begins on the 4KA7 modules but is primarily located on the 4KB7 module. The add network on each 4KA7 module combines the results from the three sections in separate adders. The A adder sums the three sum 0 bits, the C adder sums the three sum 1 bits, and the D adder sums the three carry 2 bits. The sums and carries produced are the complement of the true result and are recomplemented before being sent to the 4KB7 module. The data sent to the 4KB7 module from each 4KA7 module, therefore, consists of a sum 0, sum 1, sum 2, carry 1, carry 2, and carry 3 bit.

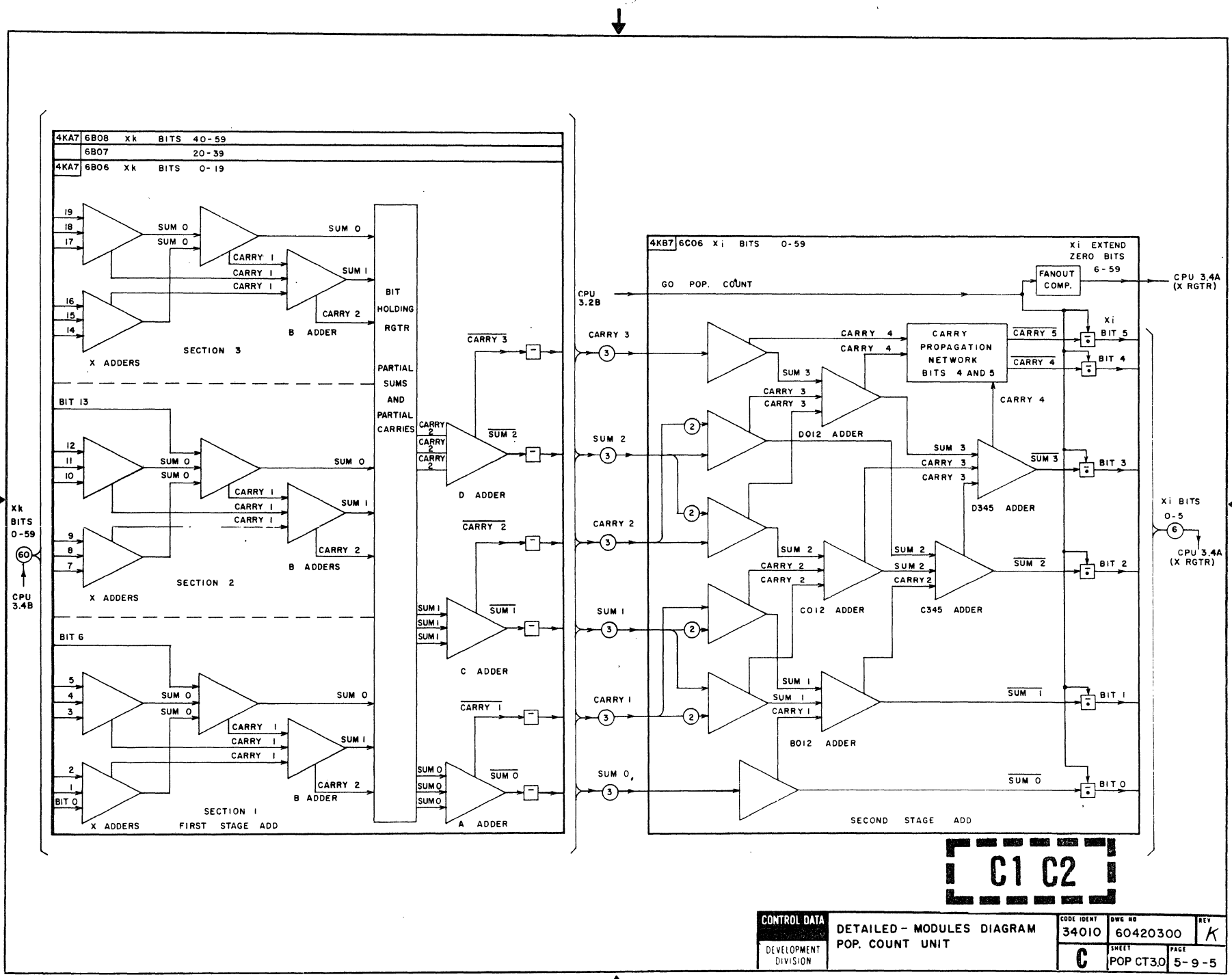
The 4KB7 module receives this data from the three 4KA7 modules and combines it in a complex add and carry propagation network. The resulting six-bit sum is the complement of the true result. The result is gated by go pop count and is recomplemented before being sent to the X register as bits 0 through 5. Go pop count is also complemented and fanned out to extend zero bits in Xi bits 6 through 59.

Go pop count is set in the population count unit when a 47 instruction issues from the CIW register.

POP CT 3.0 TEST POINTS

Module	Location	Test Point	Description	Module	Location	Test Point	Description	Module	Location	Test Point	Description
4KA7*	6B06-08	41	Section 1 sum 0 hldg rgtr								
		51	Section 1 sum 1 hldg rgtr								
		61	Section 1 carry 2 hldg rgtr								
		43	Section 2 sum 0 hldg rgtr								
		53	Section 2 sum 1 hldg rgtr								
		63	Section 2 carry 2 hldg rgtr								
		46	Section 3 sum 0 hldg rgtr								
		56	Section 3 sum 1 hldg rgtr								
		66	Section 3 carry 2 hldg rgtr								
		73	25-nanosecond clock								
4KB7*	6C06	71-75	Go pop count								

C3 C4


4KA7 6B08 Xk BITS 40-59
6B07 20-39
4KA7 6B06 Xk BITS 0-19
19
18
17
SUM 0
SUM 0
SUM 0
CARRY 1
CARRY 1
CARRY 1
SUM 1
16
15
14
CARRY 2
B ADDER
X ADDERS
SECTION 3
BIT HOLDING RGTR
PARTIAL SUMS AND PARTIAL CARRIES
BIT 13
12
11
10
SUM 0
SUM 0
SUM 0
CARRY 1
CARRY 1
CARRY 1
SUM 1
9
8
7
CARRY 2
B ADDERS
X ADDERS
SECTION 2
BIT 6
5
4
3
SUM 0
SUM 0
SUM 0
CARRY 1
CARRY 1
CARRY 1
SUM 1
2
1
BIT 0
CARRY 2
B ADDER
X ADDERS
SECTION 1
FIRST STAGE ADD
CARRY 3
CARRY 2
CARRY 2
CARRY 2
D ADDER
SUM 2
SUM 1
SUM 1
SUM 1
C ADDER
SUM 1
SUM 0
SUM 0
SUM 0
A ADDER
SUM 0
Xk BITS 0-59
60
CPU 3.4B
CPU 3.2B
CARRY 3
SUM 2
CARRY 2
SUM 1
CARRY 1
SUM 0
3
4KB7 6C06 Xi BITS 0-59
Xi EXTEND ZERO BITS 6-59
GO POP. COUNT
FANOUT COMP.
CPU 3.4A (X RGTR)
CARRY 4
CARRY 4
CARRY PROPAGATION NETWORK BITS 4 AND 5
CARRY 5
CARRY 4
Xi BIT 5
BIT 4
SUM 3
CARRY 3
CARRY 3
D012 ADDER
CARRY 4
SUM 3
CARRY 3
CARRY 3
D345 ADDER
SUM 3
BIT 3
SUM 2
CARRY 2
CARRY 2
C012 ADDER
SUM 2
SUM 2
CARRY 2
C345 ADDER
SUM 2
BIT 2
SUM 1
SUM 1
CARRY 1
B012 ADDER
SUM 1
BIT 1
SUM 0
BIT 0
SECOND STAGE ADD
Xi BITS 0-5
6
CPU 3.4A (X RGTR)
2
C1 C2
CONTROL DATA
DEVELOPMENT DIVISION
DETAILED - MODULES DIAGRAM
POP. COUNT UNIT
CODE IDENT 34010
DWG NO 60420300
REV K
C
SHEET POP CT3.0
PAGE 5-9-5

4KA7 MODULE

The three 4KA7 modules form the first stage of the add network in the population count unit and hold the results of this add in a 27-bit holding register. These 27 bits are further reduced to 18 bits before being sent to the second stage of the add in the 4KB7 module.

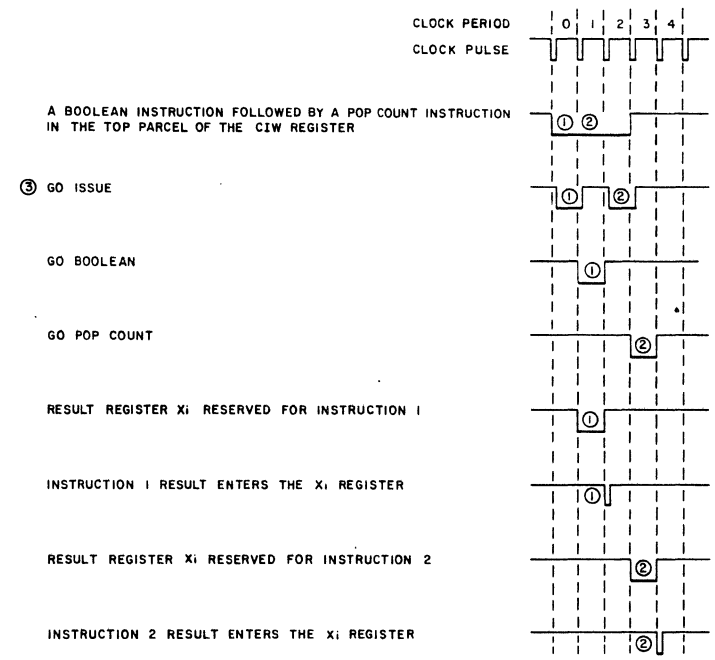
Each 4KA7 module operates on 20 bits of the 60-bit operand. Within a module, the 20 bits are divided into three sections. Sections 1 and 2 contain seven bits each, and section 3 contains six bits.

The six or seven operand bits for each section enter a group of three three-bit adders called X adders. The first two X adders in each section add six operand bits to produce two sum 0 bits and two carry 1 bits. In sections 1 and 2, the third X adder sums two of the sum 0 bits just generated (for example, A0 and A1) with the seventh operand bit (X06) to produce a sum 0 bit (A2) and another carry 1 bit (B2). In section 3, the third X adder sums the two sum 0 bits (A6 and A7) from the first two adders to generate a sum 0 bit (A8) and a carry 1 bit (B8).

The three sum 0 bits (A2, A5, and A8) resulting from the three sections are held in the bit holding register. At this point, the X adders have also generated three carry 1 bits for each section. These three carry 1 bits enter a B adder and are reduced to a sum 1 and a carry 2 bit for each section, which are also held in the bit holding register. There are now three sum 0 bits, three sum 1 bits, and three carry 2 bits in the bit holding register for each 4KA7 module.

The sum 0, sum 1, and carry 2 bits held in the bit holding registers for each section form an octal number that represents the number of one bits summed in that section.

From the holding register, the three sum 0 bits enter an A adder that sums them to produce a $\overline{\text{sum 0}}$ bit and a $\overline{\text{carry 1}}$ bit. The three sum 1 bits enter a C adder to become a $\overline{\text{sum 1}}$ bit and a $\overline{\text{carry 2}}$ bit, and the three carry 2 bits enter a D adder to become a $\overline{\text{sum 2}}$ and a $\overline{\text{carry 3}}$ bit. These bits are all complemented and sent to the 4KB7 module.



NOTES:

① 112jk Xj + Xk TO X2 (ASSUMING NO ISSUE CONFLICTS EXIST)

② 47ij2 COUNT OF ONES IN X2 TO Xi (X2 OPERAND REGISTER CONFLICTS WITH INSTRUCTION 2 DESTINATION REGISTER).

③ ENABLE ISSUE IS DELAYED 4 CLOCK PERIODS (THEREFORE, GO ISSUE IS DELAYED A MINIMUM OF 4 CLOCK PERIODS) IN THE AA120-C. THIS DELAY IS REMOVED BY INSTALLING OPTION AT364-A.

D9 D10

CONTROL DATA	TIMING DIAGRAM	CODE IDENT	DWG NO	REV
DEVELOPMENT DIVISION	POPULATION COUNT INSTRUCTION	34010	60420300	R
		C	SHEET	PAGE 5-9-11

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.

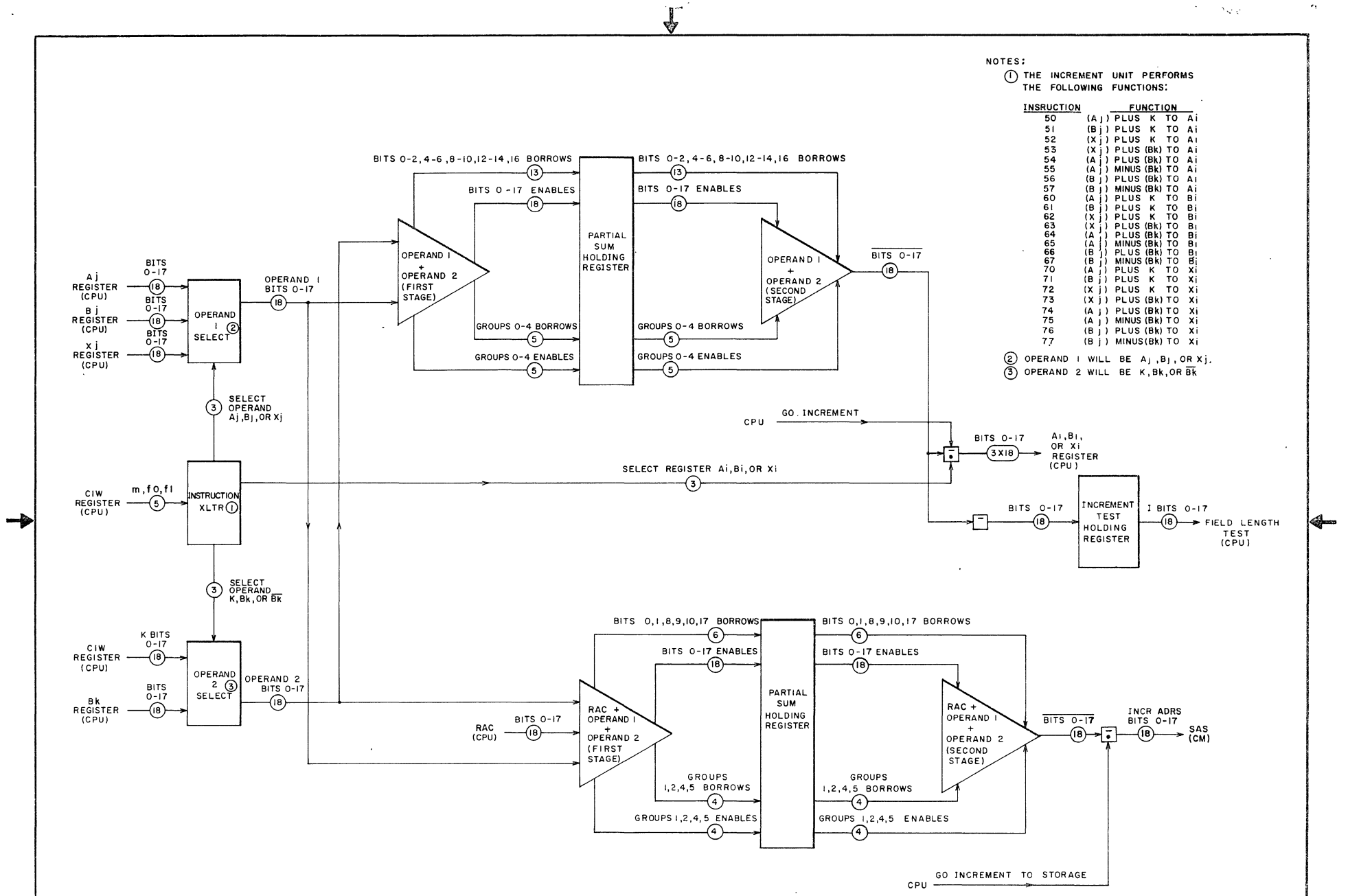
DESTINATION REGISTER SELECTION

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.

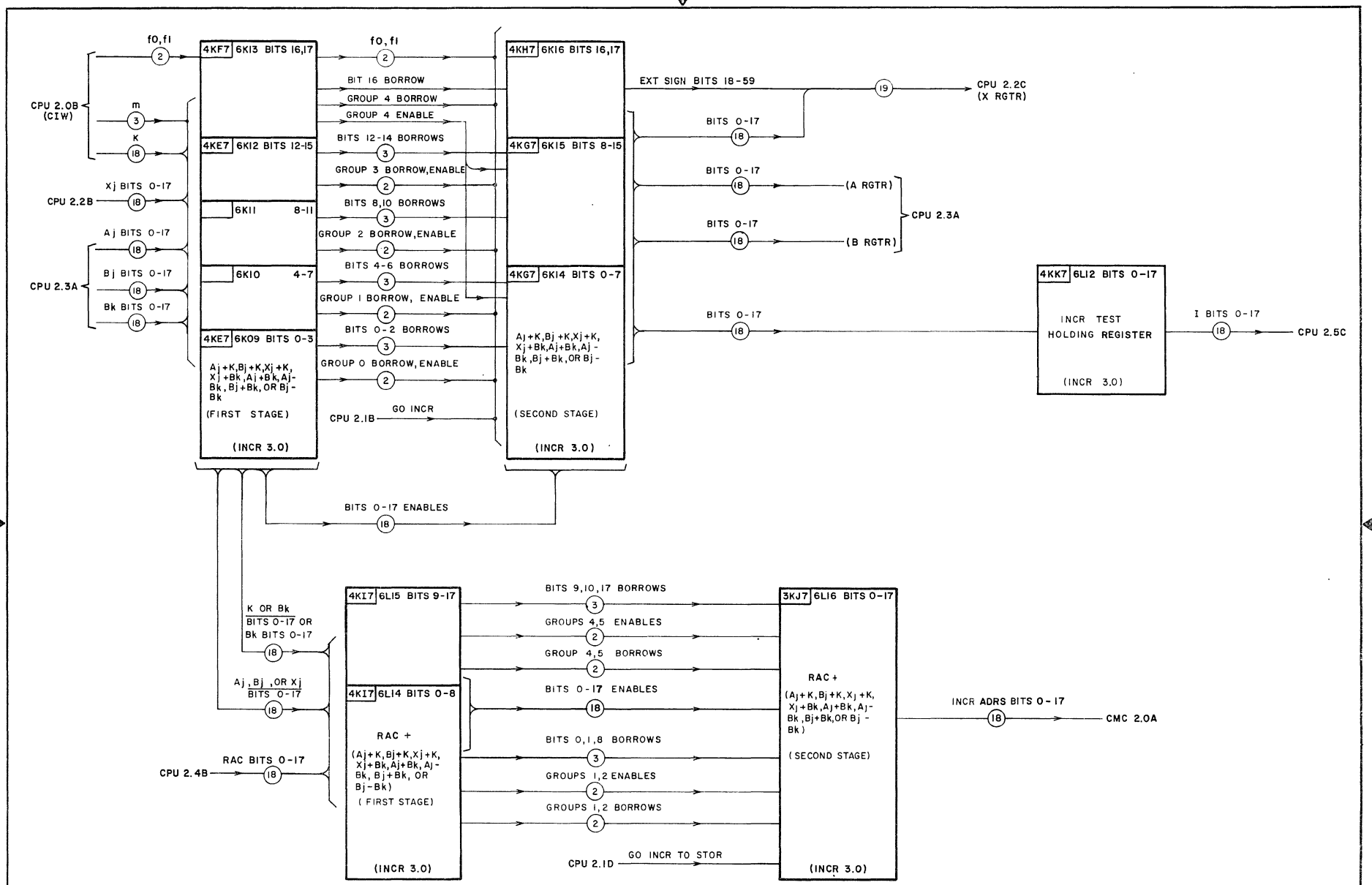
The Ai register receives the incremented operand during 50 through 57 instructions (f designator bits equal X01). When these instructions occur, the CPU generates a go increment to storage signal. This signal gates the incremented operand (plus RAC) to the SAS. The Ai register and the SAS receive increment data simultaneously. If the i designator in the CIW register has an octal value of zero, the go increment to storage signal is not generated.

The Bi register receives the incremented operand during 60 through 67 instructions (f designator bits equal X10).

The Xi 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 Xi register.



CONTROL DATA DEVELOPMENT DIVISION	PRIMARY BLOCK DIAGRAM INCREMENT UNIT	CODE IDENT 34010	DWG NO 60420300	REV Y
		C	SHEET INCR. 1.0A	PAGE 5-10-1



CONTROL DATA DEVELOPMENT DIVISION	SECONDARY BLOCK DIAGRAM INCREMENT UNIT	CODE IDENT 34010	DWG NO 60420300	REV Y
		C	SHEET INCR 2.0A	PAGE 5-10-3

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. 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 (4KE7 and 4KF7 modules) from five different input paths. One group of data paths consists of inputs from the Aj, Bj, and Xj registers. These 18-bit operands are complemented after entering the 4KE7 and 4KF7 modules. 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 complement of the K field in the CIW register, the Bk register data, and the complement of the Bk register data. 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 on the 4KE7, 4KF7, 4KG7, and 4KH7 modules. 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 (4KE7 and 4KF7 modules). The computation is completed in the second stage of the adder (4KG7 and 4KH7 modules).

A second computation is performed simultaneously with the one described previously. This computation is performed on the 4KI7 and 4KJ7 modules. 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 (4KI7 modules). The computation is completed in the second stage of the adder (3KJ7 module).

INCREMENT TEST HOLDING REGISTER

An 18-bit incremented operand is sent to the increment test holding register (4KK7 module) every clock period. The content of this register is compared with FLC in the 4LH7 module. 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 4LE7 module 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 f designator bits are fanned out in the 4KF7 module to a translator in the 4KG7 and 4KH7 modules.

The Ai register receives the incremented operand during 50 through 57 instructions (f designator bits equal X01). When these instructions occur, the 4HG7 module generates a go increment to storage signal. This signal gates the incremented operand (plus RAC) to the SAS. The SAS can accept an address at a maximum rate of one every 50 nanoseconds. The Ai register and the SAS receive increment data simultaneously. If the i designator in the CIW register has an octal value of zero, the go increment to storage is not generated.

The Bi register receives the incremented operand during 60 through 67 instructions (f designator bits equal X10).

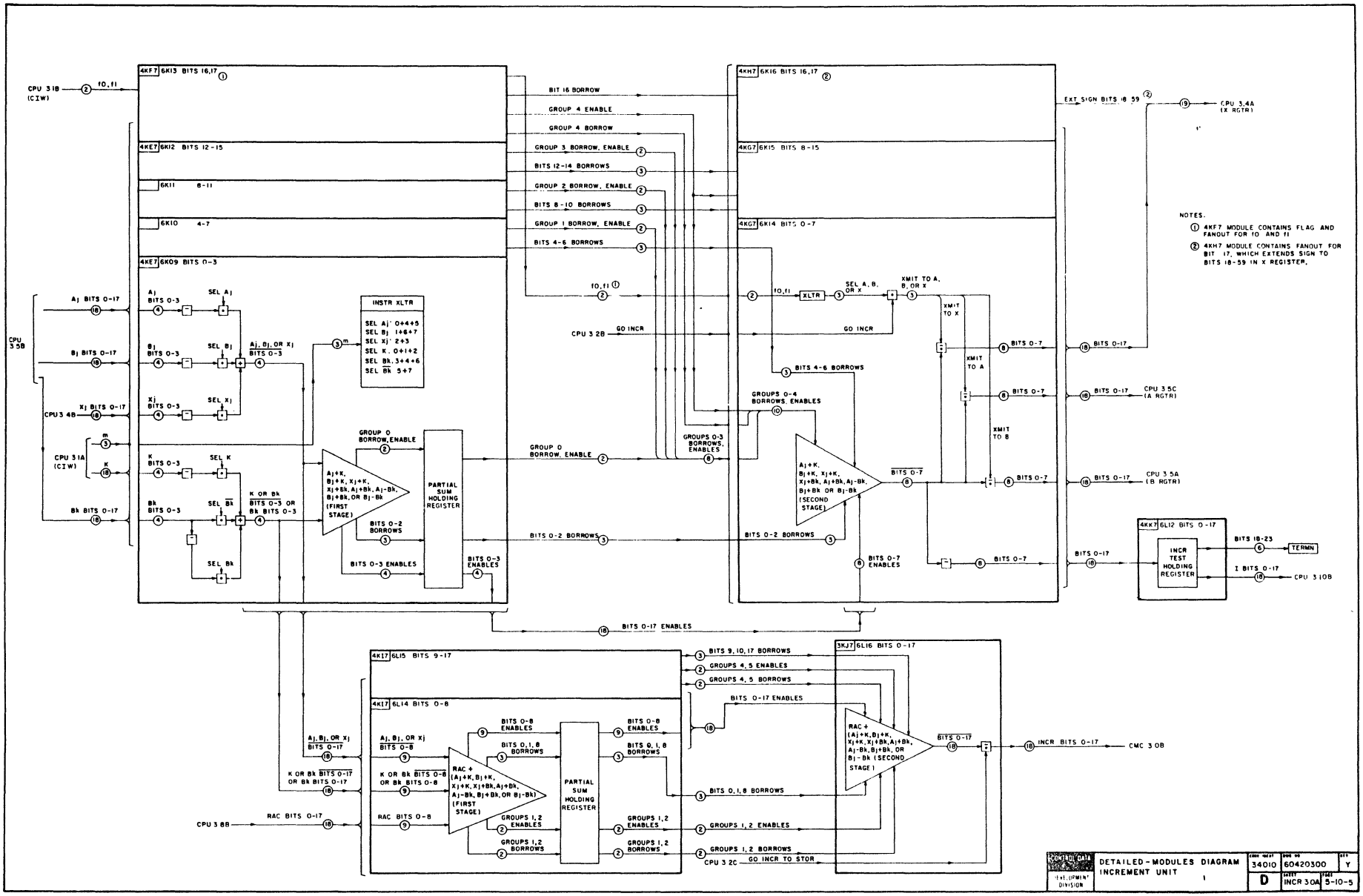
The Xi register receives the incremented operand during 70 through 77 instructions (f designator bits equal X11). When these instruction occur, the sign bit is extended in the Xi register. A fanout (4KH7 module) extends the sign of bits 17 to bit positions 18 through 59.

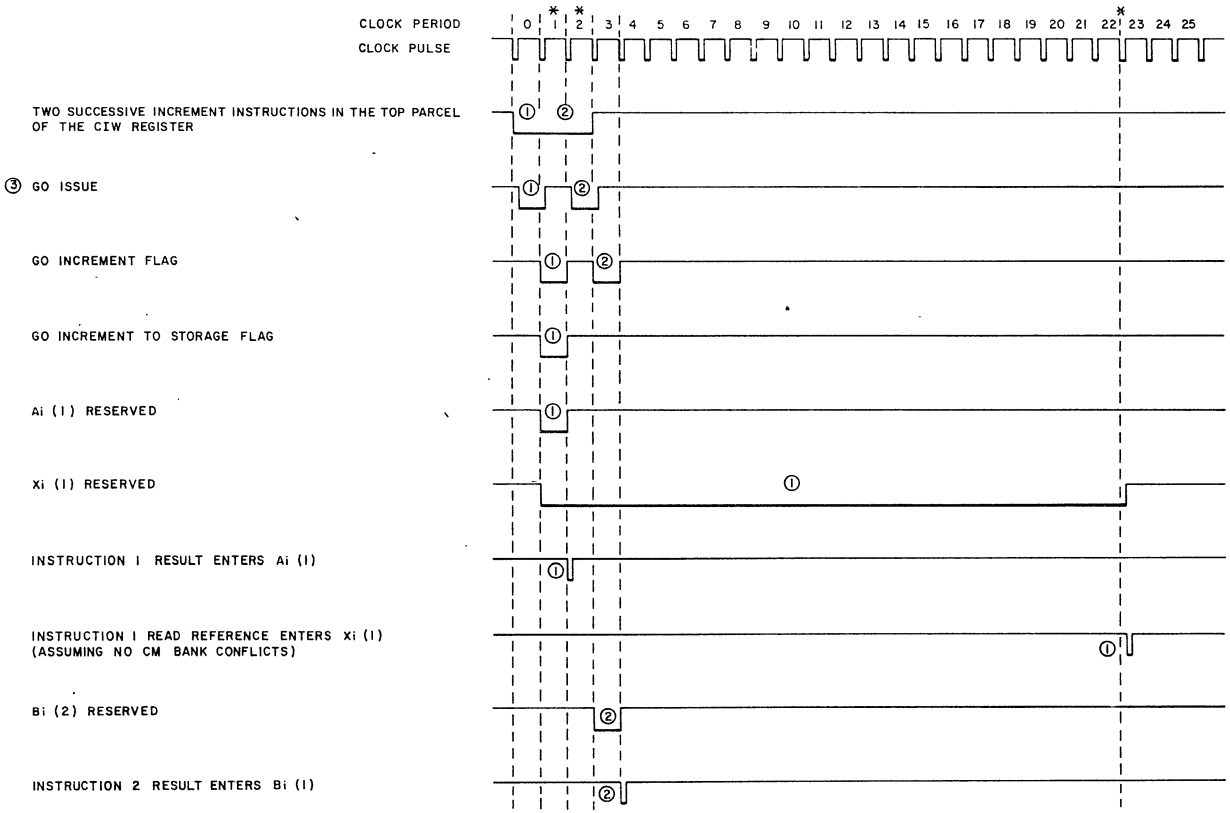
INCR 3.0A TEST POINTS

Module	Location	Test Point	Description
4KE7*	6K09-12	53	Bit N enable
		52	Bit N+1 enable
		54	N+2
		55	N+3
		64	Bit N borrow
		62	Bit N+1 borrow
		65	N+2
		41	Group enable
		44	Group borrow
		76	Select Xj
		75	Select Aj
		71	Select K
		72	Select Bk
		61	Select Bj
		51	Select $\overline{Bj}$
		66	25-nanosecond clock
4KF7*	6K13	36	Bit 16 enable
		34	Bit 17 enable
		32	Bit 16 borrow
		26	Group 4 enable
		21	Group 4 borrow
		56	f0 flag
		51	f1 flag
		63	25-nanosecond clock
4KG7*	6K14-15	63	Transmit to X
		67	Transmit to B
		65	Transmit to A
4KH7*	6K16	65	Transmit to X
		66	Transmit to B
		64	Transmit to A
4KI7*	6L14-15	04	Bit N enable
		14	Bit N+1 enable
		13	Bit N+2 enable
		05	Bit N borrow
		06	Bit N+1 borrow
		02	Bit N+8 borrow
3KJ7*	6L16		(No signal/rgtr TPs)

Module	Location	Test Point	Description
4KK7	6L12	01	I bit 0
		04	1
		03	2
		06	3
		11	4
		14	5
		12	6
		16	7
		21	8
		24	9
		23	10
		26	11
		46	12
		43	13
		45	14
		41	15
		56	16
		53	17
		33, 34	25-nanosecond clock

Module	Location	Test Point	Description


4KF7 6KI3 BITS 16,17 ①
4KE7 6KI2 BITS 12-15
6KI1 8-11
6KI0 4-7
4KE7 6KO9 BITS 0-3
4KH7 6KI6 BITS 16,17 ②
4KG7 6KI5 BITS 8-15
4KG7 6KI4 BITS 0-7
CPU 3 1B (CIW)
f0, f1
BIT 16 BORROW
GROUP 4 ENABLE
GROUP 4 BORROW
GROUP 3 BORROW, ENABLE
BITS 12-14 BORROWS
GROUP 2 BORROW, ENABLE
BITS 8-10 BORROWS
GROUP 1 BORROW, ENABLE
BITS 4-6 BORROWS
EXT SIGN BITS 18 59
CPU 3.4A (X RGTR)
NOTES.
① 4KF7 MODULE CONTAINS FLAG AND FANOUT FOR f0 AND f1
② 4KH7 MODULE CONTAINS FANOUT FOR BIT 17, WHICH EXTENDS SIGN TO BITS 18-59 IN X REGISTER.
Aj BITS 0-17
Bj BITS 0-17
Xj BITS 0-17
Bk BITS 0-17
CPU 3 5B
CPU 3 4B
CPU 3 1A (CIW)
Aj BITS 0-3
Bj BITS 0-3
Xj BITS 0-3
K BITS 0-3
Bk BITS 0-3
SEL Aj
SEL Bj
SEL Xj
SEL K
SEL Bk
Aj, Bj, OR Xj BITS 0-3
K OR Bk BITS 0-3 OR Bk BITS 0-3
INSTR XLTR
SEL Aj: 0+4+5
SEL Bj: 1+6+7
SEL Xj: 2+3
SEL K: 0+1+2
SEL Bk: 3+4+6
SEL Bk: 5+7
GROUP 0 BORROW, ENABLE
Aj+K, Bj+K, Xj+K, Xj+Bk, Aj+Bk, Aj-Bk, Bj+Bk, OR Bj-Bk (FIRST STAGE)
BITS 0-2 BORROWS
BITS 0-3 ENABLES
PARTIAL SUM HOLDING REGISTER
BITS 0-3 ENABLES
GROUP 0 BORROW, ENABLE
BITS 0-2 BORROWS
GROUPS 0-3 BORROWS, ENABLES
BITS 4-6 BORROWS
GROUPS 0-4 BORROWS, ENABLES
Aj+K, Bj+K, Xj+K, Xj+Bk, Aj+Bk, Aj-Bk, Bj+Bk OR Bj-Bk (SECOND STAGE)
BITS 0-7
BITS 0-2 BORROWS
BITS 0-7 ENABLES
CPU 3 2B
GO INCR
f0, f1 ①
XLTR
SEL A, B, OR X
XMIT TO A, B, OR X
XMIT TO X
XMIT TO A
XMIT TO B
BITS 0-7
BITS 0-17
CPU 3.5C (A RGTR)
CPU 3.5A (B RGTR)
4KK7 6L12 BITS 0-17
INCR TEST HOLDING REGISTER
BITS 18-23
TERMN
I BITS 0-17
CPU 3 1OB
BITS 0-17 ENABLES
4K17 6L15 BITS 9-17
4K17 6L14 BITS 0-8
5KJ7 6L16 BITS 0-17
BITS 9, 10, 17 BORROWS
GROUPS 4, 5 ENABLES
GROUPS 4, 5 BORROWS
BITS 0-17 ENABLES
Aj, Bj, OR Xj BITS 0-8
K OR Bk BITS 0-8 OR Bk BITS 0-8
RAC BITS 0-8
RAC + (Aj+K, Bj+K, Xj+K, Xj+Bk, Aj+Bk, Aj-Bk, Bj+Bk, OR Bj-Bk) (FIRST STAGE)
BITS 0, 1, 8 BORROWS
GROUPS 1, 2 ENABLES
GROUPS 1, 2 BORROWS
PARTIAL SUM HOLDING REGISTER
BITS 0-8 ENABLES
BITS 0, 1, 8 BORROWS
GROUPS 1, 2 ENABLES
GROUPS 1, 2 BORROWS
RAC + (Aj+K, Bj+K, Xj+K, Xj+Bk, Aj+Bk, Aj-Bk, Bj+Bk, OR Bj-Bk (SECOND STAGE)
BITS 0-17
INCR BITS 0-17
CMC 3 OB
K OR Bk BITS 0-17 OR Bk BITS 0-17
RAC BITS 0-17
CPU 3 8B
CPU 3 2C
GO INCR TO STOR
CONTROL DATA EQUIPMENT DIVISION
DETAILED - MODULES DIAGRAM
INCREMENT UNIT 1
CODE WORD 34010
SUB WD 60420300
BITS Y
D
PAGE INCR 30A
PAGE 5-10-5



NOTES:

* GO ISSUE FOR INCREMENT MAY BE DELAYED ONE CLOCK PERIOD IF MEMORY ENABLE IS NOT UP.

① INSTRUCTION 1

② INSTRUCTION 2

③ ENABLE ISSUE IS DELAYED 4 CLOCK PERIODS (THEREFORE, GO ISSUE IS DELAYED A MINIMUM OF 4 CLOCK PERIODS) IN THE AA120-C. THIS DELAY IS REMOVED BY INSTALLING OPTION AT364-A.

H7 H8

CONTROL DATA	TIMING DIAGRAM INCREMENT INSTRUCTION	CODE IDENT 34010	DWG NO 60420300	REV R
DEVELOPMENT DIVISION		C	SHEET	PAGE 5-10-21

PART 10B

INCREMENT UNIT

BLOCK DIAGRAMS

(Models 865, 875)

The increment unit executes CPU instructions 50 through 77. These instructions involve adds or subtracts of two selected 18-bit operands from the A, B, X, and CIW registers. The increment unit sends the result to an A register (50 through 57 instructions), a B register (60 through 67 instructions), or an X register (70 through 77 instructions). During 50 through 57 instructions which involve a memory reference, the increment unit also sends the result plus the 22-bit RAC to the SAS in CMC.

INPUT OPERAND SELECTION

Operand 1 and 2 select networks determine two operands for computation based on the value of the m designator in the CIW register. The two selected operands enter the increment and increment plus RAC adders simultaneously.

INCREMENT (18-BIT) ADDER

The 18-bit increment adder produces a one's complement sum or difference of the selected operands. The first stage generates bit/group borrows and enables stored by the partial sum holding register. The second stage completes the computation.

INCREMENT PLUS RAC (22-BIT) ADDER

The 22-bit increment plus RAC adder produces a one's complement sum or difference of the 18-bit selected operands and a two's complement sum of the result plus the 22-bit RAC. The first stage generates group borrows and enables, partial sums and carries, and bit enables stored by the partial sum holding register. The second stage completes the computation.

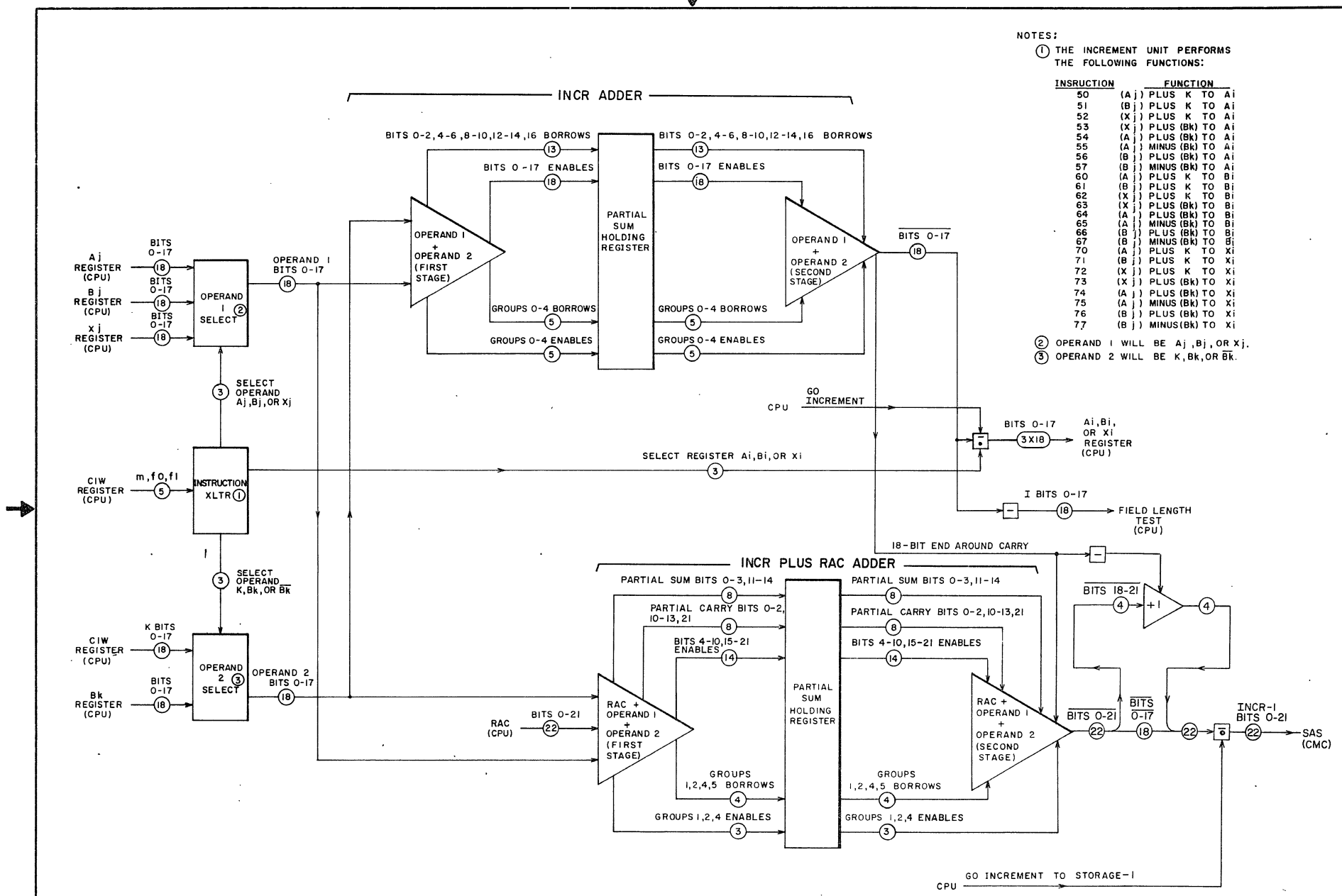
The second stage monitors the 18-bit adder for end-around carry. If the sum or difference does not produce an end-around carry, a correction factor is added to the 22-bit result. Correction is required because the 18-bit adder can produce an end-around carry, which should be a plus one added to the least significant bit (bit 0) and not passed from bit 17 to 18 of the 22-bit adder. (A carry that passes from bit 17 to 18 with no correction causes the resulting address to be high by 007777777.) The second stage produces the correction factor by blocking the end-around carry from the 18-bit adder and adding plus 1 to complemented bits 18 through 21.

INCREMENT FIELD LENGTH TEST

The increment unit sends I bits 0 through 17 to a field length tester in the CPU every clock period. When this increment operand is equal to or greater than FLC, CMC blocks the CM write reference or 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 CPU gates the increment operand to a destination register by generating a go increment signal. The lower 2 bits of the f designator from the CIW register determine whether an Ai (50 through 57 instructions), Bi (60 through 67 instructions), or Xi (70 through 77 instructions) register is selected. During 50 through 57 instructions with i not equal to zero, the CPU also gates the increment operand plus RAC to the SAS in CMC by generating a go increment to storage-1 signal.



NOTES:

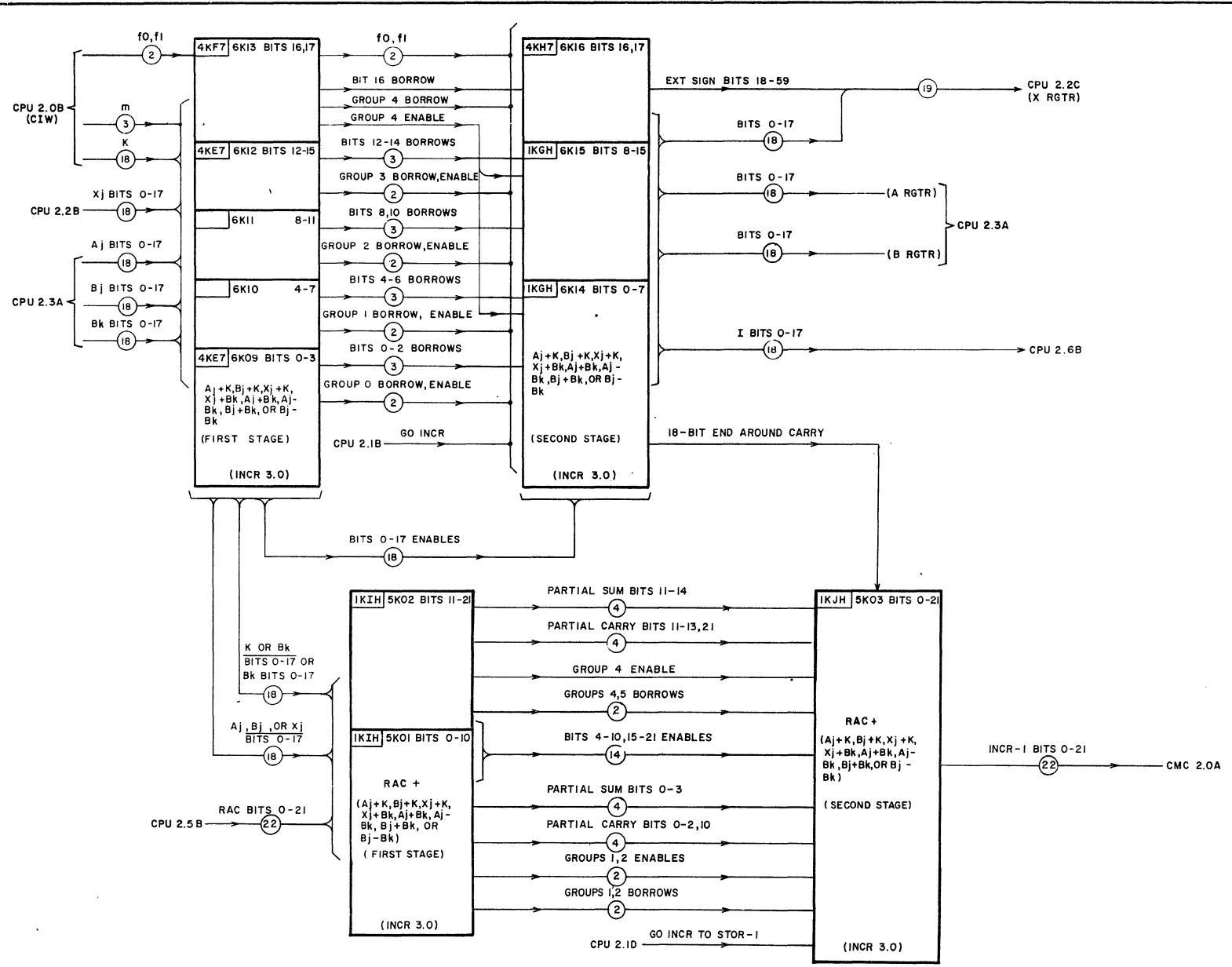
(1) THE INCREMENT UNIT PERFORMS THE FOLLOWING FUNCTIONS:

INSRUCTION	FUNCTION
50	(Aj) PLUS K TO Ai
51	(Bj) PLUS K TO Ai
52	(Xj) PLUS K TO Ai
53	(Xj) PLUS (Bk) TO Ai
54	(Aj) PLUS (Bk) TO Ai
55	(Aj) MINUS (Bk) TO Ai
56	(Bj) PLUS (Bk) TO Ai
57	(Bj) MINUS (Bk) TO Ai
60	(Aj) PLUS K TO Bi
61	(Bj) PLUS K TO Bi
62	(Xj) PLUS K TO Bi
63	(Xj) PLUS (Bk) TO Bi
64	(Aj) PLUS (Bk) TO Bi
65	(Aj) MINUS (Bk) TO Bi
66	(Bj) PLUS (Bk) TO Bi
67	(Bj) MINUS (Bk) TO Bi
70	(Aj) PLUS K TO Xi
71	(Bj) PLUS K TO Xi
72	(Xj) PLUS K TO Xi
73	(Xj) PLUS (Bk) TO Xi
74	(Aj) PLUS (Bk) TO Xi
75	(Aj) MINUS (Bk) TO Xi
76	(Bj) PLUS (Bk) TO Xi
77	(Bj) MINUS(Bk) TO Xi

(2) OPERAND 1 WILL BE Aj, Bj, OR Xj.

(3) OPERAND 2 WILL BE K, Bk, OR $\overline{Bk}$.

CONTROL DATA DEVELOPMENT DIVISION	PRIMARY BLOCK DIAGRAM INCREMENT UNIT	CODE IDENT 34010	DWR NO 60420300	REV Y
		C	SHEET INCR 1.0B	PAGE 5-10-7



CONTROL DATA	SECONDARY BLOCK DIAGRAM	CODE IDENT	DWG NO	REV
DEVELOPMENT DIVISION	INCREMENT UNIT	34010	60420300	Y
		C	SHEET INCR 2.08	PAGE 5-10-9

The increment unit executes CPU instructions 50 through 77. These instructions involve adds or subtracts of two selected 18-bit operands from the A, B, X, and CIW registers. The increment unit sends the result to an A register (50 through 57 instructions), a B register (60 through 67 instructions), or an X register (70 through 77 instructions). During 50 through 57 instructions which involve a memory reference, the increment unit also sends the result plus the 22-bit RAC to the SAS in CMC.

INPUT OPERAND SELECTION

The 4KE7 and 4KF7 modules determine two operands for computation based on the value of the m designator in the CIW register. The two selected operands enter the increment and increment plus RAC adders simultaneously.

INCREMENT (18-BIT) ADDER

The 18-bit increment adder produces a one's complement sum or difference of the 18-bit selected operands and a two's complement sum of the result plus the 22-bit RAC. The first stage (4KE7 and 4KF7 modules) generates bit/group borrows an enables stored by the partial sum holding register. The second stage (1KGH and 4KH7 modules) completes the computation.

INCREMENT PLUS RAC (22-BIT) ADDER

The 22-bit increment plus RAC adder produces a one's complement sum or difference of the 18-bit selected operands and a two's complement sum of the result plus the 22-bit RAC. The first stage (1KIH modules) generates group borrows and enables, partial sums and carries, and bit enables stored by the partial sum holding register. The second stage (1KJH module) completes the computation.

The second stage monitors the 18-bit adder for end-around carry. If the sum or difference does not produce an end-around carry, a correction factor is added to the 22-bit result. Correction is required because the 18-bit adder can produce an end-around carry, which should be a plus 1 added to the least significant bit (bit 0) and not passed from bit 17 to 18 of the 22-bit adder. (A carry that passes from bit 17 to 18 with no correction causes the resulting address to be high by 00777777.) The second stage produces the correction factor by blocking the end-around carry from the 18-bit adder and adding plus 1 to complemented bits 18 through 21.

INCREMENT FIELD LENGTH TEST

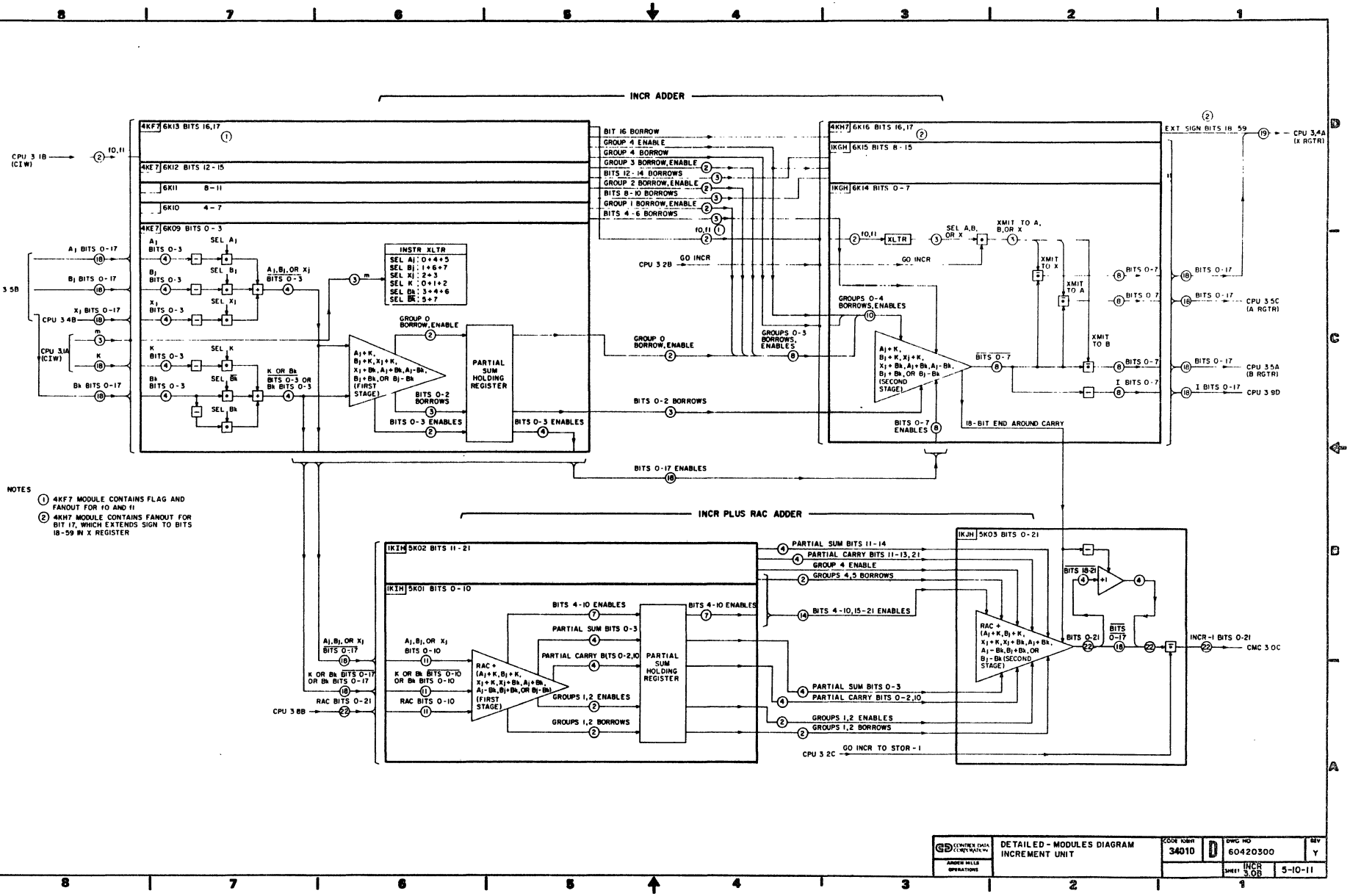
The 1KGH and 4KH7 modules send I bits 0 through 17 to a field length tester in the CPU every clock period. When this increment operand is equal to or greater than FLC, CMC blocks the CM write reference or returns a zero word for a CM read reference. This occurs only when a 50 through 57 instruction causes a CM reference.

DESTINATION REGISTER SELECTION

The CPU gates the increment operand to a destination register by sending a go increment signal to the 1KGH and 4KH7 modules. The lower 2 bits of the f designator from the CIW register determine whether an Ai (50 through 57 instructions), Bi (60 through 67 instructions), or Xi (70 through 77 instructions) register is selected. During 50 through 57 instructions with i not equal to zero, the CPU also gates the increment operand plus RAC to the SAS in CMC by sending a go increment to storage -1 signal to the 1KJH module.

INCR 3.0B TEST POINTS

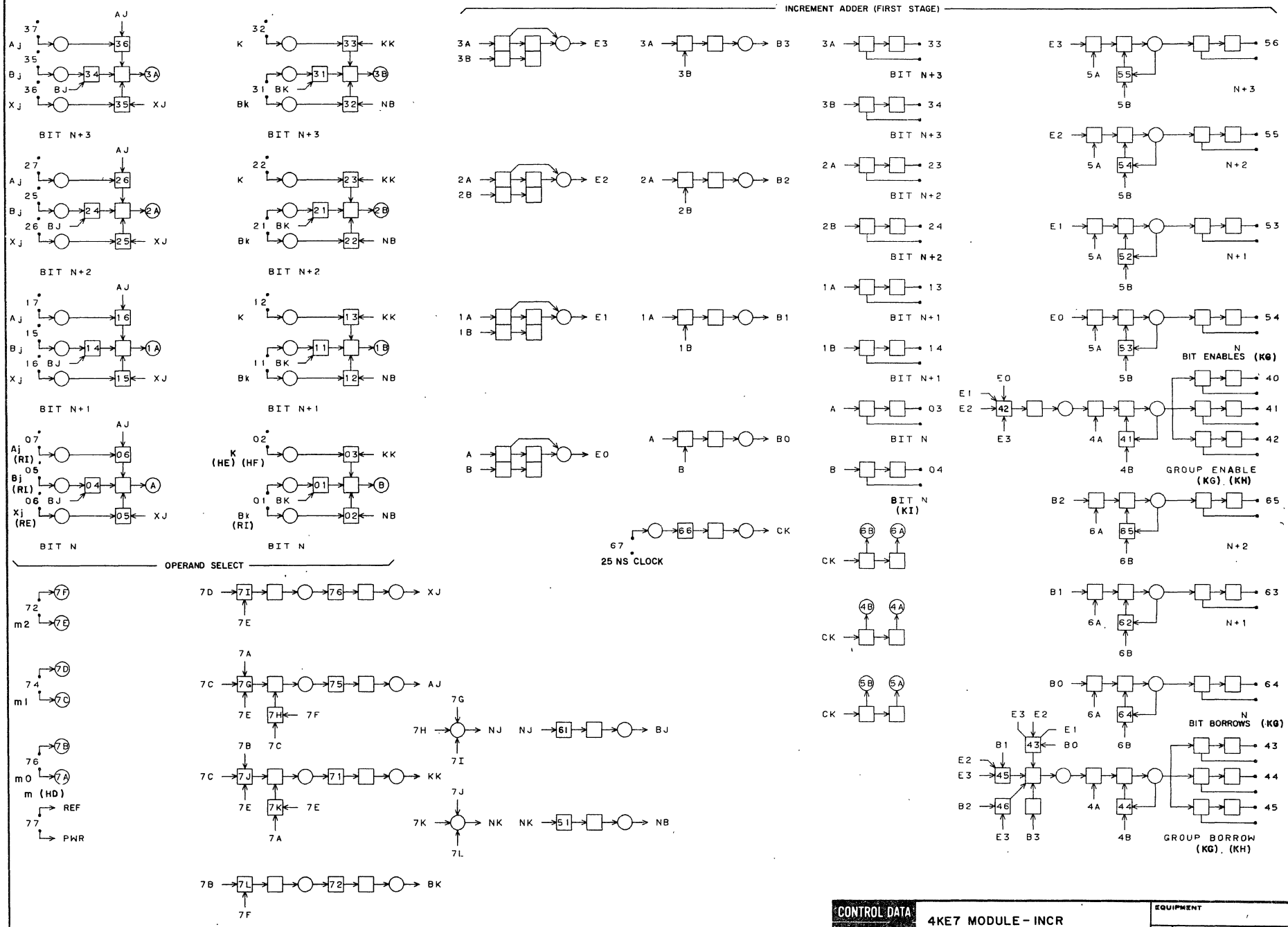
Module	Location	Test Point	Description
4KE7	6K09-12	53	Bit N enable
		52	Bit N+1 enable
		54	N+2
		55	N+3
		64	Bit N borrow
		62	Bit N+1 borrow
		65	N+2 borrow
		41	Group enable
		44	Group borrow
		76	Select Xj
		75	Select Aj
		71	Select K
		72	Select Bk
		61	Select Bj
		51	Select $\overline{Bj}$
		66	25-nanosecond clock
4KF7*	6K13	36	Bit 16 enable
		34	Bit 17 enable
		32	Bit 16 borrow
		26	Group 4 enable
		21	Group 4 borrow
		56	f0 flag
		51	f1 flag
		63	25-nanosecond clock
4KG7*	6K14-15	63	Transmit to X
		67	Transmit to B
		65	Transmit to A
4KH7*	6K16	65	Transmit to X
		66	Transmit to B
		64	Transmit to A
1KIH	5K01-02	05	Partial Carry Bit 0/11
		02	1/12
		13	2/13
		55	10/21
		03	Partial Sum Bit 0/11
		04	1/12
		16	2/13
1KJH	5K03	15	Go Incr to Stor-1


INCR ADDER
4KF7 6K13 BITS 16,17
4KE7 6K12 BITS 12-15
6K11 8-11
6K10 4-7
4KE7 6K09 BITS 0-3
CPU 3 1B (CIW)
I0,I1
Aj BITS 0-17
Bj BITS 0-17
CPU 3 5B
Xj BITS 0-17
CPU 3 4B
CPU 3 1A (CIW)
K
Bk BITS 0-17
SEL Aj
SEL Bj
SEL Xj
SEL K
SEL Bk
Aj,Bj,OR Xj BITS 0-3
K OR Bk BITS 0-3 OR Bk BITS 0-3
INSTR XLTR
SEL Aj: 0+4+5
SEL Bj: 1+6+7
SEL Xj: 2+3
SEL K: 0+1+2
SEL Bk: 3+4+6
SEL Bk: 5+7
Aj+K, Bj+K, Xj+K, Xj+Bk, Aj+Bk, Aj-Bk, Bj+Bk, OR Bj-Bk (FIRST STAGE)
GROUP 0 BORROW, ENABLE
BITS 0-2 BORROWS
BITS 0-3 ENABLES
PARTIAL SUM HOLDING REGISTER
BITS 0-3 ENABLES
BIT 16 BORROW
GROUP 4 ENABLE
GROUP 4 BORROW
GROUP 3 BORROW, ENABLE
BITS 12-14 BORROWS
GROUP 2 BORROW, ENABLE
BITS 8-10 BORROWS
GROUP 1 BORROW, ENABLE
BITS 4-6 BORROWS
GO INCR
CPU 3 2B
GROUP 0 BORROW, ENABLE
BITS 0-2 BORROWS
BITS 0-17 ENABLES
4KH7 6K16 BITS 16,17
1KGH 6K15 BITS 8-15
1KGH 6K14 BITS 0-7
XLTR
SEL A,B, OR X
XMIT TO A, B, OR X
XMIT TO X
XMIT TO A
GROUPS 0-4 BORROWS, ENABLES
GROUPS 0-3 BORROWS, ENABLES
Aj+K, Bj+K, Xj+K, Xj+Bk, Aj+Bk, Aj-Bk, Bj+Bk, OR Bj-Bk (SECOND STAGE)
BITS 0-7
XMIT TO B
I BITS 0-7
BITS 0-7 ENABLES
18-BIT END AROUND CARRY
EXT SIGN BITS 18,59
BITS 0-17
CPU 3,4A (X RGTR)
CPU 3 5C (A RGTR)
CPU 3 5A (B RGTR)
CPU 3 9D
NOTES
(1) 4KF7 MODULE CONTAINS FLAG AND FANOUT FOR I0 AND I1
(2) 4KH7 MODULE CONTAINS FANOUT FOR BIT 17, WHICH EXTENDS SIGN TO BITS 18-59 IN X REGISTER
INCR PLUS RAC ADDER
1KIH 5K02 BITS 11-21
1KIH 5K01 BITS 0-10
1KJH 5K03 BITS 0-21
Aj,Bj,OR Xj BITS 0-17
K OR Bk BITS 0-17 OR Bk BITS 0-17
RAC BITS 0-21
CPU 3 8B
Aj,Bj,OR Xj BITS 0-10
K OR Bk BITS 0-10 OR Bk BITS 0-10
RAC BITS 0-10
RAC + (Aj+K, Bj+K, Xj+K, Xj+Bk, Aj+Bk, Aj-Bk, Bj+Bk, OR Bj-Bk (FIRST STAGE)
BITS 4-10 ENABLES
PARTIAL SUM BITS 0-3
PARTIAL CARRY BITS 0-2,10
GROUPS 1,2 ENABLES
GROUPS 1,2 BORROWS
PARTIAL SUM HOLDING REGISTER
PARTIAL SUM BITS 11-14
PARTIAL CARRY BITS 11-13,21
GROUP 4 ENABLE
GROUPS 4,5 BORROWS
BITS 4-10,15-21 ENABLES
PARTIAL SUM BITS 0-3
PARTIAL CARRY BITS 0-2,10
GROUPS 1,2 ENABLES
GROUPS 1,2 BORROWS
GO INCR TO STOR-1
CPU 3 2C
RAC + (Aj+K, Bj+K, Xj+K, Xj+Bk, Aj+Bk, Aj-Bk, Bj+Bk, OR Bj-Bk (SECOND STAGE)
BITS 18-21
+1
BITS 0-21
BITS 0-17
INCR-1 BITS 0-21
CMC 3 0C
GENERAL DYNAMICS ARMEN MILLS OPERATIONS
DETAILED - MODULES DIAGRAM
INCREMENT UNIT
CODE IDENT 34010
SIZE D
DWG NO 60420300
REV Y
SHEET INCR 3.0B
5-10-11

PART 10C

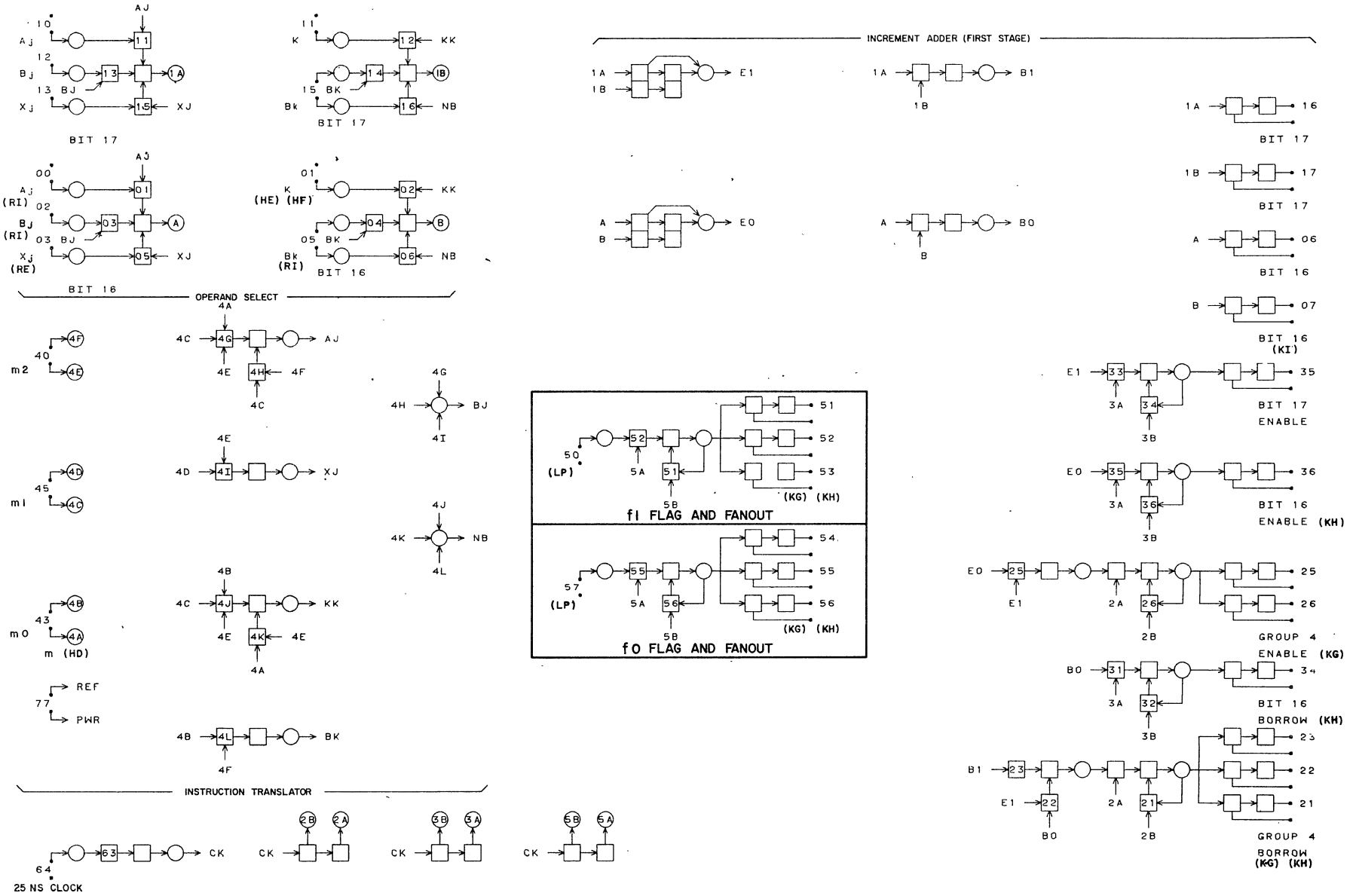
INCREMENT UNIT

LOGIC AND TIMING DIAGRAMS



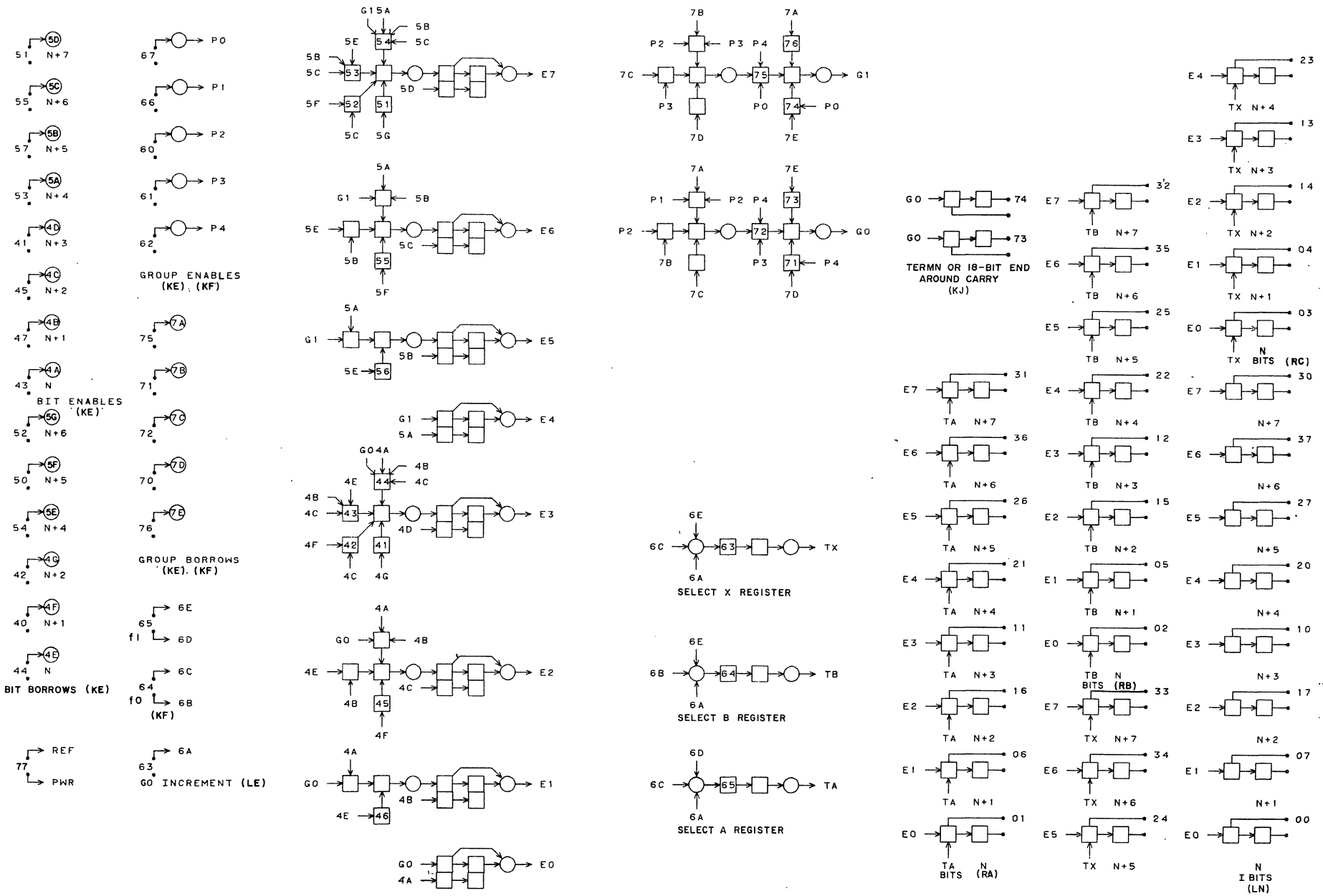
CONTROL DATA CORPORATION DEVELOPMENT DIVISION	4KE7 MODULE - INCR INCREMENT ADDER BITS 0-15 (FIRST STAGE)	EQUIPMENT	
		SIZE C	DRAWING NO. 60420300
		REV Y	
		SHEET	PAGE 5-10-13

6K09-12

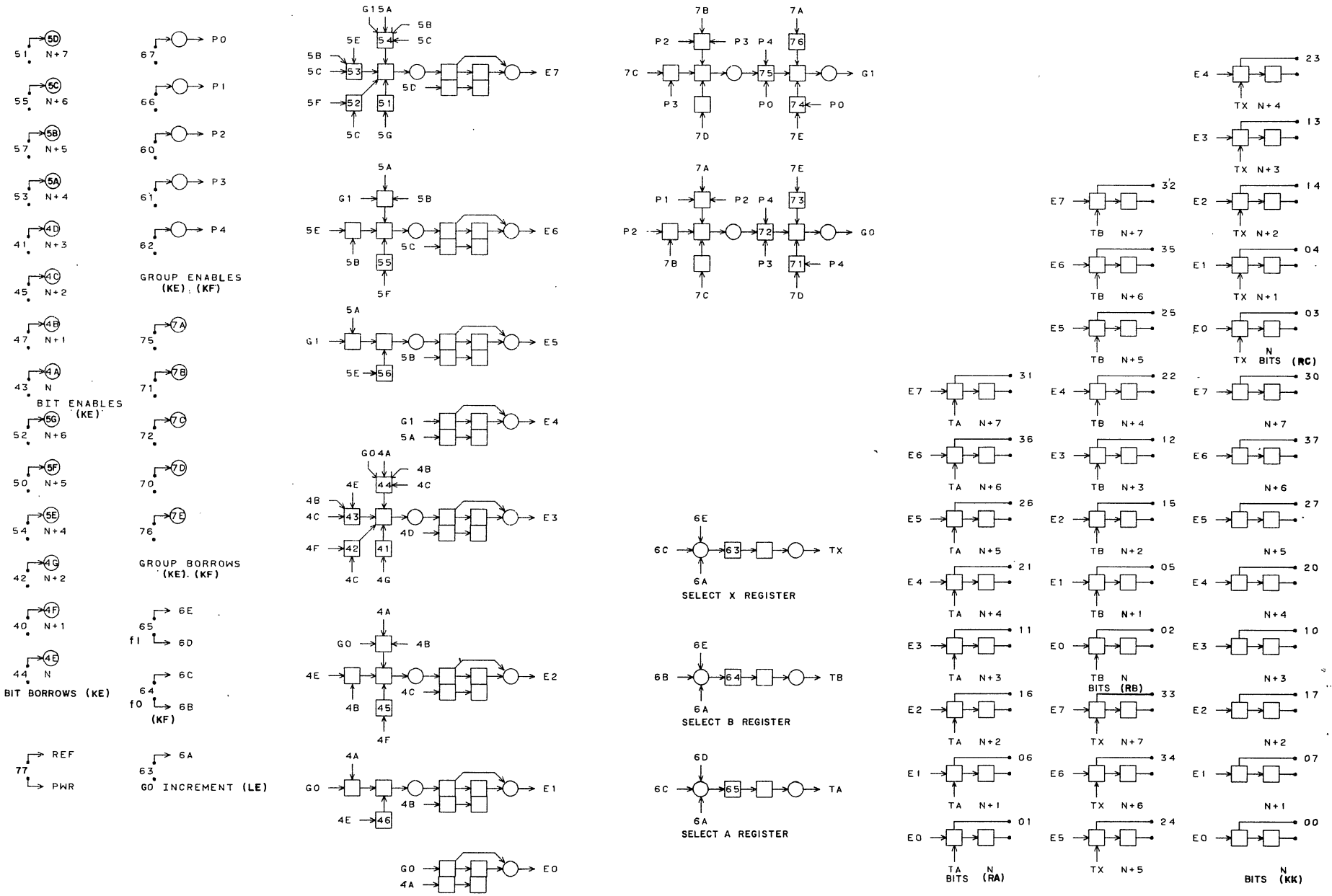


CONTROL DATA	4KF7 MODULE - INCR INCREMENT ADDER BITS 16,17 (FIRST STAGE)	EQUIPMENT
DEVELOPMENT DIVISION		SIZE C · DRAWING NO 60420300 · REV. · SHEET · PAGE 5-10-16

6K13

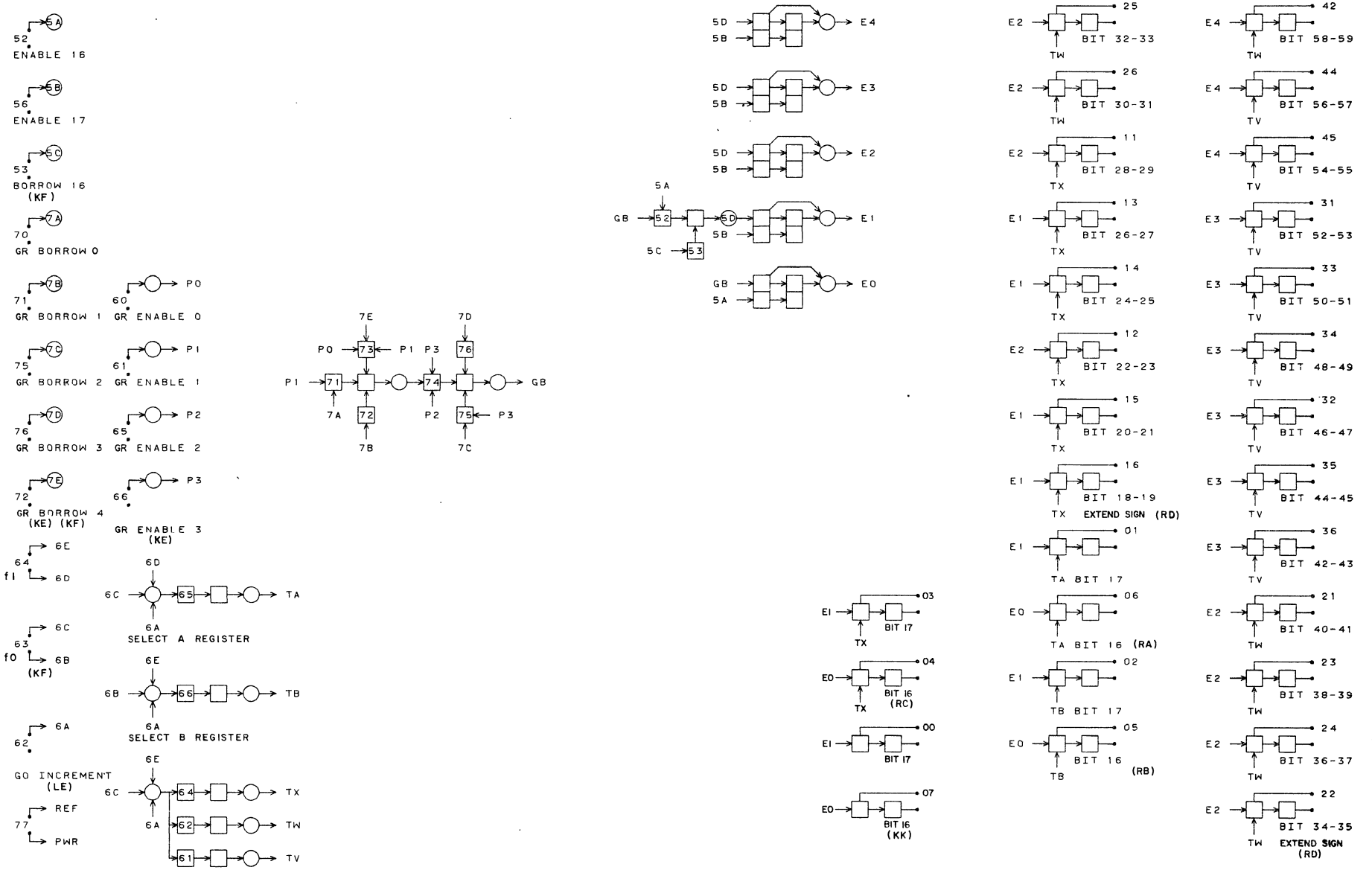


CONTROL DATA	IKGH MODULE - INCR INCREMENT ADDER BITS 0-15 (SECOND STAGE) (MODEL 8 X 5)	EQUIPMENT
DEVELOPMENT DIVISION		SIZE C · DRAWING NO. 60420300 · REV. Y
		SHEET · PAGE 5-10-17



CONTROL DATA	4KG7 MODULE - INCR	EQUIPMENT
DEVELOPMENT DIVISION	INCREMENT ADDER	SIZE C · DRAWING NO. 60420300 · REV. Y
	BITS 0-15 (SECOND STAGE) (MODELS 175,7X0)	SHEET · PAGE 6-10-19

6K14-15



CONTROL DATA	4KH7 MODULE – INCR	EQUIPMENT	
DEVELOPMENT DIVISION	INCREMENT ADDER	SIZE C	DRAWING NO 60420300 REV. Y
	BITS 16, 17 (SECOND STAGE)	SHEET	PAGE 5-10-21

Two 1KIH modules and a 1KJH module add the 22-bit RAC to the sum or difference of two 18-bit operands. The 1KIH modules generate bit enables, group enables, group borrows, partial sums, and partial carries. These signals proceed to the 1KJH module where the computation is completed.

The following example shows a pencil-and-paper addition for values which require correction.

```
Aj          1 2 3 4 5 6
K           6 5 4 3 2 2
           -0 0 0 0 0 0      18-bit sum with end around
                       1 ←   carry correction
            0 0 0 0 0 1
RAC       0 3 0 0 0 0 0 0
          0 3 0 0 0 0 0 1     22-bit result
```

The following table shows the conditions of the terms and output signals when the three operands are assigned values of the above example. Refer to the 1KJH module for the final result of this example.

	Groups	5			4				3				2			1				0			
	Bits	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Input Signals	$\overline{Aj}$=654321	F1	F1	F1	F1	1	1	0	1	0	1	1	0	0	0	1	1	0	1	0	0	0	1
	$\overline{K}$=123455	F1	F1	F1	F1	0	0	1	0	1	0	0	1	1	1	0	0	1	0	1	1	0	1
	$\overline{RAC}$=4777777	F0	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	PS10-PS0 terms	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
	PS3-PS0 terms	X	X	X	X	X	X	X	0	0	0	0	X	X	X	X	X	X	X	0	0	1	1
	PC10-PC0 terms	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
	PC10,PC2-PC0 terms	1	X	X	X	X	X	X	X	1	1	1	1	X	X	X	X	X	X	X	1	0	1
Output Signals	E10-E4 terms	1	0	1	1	1	1	1	X	X	X	X	1	1	1	1	1	1	1	X	X	X	X
	Group enables		Termn			1				X				1			1				X		
	Group borrows		1			0				X				0			0				X		

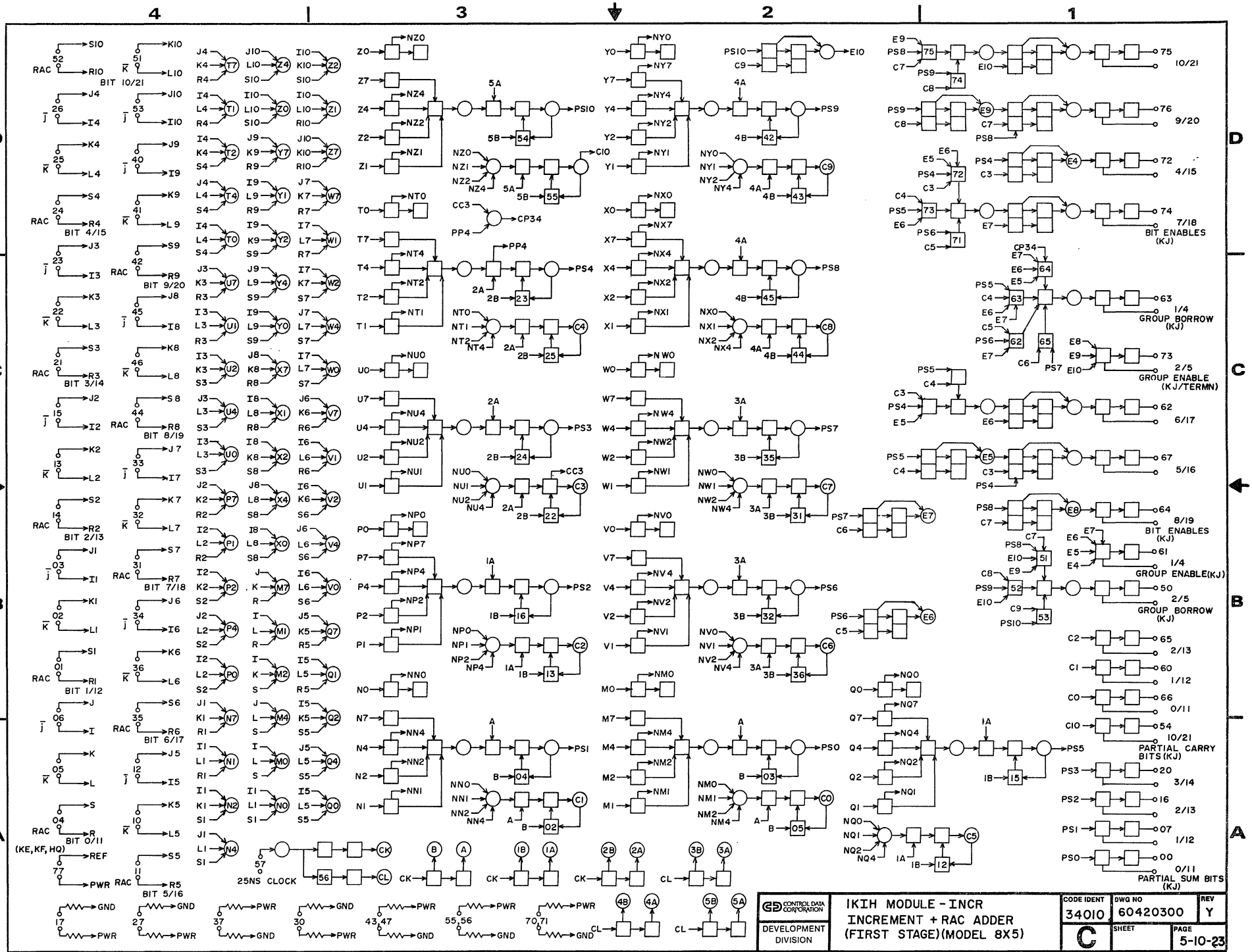
F=Forced

Partial sums PS0 through PS10 are determined by the following truth table.

$\overline{RAC}$	$\overline{j}$	$\overline{K}$	Condition
1	1	1	1
1	1	0	0
1	0	1	0
1	0	0	1
0	1	1	0
0	1	0	1
0	0	1	1
0	0	0	0

Partial carries PC0 through PC10 are determined by the following truth table.

$\overline{RAC}$	$\overline{j}$	$\overline{K}$	Condition
1	1	1	1
1	1	0	1
1	0	1	1
1	0	0	0
0	1	1	1
0	1	0	0
0	0	1	0
0	0	0	0


4
3
2
1
D
C
B
A
BIT 10/21
BIT 4/15
BIT 9/20
BIT 3/14
BIT 8/19
BIT 2/13
BIT 7/18
BIT 1/12
BIT 6/17
BIT 0/11
BIT 5/16
(KE, KF, HQ)
25NS CLOCK
7/18 BIT ENABLES (KJ)
1/4 GROUP BORROW (KJ)
2/5 GROUP ENABLE (KJ/TERMN)
8/19 BIT ENABLES (KJ)
1/4 GROUP ENABLE(KJ)
2/5 GROUP BORROW (KJ)
PARTIAL CARRY BITS(KJ)
PARTIAL SUM BITS (KJ)
CONTROL DATA CORPORATION
DEVELOPMENT DIVISION
IKIH MODULE - INCR INCREMENT + RAC ADDER (FIRST STAGE)(MODEL 8X5)
CODE IDENT 34010
DWG NO 60420300
REV Y
C
SHEET
PAGE 5-10-23
5K01-02

Two 4KI7 modules and a 3KJ7 module add RAC to the sum or difference of two 18-bit operands. Each 4KI7 module performs a partial computation on nine bits of the three operands. The upper input pin pairs receive complemented data from the Aj, Bj, or Xj register. The middle input pin pairs receive complemented data from the CIW register K designator or complementary (true or false) data from the Bk register. The lower input pin pairs receive data from the RAC register. The KI modules generate bit enables, bit borrows, group enables, and group borrows. These signals are sent to the 3KJ7 module where the computation is completed.

The following table shows the condition of the terms and output signals when the three operands are assigned arbitrary values. The output signals cause the 3KJ7 module to provide a result of 00236. (Note that Aj and K enter the 4KI7 modules in a complemented form.)

Groups	5			4			3			2			1			0		
Bits	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$\overline{RAC}$ = 777600	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
$\overline{Aj}$ = 777752	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	1	0
$\overline{K}$ = 777765	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	1
A8-A1 terms	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	X
B8-B1 terms	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	X
C8-C3 terms	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	X	X	X
Bit enables	1	1	0	1	1	0	1	1	1	1	0	1	0	1	1	1	1	1
Bit borrows	1	X	X	X	X	X	X	1	1	1	X	X	X	X	X	X	0	0
Group enables		0			0			X			0			0			X	
Group borrows		1			1			X			1			0			X	

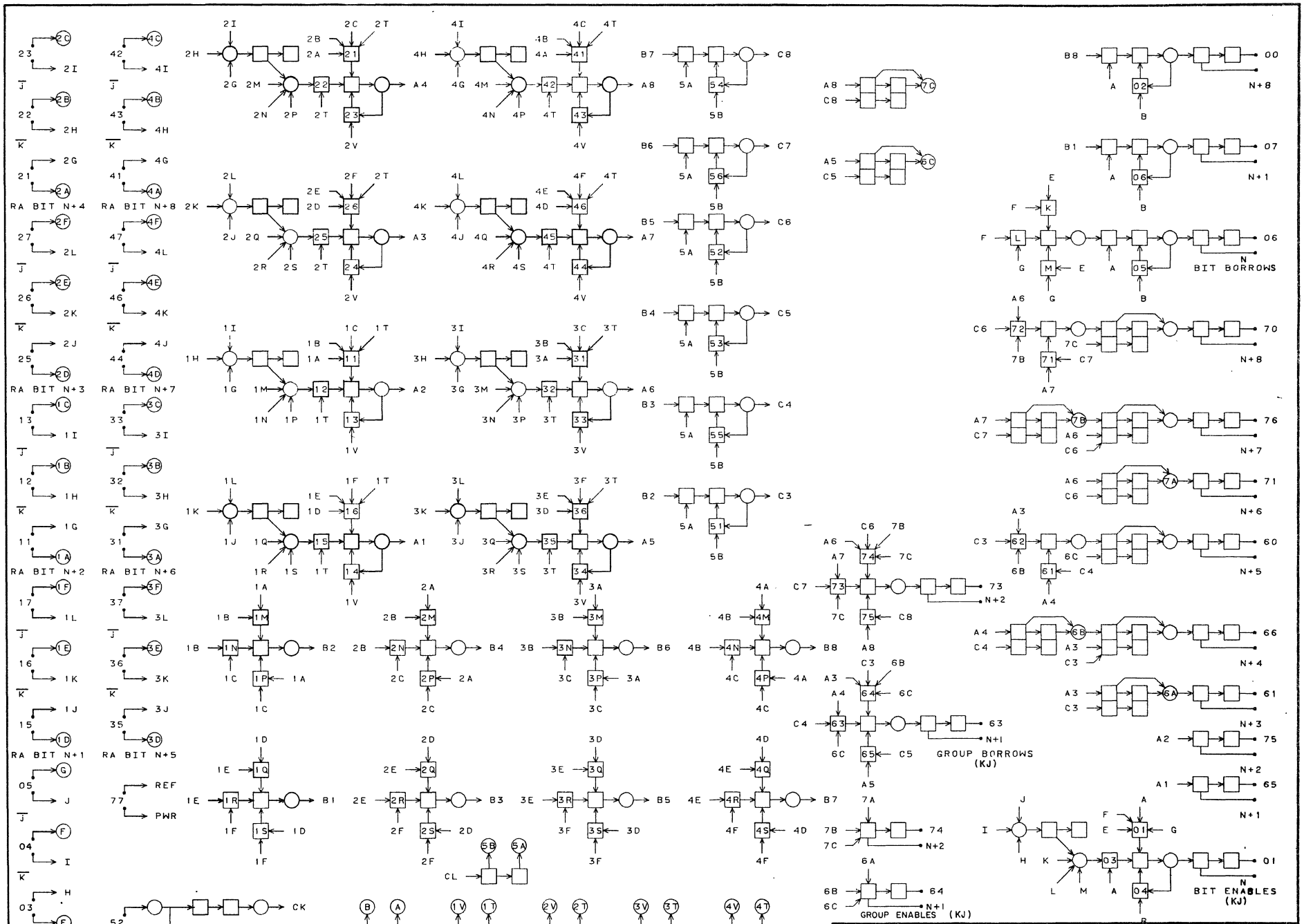
The A1 through A8, B1 through B8, and C3 through C8 terms are the key to determining the enables and borrows for three given input operands. Bit N, N+1, N+2 enables, and A1 through A8 terms are determined by the following truth table.

$\overline{RA}$	$\overline{j}$	$\overline{K}$	Condition
1	1	1	1
1	1	0	0
1	0	1	0
1	0	0	1
0	1	1	0
0	1	0	1
0	0	1	1
0	0	0	0

Bit N, N+1, N+8 borrows, and B1 through B8 terms are determined by the following truth table.

$\overline{RA}$	$\overline{j}$	$\overline{K}$	Condition
1	1	1	0
1	1	0	1
1	0	1	1
1	0	0	1
0	1	1	0
0	1	0	0
0	0	1	0
0	0	0	1

The C3 through C8 terms are the B2 through B7 terms left-shifted one bit position.



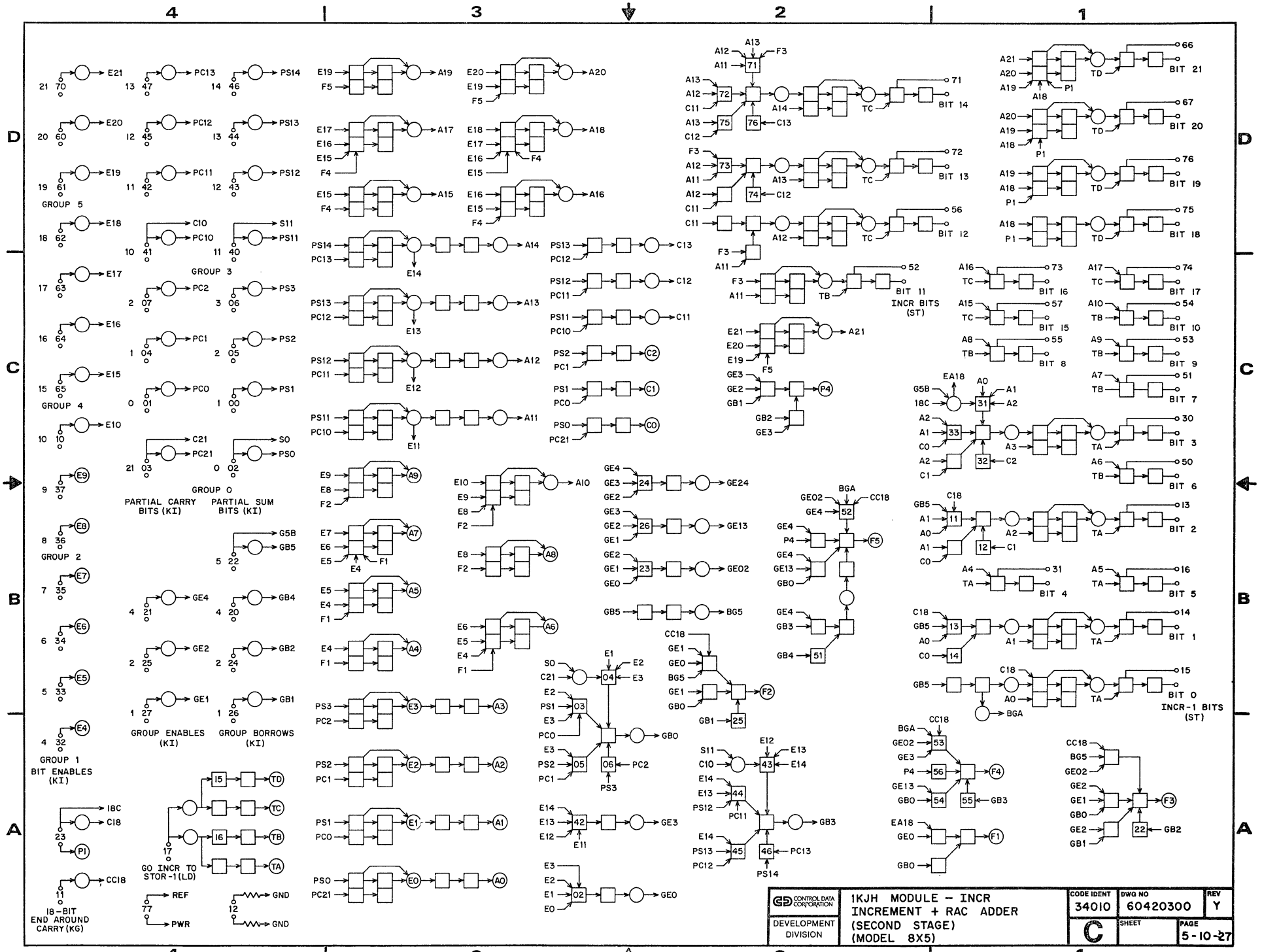
CONTROL DATA CORPORATION	4KI7 MODULE - INCR INCREMENT + RAC ADDER (FIRST STAGE) (MODELS 175, 7X0)	EQUIPMENT	
DEVELOPMENT DIVISION		SIZE C	DRAWING NO 60420300 REV. Y
		SHEET	PAGE 5-10-25

6L14-15

Two 1KIH modules and a 1KJH module add the 22-bit RAC to the sum or difference of two 18-bit operands. The 1KJH module completes the computation and sends the result to SAS when go increment to storage-1 is present. The result is based on input signals received from the 1KIH modules.

The following table shows the condition of various terms when the input signals are assigned an arbitrary value. (Note that the input signals are the same as the output signals shown in the 1KIH module table.)

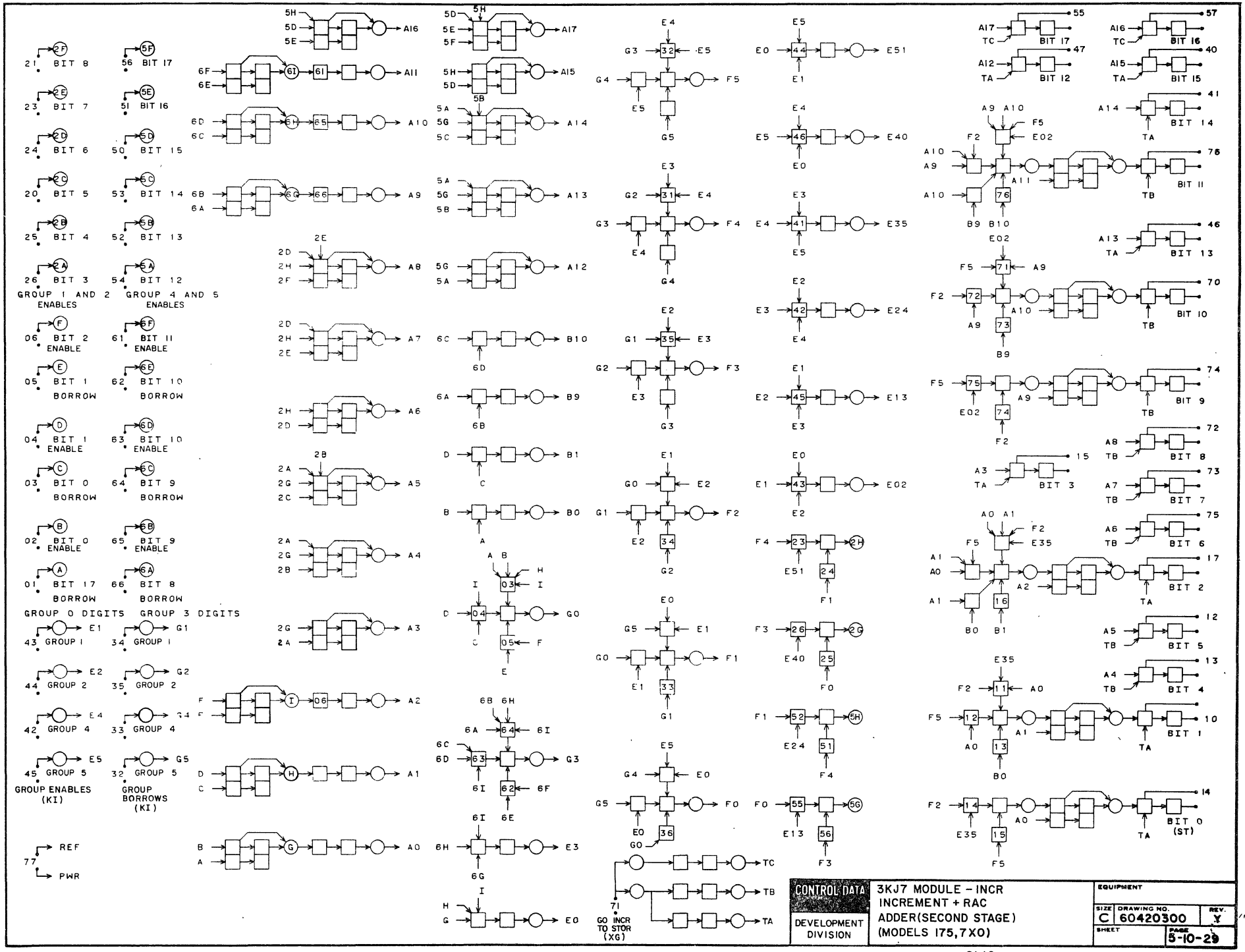
	Groups	5			4				3				2			1				0			
	Bits	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Input Signals	E10-E4 terms	1	0	1	1	1	1	1	X	X	X	X	1	1	1	1	1	1	1	X	X	X	X
	PS3-PS0 terms	X	X	X	X	X	X	X	0	0	0	0	X	X	X	X	X	X	X	0	0	1	1
	PC10,PC2-PC0 terms	1	X	X	X	X	X	X	X	1	1	1	1	X	X	X	X	X	X	X	1	0	1
	Group enables		N/A				1				X				1			1				X	
	Group borrows		1				0				X				0			0				X	
	18-bit end around carry	1	1	1	1																		
Internal Terms	E3-E0 terms	X	X	X	X	X	X	X	1	1	1	1	X	X	X	X	X	X	X	1	0	0	0
	C13-C11,C2-C0 terms	X	X	X	X	X	X	X	X	0	0	0	X	X	X	X	X	X	X	X	0	1	1
	Group enable		X				X				1				X			X				0	
	GE2-4,GE1-3,GE0-2 terms				2-4=1					1-3=1				0-2=0									
	F5-F1 terms		0				0				0				0								
	A21-A0 terms	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
Output	Incr-1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
	Corrected result	0		3			0				0				0			0				1	


GROUP 5
GROUP 4
GROUP 3
GROUP 2
GROUP 1
BIT ENABLES (KI)
GROUP 0
PARTIAL CARRY BITS (KI)
PARTIAL SUM BITS (KI)
GROUP ENABLES (KI)
GROUP BORROWS (KI)
18-BIT END AROUND CARRY (KG)
GO INCR TO STOR-1 (LD)
REF
PWR
GND
INCR BITS (ST)
INCR-1 BITS (ST)
BGA
CONTROL DATA CORPORATION
DEVELOPMENT DIVISION
1KJH MODULE - INCR
INCREMENT + RAC ADDER
(SECOND STAGE)
(MODEL 8X5)
CODE IDENT 34010
DWG NO 60420300
REV Y
SHEET C
PAGE 5-10-27
5K03

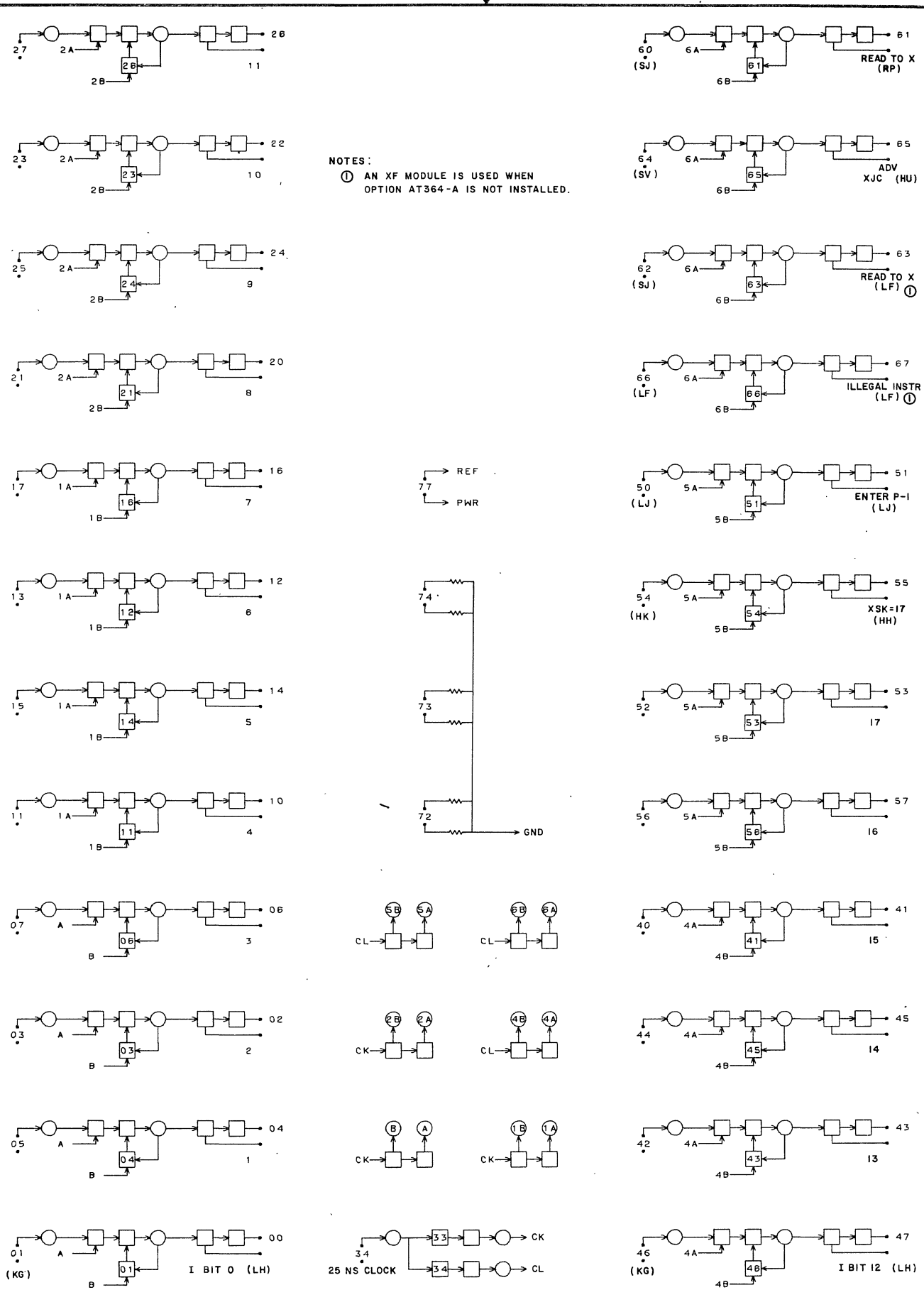
3KJ7 MODULE

Two 4KI7 modules and a 3KJ7 module add RAC to the sum or difference of two 18-bit operands. The 3KJ7 module completes the computation and sends the result to the SAS when the go increment to storage signal is present. This result is based on the condition of bits 0 through 15 enables; bits 0, 1, 8, 9, 10, 17 borrows; groups 1, 2, 4, 5 enables; and groups 1, 2, 4, 5 borrows. These input signals are generated by the 4KI7 modules. The following table shows the condition of various cage wire terms when the input signals are assigned an arbitrary value. (Note that the input signals are the same as the output signals shown on the 4KI7 module table.)

Groups	5			4			3			2			1			0		
Bits	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit enables	1	1	0	1	1	0	1	1	1	1	0	1	0	1	1	1	1	1
Bit borrows	1	X	X	X	X	X	X	1	1	1	X	X	X	X	X	X	0	0
Group enables		0			0			X			0			0			X	
Group borrows		1			1			X			1			0			X	
A15-A0 terms	X	X	1	1	1	1	0	0	0	1	0	1	1	0	0	1	1	0
B10, 9, 1, 0 terms	X	X	X	X	X	X	X	1	1	X	X	X	X	X	X	X	0	1
E5-E0 terms	X	X	X	X	X	X	X	X	X	X	X	X	0	0	0	0	0	0
F5-F0 terms	X	X	X	X	X	X	X	X	X	X	X	X	1	1	1	1	0	1
G5-G0 terms	X	X	X	X	X	X	X	X	X	X	X	X	1	1	1	1	0	1
Result = 000236	X	X	0	0	0	0	0	0	0	0	1	0	0	1	1	1	1	0



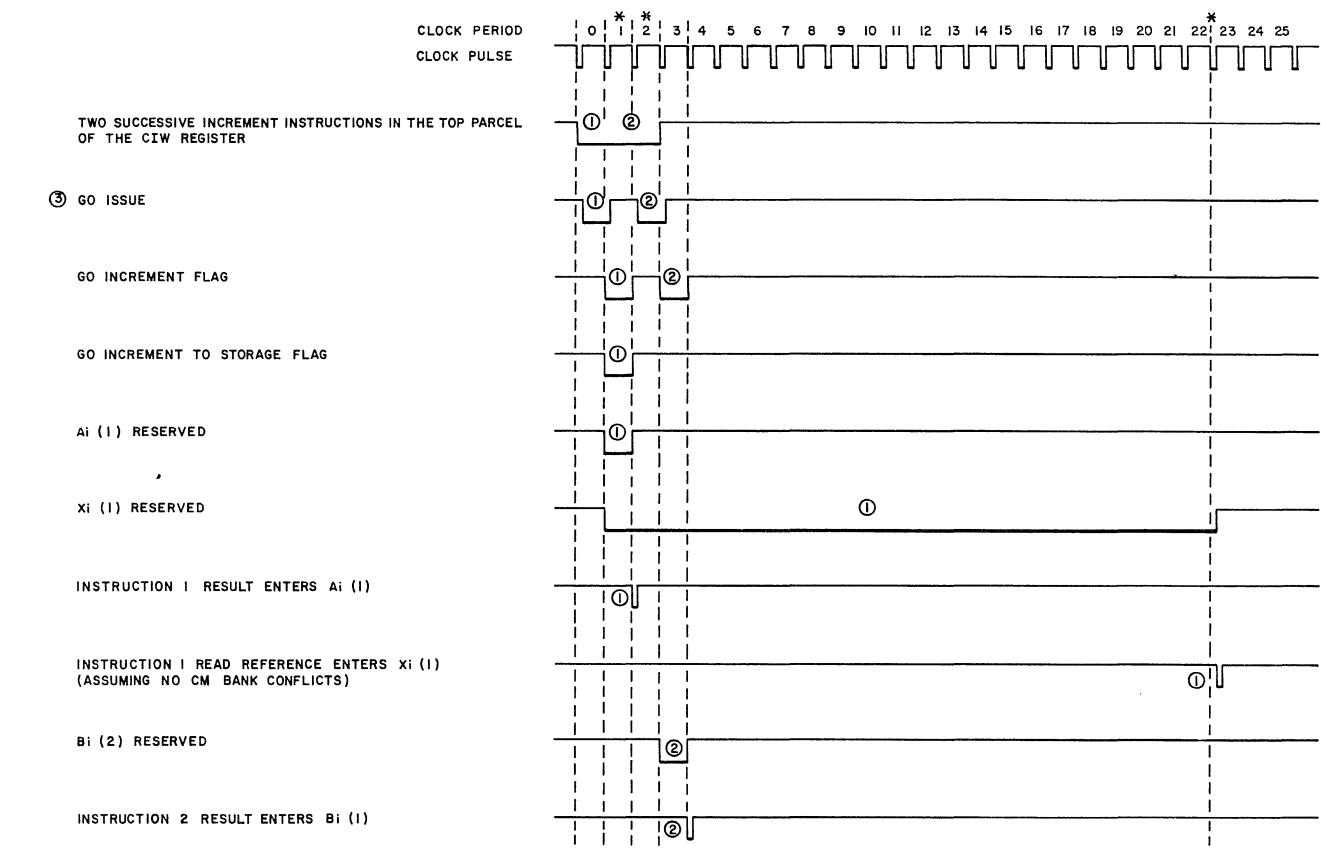
CONTROL DATA	3KJ7 MODULE – INCR INCREMENT + RAC ADDER(SECOND STAGE) (MODELS 175,7X0)	EQUIPMENT
DEVELOPMENT DIVISION		SIZE C · DRAWING NO. 60420300 · REV. Y
		SHEET · PAGE 5-10-29



CONTROL DATA DEVELOPMENT DIVISION

4KK7 MODULE - INCR
INCR TEST HLDG REGISTER
(MODEL 175,7X0)

CODE IDENT 34010 DRAG NO 60420300 SIZE C SHEET 6L12 REV Y 5-10-71



NOTES:

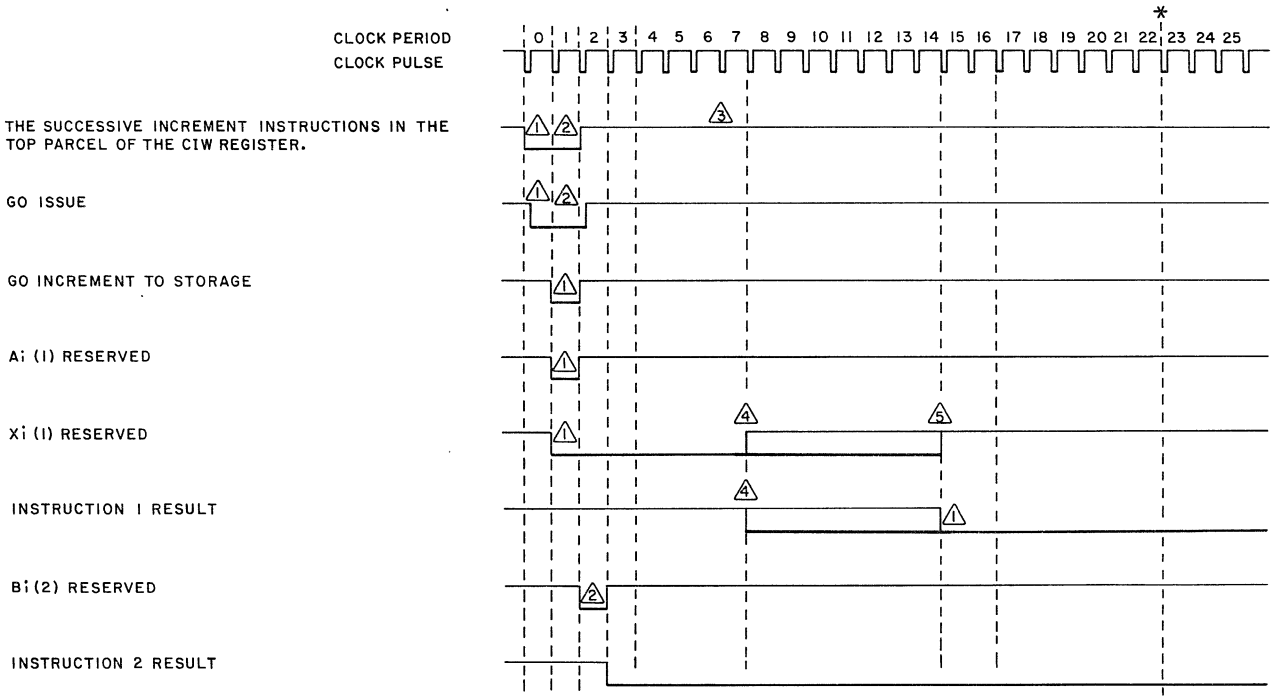
* GO ISSUE FOR INCREMENT MAY BE DELAYED ONE CLOCK PERIOD IF MEMORY ENABLE IS NOT UP.

① INSTRUCTION 1

② INSTRUCTION 2

③ ENABLE ISSUE IS DELAYED 4 CLOCK PERIODS (THEREFORE, GO ISSUE IS DELAYED A MINIMUM OF 4 CLOCK PERIODS) IN THE AA120-C. THIS DELAY IS REMOVED BY INSTALLING OPTION AT364-A.

CONTROL DATA DEVELOPMENT DIVISION	TIMING DIAGRAM INCREMENT INSTRUCTION (MODELS 175, 7X0)	CODE IDENT 34010	DWG NO 60420300	REV Y
		SIZE C	SHEET	PAGE 5-10-33



NOTES:

1. INSTRUCTION 1
2. INSTRUCTION 2
3. ISSUE MAY WAIT ONE CLOCK PERIOD FOR MEMORY ENABLE ON SINGLE/DUAL MODEL 875.
4. SINGLE/DUAL MODEL 875.
5. SINGLE/DUAL MODEL 865.

CONTROL DATA CORPORATION / DEVELOPMENT DIVISION	TIMING DIAGRAM INCREMENT INSTRUCTION (MODEL 8X5)	CODE IDENT C	DWG NO 60420300	REV Y
			SHEET	PAGE 5-10-35

COMMENT SHEET

MANUAL TITLE: CDC CYBER 170 Models 175, 740, 750, 760, 865, 875 Functional Units Hardware Maintenance Manual

PUBLICATION NO.: 60420300 **REVISION:** AA

NAME: _____

COMPANY: _____

STREET ADDRESS: _____

CITY: _____ **STATE:** _____ **ZIP CODE:** _____

This form is not intended to be used as an order blank. Control Data Corporation welcomes your evaluation of this manual. Please indicate any errors, suggested additions or deletions, or general comments below (please include page number references).

☐ Please Reply ☐ No Reply Necessary

CUT ALONG LINE

AA3419 REV. 4/79 PRINTED IN U.S.A.

NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

FOLD FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 8241 MINNEAPOLIS, MINN.

POSTAGE WILL BE PAID BY

CONTROL DATA CORPORATION

Publications and Graphics Division
ARH219
4201 North Lexington Avenue
Saint Paul, Minnesota 55112

CUT ALONG LINE

FOLD FOLD

CORPORATE HEADQUARTERS, P.O. BOX 0, MINNEAPOLIS, MINN. 55440
SALES OFFICES AND SERVICE CENTERS IN MAJOR CITIES THROUGHOUT THE WORLD

LITHO IN U.S.A.

CONTROL DATA CORPORATION