Burroughs

B 5281

CENTRAL PROCESSOR

TECHNICAL MANUAL

B 5000 DATA PROCESSING SYSTEM

PRINTED IN U.S.A. 2-16-65 B 5281.51

✓ CHANGES OR ADDITIONS

On "Revised" pages, the check mark (✓) shown to the left of items or subject titles indicates changes or additions since last issue.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

LIST OF ILLUSTRATIONS

LIST OF TABLES

SECTION 1

PREVENTIVE MAINTENANCE

1.1 DAILY

1. Run selected Test Routines or MTR if available. The selection would be based on:

   a. Possible troubles encountered during operation or P.M.

   b. Running all Test Routines available in rotation as soon as reasonable.

1.2 WEEKLY

1. Check that each fan is operating.

1.3 MONTHLY

1. Check Variable Bias Level. Refer to Subject 3.1.

2. Inspect (and replace, if necessary) air filters.

3. Check clock width and frequency. Refer to Central Control Technical Manual, Subject 3.2.

1.4 QUARTERLY

1. Clean fan screens

2. Check regulator output -12V, -1.2V, -4.5V according to Subject 5.6 of the Power Supply Manual.

1.5 SEMIANNUALLY

Lubricate Rotron Muffin fans with Anderol L-826 using special oil injector.

Oil Injector ..... Part No. 11838588
Oil ..... Part No. 11838596

PROCEDURE

The exhaust fans are lubricated by inserting the Oil Injector needle through a self-sealing rubber cap located in the center of the motor hub.

Note that on most units a Gold Seal label is mounted over the rubber plug; this series of fans is called the Gold Seal series.

1. Remove and clean the fan grill as necessary.

2. Remove air from Oil Injector by holding the needle up and pressing on the plunger.

3. Place Oil Injector needle at the center of circle marked on the Gold Label (on the 034 series place needle approximately 1/8" from the edge of the rubber cap).

4. Position the needle at an angle of approximately 45° to the surface of the label and point it toward the center of the rubber cap.

5. Pierce the label and the concealed self-sealing rubber cap located under the label.

6. Insert the needle approximately 1/4".

7. Depress the plunger of the Oil Injector to allow approximately 1/16" of oil to escape. Rotating the fan will relieve air pressure and allow oil to flow into the oil chamber.

SECTION 2

TROUBLESHOOTING

2.1 TEST ROUTINES

INTRODUCTION

The following is a list of test routines for the B 5500 Data Processing System. For a detail description of each test, refer to Test Routine Manual (1198594 2).

TR 5203	Compare Operators Test
TR 5204	Bit Operators Test
TR 5205	Relative Address Test
TR 5206	Logical Operators Test
TR 5207	Integer Stores Test
TR 5221	Core Memory Test
TR 5222	Drum Memory Test
TR 5230	Character Mode Test
TR 5331	Arithmetic Operators Test
TR 5332	Secl/Fetch Test
TR 5334	Sub-Level Test
TR 5336	Varf Lodl Store Test
TR 5340	I/O Test
TR 5555	Magnetic Tape Unit Test
TR 5556	S.P.O. Keyboard Test
TR 5557	Magnetic Tape Compatibility Test
TR 5558	Interaction Test
TR 5559	Paper Tape Reader Test
TR 5560	Paper Tape Punch Test
TR 5600	Data Com Control Sub-System Test
TR 5901	Chaining Routine
TR 5902	Card Lister Routine
TR 5903	I/O Utility Routine
TR 9995	Loader
TR 9996	Sub-Routine Test
TR 9997	Mantissa Adder Test
TR 9998	Control State And Interrupt Test

A-11173549-A Speed Up Test

A-11159258-E Disc File

SW 11169901 Disc Test Routine Part 1

SW 11169919 Disc Test Routine Part 2

SW 11169927 Disc Test Routine Part 3

SW A-11173426-B Marginal Tape Test

2.2 TEST SWITCHES AND INDICATORS

A description of the Test Switches and Indicators will be found in the D&D Manual, Subject 2.2.

Printed in U.S.A.

2.3 SPECIAL TOOLS

In addition to the normal tools provided for maintenance of the B 5000 system, the following special tools are also provided:

1. Diode-stick cutters.
2. Wire wrapping tools.
3. Cable-connector-pin insertion and removal tools.
4. Cover-removal tool.
5. Package handles.

DIODE STICK CUTTING TOOL (P/N 11838109)

The Diode Sticks provided as spares are uncut. The diode stick cutter is a plier-like device which can be used to cut the diode sticks as needed. Care must be taken when using the cutter to keep from breaking the bond between the diodes or resistors and the common bus. The diode stick tool must not be used for any other purpose.

WIRE UNWRAPPING TOOL (P/N 11838058)

The hand unwrapping tool (see Figure 2.3-1) is used when it is necessary to remove a wire from a pin. The tool has two ends; one end is for wires which are wrapped in a clockwise direction; the other end is for wires which are wrapped in a counter-clockwise direction. To use this tool, proceed as follows:

1. Determine the direction of wrap and insert the appropriate end of the tool over the pin.
2. Rotate the tool until the wire is sufficiently uncoiled so that it can be removed from the pin.

FIGURE 2.3-1 DUAL, RIGHT AND LEFT HAND, UNWRAPPING TOOL

WIRE WRAPPING TOOL (P/N 1183804-2)

The wire wrapping tool is a hand-wrapping tool and is shown in Figure 2.3-2. The tool will wrap a standard field change wire.

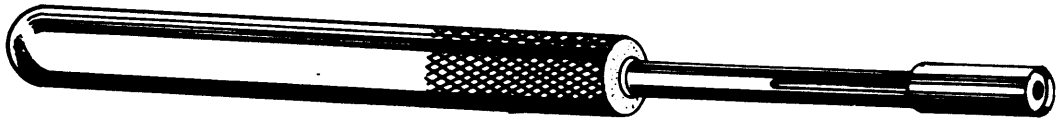


FIGURE 2.3-2 WIRE WRAPPING TOOL

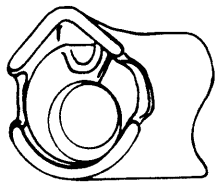
Figure 2.3-3, A through F, shows the steps used to wrap a connection. If a wire was previously wrapped, the portion of the wire which was wrapped cannot be used again. If the old wire is not long enough to strip off enough insulation to permit another wrap, a new wire must be routed in its place. To wrap a new wire proceed as follows:

1. Remove the insulation from the end of the wire. Approximately 1-1/2" of wire is required for a six-turn connection of 24-gauge wire.

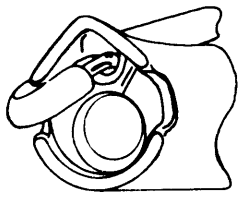
2. Place the tool over the wire as shown in Figure 2.3-3B.

3. Anchor the wire as shown in Figure 2.3-3C and insert the tool over the pin as shown in Figure 2.3-3D.

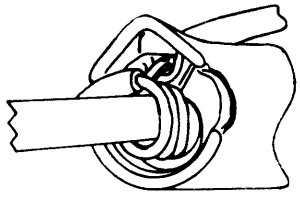
4. Rotate the tool in a clockwise direction. The wire will wrap around the pin as shown in Figure 2.3-3E and F. Too much pressure will cause the wire to bunch.



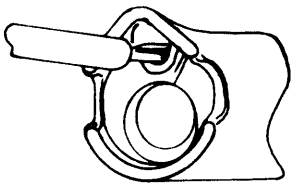
A-BIT AND SLEEVE



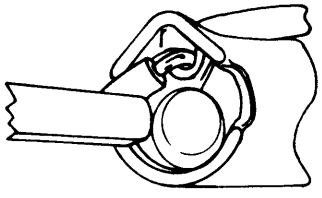
C-WIRE ANCHORING



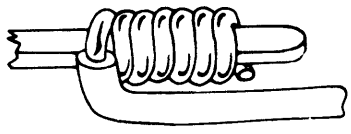
E-WRAPPING



B-WIRE INSERTION



D-TERMINAL INSERTION



F-FINISHED CONNECTION

FIGURE 2.3-3 WIRE WRAPPING

CABLE-CONNECTOR PIN INSERTION TOOL (P/N 11838075

Refer to Subject 4.3 of this manual for a description of how to use this tool.

CABLE-CONNECTOR PIN REMOVAL TOOL (P/N 11838067)

Refer to Subject 4.3 of this manual for a description of how to use this tool.

COVER REMOVAL TOOL (P/N 80551)

The Cover Removal tool is a 3/8" Allen-set-screw wrench. The short end is cut off to approximately 1/2" and a plastic handle is inserted on the long portion of the wrench.

PACKAGE HANDLES (P/N 77213)

For those packages or sticks which are not provided with handles special non-conducting handles are available. These handles must be removed from the package before the gates are closed.

2.4 RIN INDEX

RIN INDEX FOR THE B 5281 PROCESSOR UNIT (78494)

RIN NO.	INSTAL. TIME IN HOURS	PRE-REQUISITE	UNITS EFFECTED	DESCRIPTION
5004	2.0		102 ⇒ 147	Improves manner in which cables are secured.
5024	1.0		102 ⇒ 147	Replaces end panel washers with proper size.
5028	0.5		102 ⇒ 159	Extends lower limit of the -4.5V regulator.
5043	4.0		102 ⇒ 147	Removes a portion of the frame to relieve cable stress.
5044	0.5		102 ⇒ 147	Adds missing ground wire at location EEB1Y0
5047	4.0		102 ⇒ 147	Changes J00ZD1 input logic to function for word branch operators.
5051	4.0		102 ⇒ 159	Make stick cut changes, making hardware agree with documents.
5052	1.0		102 ⇒ 147	Replace clock cables with standard types.
5076	0.1 per cable		102 ⇒ 147	Key pin support for all yellow and green single key pin cable connectors.
5078	1.0		102 ⇒ UP	Change to double driver package to eliminate oscillations.
5094	4.0		102 ⇒ UP	Plug-in heatsink replacement to prevent the shorting of the collectors in common heatsinks.
5094S1	1.0		102 ⇒ UP	Supply transistors for installation of RIN 5094.
5100	4.0		102 ⇒ UP	Provide a quad connector retaining device to insure proper seating of the connector.
5110	2.5		102 ⇒ UP	"Z" level corrections for "D" Rack.

RIN NO.	INSTAL. TIME IN HOURS	PRE-REQUISITE	UNITS EFFECTED	DESCRIPTION
5111	3.0		102 ⇒ UP	"Z" level corrections for "E" Rack.
5112	0.5		102 ⇒ UP	"Z" level corrections for "J" Rack.
5117	0.5		102 ⇒ UP	Release a revised Equation Book.
5119	0.8		102 ⇒ UP	Removes extra termination of E04F'.

SECTION 3

ADJUSTMENTS

3.1 VARIABLE BIAS

Set the false level of the clock pulse to -0.5V by adjusting the variable bias packages. See chart below for locations:

A	AC7 A2	CC7 A2	EC7 A2
B	AC8 A2	CC8 A2	EC7 A2
J	CC8 A2	EC8 A2	
D	AC8 A2	CC8 A2	EC8 A2
E	AB8 N2	EB8 N2	

Reference: DA Pages

A Rack 65.96.87.0
B Rack 65.96.86.0
J Rack 65.96.88.0
D Rack 65.96.89.0
E Rack 65.96.85.0

NOTE

Verify that clock width has been properly adjusted prior to making the above adjustment. See Central Control Manual, Subject 3.2.

The above checks can be made at one point on each Clock Driver. Put scope on the following pins to see both clock width and Variable Bias level. At this point, Variable Bias level should be -0.6V.

A Rack	B Rack	J Rack	D Rack	E Rack
AA C6 B7	BA C7 B7	JC C7 B7	DA C7 B7	EA B7 P7
AC C6 B7	BC C7 B7	JE C7 B7	DC C7 B7	EE B7 P7
AE C6 B7	BE C6 B7		DE C7 B7	

SECTION 4

ASSEMBLY AND DISASSEMBLY

4.1 REGULATOR

See Section 4 of the Power Supply Manual.

4.2 WIRE WRAPPED PINS

REMOVAL

1. Remove wires with unwrapping tool.
2. If pin is bent, straighten it with long nose pliers.
3. Push on pin from the wire side with long nose pliers. As soon as the pin clears the block (package side), grasp the pin with the pliers and pull it out.
4. If the pin is broken off flush with the pin block, use a small drift punch or another pin held with pliers to drive the pin out.

REPLACEMENT

1. Insert the pin in the block from the package side of the gate. Make sure that the pin is inserted correctly (the contact side of the pin points away from the slot on the side of the pin hole).
2. Take the long-nose pliers and pull on the pin from the wire side until the pin is even with adjacent pins. Do not pull it too far or the pin block may be damaged.

NOTE

The M row pins are U shaped and do not extend through to the package side.

4.3 PINS IN WINCHESTER PLUG

INTRODUCTION

The pins in the Winchester plugs are held in place by a circular spring clip which clips on at the approximate center of the pin. When the pin is fully inserted, the spring clip expands into a groove in the block. This holds the spring in place.

REMOVAL

1. Obtain the cable-pin removal tool (P/N 11838067)

2. Slide the removal tool over the end of the pin until contact is made with the spring clip. Do not put any side strain on the removal tool since there is danger of breaking it. Apply slight pressure to the tool and the pin should become free.

REPLACEMENT

1. Use a crimper to connect the pin to its wire.

2. Insert the pin into the plug using the cable-pin insertion tool (P/N 11838075). Take care not to damage the spring clip.

4.4 PACKAGES

INTRODUCTION

Packages, diode sticks, and resistor sticks are removable. Handles are provided which fit into the package extensions.

CAUTION

Some handles must be removed before the gate is closed or the package may be damaged. See Subject 2.3.

WARNING

Power must be OFF before removing any element.

SECTION 5

INSTALLATION

5.1 CABLING - PROCESSOR A

INTRODUCTION

The listings in the following table show the cables which are required to be plugged in during installation of Central Processor A. Power quads to each gate should not be plugged in until the regulators have been checked out. Some cables in the listing may have been previously installed as part of system check out.

TABLE 5.1-1 INSTALLATION CABLING - PROCESSOR A

CABLE NO.	FUNCTION	FROM		TO		VIA TRAY
		UNIT	CONNECTOR	UNIT	CONNECTOR	
7-A	INTERRUPT & CONTROL	CC	EA C0 A7	PA	DA C0 N2	8F
8-AA	DP-MEMORY WRITE EXCHANGE	PA	DE C0-N7	CC	BA A0 A2	7F
8-AB	DP-MEMORY WRITE EXCHANGE	PA	DC B0 A7	CC	BA A0 N2	5F
9-AA	DP-MEMORY READ EXCHANGE	CC	AC A0 A7	PA	EE D0 A2	7F
9-AB	DP-MEMORY READ EXCHANGE	CC	AC A0 N7	PA	EE C0 N2	5F
10-A	INTERRUPT & CONTROL	PA	EA D0 A7	CC	EA C0 A2	7F
24-26	HEAT & EXC. CURRENT SENSE	PA	BB A3 Y9	PA VOLT. REG.	CS J1 01	NT
					CS J1 02	NT
					CT K1 02	NT
25-5	POWER (GROUND)	D & D	DA J1 08	PA	CU K1 07	2R
25-6	POWER (GROUND)	D & D	DA J1 08	PA	CA K1 07	2R
25-25	POWER (+20V)	D & D	DB L4 03	PA	CS K1 12	2F
25-34	POWER (+50V)	D & D	DB L1 03	PA	CT K1 04	2F
25-43	POWER (-33V)	D & D	DB J2 03	PA	CT K1 01	2F
45-A	INDICATOR INPUT	PA	AB B4 N7	D & D	AN J1	4F
46-A	INDICATOR INPUT & MANUAL CONTROL	PA	AD B4 N7	D & D	AN K1	4F
47-A	INDICATOR INPUT & MANUAL CONTROL	PA	AF B4 N7	D & D	AN L1	4F
48-A	INDICATOR INPUT & MANUAL CONTROL	PA	BF A4 A7	D & D	AN R1	4F
49-A	INDICATOR INPUT & MANUAL CONTROL	PA	BD D4 N7	D & D	AN S1	4F
50-A	INDICATOR INPUT & MANUAL CONTROL	PA	BB B4 N7	D & D	AT J1	4F
51-A	INDICATOR INPUT	PA	JE D0 N2	D & D	AT S1	4F
52-A	INDICATOR INPUT & MANUAL CONTROL	PA	JC C0 N2	D & D	AN N1	4F
53-A	INDICATOR INPUT & MANUAL CONTROL	PA	DE B0 N2	D & D	AT K1	4F
54-A	MANUAL CONTROL	D & D	AN M1	PA	AB C4 N7	4F
55-A	MANUAL CONTROL	D & D	AT R1	PA	JE C0 N2	4R

TABLE 5.1-1 (CONTINUED)

CABLE NO.	FUNCTION	FROM		TO		VIA TRAY
		UNIT	CONNECTOR	UNIT	CONNECTOR	
56-A	INDICATOR INPUT & MANUAL CONTROL	PA	DC A0 A2	D & D	AN P1	4R
57-A	INDICATOR INPUT & MANUAL CONTROL	PA	DA B0 N2	D & D	AT L1	4R
58-A	INDICATOR INPUT & MANUAL CONTROL	PA	EA B0 N2	D & D	AT M1	4R
59-A	MANUAL CONTROL & TEST SWITCH	D & D	AT T1	PA	EA C0 A2	4R
60-A	MANUAL CONTROL	D & D	AT N1	PA	EE B0 A2	4R
61-A	INDICATOR INPUT	PA	EE B0 N2	D & D	AT P1	4R
144-1A	POWER (GATE J)	PA	JC B0 N7	REG.	CS 07	NT
144-2A	POWER (GATE J)	PA	JE B0 N7	REG.	CS 08	NT
145-A	POWER (GATE A)	PA	AB B4 N2	REG.	CS 01	NT
146-A	POWER (GATE A)	PA	AD B4 N2	REG.	CS 02	NT
147-A	POWER (GATE A)	PA	AF B4 N2	REG.	CS 03	NT
148-A	POWER (GATE B)	PA	BB B4 N2	REG.	CS 04	NT
149-A	POWER (GATE B)	PA	BD B4 N2	REG.	CS 05	NT
150-A	POWER (GATE B)	PA	BF B4 N2	REG.	CS 06	NT
151-A	POWER (GATE D)	PA	DA B0 N7	REG.	CS 11	NT
152-A	POWER (GATE D)	PA	DC B0 N7	REG.	CS 12	NT
153-A	POWER (GATE D)	PA	DE B0 N7	REG.	CS 13	NT
154-A	POWER (GATE E)	PA	EA B0 N7	REG.	CS 14	NT
155-A	POWER (GATE E)	PA	EC B0 N7	REG.	CS 15	NT
156-A	POWER (GATE E)	PA	EE B0 N7	REG.	CS 16	NT
175	CLOCK (GATE A)	CC	EA C2 L6	PA	AA C5 L9	2R
					AC C5 L9	2R
					AE C5 L9	2R
176	CLOCK (GATE B)	CC	EA C2 L7	PA	BA C6 L9	2R
					BC C6 L9	2R
					BE C5 L9	2R
177	CLOCK (GATE D)	CC	EA C2 L8	PA	DA C6 L9	2R
					DC C6 L9	2R
					DE C6 L9	2R
178	CLOCK (GATE E)	CC	EA C2 L9	PA	EA B6 Y9	2R
					EE B6 Y9	2R
179	CLOCK (GATE J)	CC	EA C2 L5	PA	JC C6 L9	2R
					JE C6 L9	2R
197-1	GROUND (GATE A)	PA	GROUND BUS	REG.	CU K1 01	NT
197-2	GROUND (GATE B)	PA	GROUND BUS	REG.	CU K1 02	NT
197-3	GROUND (GATE D)	PA	GROUND BUS	REG.	CU K1 02	NT
197-4	GROUND (GATE E)	PA	GROUND BUS	REG.	CU K1 01	NT
197-17	GROUND (GATE J)	PA	GROUND BUS	REG.	CA J1 01	2R
*205	POWER (-19V)	PS	DA K2 B2	REG.	CA K1 04	NT
229	115V FAN	I/O-SS	FA K1 06/07	PA	FA K1 06/07	NT
231	115V CONVENIENCE	D & D	DF P1 03/04	PA	HB L1/L2 01/02	NT

* NORMALLY INSTALLED AS INDICATED IN SECTION 5.2 OF THE POWER SUPPLY MANUAL.

5.2 CABLING - PROCESSOR B

INTRODUCTION

The table listed below shows cabling for Central Processor B. The power quads should not be plugged in until the regulators have been checked out.

TABLE 5.2-1 INSTALLATION CABLING - PROCESSOR B

CABLE NO.	FUNCTION	FROM		TO		VIA TRAY
		UNIT	CONNECTOR	UNIT	CONNECTOR	
7-B	INTERRUPT & CONTROL	CC	EA C0 N7	PB	DA C0 N2	2F
8-BA	MEMORY WRITE EXCHANGE	PB	DE C0 N7	CC	BE A0 A2	8F
8-BB	MEMORY WRITE EXCHANGE	PB	DC B0 A7	CC	BE A0 N2	5F
9-BA	MEMORY READ EXCHANGE	CC	AE A0 A7	PB	EE D0 A2	8F
9-BB	MEMORY READ EXCHANGE	CC	AE A0 N7	PB	EE C0 N2	5F
10-B	INTERRUPT & CONTROL	PB	EA D0 A7	CC	EA C0 N2	3F
24-27	HEAT & EXCESS CURRENT SENSE	PB	BB A3 Y9	PB VOLT REG.	CS J1 01	NT
					CS J1 02	NT
					CS K1 02	NT
25-7	POWER (GROUND)	D & D	DA J1 07	PB	CU K1 07	2R
25-8	POWER (GROUND)	D & D	DA J1 07	PB	CA K1 07	2R
25-26	POWER (+20V)	D & D	DB L4 02	PB	CS K1 12	2F
25-35	POWER (+50V)	D & D	DB L1 02	PB	CT K1 04	2F
25-44	POWER (-33V)	D & D	DB J2 02	PB	CT K1 01	2F
45-B	INDICATOR INPUT	PB	AB B4 N7	D & D	AN J1	4R
46-B	INDICATOR INPUT & MANUAL CONTROL	PB	AB B4 N7	D & D	AN K1	4R
47-B	INDICATOR INPUT & MANUAL CONTROL	PB	AF B4 N7	D & D	AN L1	4R
48-B	INDICATOR INPUT & MANUAL CONTROL	PB	BF A4 A7	D & D	AN R1	4R
49-B	INDICATOR INPUT & MANUAL CONTROL	PB	BD D4 N7	D & D	AN S1	4R
50-B	INDICATOR INPUT & MANUAL CONTROL	PB	BB B4 N7	D & D	AT J1	4R
51-B	INDICATOR INPUT	PB	JE D0 N2	D & D	AT S1	4R
52-B	INDICATOR INPUT & MANUAL CONTROL	PB	JC C0 N2	D & D	AN N1	4R
53-B	INDICATOR INPUT & MANUAL CONTROL	PB	DE B0 N2	D & D	AT K1	4R
54-B	MANUAL CONTROL	D & D	AN M1	PB	AB C4 N7	4R
55-B	MANUAL CONTROL	D & D	AT R1	PB	JE C0 N2	4F
56-B	INDICATOR INPUT & MANUAL CONTROL	PB	DC A0 A2	D & D	AN P1	4F
57-B	INDICATOR INPUT & MANUAL CONTROL	PB	DA B0 N2	D & D	AT L1	4F
58-B	INDICATOR INPUT	PB	EA B0 N2	D & D	AT M1	4F
59-B	MANUAL CONTROL & TEST SWITCH	D & D	AT T1	PB	EA C0 A2	4F
60-B	MANUAL CONTROL	D & D	AT N1	PB	EE B0 A2	4F
61-B	INDICATOR INPUT	PB	EE B0 N2	D & D	AT P1	4F
144-1B	POWER (GATE J)	PB	JC B0 N7	REG.	CS 07	NT
144-2B	POWER (GATE J)	PB	JE B0 N7	REG.	CS 08	NT
145-B	POWER (GATE A)	PB	AB B4 N2	REG.	CS 01	NT

Continued on next page

TABLE 5.2-1 (CONTINUED)

CABLE NO.	FUNCTION	FROM		TO		VIA TRAY
		UNIT	CONNECTOR	UNIT	CONNECTOR	
146-B	POWER (GATE A)	PB	AD B4 N2	REG.	CS 02	NT
147-B	POWER (GATE A)	PB	AF B4 N2	REG.	CS 03	NT
148-B	POWER (GATE B)	PB	BB B4 N2	REG.	CS 04	NT
149-B	POWER (GATE B)	PB	BD B4 N2	REG.	CS 05	NT
150-B	POWER (GATE B)	PB	BF B4 N2	REG.	CS 06	NT
151-B	POWER (GATE D)	PB	DA B0 N7	REG.	CS 11	NT
152-B	POWER (GATE D)	PB	DC B0 N7	REG.	CS 12	NT
153-B	POWER (GATE D)	PB	DE B0 N7	REG.	CS 13	NT
154-B	POWER (GATE E)	PB	EA B0 N7	REG.	CS 14	NT
155-B	POWER (GATE E)	PB	EC B0 N7	REG.	CS 15	NT
156-B	POWER (GATE E)	PB	EE B0 N7	REG.	CS 16	NT
180	CLOCK (GATE A)	CC	EA C2 Y6	PB	AA C5 L9	2R
					AC C5 L9	2R
					AE C5 L9	2R
181	CLOCK (GATE B)	CC	EA C2 Y7	PB	BA C6 L9	2R
					BC C6 L9	2R
					BE C5 L9	2R
182	CLOCK (GATE D)	CC	EA C2 Y8	PB	DA C6 L9	2R
					DC C6 L9	2R
					DE C6 L9	2R
183	CLOCK (GATE E)	CC	EA C2 Y9	PB	EA B6 Y9	2R
					EE B6 Y9	2R
184	CLOCK (GATE J)	CC	EA C2 Y5	PB	JC C6 L9	2R
					JE C6 L9	2R
197-5	GROUND (GATE A)	PB	GROUND BUS	REG.	CU K1 01	NT
197-6	GROUND (GATE B)	PB	GROUND BUS	REG.	CU K1 02	NT
197-7	GROUND (GATE D)	PB	GROUND BUS	REG.	CU K1 02	NT
197-8	GROUND (GATE E)	PB	GROUND BUS	REG.	CU K1 01	NT
197-18	GROUND (GATE J)	PB	GROUND BUS	REG.	CA K1 04	NT
*206	POWER (-19V)	PS	DA M7 B2	PB	CA J1 01	2R
230	115VAC FAN	M-SS2	FA K1 06/07	PB	FA K1 06/07	NT
231	115VAC CONVENIENCE	D & D	DF P1 03/04	PB	HBL1/L2 01/02	NT
232	115VAC CONVENIENCE	M-SS2	HBL1/L2 01/02	PB	HB L1 01/02	NT

* NORMALLY INSTALLED AS INDICATED IN SECTION 5.2 OF THE POWER SUPPLY MANUAL.

NOTE

Refer to D & D DA's Page 56.02.03.0 for sensing wire deletion with two Processors.

SECTION 6

CIRCUIT ANALYSIS

See Power Supply Manual, Section 6.

Printed in U.S.A.

FUNCTIONAL DESCRIPTION

7.1 SINGLE LENGTH ADD (AD1L) - SINGLE LENGTH SUBTRACT (SU1L)

PURPOSE

Algebraically ADD or SUBTRACT the two Operands in the top of the stack. When the operation is complete, the "B" Register will contain the sum or difference. The "A" Register will be set to EMPTY, the "B" Register is set to FULL and the "B" Register Flag bit is set to ZERO.

SUMMARY OF OPERATION

When the "A" Register is equal to zero (W06L), the answer is contained in the "B" Register.

When the "B" Register is zero and the "A" Register is NOT zero ($\overline{W06L} \cdot W07L$), the "B" Register is replaced by the contents of the "A" Register.

If the "A" and "B" Registers equal zero ($W06L \cdot W07L$), the answer will equal zero and it will be contained in the "B" Register. Thus, all of the "B" Register is cleared. In either case, the operation is terminated.

Providing that the Mantissa signs and the exponents of the Operands are equal, the Mantissas are added and the sum placed in the "B" Register. If the sum exceeds 13 octal digits, the Mantissa of the sum is shifted right one octal place, rounded, and the exponent algebraically increased by one.

When the exponents of the Operands are equal but the Mantissa signs are unequal, the difference of the Mantissas with appropriate sign is placed in the "B" Register. If the exponents of the Operands are unequal, the Operands are aligned. When the alignment causes the smaller Operand to be shifted right 14 or more octal places, the larger Operand is the result. If the alignment causes less than 14 octal shifts, the last digit shifted out is checked for four or greater. Should it be four or greater, a round will result during addition.

If the signs of the Operand are equal, the Mantissas are added and the sum placed in the "B" Register. In case the sum does not exceed 13 octal digits, the last digit shifted out of the Register is used for rounding the result. When the sum is 14 digits, the Mantissa in the "B" Register is rounded to 13 digits.

When the signs of the Operand are unequal, the digits are complemented as they are shifted out of the Register during alignment. In effect, the equivalent of a 15 digit subtraction occurs in this latter case and the result is rounded to the 13 most significant digits of the 15 digit result.

When the result has an exponent of greater than +63, the Exponent Overflow bit is set in the Interrupt Register. The "B" register contains the correct Mantissa; Mantissa sign, and Exponent sign. The magnitude of the correct exponent is contained in the Exponent Field of the "B" register Modulo 64.

7.2 SINGLE LENGTH DIVIDE (DV1L)

PURPOSE

The Single Precision Divide Operator (DV1L) will divide the number in the "B" Register by the number in the "A" Register. The quotient is found in the "B" Register. Fourteen significant quotient digits are developed and then rounded to thirteen.

If the Mantissa of the number in the "B" Register is zero, the "B" Register is cleared. If the Mantissa of the number in the "A" Register is zero, the Divide by Zero Interrupt is set. In either case, the operation is terminated.

When the exponent overflows or underflows, the appropriate Interrupt bit is set. The correct exponent is found in the "B" Register, Modulo 64.

SUMMARY OF OPERATIONS

During stack adjustment, mark the "A" Register EMPTY. This is because the divisor will not be considered valid information after the operation is completed. Also, clear the Flag bit in the "B" Register to mark the information as an Operand. After the Registers have been LOADED, normalization will begin.

Set the Divide by Zero Interrupt if the "A" Register Mantissa is zero. Terminate the Operator if either Operand is zero. If neither Operand is zero, check the 13th octade. If it is zero, shift the Mantissa left and decrease the exponent of the associated word until the 13th octade is NOT zero.

When both Mantissas are normalized, the first cycle of the divide operation is allowed. If the sign of the "A" Register Mantissa is negative, complement the sign of the "B" Register Mantissa. In any divide operation opposite signs result in a negative result and similar signs result in a positive result.

Set Q02F ON to initiate exponent adjustments and complement the sign of the "A" Register exponent to cause the algebraic sum of the two exponents to ultimately result in a difference operation.

The process of division is basically similar to the usual principle of repetitive subtraction. The operation starts out in subtraction and increases the quotient digit until the remainder goes below zero. The cycle then uses the quotient predictor to gate the operation either into the same subtract cycle or, into an add to the complemented remainder. In each case, this forms the quotient digit in the fewest arithmetic operations.

When a digit has been formed in the 13th position of the quotient, the 14th is formed, examined and operations are terminated. If the last digit is 4 or more, the quotient is rounded.

The 42 bits of the "A" and "B" Registers are added and placed in the "B" Register. Since the "A" Register is a complement, this is the equivalent of a subtraction.

If the subtraction is successful, the remainder of the decrement is above zero. Count the quotient digit up one for each operation that will not cause the remainder to go below zero. When the operation causes it to go below zero, the combination will force the remainder to go into complement form. This indicates that the formation of the quotient digit is now completed.

The quotient predictor establishes whether the next quotient digit will be 4 or more (W04L), or, if it will be less than 4 ($\overline{\text{W04L}}$). If it is 4 or more, the operation will add a true "A" to a complement remainder (effectively a subtraction) and will count the next quotient digit down. When the sum of a complemented remainder and the true "A" overflow, the corrections to the quotient digit have been made and the remainder is in true form. This type of cycle is best thought of as a corrective adjustment of the quotient digit based on the number of times the add cycle proves the assumed value of the quotient digit to be in error.

If the quotient digit is less than 4, the operation is best formed by the subtract operation which was just exited. The remainder will be in complement form and shifted to the left one place to re-position it for the subtract cycle to follow.

If the quotient predictor shows the next digit will be zero, the cycle will continue shifting the "B" and "X" Registers and counting the "N" Register.

If there has been sufficient pulses, X13L vanishes and is replaced by $\overline{\text{X13L}}$. $\overline{\text{X13L}}$ signals a normalized quotient. Since all non-integer arithmetic operators must have normalized Mantissas, the divide operation will form the first significant quotient digit in either the first or second octade position of the temporary storage "X" Register.

When the quotient digit is in the first position, 13 shifts will bring it into X13L and the 14th digit will be formed. If it is in the second position, 14 shifts will be needed and the 15th digit will be formed. If the count of shifts in the "N" Register is 14, the first quotient digit formed is zero, and the existing value is 8 times larger than was expected. Therefore, reduce the exponent by one.

The rounding of the Mantissa will be an add of one into the "B" Register. If the exponent sign is negative and the high order bit is ON, set the Exponent Underflow Interrupt. If the exponent sign is positive and the high order exponent bit is ON, set the Exponent Overflow Interrupt.

7.3 SINGLE LENGTH MULTIPLY (MUIL)

PURPOSE

To algebraically MULTIPLY two Operands in the "A" and "B" Registers and leave the product in the "B" Register.

SUMMARY OF OPERATIONS

Adjust the stack as required to LOAD the two Operands into the "A" and "B" Registers. Mark the "A" Register as UNOCCUPIED and clear the Flag bit of the "B" Register. This insures that the "A" Register at the end of the operation is marked as EMPTY and that the final result is flagged as an Operand.

With the common requirement that both the "A" and "B" Registers be LOADED and a Single Precision Multiply Operation is being performed (MU1L); clear the "B" Register and terminate the operation if either the "A" or "B" Mantissa equals zero (W06L + W07L). When one of the multipliers is zero, the answer is zero.

If the "A" and "B" Registers are LOADED, the Mantissas of the "A" and "B" Registers are NOT zero ($\overline{W06L} \cdot \overline{W07L}$), and a multiply operation with the exponents of the "A" and "B" Registers NOT equal to zero (MU1L $\cdot$ $\overline{W71L}$ + MU1L $\cdot$ $\overline{W72L}$), the "A" and "B" Registers are normalized. When the most significant octade is equal to zero; shift the Mantissa left by octades, set the low order octade equal to zero, and count the exponent down one. This shift and count down of the exponent will continue until the Mantissa is normalized.

If the exponents of the "A" and "B" Registers are zero (J93L), turn ON Q05F to record the fact that the two Operands are integers (non-floating point). If the result is integer, do NOT normalize.

When both Operands are integers, or both Operands are normalized; initiate Multiply. The multiplier ("B" Mantissa) is shifted to the "X" Register. The Mantissa of the "B" Register is cleared to assemble the partial-final product.

The "B" Register exponent is counted plus one. This is to give the 13th additional count of the exponent. In a Multiply operation, if 13 digits are multiplied by 13 digits, the answer is 26 digits in length. However, since the 13 least significant digits of the final answer are lost, add thirteen to the exponent. One is added here and 12 more in the exponent Add operation.

Q02F going ON starts the Exponent Add Operation which goes on concurrently and independently with the Multiply. It terminates when the exponents have been added together.

Because the Exponent Add Operation is common to several operators, a separate instruction sheet has been made up for it. Refer to Subject 7.7, Pages 2 and 3.

If the least significant bit of the multiplier is OFF ($\overline{B01F}$) and sensed at the same time as we shift, it indicates that the multiplier is an even number. The multiplicand is then doubled by shifting it left by one bit position.

The least significant bit of the "A" Register (A01F) is set to zero and the status of the multiplicand is recorded by using the three least significant bits of the "M" Register as a counter. Count these up by one. This records the fact that the multiplicand has been doubled (2A).

Should the "N" Register be equal to zero (NEZL), set the high order digit of the "X" Register to zero. This insertion of a zero happens only on the first shifts of the "X" Register and is a flag between the multiplier digits and the digits of the answer that will be shifted into the "X" Register as developed.

In the case of a zero multiplier, just shift the "B" and "X" Registers until a non-zero multiplier is in X1.

When the multiplier is equal to one, add the contents of the "B" Register and the "A" Register and place the result back in the "B" Register.

If the multiplier digit in the "X" Register is 2 or 4, the left shift of the "A" Register Level (ALSL) is true and the least significant bit of the "M" Register is OFF ($\overline{\text{M01F}}$). This indicates that the multiplicand is in its original state (A) or in the complement of the original state ($\overline{\text{A}}$). The multiplicand is doubled by shifting "A" and the extension of "A" left one bit. The multiplicand counter M01 thru M03 is counted up one to record the new state of "A" (2A, $\overline{\text{2A}}$ or 4A, $\overline{\text{4A}}$).

If the multiplier digit is 3; do the first add, then double the multiplicand for the second add cycle.

If the right shift of "A" Level (ARSL) is true; shift "A" and the extension of "A" (M04 thru M06) right one bit. This halves the multiplicand. Count the multiplicand counter (M03 thru M01) down by one to record the status of the multiplicand. ARSL is formed by a combination of the state of the multiplicand and the value of the multiplier digit.

If the complement of "A" shift level is true (ACSL), this will indicate the next multiplier digit is -0 thru -4. Complement "A", the extension of "A" (M06 thru M04), and the Carry Flip-flop (Q01F). ACSL is developed from the combination of the state of the multiplicand (2A, A etc.), and the status of the current multiplier digit in the "X" Register.

Unconditionally, add the partial product in the "B" Register to the multiplicand in the "A" Register and place the results back in the "B" Register. The "A" Register remains the same.

If the multiplier has not been exhausted ($\overline{\text{N14L}}$), then shift the multiplier digit right by one to bring the next multiplier digit into the sensory octade (X03 thru X01). Shift the "B" Register right by octades, shift the least significant octade of "B" to the "X" Register, and shift the extension of the "B" Register (M10 thru M8) to the high octade of the "B" Register (B13). This shift in conjunction with the shift of the "X" Register, shifts the partial product and the multiplier digits right. This has the same effect as placing the next partial product left, which is normally done when multiplying.

The current multiplier digit will have already been sensed and action taken accordingly.

"N" Register is counted up one to keep track of the number of multiplier digits used. When all 13 multiplier digits have been accounted for and there is no carry; or, if

if the "N" Register has counted to 14 (N14L) indicating that all of the multiplier digits have been used plus a CARRY ADD; transfer the control to normalize and exit.

If the final result in the "B" and "X" Registers is not normalized (B13L) and it was a non-integer multiply ($\overline{Q05F}$); then normalize the "B" Register by shifting the "B" and "X" Registers right until the "B" Register is normalized. Count the exponent of the "B" Register down by one for each shift.

If the "B" Register is normalized ($\overline{B13L}$) and there is a digit in X13 that is 4 or greater (X30F); turn ON Q01F to accomplish a round operation.

If the multiply was an Integer Multiply Operation (Q05F) and the Mantissa of the "B" Register is zero (W07L) then transfer the contents of the "X" Register to the "B" Register, clear the exponent of the "B" Register and clear the "M" Register.

Clear the "A" Register in preparation for the final add (only necessary to complete the round operation).

If an Integer Multiply and the "B" Register Mantissa is zero (Q05F • W07L), or if a Non-Integer Multiply and the "B" Register Mantissa is normalized ($\overline{B13L}$); terminate the operation.

If the Exponent Add Operation or the normalizing resulted in an Exponent Overflow (M11F) and the sign of the exponent is negative (B46F); set the Exponent Underflow Interrupt Bit in the Interrupt Address Register.

If the Exponent Add Operation resulted in an Exponent Overflow (M11F) and the sign of the "B" Register exponent is positive ($\overline{B46F}$); set the Exponent Overflow Bit in the Interrupt Address Register.

7.4 INTEGER DIVIDE (DV3L)

PURPOSE

The Integer Divide Operator (DV3L) will divide the number in the "B" Register by the number in the "A" Register. The quotient is found in the "B" Register. Only the digits which are integers are developed in the quotient. For all conditions, the "A" Register is marked EMPTY, the "B" Register is marked FULL and the "B" Register Flag bit is reset.

The "B" Register is cleared when the Mantissa in "B" is zero. The Divide by Zero Interrupt is set when the Mantissa in the "A" Register is zero. In either case, the operation is terminated.

Normalized Operands are checked for exponent value and if the exponent of the number in the "A" Register is larger, the "B" Register is cleared and the operation terminated. If the exponents are equal or the "A" Register exponent is less than the "B" Register exponent, divide is initiated and develops either the integer portion or 13 digits of the quotient, whichever is less. If the exponents at the end of the operation are not equal, the Integer Overflow Interrupt is set and normal floating point expression of the result is accomplished. If they are equal, the exponent of the "B" Register is set to zero.

SUMMARY OF OPERATION

Stack adjustment and normalization takes place, if required.

If the exponent of the number in the "A" Register is still greater than that in the "B" Register, the answer is zero, so terminate the operator. If it is equal to or less than that in the "B" Register, initiate divide and proceed as described in Single Precision Divide with the exception of the exponential conditions listed above.

The divide cycle may be terminated at any point within the operation if the exponential values so indicate. No exponential arithmetic need be performed at this time.

The basic comparison of exponents is accomplished prior to or during the cycle. If, at any time when a divide operation is completed and the X12 position is zero, the quotient is not yet normalized. Therefore, the exponent should be reduced to keep track of the exponential value of the remainder by counting down the exponent of the "B" Register.

If the exponents become equal, or an X12 position has a value other than zero shifted in; the last position of the quotient will be formed and the operation will be set to terminate.

If the exponent of the "B" Register is still greater than that of the "A" Register, 13 positions of quotient are not enough for an integer expression. Therefore, set

the Integer Overflow Interrupt, change the sign of the "A" Register exponent for proper exponent arithmetic, and set Q02F to initiate it.

If the exponents are equal, clear the exponent portion of the "B" and "M" Registers. An integer result has been obtained if Q04F is OFF. Exponential Interrupts are established as in Single Precision Divide.

7.5 REMAINDER DIVIDE (DV4L)

PURPOSE

The Remainder Divide Operator (DV4L) will divide the Operand in the "B" Register by the Operand in the "A" Register. Upon the successful completion of the division, the remainder will be found in the "B" Register. For all conditions the "A" Register is marked EMPTY, the "B" Register is marked FULL, and the "B" Register Flag bit is reset.

The "B" Register is cleared when the Mantissa in the "B" Register is zero. The Divide by Zero Interrupt is set when the Mantissa in the "A" Register is zero. In either case, the operation is terminated.

When the Mantissa of the remainder is zero, the "B" Register is cleared. If a non-integer quotient is developed, the Integer Overflow Interrupt is set and the "B" Register is cleared. Normal floating point expression of the remainder is accomplished including setting of Exponential Interrupts.

SUMMARY OF OPERATION

The Remainder Divide Operator is identical in function to the Integer Divide Operator with exceptions in the handling of the quotient and certain exponential functions.

Stack adjustment and normalization of Operands takes place if required. Operations examine exponents if the Operands are normalized. Then, if the exponent of the Operand in the "A" Register is still greater than that in the "B" Register, initiate Divide and proceed as described in Integer Divide.

The exceptions to Integer Divide are as follows:

1. Do NOT alter the sign of the Mantissa in the "B" Register because the remainder sign is not affected by the operations.

2. When the exponents become equal and the next quotient digit should be obtained by adding, the remainder is in complement form. Perform an Add Restore Operation. Clear the "A" Register to set up the add.

3. An integer quotient cannot be calculated when the X12 position has a value other than zero shifted into it. Therefore, clear the "B", "Q" and "A" Registers, and set the Integer Overflow Interrupt. Exponential Interrupts are established as in Single Precision Divide.

EXPONENTIAL ARITHMETIC OPERATION

In any Multiply or Divide (except Remainder Divide), all that is required for parallel functioning of the process of exponent Add or Subtract operations is that the Operator Logic sets Q02F.

Note that some of the values of the "J" Register require Q02F to be OFF for the operator to proceed. This will assure that the pulses required to bring about the corrected exponent values are taken before the operator is exited.

The Exponent Arithmetic consists of an algebraic addition of the contents of 2 binary counters. They are the "A" register exponent [M07F & A (45 thru 40)] and the "B" register exponent [M11F & B (45 thru 40)].

The algebraic addition is of the serial type. The value of the "A" Register bits are used to increment or decrement the "B" Register. In addition, during the exponent arithmetic operations, a constant amount (+12 for Multiply; -12 for Divide) must be added algebraically to the "B" Register.

The exponent arithmetic is performed under control of a binary counter which is the "M" Counter. This counter has two output levels: W15L; M = 2 or 4; and W16L; M ≠ 2, 4 or 9.

As soon as Q02F is set, the exponent arithmetic is started. At first the "M" Counter points to zero, and W16L is true. If A40F is ON, "B" is incremented or decremented depending upon whether the signs of the exponents are equal or not. Simultaneously, A40F is reset. With A40F OFF, "M" is counted up one, "A" and "B" are shifted right one bit, and B40F (first bit of the result) is shifted into M07F. Refer to Figure 7.7-1.

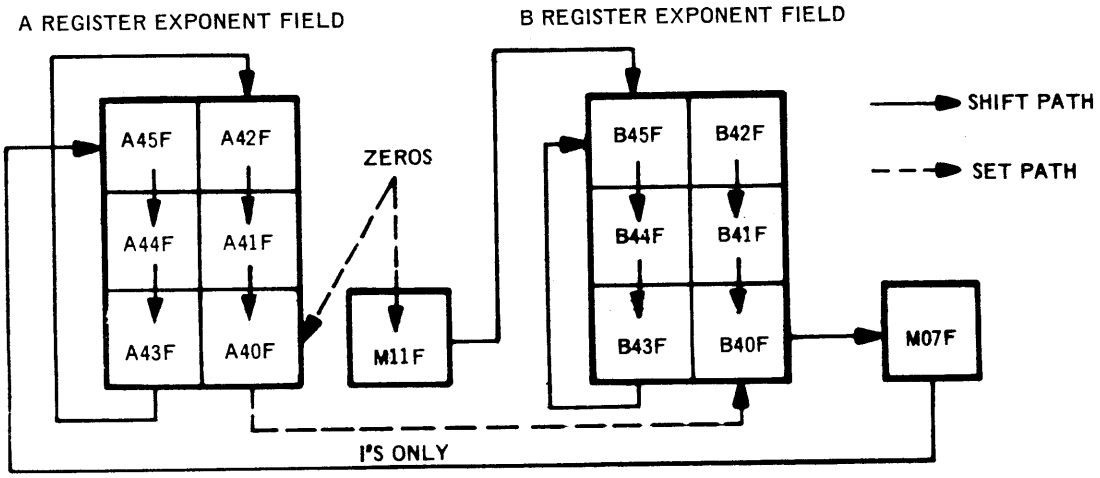


FIGURE 7.7-1 EXPONENT ADD SHIFT PATHS

The state of B40F is stored in M07F. A40F is lost, but its state was inserted into B40F before the shift. Now the cycle repeats, BUT, since all bits have retained their relative position with respect to one another; the operation has a binary value twice that of the last one. Each shift and count clears the bit in A40F and inserts its state in B40F as indicated for each binary bit value.

When the "M" Counter reaches 2 or 4, B40F will be incremented unconditionally if the operation is either Multiply and "B" exponent positive, or, Divide and "B" exponent negative. B40F will be decremented if the operation is either Multiply and "B" exponent negative, or, Divide and "B" exponent positive.

It should be noted that the arithmetic logical levels will not be shown on flow charts but they are implied by the flow chart on which the operation appears.

7.6 DOUBLE LENGTH ADD/SUBTRACT (AD2L-SU2L)

PURPOSE

The Double Length Operand in the "A" and "B" Registers is algebraically added to or subtracted from the Double Length Operand addressed by the "S" Register. The double length result is left in the "A" and "B" Registers and the "S" Register is reduced by two. Bit positions 48 thru 40 (flag, signs, and exponent) of the least significant word of the double length result are reset. All non-zero results are normalized. For a Subtract operation, the sign of the Mantissa in the "A" Register is reversed and an Add Operation is performed.

PROVISIONS

S_1 e_1 E_1 M_1 m_1 = Operand at top of stack

S_2 e_2 E_2 M_2 m_2 = Other Operand

S e E M m = Result

S = Mantissa sign

e = Exponent sign

E = Exponent value

M = High order half of Mantissa

m = Low order half of Mantissa

SUMMARY OF OPERATION

Adjust the stack so that the smaller Mantissas "m_1 and m_2" are added first. Should both registers be occupied, either on entry or after stack adjustment; await completion of any loading before starting exponent comparison. At the completion of this state, the registers will have the following conditions:

1. "A" Register - Most significant half of 2nd Operand (M_2).
2. "B" Register - Most significant half of 1st Operand (M_1).
3. "X" Register - Least significant half of 1st Operand (m_1).

EXPONENTS EQUAL

Exponents are only in the most significant half of the 2 Operands.

Place the least significant Mantissa (m_1) of the 1st Operand into the "A" Register and store the most significant Mantissa (M_2) of the 2nd Operand in the "X" Register. LOAD the least significant Mantissa (m_2) of the 2nd Operand into the "B" Register. Count the Stack Address down by one to address the 2nd half Mantissa (m_2).

EXPONENTS UNEQUAL

"B" Register Not Normalized

If the exponent of the "B" Register is greater than the exponent of the "A" Register and the Mantissa of the "B" Register is not normalized, an attempt will be made to equalize the exponents by normalizing the "B" Register.

"A" Register Normalized - Scale "B" Register

If the exponent of the "A" Register is greater than the exponent of the "B" Register and the "A" Register is normalized, then the "B" Register is scaled and its exponent counted up in an attempt to equalize the exponents.

Operand Exchange Required

If the exponent of the "B" Register is greater than the exponent of the "A" Register and the "B" Register is normalized; or, if the exponent of the "A" Register is greater than the exponent of the "B" Register and the "A" Register is NOT normalized; then an Operand Interchange must take place to allow scaling or normalizing as necessary. This is done by placing the proper values in the "B", "X" and "A" Registers.

If the Exponent of the "B" Register is greater than the exponent of the "A" Register and the "B" Register is normalized, if the Operands have been interchanged and either the "B" Register Mantissa is NOT zero or, the "N" Register count is NOT 14; then a second Operand interchange is initiated.

The value of the digits shifted out by scaling is checked. If the digit to be shifted out is 4 or greater (X03F), set the logical toggle Q01F to prepare for a possible round.

When the "N" Register equals 14 and the "B" Register Mantissa is zero, the double length Operand is zero. The exponent of the "A" Register is placed in the "B" Register. This will equalize the exponents. When the exponents are equal and either 0 or 2 Operand interchanges have been accomplished, add the 39 bits of the "B" Register Mantissa to the 39 bits of the "A" Register Mantissa and place the results back into the 39 bits of the "B" Register. Place any CARRY into the extension of the "B" Register (M10 thru M08). These operations will now place the addition of the two minor Mantissas into the "B" Register and place the exponent of the "A" Register into the "B" Register. This is the exponent of the final answer and it will stay in the "B" register during the Add of the most significant Mantissas. If either 0 or 2 Operand interchanges have been accomplished, then the most significant Mantissa of the 2nd Operand (M_2) must be addressed by increasing the "S" Register.

If there had been a carry out of the 13th octade (W13L), then it would be necessary to add it in on the Add operation of the major Mantissas.

Initiate a LOAD of the "A" Register if the logical toggle Q06F is OFF. Q06F being OFF indicates that there are no Multiply operations involved. The most significant half of the 2nd Operand (M_2) will be LOADED. Upon completion of the LOADING of the "A" Register, interchange the 39 bits of the "X" Register with the 39 bits of the "B" Register. This places the first half of the 1st Operand (M_1) into the "B" Register, and the result of the addition of the minor Mantissas into the "X" Register. The "S" Register is counted down by one in preparation for the final storage setting and to Add the major Mantissas (M_1 and M_2).

If the logical toggle Q03F is ON (Internal Add), or, after it is turned ON (Internal Subtract); Add the contents of the "A" and "B" Registers and place the results back into the "B" Register. Count the stack down by one (to prepare it for final storage) and transfer the contents of the "X" Register to the "A" Register Mantissa. (The "X" Register contained the result of the Add of the least significant Mantissas m_1 and m_2.)

Should there be an overflow on the Internal Add operation, scale for Overflow. Unconditionally scale the double length result in the "B" and "X" Registers by shifting the "X" Register right by octades. Also shift the least significant digit of the "B" Register into the "X" Register and shift the "B" Register (Mantissa only) right by octades. Because this scaling is a result of an overflow, a 1 is placed in the most significant digit of the "B" Register. The exponent is counted up one to keep the same relative value for the final answer set up.

Set up for final normalizations if:

1. There is an overflow on Subtract, or

2. There is no overflow on Add.

When there is no overflow on the Internal Subtract operation, decomplement the minor Mantissa for the final answer. Unconditionally set the logical toggle Q02F to keep the Internal Subtract condition. Place the complement of the "A" into the "A" Register, shift the Mantissa of the "B" to the "X" Register and clear the "B" Register. Q01F is turned ON to be added into the decomplemented Mantissa. The exponent of the result is in the "B" Register at this time and this portion of the "B" Register remains undisturbed. Add the decomplemented "A" Register to the cleared "B" Mantissa and Q01F, then place the result back into the "B" Register. This places the decomplemented least significant result in the "B" Register.

Add Q01F to this decomplement. If there is no overflow, turn OFF Q01F. If there is an overflow, leave Q01F ON so that it may be added into the decomplement of the most significant Mantissas.

Transfer the "X" register which contains the result of the Add of the most significant Mantissas into the "A" register. This transfer prepares for the decomplement of the most significant Mantissa.

Unconditionally turn Q02F OFF and place the complement of the "A" Mantissa back into the "A" Register. With the logical toggle Q02F ON for the first clock pulse, transfer the "B" Mantissa (least significant result) into the "X" Register and clear the Mantissa portion of the "B" Register.

With Q02F OFF (2nd clock pulse), Add the decomplemented "A" Mantissa to the cleared "B" Mantissa plus any CARRY from the previous decomplement (Q01F) and place the result back into the "B" Register. This places the result of the most significant Operands into the "B" Register with the result of the least significant Operands in the "X" Register. Complement the sign of the Mantissa to complete the answer set up.

Shift the "B" and "X" Registers left, by octades, if the result is NOT normalized. Set 0 into the least significant end and count down the exponent by one and the "N" Register up by one for each shift until either the result is normalized or the answer is discovered to be zero.

When the result is normalized or, if already normalized on entry into this state; transfer the contents of the "B" Register to the "A" Register (all 48 bits), and transfer the 39 bits of the "X" Register into the 39 bits of the "B" Register. This places the double length result in the "A" and "B" Registers, with the exponent signs and the most significant half of the result in the "A" Register.

If the sign of the exponent is negative and there has been an exponent underflow, set the Underflow Interrupt Bit ON in the Interrupt Address Register.

If the sign of the exponent is positive and there has been an exponent overflow, set the Exponent Overflow Bit ON in the Interrupt Address Register.

Unconditionally set the flag, signs, and exponent of the 2nd word of the result equal to ZERO. Allow fetch and terminate the operation.

7.7 DOUBLE LENGTH DIVIDE (DV2L)

GENERAL DESCRIPTION

The double length Operand in the two top locations of the stack are algebraically divided by the double length Operand contained in the "A" and "B" Registers. The double length result is found in the "A" and "B" Registers and the new top of the stack is the location below the original Operands.

Bit positions 48 thru 40 of the 2nd word of the result are set to zero. The Flag bit of the 1st word is set to zero. Both Registers are marked FULL.

If the Divisor is zero, the Divide by Zero Interrupt is set and the double length dividend addressed in the top of the stack is placed in the "A" and "B" Registers. When the dividend is zero, the "A" and "B" Registers are cleared. In either case, the operation is terminated.

The correct Exponent, Exponent Sign, Mantissa Sign, and 26 digits of quotient are developed. Exponential Overflow or Underflow Interrupts are set if the condition exists. The correct exponent Modulo 64 is found in exponent field.

7.8 DOUBLE LENGTH MULTIPLY (MU2L)

GENERAL DESCRIPTION

The double length Operand in the "A" and "B" Registers is algebraically multiplied by the double length Operand addressed by the "S" Register. The double length result is left in the "A" and "B" Registers and the "S" Register is reduced by two. The least significant word of the double length result is contained in bits 39 thru 1 of the "B" Register. The remainder of the word, bits 40 thru 48, are zero. The Flag Bit of the most significant word of the double length result is set to zero. All non-zero results are normalized and the "A" and "B" Registers are marked OCCUPIED.

Both double length Operands are normalized upon entry. If either Operand has a Mantissa of zero, the "A" and "B" Registers are set to all zeros and the operation is terminated.

The normalized double length Mantissas of the two Operands are multiplied if neither double length Operand has a Mantissa of zero. Twenty-seven digits of the 52 digit product are retained. The product is normalized and truncated to a 26 digit result.

When the exponent of the result is greater than +63 or less than -63, the Exponent Overflow Bit or Exponent Underflow Bit is set respectively in the Interrupt Register. The result in the "A" and "B" Registers contains the correct double length Mantissa, Mantissa Sign, and Exponent Sign. The magnitude of the correct exponent is contained in the exponent field of the "A" Register and is Modulo 64.

PROVISIONS

$S_1\ e_1\ E_1\ M_1\ m_1$ = Initial double length multiplier (two top Operands in stack).

$S_2\ e_2\ E_2\ M_2\ m_2$ = Initial double length multiplicand (3rd & 4th Operands in the stack).

$S_1\ e_3\ E_3\ M_3\ m_3$ = Normalized double length multiplier with adjusted exponent.

$S_2\ e_4\ E_4\ M_4\ m_4$ = Normalized double length multiplicand with adjusted exponent.

$M_6\ \&\ m_6 = M_4 \times m_3$ (1st Partial Product)

$M_7\ \&\ m_7 = M_3 \times m_4 + m_6$ (2nd Partial Product)

$M_8\ \&\ m_8 = M_3 \times M_4 + M_7$ (3rd Partial Product)

$M_9\ \&\ m_9 = M_8\ \&\ m_8 + M_6$ (4th Partial Product)

$S\ e\ E\ M\ m = M_9\ \&\ m_9 + m_7$ (Final Result)

S = Mantissa Sign

e = Exponent Sign

E = Exponent

M = High order half of Mantissa

m = Low order half of Mantissa

SUMMARY OF OPERATION

The execution of the Double Precision Multiply consists of three operators which are MU1L, MU2L and AD2L. The execution of J = 5, 6 and 7 is a part of Single Precision Multiply. The execution of J = 8 occurs in 4 sub-cycles. During the 4th sub-cycle, the operator is changed from a Double Precision Multiply to a Double Precision Add. The execution then continues thru J = 9, 10, 11, 14 and 15 as a Double Precision Add.

The double length multiplier (M_1 & m_1) is obtained from the stack to allow normalization in the "B" and "X" Registers. When the normalization of the double multiplier has been completed, the least significant word (m_3) is stored back into the former location of the least significant word of the double multiply (m_1). The most significant normalized word of the double multiplier M_3 is stored into Memory at the original M_1 Address of the stack. The adjusted exponent of M_3 is transferred from the "B" Register to the exponent position of the "A" Register. The double length multiplicand (M_2 & m_2) is obtained at this time.

The "E" Register will initiate a LOAD of the "B" Register with the least significant word of the double multiplicand (m_2) with EWZL received from the store of M_3. The Memory operation initiated at this time results in another Memory Access which will now LOAD the "B" Register with the most significant word of the double multiplicand (M_2). At the same time, the least significant word of the double multiplicand (m_2) is transferred to the "X" Register.

When the double length multiplicand (M_2 & m_2) is normalized, an exponent Add is initiated and the sign of the result is set. If either Operand turns out to be zero when normalization is attempted, exit is made to an ANSWER = 0 operation. The normalized most significant word of the double multiplicand (M_4) is stored in the location of M_2 in the stack.

The "B" and "X" Registers are shifted to the left to normalize the double length multiplier or multiplicand. As the Registers are shifted to the left, the "N" Register is incremented to record the shift, a zero is set into the low order position of the "X" Register (X1) and the exponent of the most significant word is decremented (M [11]&B [45 ⇒ 40] -1) for each shift. This is only true when Q05F is OFF. When Q05F is ON, it indicates that the Double Precision Multiply has been initiated from Double Precision Divide, and the exponent is NOT counted down.

The above operation continues until the Operands are normalized (indicated by significance in B13), or the "N" Register shift counter equals 13, and the "B" Register Mantissa equals zero (N13L • W07L) in which case the Operand is zero.

During the normalizing of M_2 and m_2, the sign bit of the "B" Register (B47F) is complemented if the sign bit of the "A" Register (A47F) is ON. This action sets the proper sign of the result into the sign bit position of the "B" Register. (Multiplication - Like signs = Positive, Unlike signs = Negative.) The normalized least significant word of the double multiplicand (m_4) is stored in the stack at the Address of the original word m_2.

When the Memory Write Access has been obtained (MWOF) from the store of the "A" Register, the contents of the "B" Register (M_4) are transferred to the "A" Register. This results in M_4 now being in the "A" Register, and m_3 is in the "X" Register. An exponent Add is initiated by setting the logical flip-flop Q02F ON. At the same time, the exponent of M_4 in the "B" Register is incremented by one.

Sub-Cycle 1: Initiate Single Precision Multiply

1. Multiplicand = M_4 (located in the "A" Register)
2. Multiplier = m_3 (located in the "X" Register)
3. M_6 & m_6 = M_4 x m_3 (1st Partial Product)
4. Initiate store of S e E M_6
5. A READ of m_4 from the stack is initiated.
6. Initiate Sub-Cycle 2 by the reading of M_3 from the stack.

The least significant half of the 1st Partial Product (m_6) is transferred from the "X" Register to the "B" Register. The least significant half of the original multiplicand (m_4) is transferred from the "A" Register to the "X" Register.

Sub-Cycle 2: Initiate Single Precision Multiply

1. The "B" Register contains m_6.
2. Multiplier = m_4 (located in the "X" Register).
3. Multiplicand = M_3 (located in the "A" Register).
4. M_7 & m_7 = M_3 x m_4 + m_6 (2nd Partial Product).

During the 2nd sub-cycle, the state of the high order octade of the "X" Register is temporarily stored in the "M" Register. This action retains the high order significance of the least significant half of the 2nd Partial Product (m_7). This significance will be used during the final normalization of the result.

The stack Address will be counted down by one to properly Address the LOAD of the "A" Register (M_4) which is initiated at this time.

Sub-Cycle 3: Initiate Single Precision Multiply

1. The "B" Register contains M_7.
2. Multiplicand = M_4 (located in the "A" Register).

3. Multiplier = M_3 (located in the "X" Register).

4. M_8 & m_8 = M_3 x M_4 + M_7 (3rd Partial Product).

5. Initiate a READ of S e E M_6 from the stack.

The operator level is changed from Double Precision Multiply to Double Precision Add. The least significant part of the 3rd Partial Product (m_8) is transferred from the "X" to the "B" Register. The most significant part of the same product is transferred from the "B" to the "X" Register. This is in preparation of the Add of m_8 to M_6.

Sub-Cycle 4: Initiate Double Precision Add

The following action takes place after the "T" Register is changed from MU2L to AD2L.

Part of the logic contained in the Double Precision Add portion of the flow chart does not apply to the Double Precision Add function of the Double Precision Multiply.

1. An addition of the "B" Register (M_6) plus the "A" Register (m_8) takes place.

2. The result of this addition is the least significant half of the 4th Partial Product (m_9).

   a. At this time, the sign of the exponent and the exponent answer e E which were temporarily stored in the "A" Register, are transferred to the "B" Register in preparation for setting up the final answer.

   b. Also at this time with Q06F ON, the "B" and "X" Registers are interchanged. This places the most significant half of the 4th Partial Product (M_8) into the "B" Register, and the least significant half of the 5th Partial Product (m_9) into the "X" Register.

3. With Q03F ON (Internal Add), the 39 bit Mantissa of the "B" Register is added to the 39 bit Mantissa of the "A" Register plus Q01F (CARRY). The result is placed back into the "B" Register Mantissa.

   a. This operation adds M_8 to zero plus Q01F, resulting in M_9 (the most significant half of the 4th Partial Product.

4. The least significant half of the 5th Partial Product (m_9) is transferred from the "X" to the "A" Register.

5. The Stack Address is decremented by one, resulting in the final address (the next location in sequence following the four Operands used during this operator).

6. The most significant half of the 4th Partial Product (M_9) is scaled if there was an overflow during the addition.

7. The "B" Register exponent (B45 ⇒ B40) is incremented by one to record the change in the value of the Mantissa.

8. If the "B" and "X" Registers must be normalized, the high order octade of the least significant half of the 3rd Partial Product (m_7), which was temporarily stored in the "M" Register, is now transferred to the least significant octade in the "X" Register.

9. The final double length answer is placed in the proper Registers.

10. The operator is terminated with a possible Interrupt if the exponent either overflowed or underflowed.

11. If during normalization of the Operands, or during the Double Precision Add, and Operand is determined to be equal to zero; the "A", "B" and "M" Registers are cleared and the operator is terminated.

7.9 LOGICAL AND (LOAL)

PURPOSE

If a bit position is on in both the "A" and "B" registers, the bit will remain on in the "A" register. The "A" register Flag Bit is set to the state of the "B" register Flag Bit.

SUMMARY OF OPERATION

Stack adjustment takes place to load the "A" and "B" registers.

When both registers are full, the False sides of the "B" register are gated to the corresponding KCL inputs of the "A" register.

The "B" register is marked empty.

A48F is replaced by B48F.

Exit the operator.

7.10 LOGICAL OR (LOOL)

PURPOSE

For all bit positions of the "A" register, except the Flag Bit, set the bit to one if the corresponding bit position in either the "A" or the "B" register is one. The Flag Bit of the "A" register is set to the state of the Flag Bit of the "B" register.

SUMMARY OF OPERATION

The stack is adjusted to load the "A" and "B" registers if necessary.

When both registers are loaded, the True outputs of the "B" register bit positions are gated to the corresponding JCL inputs of the "A" register.

A48F is replaced by B48F and the "B" register is marked empty.

7.11 LOGICAL EQUIVALENCE (LOEL)

PURPOSE

The function of this operator sets a one in each position of the "B" Register, except the Flag bit, when the corresponding bit positions of the "A" and "B" Registers are equal. A zero is set in each position of the "B" Register, except the Flag Bit when the corresponding bit positions of the "A" and "B" Register are not equal.

SUMMARY OF OPERATION

Upon entry into the operator the "A" and "B" Registers are loaded if not already loaded.

The "A" and "B" Registers are shifted left octally with the "N" Register counting the shifts. This allows each octade to pass through A16 and B16 where the comparison of bits is made. The result of this comparison, which is set into B1, is shifted at the same time.

The comparison which takes place is a bit comparison, A48 and B48, A47 and B47, A46 and B46, and if the bits are alike, both zero or both one, the corresponding bit position in B1 (B01F, B02F, B03F) is set to one. If the bits are not equal, the corresponding bit position in B1 (B01, B02, B03F) is set to zero.

The Flag bit of the word in the "B" Register (B48F) is retained by setting B03F to the state of B48F when comparing the octade containing the Flag bit (B48F•N=0).

When the "N" Register equals 15, the words in "A" and "B" have been compared and the result set into the "B" Register has been shifted a sufficient number of times to place each octade in its position with the Flag bit in B48. Terminate the operator and mark the "A" Register empty.

7.12 LOGICAL NEGATE (LONL)

PURPOSE

This operator will compliment every bit position of the "A" register except the flag bit.

SUMMARY OF OPERATION

Load the "A" register if necessary.

With the "A" register loaded, compliment every bit in "A" except A48F.

Q01F is complimented redundently.

7.13 COMPARE OPERATORS:

COMPARE:	B Greater than A	BGAL - 0225
	B Greater than or Equal to A	BGEL - 0125
	B Equal to A	BEQL - 4425
	B Less than or Equal to A	BLEL - 4125
	B Less than A	BLAL - 4225
	B Not Equal to A	BNEL - 0425

PURPOSE

The relational operators perform comparisons on the two top operands in the Stack. The operands are removed from the stack and the results of the comparison are placed in the top of the stack. The operands may be in an unnormalized form and the required normalizing will take place in the comparison operation. For the relational operators; operands of zero; minus zero and a zero mantissa, with a non-zero exponent, are considered equal. Flag bits are ignored.

SUMMARY OF OPERATION

The operand in the "B" register is algebraically compared with the operand in the "A" register. Depending on the operator level, if the value of the operand in the "B" register is algebraically greater than, greater than or equal to, equal to, less than or equal to, less than, not equal to, the value of the operand in the "A" register, the low order bit of the "B" register is set to one and all other bits of the "B" register are set to zero. Otherwise, all bits of the "B" register are set to zero. The "A" register is set to empty.

7.14 COMPARE FIELD EQUAL (CFEL) - COMPARE FIELD LOW (CFLL)

PURPOSE

Compare a field of bits in the two top words of the stack. The number of bits to be compared is contained in the repeat-count field of the operator. If the repeat-count field is zero, the compare is true and the "A" register so marked.

SUMMARY OF OPERATION (COMPARE FIELD EQUAL)

A field in the "A" register, starting at the bit position addressed by the "G" and "H" registers, is compared with a corresponding length field in the "B" register, starting at the bit position addressed by the "K" and "V" registers and proceeding towards the low order bit positions.

The length of the fields in the registers is specified by the six high order bits of the operator. The comparison is terminated by the comparison of the number of bits specified or the comparison of the low order bit of either register.

If all of the corresponding bits of the fields compared are equal, the low order bit of the "A" register is set to one, and the rest of the "A" register is set to zero. If any of the corresponding bit positions of the fields compared are not equal, all bit positions of the "A" register are set to zero. The contents of the "B", "G", "H", "K", and "V" registers are restored to their original value.

SUMMARY OF OPERATION (COMPARE FIELD LOW)

A field in the "A" register, starting at the bit position addressed by the "G" and "H" registers, is compared with a field in the "B" register, starting at the bit position addressed by the "K" and "V" registers and proceeding towards the low order bit positions.

The length of the fields in the registers is specified by the six high order bits of the operator. The comparison is terminated by the comparison of the number of bits specified or by the comparison of the low order bit position of either register.

If the magnitude of the field compared in the "B" register is less than the magnitude of the field compared in the "A" register, the low order bit of the "A" register is set to one and all other bit positions of the "A" register are set to zero; otherwise, all bit positions of the "A" register are set to zero.

The contents of the "B", "G", "H", "K", and "V" registers are restored to their original values.

7.15 SYLL. BRANCH FWD. COND. (BFCL), SYLL. BRANCH BKWD. COND. (BBCL), WORD BRANCH FWD. COND. (JFCL), WORD BRANCH BKWD. COND. (JBUL)

PURPOSE

To branch the program forward or backward "n" number of program words or syllables. The condition required to branch is B01F=0. If B01F=1 no branch takes place and the program continues with the next syllable in sequence. If B01F=0 and the word in "A" is a Descriptor (A48F=1) branch to the address contained in the 15 least significant bits of "A". If B01F=0 and the word in "A" is an operand, branch to the relative address defined by the 10 least significant bits in "A".

SUMMARY OF OPERATION

If the "A" or "B" Register is not loaded, adjust "A" and "B" Registers and the stack until "A" and "B" Registers are loaded. When the "A" and "B" Registers are loaded, or if occupied on entry, the status of B01F is checked to determine whether a branch is called for. If the least significant bit of "B" is ON, set AROF and BROF to zero. Since there will be no branch, the words in A and B are no longer valid. Allow a normal Fetch.

If "A" and "B" Registers are occupied and the least significant bit of "B" is OFF (AROF•BROF•$\overline{\text{B01F}}$) allow the branch operator to continue.

If "A" is a Descriptor and is present in Core Memory (A48F•A46F), transfer the contents of "A" to "B" so the address portion of "A" can be transferred from "B" to "C" Register. The contents of B15 ⇒ B1 are transferred to "C" Register. A Fetch from this address is initiated.

Set "L" to zero to unconditionally branch to the first syllable of the word addressed by the Descriptor.

If the Descriptor is not in Core Memory, set the Presence Interrupt condition and terminate the operator.

If the word in "A" Register is an Operand, the contents of "C" Register are transferred to "M" Register. "L" and "M" Registers are then modified by the 12 least significant bits of "A" Register (A12 A1). The modification is accomplished by using the 10-bit address adder plus incrementing "L" Register as indicated by the 2 least significant bits of "A" Register. Associated with the counting of "L" is carry logic which increments or decrements the word address in "M" when the syllable address (L) is counted beyond word length. Since all branching in word mode is referenced to the next word in sequence a final count L + 1 is executed for the branch backward operators as well as the branch forward operators. The operator inspects the least significant bit in "A". If this bit = 1 then "L" Register is counted up or down by one and the "A" Register shifts right 1 binary place. Again the operator inspects A01F but this time, (if A10F = 1) the bit has a weight of 2 and "L" is counted up or down twice. If Word Branch is desired by finding T11F ON, the "L" modifications above are by-passed. The entire "A" Register is added or subtracted as below, "L" is reset and normal branch operations are allowed.

Add the contents of "A" [10 ➧ 1] + Q01F to the contents of "M" [10 ➧ 1] and leave the result in "M" [10 ➧ 1]. Transfer the contents of the modified "M" Register to "B" in preparation for the "B" to "C" transfer. Transfer the 15 least significant bits of "B" to "C". "C" Register will now contain the modified address. Allow a Fetch from the address indicated by "C". The Fetch will load "P" Register with the new group of syllables. Terminate the operator.

7.16 SYLL. BRANCH FWD. UNCOND. (BFUL), SYLL. BRANCH BKWD. UNCOND. (BBCL), WORD BRANCH FWD. UNCOND. (JFUL), WORD BRANCH BKWD. UNCOND. (JBUL)

PURPOSE

Branch the program forward or backward "n" number of program words or syllables. "n" is contained in 12 least significant bits of an operand in "A" Register. "n" can be a value of 0 thru 4096. If the word in "A" Register is a descriptor, branch to the address contained in the 15 least significant bits of "A".

SUMMARY OF OPERATIONS

If the "A" Register is not loaded, adjust "A" and "B" Registers and the stack until "A" Register is loaded. When the "A" Register is loaded, or if occupied on entry, the status of "A" is checked to determine whether it is a Descriptor or an Operand. If it is a Descriptor, the Presence bit is checked. If the Descriptor is not in Core Memory, set the Interrupt condition and terminate the operator. If A46F is ON, interchange "A" and "B" Registers, so the address portion of "A" can be transferred from "B" to "C" Register. Set "L" to zero to unconditionally branch to the first syllable of the address contained in the Descriptor. The contents of B15 ⇒ B1 are transferred to "C" Register. A Fetch from this address is initiated, the original contents of "B" Register are transferred from "A" to "B" Register and the operator terminates. If "A" Register is occupied and contains an Operand (AROF• $\overline{\text{A48F}}$), transfer "C" Register to "M" Register for modification by the Operand.

BRANCH FORWARD UNCONDITIONAL

The modification occurring at this time uses only the two low order bits of "A" Register. They are used to increment "L" to the correct syllable within an address. Shift "A" right one bit. When executing BFUL and the low order bit of "A" is ON (A01F), increment "L" one. A01F would then contain the 2 bit of A1 octade indicating that "L" must be counted up 2 if the bit is ON. A39 ⇒ A1 is shifted right one bit because N02L now exists, which shifts A03F into A01F, and "N" is counted up one. N03L will exist. "L" will be incremented when the final shift of "A" occurs (N03L) to point "L" at the correct syllable. If "L" was not incremented at N03L time, it would be pointing at one less than the desired syllable. If "L" is equal to 3, increment "M" Register one. Set Q09F to one to set up the logic for the address adder which will be used to add the contents of A10F ⇒ A01F to "M" Register.

BRANCH BACKWARD

If the least significant digit of "A" is ON (A01F) and "N" is not equal to 3 ($\overline{\text{N03L}}$) decrement "L" one. Shift "A" right one bit. The bit now in A01F has a value of 2. If A01F is ON, decrement L by 2. If at any time during decrementing "L" becomes

zero, decrement "M" one to obtain the next lower address. Shift "A" once more to place the original A03F in A01F for the address add portion of the operator. Set Q09F to one to set up the address adder logic which will add the complement of "A" to "M". Complement "A" in preparation for the subtract of "A" from "M".

BRANCHING FORWARD OR BACKWARD

The increment is computed from the next syllable in sequence, therefore, the "L" Register (and "M", if L=3) is incremented +1 after "L" and "M" have been counted. Add the contents of "A" and Q01F to the contents of "M" and leave the result in M10 through M1.

Transfer the contents of "B" to "A" for temporary storage so "B" can be used to transfer the modified "M" address to "C".

WORD BRANCH

If Word Branch is specified (by T11F), by-pass the syllable modifications above. The entire field is added (or subtracted) as normally done after "L" modifications and the Branch proceeds as follows.

Transfer the contents of the modified "M" Register to "B" in preparation for the "B" to "C" transfer. Allow a Fetch from the address indicated by "C". The Fetch will load "P" Register with the new group of syllables. Restore "B" with its original contents which were stored in "A". Terminate the operator.

7.17 BRANCH RETURN (RJPL)

GENERAL DESCRIPTION

Branch to an address contained in the top word of the stack and return from the Mark Stack Control Word, the entry stack location to the "F" Register and the base address of the Program Reference Table to the "R" Register. Set the Mark Stack Flip-flop and the Sub-Level Flip-flop to the state contained in the Mark Stack Control Word.

SUMMARY OF OPERATION

If both "A" and "B" Registers are empty ($\overline{AROF} \cdot \overline{BROF}$), initiate a load "B" Memory Access and turn logical toggle Q03F on. (This is to indicate that the stack address must be adjusted prior to terminating if a Presence Interrupt is set.) Mark "B" as occupied. If "A" is loaded on entry, transfer the contents of "A" to "B". If the Presence bit is off ($\overline{B46F}$), set the Presence Bit Interrupt and if Q03F is on, decrement the stack address (S-1). Allow the Syllable Execute Complete Level and go to Interrupt. If the Presence bit is on (B46F), transfer the return stack location field in "B" (B30 ⇒ B16) to the "S" Register and initiate Load of "B" from "S" address. Clear the "L" Register.

Transfer the branch address field from "B" (B15 ⇒ B1) to the "C" Register and initiate a Branch operation. Branch will now occur. Clear the Mark Stack Flip-flop (MSFF) and the Sub-Level Flip-flop (SALF).

Upon completion of all Memory Accesses (Fetch excluded), transfer the entry stack location field from the Mark Stack Control Word (MSCW) in "B" to the "F" Register. Decrease the stack address by one (S-1) and mark "B" as empty.

Transfer the base address of the Program Reference Table from the MSCW to the "R" Register. Allow the Syllable Execute Complete Level. If at this time the mark stack bit is on (B32F) in the Mark Stack Control Word, set the Mark Stack Flip-flop, and if the Sub-Level Bit is on (B31F), set the Sub-Level Flip-flop.

7.18 STORE DEST/NON DEST (BSDL-BSNL)

GENERAL DESCRIPTION

Store and discard or store and preserve information in the "B" register in a location either established by or augmented by information in the "A" register in conjunction with base registers "R" or "F".

SUMMARY OF OPERATION

Load both "A" and "B" registers if not already loaded on entry. If the Flag bit of "A" is ON and the presence bit is ON, transfer the 15 low order bits of "A" register to "M" register and store "B: in that address. If the Presence bit is OFF, set Presence Interrupt and terminate.

If the Flag bit is OFF, Sub-Level Flip-flop is ON and A10F is ON, check the Mark Stack Flip-flop. If the MSFF is OFF, use "F" register as a base by transferring it to "M". If MSFF is ON, use the 15 bits in positions 30 ⇒ 16 of the word in R+7 location. This is done by a special memory cycle which reads only these bits directly into the "M" register. If in addition to SALF and A10F, A09F and A08F are ON; complement "A" register bits 7 ⇒ 1 to subtract the A value from the "F" or R+7 register base. Any other configuration uses the "R" register as a base. The value as set up in "A" is added to the base and the word is stored. If Variant Flip-flop is ON, set SALF and clear VARF to resume Sub-Level Operations. If the store is destructive, mark "B" empty.

7.19 INTEGER STORE DEST. (ISDL), INTEGER STORE NON-DEST. (ISNL), COND. INTEGER STORE DEST. (CSDL), COND. INTEGER STORE NON-DEST. (CSNL)

PURPOSE

Integer Store verifies the integer value of the word to be stored prior to storing. Storage method is identical to Store Operators. If integer values are not available, the Integer Overflow Interrupt is set.

SUMMARY OF OPERATION

Load both "A" and "B" Registers if not already loaded on entry. If the Flag bit is ON and the Presence bit is ON, transfer the 15 low order bits of "A" Register to "M" Register. If the Presence bit is OFF, set Presence Interrupt and terminate the operator. If the Flag bit is OFF, Sub-Level Flip-flop is ON and A10F is ON, check the Mark Stack Flip-flop. If the MSFF is OFF, use the "F" Register as the base by transferring it to "M" Register. If the MSFF is ON, use the 15 bits in position 30 ⇒ 16 of the word in location R+7. This is placed in "M" Register by a special Memory Read cycle. If, in addition to SALF and A10F, A09F and A08F are ON, complement the "A" Register bits to subtract them from the base Register "F" or R+7 value. Any other configuration uses the "R" Register as a base. The value as established in "A" Register is added to "M" Register. At the same time, either if the word in "B" Register has a zero exponent or the operator is conditional store and the Integer Bit (A29F) is OFF in the word "A" Register, store the word in "B" Register in the address established in "M" Register after the add function. If VARF is ON, set SALF and clear VARF to resume sub-level operations.

If the word in "B" Register does not have a zero exponent and either the Integer Bit in the word in the "A" Register is ON (A29F) or either of the two non-conditional Store operators are involved, try to normalize the value in "B" Register by shifting right the octal value and reducing the exponent in the word. If the entire Mantissa becomes zero value by shifting out the significant digits, then clear "B" Register, set the Integer Overflow Interrupt and terminate. If the exponent becomes zero due to reduction of the exponent value, check the values of the bits in the decade being shifted out. If 4 or more on positive integers or 3 or less on negative integers, add one to the value of the Mantissa as a round adjustment, then store the word in the location previously made up. Terminate the operator.

7.20 DIAL A, DIAL B (DIAL, DIBL)

PURPOSE

To set the Bit and Character Pointers of either "A" or "B" Registers to the value contained in the Repeat Count Field, unless that value is zero. If DIAL A and the Repeat Count Field is zero, treat it as a no-op. If DIAL B and the Repeat Count Field is zero, this sets the Variant Flip-flop and resets the Sub-Level Flip-flop. This gives programmatic control to the size of the PRT storage area in Sub-Level operation. This is based on the philosophy that certain of the requirements of programing will need extended space for PRT and this allows full indexing of 1023 words in this area by removing the sub-level limitation, then each operator doing the indexing will restore it on exit from that operator.

SUMMARY OF OPERATION

If DIAL A and the Repeat Count Field is not equal to zero ($\overline{TEZL}$), transfer the contents of the Repeat Count Field (T12 ⇒ T07) to "G" and "H" Registers, with G ⇐ (T12F ⇒ T10F) and H ⇐ (T09F ⇒ T07F). Allow the Syllable Execute Complete Level (SECL).

If DIAL B and the Repeat Count Field is not equal to zero ($\overline{TEZL}$), transfer the contents of the Repeat Count Field (T12 ⇒ T07) to the "K" and "V" Registers, with K ⇐ (T12F ⇒ T10F) and V ⇐ (T09F ⇒ T07F). Allow the Syllable Execute Complete Level (SECL).

If DIAL B and "T" is zero, reset SALF. If "T" is zero and SALF is on, set Variant flip-flop (VARF). Allow SECL.

NOTE

DIAL B zero (value) is now defined as a separate operator. It is called Set Variant Operator. DIAL A zero is also redefined as No-Op Operator.

7.21 TRANSFER BITS (TRFL)

PURPOSE

Transfer a field of bits from the source string to the destination string. If the field is zero, this operator is defined as the Delete Top Of The Stack operator.

SUMMARY OF OPERATION

The top address of the stack ("A" Register) is marked unoccupied and the execution cycle is ended if the repeat-count field of "T" is at zero (TEZL), and "A" and "B" are occupied (AROF•BROF). Due to this action, this configuration of the operator is re-defined as DELETE TOP OF THE STACK operator.

The character and bit pointers of "A" and "B" are stored in "X". Q07F is turned on and "N" is set to 14. Q01F is turned on to prevent setting "N" to 14 with a future clock pulse. If the "A" register is occupied, the "A" register character, being pointed at by "G", is shifted to "Y".

The "B" Register is shifted right and circulated, one octade, and "N" is decremented by 1 to tally the shift, if "K" is pointing at one of the LAST four characters of "B", and either "K" is not equal to "N" or there have been an odd number of right shifts of "B". The "B" Register is shifted left and circulated one octade and "N" is incremented by 1 to tally the shift, if "K" is not equal to "N" and "K" is pointing at one of the FIRST four characters of "B".

As soon as KENL is true ("K" equals "N"), the first character whose bits are to be transferred is shifted to the output alignment station of "B" (first and second octades).

Both Q04F and Q03F are off as long as the Bit Pointers of "A" and "B" are not pointing to the last bits of their respective characters. If HE5L is true, "H" is pointing at the last bit of the character being pointed at by "G". "H" and "G" are associated with the "A" Register. Q04F is set to 1 to permit reloading "Y" at the next clock pulse. "Y" is cleared for the same reason. If Q04F is on, the "G" character of "A" is shifted to "Y" and Q04F is turned off.

Q03F is turned on if VE5L is true. VE5L is true if "V" is pointing at the last bit of the character being pointed at by "K". "V" and "K" are associated with the "B" register. If Q03F is on, the "B" register is shifted left and circulated one octade. "N" is counted up by one to tally the shift. The bit pointer for the source string ("H") indicates which bit of "Y" to transfer to the B1 and B2 octades of "B". The "V" register indicates which bit position of the B1 and B2 octades is to receive the bit from "Y".

"H" is counted up by one to point to the next higher bit position of "Y". "V" is counted up by one to point to the next higher bit position of B1 and B2. "T" is counted down by 1 to tally the previous bit transfer.

If VE5L is true, "V" is pointing at the last bit of a character. Hence, "K" is counted up by one to point at the next character of the destination string.

If HE5L is true, "H" is pointing at the last bit of a character. Hence, "G" is counted up by one to point at the next character of the source-string. The "B" Register is shifted left and circulated, one octade, to place the most significant character into B1 and B2. "N" is counted up by one to tally the shift. Q03F is reset. End the bit transfer if the last bit of the word in "A" or "B" is being pointed to or if the last bit to be transferred is about to complete its transfer movement.

The "G", "H", "K", and "V" Register settings which were formerly stored in "X" are returned to their respective Registers. AROF is cleared to indicate that the top address of the stack no longer contains valid information.

If N08F is on and N14L is false, the "B" Register is shifted left and circulated and "N" is counted up to tally the shifts. If N08F is off and N14L is false, the "B" Register is shifted right and circulated and "N" is counted down to tally the shifts. This continues until N14F is true. The net result is returning the word in "B" to its alignment at the beginning of the operation. End the operator.

7.22 SET FLAG BIT (SFBL), RESET FLAG BIT (RFBL)

PURPOSE

Mark the word in the top of the Stack as an OPERAND or as a DESCRIPTOR.

SUMMARY OF OPERATION

Adjust the Stack until the "A" register is occupied.

If the operator is SFBL, set A48F to one and Exit the operator.

If the operator is RFBL, reset A48F and Exit the operator.

7.23 TEST FLAG BIT (TFBL)

PURPOSE

To check the top word of the stack to see if it is marked as an operand. If it is an operand, set "A" to integer 1 otherwise set "A" to 0.

SUMMARY OF OPERATION

Upon entry into the operator the "A" and "B" registers are checked. If the top word of the stack is not in "B", adjust stack until it is in "B". "A" will be cleared. Once the top word of the stack is in "B" register, check it for Flag bit "off". If it is off, set least significant digit of "A" equal to "1" and terminate the operation.

7.24 RESET SIGN BIT (MSPL) - SET SIGN BIT (MSNL) - CHANGE SIGN BIT (CSSL)

PURPOSE

To set the Sign Bit of the word in the "A" register to either one or zero, when executing a Set Sign Bit or Reset Sign Bit Operator respectively.

SUMMARY OF OPERATION

Adjust the stack if required. If a Reset Sign Bit Operator is being executed, set A47F to zero.

If a Set Sign Bit Operator is being executed, set A47F to one.

7.25 MARK STACK (MSOL)

PURPOSE

To load a Mark Stack Control Word into the Stack, to mark the limits of the Stack for inserting parameters and to provide a return to the program in process.

SUMMARY OF OPERATION

The contents, if any, of the "A" and "B" registers are pushed into the core portion of the Stack with Stack adjustment. The "B" register is cleared and a Mark Stack Control Word is then constructed in the "B" register as follows.

The present setting of the "R" register is transferred to bits 34 through 42 of the "B" register. The present contents of the "F" register are transferred to bits 16 through 30 of the "B" register. The present setting of the Mark Stack Flip-flop (MSFF) and Sub Program Level Flip-flop (SALF) are transferred to bits 31 and 32 respectively, of the "B" register. The "B" register is marked as a Control Word by setting bits 48 and 47. With the use of the "S" register the contents of the "B" register are then pushed into the core stack to become the top word of the core stack.

The "F" register is then set to the core address that the MSCW was stored into.

If MSFF is reset and SALF is set, the contents of the "B" register are also stored in the cell addressed by the contents of the "R" register plus seven.

MSFF is set to the One state if not already set.

The "A" and "B" registers are marked as invalid through the resetting of AROF and BROF. The syllable is then terminated.

7.26 ENTER CHARACTER MODE

PURPOSE

In Line Character Mode Entry will allow Character Mode segments to be executed from a Word Mode program without the necessity of using a Program Descriptor.

This is especially useful in COBOL programs executing MOVE verbs.

SUMMARY OF OPERATION

If the "B" register is occupied, its contents are transferred to the "A" register and also stored in the Stack.

A Return Control Word (RCW) is constructed and pushed into the Stack.

Sub Level Flip-flop (SALF) is set to establish the proper indexing in Character Mode.

The Mark Stack Flip-flop (MSFF) is cleared as it is not used in Character Mode.

The current "S" register contents are stored in the "F" register and also in "X" 30⇒16 for proper reference in Character Mode.

The "R" register is cleared since it is not used for addressing in Character Mode.

CWMF is set to one to establish the operation as Character Mode.

The "S" register is replaced by bits 30 ⇒ 16 of the "X" register.

The "A" register is placed back into the "B" register. This was the top of the Stack and contains the Destination Address.

The operation is changed to a Recall Destination Address with the "J" register equal to 2. The "A" register is marked as empty and the "K" and "V" registers are cleared.

7.27 RETURN NORMAL (RNML) - RETURN SPECIAL (RSPL)

PURPOSE

The function of the Return Normal/Special Operator is to facilitate returning to the main routine or a higher sub-level following the completion of a sub-routine. The Return Normal Operator is used in sub-routines where no parameters are needed and there is no nesting of sub-routines ("F" register contains the address of the RCW).

The Return Special Operator is used in sub-routines where parameters were inserted in the sub-routine by the main program. (At the completion of the sub-routine, the RCW will be the second word in the Stack).

The Return Normal/Special Operator also places the result of the sub-routine in the "A" register.

SUMMARY OF OPERATION

Upon entry, if both the "A" and "B" registers are empty, initiate a load of the "A" register. The Stack address is decremented by one during the load. The "B" register is marked as empty to allow it to be loaded as necessary during the operation.

Load the "B" register with the Return Control Word. If the word in "A" register is an Operand or is present, proceed. If the word is a Descriptor and is not present, set the Presence Interrupt and exit the operator. The address of this RCW, when executing a Return Normal Operator is contained in the "F" register. When the operator is a Return Special, the RCW will be the second word in the Stack and will be addressed by the "S" register directly. If the operator is a Return Normal, the address contained in the "F" register will be transferred to the "S" register. The "A" register will contain the result of the sub-routine after stack adjustment.

After the "B" register has been loaded with the RCW, the Flag Bit is checked. If it is off, BROF is set to ONE and the operation is terminated. If the system is operating in Normal State (NCSF) at this time, the Flag Bit Interrupt is set.

If the RCW is a Descriptor (B48F) the top MSCW address contained in the RCW (B30 ⇒ B16) is transferred to the "S" register. The MSCW is loaded into "B" from this address. As "B" is being loaded with the MSCW, 39 bits of the RCW in "B" are transferred to the "X" register for temporary storage to retain the top MSCW address. Also the "H", "V", "G", "K", "L", and "C" registers are restored to their configurations they held prior to entry into this sub-routine. A Fetch is initiated from the new "C" register setting. Logical Flip-flop Q05F will be set to ONE if B46F is on in the RCW. The setting of Q05F will allow the Return Operator to be completed as a Descriptor Call Operator. If B46F is OFF in the RCW, the Return Operator will be completed as an Operand Call Operator.

When the MSCW is in the "B" register, the following actions may be performed:

1. If MSFF was set and the Processor was in Sub-Program Level (B32F•B31F) when the MSCW was generated, a search is made for the first MSCW (B32F'). Transfer the "F" register field of "B" to "S" and access the next lower MSCW. This process continues until the first MSCW is found. Q06F is set to One as a flag to indicate that one MSCW exists before the next Return Control Word in the Stack.

2. If the Mark Stack bit (B32F) is set; set the Mark Stack Flip-flop.

3. If the Sub-Level bit is set (B31F); set the Sub-Level Flip-flop.

4. Transfer the "F" and "R" register fields of the top MSCW to the respective registers.

5. If exit is made to Program Level (B31F'), or when the first MSCW is found (B32F'); transfer the address of the top MSCW (originally obtained from the RCW) to the "S" register.

6. When exit is made to Sub-Program Level and more than one MSCW exists in the Stack (Q06F), store the bottom MSCW in the cell addressed by R+7. This MSCW will point to the address of the Return Control Word of the sub-routine to which exit is made.

7. The Stack is decremented by one from the top MSCW address. This restores the "S" register to the top of the Stack address prior to entry into this sub-routine.

8. The "T" register is changed to a Call syllable by setting T02F to one. The type of Call (Operand or Descriptor) is dependent upon the state of Q05F. If Q05F is ON, T01F is left on for a Descriptor Call Operator. If Q05F is OFF, T01F is reset for an Operand Call Operator.

9. The "J" register is set to 2 to allow the Operand/Descriptor Call Operator to be executed commencing at J=2. (Refer to Operand/Descriptor Call Operator).

7.28 EXCHANGE (EXCL)

PURPOSE

To interchange the two top words in the stack.

SUMMARY OF OPERATION

Upon entry into the operator, the "A" and "B" registers are loaded if they are marked empty. When both registers are loaded, or if they were loaded on entry, "A" and "B" are interchanged. This completes the operation.

7.29 DUPLICATE (DUPL)

PURPOSE

To duplicate the word in the top of the stack.

SUMMARY OF OPERATION

If both the "A" and "B" registers are loaded, the "B" register contents are stored in the stack; then the contents of "A" are transferred to "B". If both registers are empty, the "B" register is loaded and then the contents of the "B" register are transferred to the "A" register. If either register is loaded and the other empty, the contents of the loaded register are transferred to the empty register. Upon completion of the operator the "A" and "B" registers will have the same contents.

7.30 DELETE TOP OF STACK (DELL)

PURPOSE

This operator will delete the top word in the Stack.

SUMMARY OF OPERATION

The Stack is adjusted so that "A" and "B" are occupied.

AROF is then reset to mark the top of the Stack as invalid.

7.31 STACK SEARCH FOR FLAG (SSFL)

PURPOSE

A Flag Bit search is necessary when a program or data segment has been overlaid. When this occurs, all descriptors in the Stack must be checked for references to the overlaid segment.

The low order 15 bits of the word in the top of the Stack are used as a base address for the search. When a word is found with the Flag bit on, it is left in the top of the Stack.

SUMMARY OF OPERATION

The "A" register is loaded from the top of the Stack if necessary. The 15 low order bits of "A" are placed in the "M" register and this address is accessed. If the word found at this address is an Operand, the "M" register is incremented by one and this address is accessed. This process continues until a word is found with its Flag bit on.

When a word is found with its Flag bit on, it is marked as a present Data Descriptor pointing to itself and the operation is terminated.

7.32 LOAD (LODL)

Purpose

To locate a word in memory using either the word in "A" register as a direct address or using the word in "A" register to augment a Base Register address and then to place the word so located into the "A" register.

SUMMARY OF OPERATION

Load "A" register from either "B" register or the top of the stack. Decrement "S" register if the latter is necessary. If the word in "A" register has the Flag bit on and the Presence bit on, transfer the 15 low order bits of the "A" register to the "M" register and place that word into "A" register directly. If the Presence bit is off, set Presence Interrupt and terminate the operator.

With the Flag bit off, the Sub-Level Flip-flop on and A10F on, check the Mark Stack Flip-flop. If the MSFF is off, use the "F" register as a base by transferring it to the "M" register. If the MSFF is on, use the 15 bits in positions 30 ⇒ 16 of the word located in R+7. This is done by a special memory cycle which reads only these bits directly into the "M" register. If in addition to SALF and A10F, A09F and A08F are on, complement the "A" register bits 7 ⇒ 1 to subtract the "A" value from the "F" or R+7 base. If A09F is on and A08F is off, use the "C" register as the base. Any other configuration uses the "R" register as the base. The Sub-Level Flip-flop is set and Variant Flip-flop reset if the Variant Flip-flop is on to restore sub-level operations. The value in "A" register is added to the base in "M" register and the word is read from memory into "A" register. The operator is terminated.

7.33 INDEX (INDL)

PURPOSE

The 15 low order bits of the "B" register are added to the 15 low order bits of the "A" register with the sum appearing in the "A" register. Any carries are lost. The remaining bits of "A" are left unchanged.

SUMMARY OF OPERATION

When both registers are full, add the 39 low order bits of "A" to the 39 low order bits of "B", together with any carry which might exist, and place the sum in the 39 low order bits of "B".

On Memory Read Access Obtained (MROF), count the "S" register down one to point at the new top of the stack. Mark "B" as empty, transfer the 15 low order bits of "B" to "M". This establishes the 15-bit sum in "M" without any carry from the high order position. Transfer the 15 bits in "M" to the 15 low order bits of "A". End the operation.

7.34 CONSTRUCT OPERAND/DESCRIPTOR CALL (MDVL/MDAL)

PURPOSE

To change the present operator in the "T" Register to an Operand or Descriptor Call Operator and identify the second word in the stack as a Data Descriptor.

SUMMARY OF OPERATION

Upon entry into the operator, the "A" and "B" Registers are loaded if empty. If the "A" and "B" Registers are occupied, an "A" to "B" and "B" to "A" exchange is accomplished.

At the completion of all Memory Accesses, (EEZL), the "B" Register is marked as occupied, the A48 bit is set and the A47 bit is reset to construct a Data Descriptor in the "A" Register. At the same time, the T2 bit is set. If the operator is a CONSTRUCT OPERAND CALL, the T1 bit is reset, but remains set if the operator is a CONSTRUCT DESCRIPTOR CALL.

At this time, the construction of either an OPERAND CALL or DESCRIPTOR CALL is completed. The "J" Register remains at 2 and the execution of an OPERAND CALL ($T02F \bullet \overline{T01F}$) or DESCRIPTOR CALL ($T02F \bullet T01F$) continues. For a further description, refer to OPERAND/DESCRIPTOR CALL flow chart and write-up.

7.35 SET VARIANT (VARL)

PURPOSE

Set the Variant Flip-flop (VARF) to one if operating in Sub-level (SALF). This gives programmatic control to the size of the PRT storage area in Sub-Level operation.

SUMMARY OF OPERATION

If the Sub-Level Flip-flop is on, set VARF to one.

Reset the Sub-Level Flip-flop and exit the operator.

7.36 NO-OP (NOPL)

PURPOSE

This operator acts as a filler to be used as an aid in de-bugging programs.

SUMMARY OF OPERATION

The first clock pulse finds "T" equal to zero, so no actions are accomplished and the operator is terminated.

7.37 VARIABLE FIELD ISOLATE (VFIL)

PURPOSE

This operator will select a field from the top word in the Stack. The selected field is placed in a word right justified, and the rest of the word is set to zero. This new word replaces the top word in the Stack.

SUMMARY OF OPERATION

The selected field may be 1 to 39 bits in length.

The starting bit of the field is defined by the "G" and "H" registers. The length of the field in characters is defined by the L variant field of the operator. The number of characters should include those characters which contain the first and last bits.

The S variant field of the operator specifies the number of bits that the field is offset from the right character boundary.

7.38 "F" AND "S" REGISTER SET/STORE (FXSL)

PURPOSE

This operator uses the two top words in the stack. Based on the two low order bits of the top word, either the "F" or "S" register will be set from or stored into the second word in the Stack.

The field transfer is under control of the two low order bits of the "A" register as follows:

A02F A01F	S ⇐ B(15 ⇒ 1)
A02F A01F'	F ⇐ B(30 ⇒ 16)
A02F' A01F	B(15 ⇒ 1) ⇐ S
A02F' A01F'	B(30 ⇒ 16) ⇐ F

SUMMARY OF OPERATION

The Stack is adjusted if necessary. In this operator, the transfer is accomplished with one clock pulse, so the "A" register is set to empty and SECL occurs. Simultaneously, check the A02F bit. If it is on, set the appropriate register from the field in the "B" register and mark the "B" register empty. If A01F is on, the "S" register is set from B(15 ⇒ 1). If it is off, the "F" register is set from B(30 ⇒ 16) and SALF is set to one. It is necessary to set SALF because a change of the "F" register re-orients the relative Stack area and only sub-program level will allow proper indexing.

If A02F is off, "F" or "S", as specified by the state of A01F, is transferred to the appropriate field of the "B" register.

7.39 LINK LIST LOOKUP (LLLL)

PURPOSE

This operator will scan the "available space link list" in the memory allocation portion of the MCP.

SUMMARY OF OPERATION

The word in the top of the Stack contains the size of the segment desired in bits 30 ⇒ 16. The second word in the Stack contains the address of the entry to the link list in bits 15 ⇒ 1.

The operator will access each link entry in turn and compare the size of the called for segment with the link entry segment.

If the size of called for segment is equal to or smaller than the link entry, the address of the link entry is placed in the top of the Stack in bits 15 ⇒ 1. The rest of the word is cleared and then marked as a descriptor.

The second word in the Stack has the next link entry address in bits 15 ⇒ 1 and the length of the list just selected in bits 30 ⇒ 16.

If the operator fails to find a list entry large enough to handle the segment, the last link list entry (Tag Word) directs the operator to Exit.

7.40 INTERROGATE PERIPHERAL STATUS (IPSL), INTERROGATE I/O CHANNEL (TIOL)

PURPOSE (INTERROGATE PERIPHERAL UNIT STATUS)

Interrogate Peripheral Unit Status will allow the MCP to determine much more rapidly which units have changed status. This bulk interrogation of Peripheral units eliminates the need for a separate I/O command for each unit.

PURPOSE (INTERROGATE I/O CHANNELS)

Interrogate I/O Channel will replace a programed search through a table to determine which I/O Channel will be used on the next I/O Operation.

SUMMARY OF OPERATIONS

Interrogate Peripheral Units

This operator places in the top of the Stack a word representing the current ready status of the peripheral equipment. One bit in the word is associated with each peripheral unit. This bit is set to one if the associated unit is ready and to zero if the associated unit is not ready.

The Stack is adjusted so that the "A" register is empty. The "A" register is set to zero. The low order bits (A31 ⇒ A01) are set to reflect the current status of the peripheral units.

The "A" register is marked occupied and the operation is terminated.

Interrogate I/O Channels

This operator interrogates the I/O Channels to determine which channel is currently in line to be assigned next. Stack adjustment occurs if necessary and a literal is placed in the top of the Stack. This literal indicates the next channel to be assigned in the following way:

LITERAL	CHANNEL
0	All channels busy
1	Channel one due for assignment
2	Channel two due for assignment
3	Channel three due for assignment
4	Channel four due for assignment

7.41 FIXED FIELD TRANSFERS

PURPOSE

These operators will facilitate the setting of the "F" and "S" registers.

SUMMARY OF OPERATION

The Fixed Field Transfer Operators use the two top words in the Stack. The contents of the specified field of the top word is transferred to the specified field of the second word.

The fixed fields are termed either, "F", which encompasses bits 30 ⇒ 16 of either word; or "Core", which encompasses bits 15 ⇒ 1 of either word. On this basis, two words with two positions in each gives four combinations for exchanging the fields.

There are four separate operators that make up the Fixed Field Transfers:

1. TRANSFER "F" FIELD TO "F" FIELD (FFXL). The contents of bits 30 ⇒ 16 of the "A" register are transferred to bits 30 ⇒ 16 of the "B" register.

2. TRANSFER "F" FIELD TO "CORE" FIELD (FCXL). The contents of bits 30 ⇒ 16 of the "A" register are transferred to bits 15 ⇒ 1 of the "B" register.

3. TRANSFER "CORE" FIELD TO "CORE" FIELD (CCXL). The contents of bits 15 ⇒ 1 of the "A" register are transferred to bits 15 ⇒ 1 of the "B" register.

4. TRANSFER "CORE" FIELD TO "F" FIELD (CFXL). The contents of bits 15 ⇒ 1 of the "A" register are transferred to bits 30 ⇒ 16 of the "B" register.

In all the above cases: the "B" register bits not transferred remain unchanged and the "A" register is marked empty.

7.42 BRANCH ON NON-ZERO FIELD AND DELETE

PURPOSE

These operators test a field of the word in the "B" register for being equal to zero. If the field is non-zero, a branch occurs.

SUMMARY OF OPERATIONS

Branch Forward on Non-Zero Field, Non-Destructive. (ZFNL)

This operator tests a field of the word in the "B" register for zero. If the field is zero, the "A" register is marked empty and the operator is terminated. If the field is not zero, the "T" register is changed to a syllable branch forward unconditional. See Note 1 for field definition.

Branch Forward on Non-Zero Field, Destructive. (ZFDL)

This operator tests a field of the word in the "B" register for zero. If the field is zero, the "A" and "B" registers are marked empty and the operator is terminated. If the field is not zero, the "B" register is marked empty and the "T" register is changed to a syllable branch forward unconditional. See Note 1 for field definition.

Branch Backwards on Non-Zero Field, Non-Destructive. (ZBNL)

This operator tests a field of the word in the "B" register for zero. If the field is zero, the "A" register is marked empty and the operator is terminated. If the field is not zero, the "T" register is changed to a syllable branch backwards unconditional. See Note 1 for field definition.

Branch Backwards on Non-Zero Field, Destructive. (ZBDL)

This operator tests a field of the word in the "B" register for zero. If the field is zero, the "A" and "B" registers are marked empty and the operator is terminated. If the field is not zero, the "B" register is marked empty and the "T" register is changed to a syllable branch backwards unconditional. See Note 1 for field definition.

NOTE 1

The starting bit of the field is determined by the "G" and "H" registers. The length of the field is determined by the 4 high order bits in the repeat field. The field may be 1 thru 15 bits in length.

If the repeat field is zero, the operator is a delete top of the stack.

7.43 BEGIN LOOP (BELL)

PURPOSE

To begin execution of a string of program syllables enclosed between a Begin Loop and an End Loop operator.

SUMMARY OF OPERATION

The present "X" register contents is transferred to the "B" register and updated with information pertaining to the loop being started. This new Loop Control Word is then stored in the "X" register and the original "X" register is stored in memory as a Loop Control Word. Restore the "B" register and exit the operator.

7.44 CALL REPEAT FIELD (CIRL)

PURPOSE

To obtain a word from memory which will be used as a repeat field. This memory location is determined by reducing the address of the Return Control Word by the repeat field of this operator.

SUMMARY OF OPERATION

The address of the Return Control Word, contained in the "F" register, is transferred to the "S" register. The "S" register is then decremented by the value of the repeat field of "T" register. The six low order bits of the word found at this address are transferred to the repeat field of the "T" register.

If this value when transferred to the repeat field is zero, the operator portion of the following syllable is ignored and a Jump Forward Unconditional operator is executed using the repeat field of the following syllable.

If this value when transferred to the repeat field is not zero, transfer of the repeat field is suppressed and the present contents of the repeat field is used for the following syllable.

7.45 CHARACTER MODE NO-OP (NOCL)

PURPOSE

To provide an operator which can be used as a program debugging aid. This operator, when executed, produces no results other than allowing SECL.

SUMMARY OF OPERATION

If upon entry the repeat field is equal to zero, exit the operator. If the repeat field is not equal to zero, an Increase Tally Operator is executed.

7.46 COMPARISON OPERATORS

Compare Equal (SEQL)
Compare Equal or Less (SLEL)
Compare Greater (SGTL)
Compare Greater Or Equal (SGEL)
Compare Less (SLTL)
Compare Not Equal (SNEL)

PURPOSE

To compare two equal length fields of alphanumeric characters. The comparison is made based on the collating sequence. The True/False Flip-flop will be set to a state which will represent the result of the comparison.

SUMMARY OF OPERATION

If the "V" register (bit pointer for the "B" register) is zero and the repeat field of the "T" register is zero, exit the operator. If the "V" register is not equal to zero, the "K" register is counted up one to ignore this partial character.

The "B" register will be in alignment when the "K" register is equal to the "N" register. Hence "B" will be shifted left or right until alignment is obtained.

After proper alignment, B16 and B15 are transferred to the "Z" register and the character in "A" register, pointed at by "G", is transferred to the "Y" register.

The actual comparison is made in the "Y" and "Z" registers and will continue until either an inequality is found or the repeat field of the "T" register is counted to zero. When a "Y" greater than "Z" condition is found, the True/False Flip-flop will be turned on and will either remain on or be turned off depending on the Comparison Operator being used.

7.47 END LOOP (ENLL)

PURPOSE

To check the repeat field portion of the Loop Control Word to determine whether the loop is to be executed again.

SUMMARY OF OPERATION

The Loop Control Word contained in the "X" register, is transferred to the "B" register. The repeat field portion of this Loop Control Word is transferred to the "T" register.

If the repeat field is not equal to zero, the Loop Control Word is distributed to appropriate registers and the loop is repeated.

If the repeat field is zero, the "B" register is loaded with the last Loop Control Word or Return Control Word. This Loop or Return Control Word is transferred to the "X" register and the operator is terminated.

7.48 EXIT CHARACTER MODE (RECL)

PURPOSE

To terminate Character Mode operation and establish Word Mode operation.

SUMMARY OF OPERATION

Access a Return Control Word using the present "F" register contents. Distribute this Return Control Word to appropriate registers and access a Mark Stack Control Word. Distribute the Mark Stack Control Word to appropriate registers, reset the Character/Word Mode Flip-flop and set the "S" register to one less than the address of the Mark Stack Control Word.

7.49 IN LINE EXIT CHARACTER MODE (ILEL)

PURPOSE

This operator is the same as the Exit Character Mode operator except that the "C" and "L" registers are not loaded from the Return Control Word. In Line Exit Character Mode is used to exit from Character Mode programs entered via an In Line entry.

SUMMARY OF OPERATION

The Return Control Word is accessed and loaded into the "B" register. Distribution of the Return Control Word and accessing of the Mark Stack Control Word continues as normal, with the exception of the "L" and "C" registers not being restored.

7.50 FIELD ADD (FADL)

FIELD SUBTRACT (FSUL)

FIELD ADD AUX. (FAXL)

FIELD SUBTRACT AUX. (FSXL)

PURPOSE

To algebraically add the contents of a source string field to the contents of a destination string field.

PROVISIONS

1. Both fields are of equal lengths, and the field lengths are specified by the repeat field of the "T" register.

2. The sign of each field is stored in the zone bits of the least significant character in each field. If the B-bit is on and the A-bit is off, the field is negative. All other combinations are positive.

3. All zone positions, other than the one associated with the least-significant character, are set to zero.

4. If the sum of the addition overflows in the destination field, the overflow is lost but the True/False Flip-flop is left on at the end of the operator to store the fact that there was an overflow.

5. If both fields contain a minus zero and a Field Add is requested or, if the source field is plus zero and the destination field is minus zero, the answer will be zero. In the other two cases there will be a plus zero answer.

6. Although the character pointers point to the most significant characters of their respective field, addition is performed from least-significant character.

SUMMARY OF OPERATION

Prior to executing a Field Add or Subtract, the two fields must be compared. The Compare Operator is used to perform this comparison. After the comparison is made, the True/False Flip-flop (TFFF) and Q03F will indicate whether the source field is greater than, less than, or equal to the destination field.

If Q03F is on, it will indicate that there was an inequality. If TFFF is on, it will indicate that the source field is greater than the destination field. If TFFF is off, it will indicate that the source field is less than the destination field. With J=5 and the comparison complete, the Field Add Flow Chart becomes the new reference document.

It is assumed that the least significant character of each field have been compared (TEZL). This causes the "J" register to be set to zero.

The "H" and "V" registers and Q03F are not cleared because FASL is true. Similarly, Q02F is not turned on. The turn on or turn off of TFFF is also inhibited because the syllable in the "T" register will either be FADL or FSUL. Q04F and Q06F are cleared. The J=5 box of the Field Add/Subtract Flow Chart contains additional occurrence for J=5 on the Compare Operators Flow Chart. If FASL is true in addition to TEZL and EEZL, T06F is turned off to make FASL false and FAXL true. This prevents SECL from becoming true at the next clock pulse.

In effect, this signals the end of the compare portion and the beginning of the actual Field Add/Subtract operator. With the "B" register in proper alignment, the character in the "A" register being pointed at by "G", is shifted to "Y". The character in the "B" registers output alignment station is shifted to the "Z" register. The sum of the "Y" and "Z" registers is shifted to the "Z" register. The "Z" register and the possible carry in Q07F is then transferred to the "B" register. This action will continue until the repeat field of the "T" register has been counted to zero.

7.51 INCREASE TALLY (INTL)

PURPOSE

This operator increases the "R" register by the amount of the repeat field. The tally is modulo 64; overflows are lost.

SUMMARY OF OPERATION

If the repeat field is zero, the operation is defined as a NO-OP. If the repeat field is not equal to zero, increase the tally by one and decrease the repeat-count field by one. When the repeat field is equal to zero (or if zero on entry) allow the Syllable Execute Complete Level (SECL).

7.52 INPUT CONVERT (ICOL)

PURPOSE

To convert "n" number of characters from 6 bit decimal characters to an octal value. The number of characters to be converted is found in the repeat field of the operator and is limited to 8 characters.

SUMMARY OF OPERATION

Upon entry into the operator, the "B" register is realigned, if required, and then stored. The "A" register is loaded if empty. If the repeat field is zero the operation will terminate. The "A" register bit pointer is cleared and its associated character and memory address increased if necessary.

With the "A" register loaded, a character at a time is transferred to the "B" register, until the number of characters as defined by the repeat field have been transferred. (The high order positions of the "T" register (T12 through T10) are cleared prior to this shifting, thus effectively limiting the maximum number of characters to eight).

When all characters have been transferred to "B", the decimal to binary conversion is started. The repeat field is set to 27 to allow the 27 shifts needed for the conversion. (For rules for decimal to binary conversion, see A-11835972). When the conversion is complete, the repeat field will be zero. Set the sign of the original word into the sign position for the mantissa of an octal word, store the converted word, and terminate the operation.

7.53 JUMP FORWARD CONDITIONAL (CFJL) JUMP FORWARD UNCONDITIONAL (FWJL)

PURPOSE

To jump over a number of consecutive program syllables as specified by the repeat field of this operator. The Conditional Jump Forward Operator can be executed only when the True/False Flip-flop is off upon entry.

SUMMARY OF OPERATION

If the repeat field is greater than four, the "C" register is counted up one and the "T" register is counted down by four. This is done to speed up the operation by jumping a word at a time.

If the repeat field is less than four, the "L" register is counted up by one and the "T" register is counted down by one. This action continues until the repeat field reaches zero.

7.54 JUMP OUT OF LOOP (JOLL)
JUMP OUT OF LOOP CONDITIONAL (CJOL)

PURPOSE

These operators are used to jump out of a repetitive program string and to terminate the repetition of that string by replacing the present "X" register with a Loop Control Word. The Jump Out Of Loop Conditional Operator can be executed only when the True/False Flip-flop is off upon entry.

SUMMARY OF OPERATION

Transfer the address of a Loop Control Word, presently contained in the "X" register, to the "S" register and load "A" with this Loop Control Word. If the repeat field is greater than four, the "C" register is counted up by one and the "T" register is counted down by four. This is done to speed up the operation by jumping a word at a time.

If the repeat field is less than four, the "L" register is counted up by one and the "T" register is counted down by one. This action continues until the repeat field reaches zero. With repeat field equal to zero, restore the "S" register and transfer the Loop Control Word to the "X" register.

7.55 JUMP REVERSE CONDITIONAL (CRJL)
JUMP REVERSE UNCONDITIONAL (REJL)

PURPOSE

To jump over a number of consecutive program syllables in the reverse direction as specified by the repeat field of this operator. The Conditional Reverse Jump Operator can be executed only when the True/False Flip-flop is off upon entry.

SUMMARY OF OPERATION

If the repeat field is greater than four, the "C" register is counted down one and the "T" register is counted down by four. This is done to speed up the operation by jumping a word at a time.

If the repeat field is less than four, the "L" register is counted down by one and the "T" register is counted down by one. This action continues until the repeat field reaches zero.

7.56 OUTPUT CONVERT

PURPOSE

To convert "n" number of characters from binary form to a decimal equivalent. If the converted field overflows, the True/False Flip-flop will be turned off.

SUMMARY OF OPERATION

Upon entry into the operator the Bit and Character Pointers, "H", "V", and "G", are cleared and the true/false flip-flop is set to one. If the bit and character registers ("H", "V" and "G") had been on, the associated character and/or address register is adjusted.

If the "B" register had been aligned and adjustment of the character register ("K") is necessary, a shift of "B" is accomplished to keep "B" aligned. The repeat field is checked, and if it is zero (TEZL) the operation terminates.

If the repeat field is not equal to zero ($\overline{\text{TEZL}}$), the contents of the "V" register (previously cleared) are transferred to T12 through T10, thus effectively limiting the number of characters to be converted to a maximum of eight.

The "A" register is loaded with the word to be converted and the "B" register, if occupied, will be stored to preserve its contents. When "A" is loaded and "B" is stored and cleared, the BINARY to DECIMAL conversion is performed.

If the conversion results in an overflow, set the true/false flip-flop. Set the sign of the word in the "B" bit of the least significant digit of the converted word. Transfer the converted word to "A" and load "B". If the number of the converted characters to be transferred is equal to eight, align "B" if necessary, and transfer the converted word to the "B" register, character by character.

If the number of characters to be transferred is less than eight, the complement of the "T" register is used to compare with the "G" register for transfer alignment.

When "G" is equal to the complement of "T" (T9 through T7) the transfer is started. If "G" is not equal to the complement of "T", increase "G" plus one with each clock pulse until "G" is equal to the complement of "T". The "G" register now points to the first character to be transferred. The transfer of character is started. Upon completion of the transfer terminate the operation.

7.57 RECALL CONTROL ADDRESS (RPAL)

PURPOSE

To obtain values for the "C" and "L" registers from a word stored in memory by the Store Control Address Operator.

SUMMARY OF OPERATION

Reduce the Return Control Word address, presently contained in the "F" register, by the value of the repeat field. This word is accessed and placed in the "B" register.

If the flag bit of the word is off, or the Presence bit is on, transfer bit positions 1 through 15 to the "C" register.

If the flag bit of the word is off, transfer bit positions 37 through 38 to the "L" register.

If the Presence bit and the flag bit are both on, set the "L" register to zero.

If the flag bit is on, the Presence bit is off and operation is in the Normal State, set the Presence Bit Interrupt.

Restore the "A" and "B" registers and exit the operator.

7.58 RECALL DESTINATION ADDRESS (RDAL)

PURPOSE

To obtain values for the "S" and "K" registers from a word stored in memory by the Store Destination Address Operator.

SUMMARY OF OPERATION

If the "B" register contains a portion of the destination string, and is aligned, the register is restored. The "B" register is then stored in the destination string.

Upon completion of the storing of the "B" register, the Return Control address contained in the "F" register is transferred to "S" and reduced by the value of the repeat field. This word is accessed and placed in the "B" register.

The 15 low order bits of the word are transferred to the "S" register. If the word accessed is an operand, the character field is transferred to "K". If the word is a descriptor with the Presence Bit off, set the presence bit interrupt.

7.59 RECALL SOURCE ADDRESS (RSAL)

PURPOSE

To obtain values for the "M" and "G" registers from a word stored in memory by the Store Source Address Operator.

SUMMARY OF OPERATION

Reduce the Return Control Word address, presently contained in the "F" register, by the value of the repeat field. This word is accessed and placed in the "B" register.

The 15 low order bits of the word are transferred to the "M" register. If the word accessed is a operand, bits 16 through 18 are transferred to the "G" register. If the word is a descriptor with the Presence Bit off, set the Presence Bit Interrupt.

7.60 RESET BIT (REBL)

PURPOSE

To reset "n" number of bits starting with the character and bit as designated by the "V" and "K" registers. The number of bits to be reset is specified by the repeat field.

SUMMARY OF OPERATION

Upon entry into the operator the repeat field is checked; if it is zero the operator is terminated. If the repeat field is not zero, the "B" register is loaded if necessary. The "B" register is then aligned so that the character containing the first bit to be reset is in the output alignment station. The character in the alignment station is then shifted to the "Z" register.

There, the bit position called out by the setting of the "V" register is reset. The repeat field is counted down and the bit pointer is counted up. If the bit pointer is to include more bits than are contained in one character, successive characters are shifted into the output alignment station. The characters that have been operated on are returned to the "B" register. If the number of bits is greater than a full word, the word in "B" is stored and a new word is brought into the "B" register.

When the repeat field has been counted down to zero, the operation is terminated and the character and bit pointers are left pointing at the next bit position.

7.61 SET BIT (SEBL)

PURPOSE

To set bits to one in the destination string starting at the position specified by the "S", "K" and "V" registers. The number of bits acted upon is specified by the repeat field.

SUMMARY OF OPERATION

Upon entry into the operator the repeat field is checked; if it is zero the operator terminates. If the repeat field is not zero, the "B" register is loaded if necessary. The "B" register is then aligned so that the character containing the first bit to be set is in the output alignment station. The character in the alignment station is then shifted to the "Z" register. There the bit position called out by the setting of the "V" register is set. The repeat field is counted down and the bit pointer is counted up. If the pointer is to include more bits than are contained in one character, successive characters are shifted into the output alignment station.

The characters that have been operated on are returned to the "B" register. If the number of bits is greater than a full word, the word in "B" is stored and a new word is brought into the "B" register.

When the repeat field has been counted down to zero, the operation is terminated and the character and bit pointers are left pointing at the next position.

7.62 SET DESTINATION ADDRESS (SDPL)

PURPOSE

To set the "S" register to an address formed by reducing the address of the Return Control Word by the amount of the repeat field.

SUMMARY OF OPERATION

Upon entry into the operator, the "B" register is restored if it is not in its original state. The "B" register is then stored in the destination string.

The Return Control Word address is transferred to the "S" register where it is decremented by the amount of the repeat field. The "K" and "V" registers are cleared and the "B" register is marked as empty. The unmodified Return Control Word address is returned to the "F" register.

7.63 SET SOURCE ADDRESS (SSPL)

PURPOSE

To set the "M" register to an address which is formed by reducing the Return Control Word address by the amount of the repeat field.

SUMMARY OF OPERATION

Transfer the Return Control Word address to the "M" register. Decrement "M" by the amount of the repeat field of the "T" register. When the repeat field equals zero, terminate the operator.

7.64 SET TALLY (SETL)

PURPOSE

To store the tally (repeat field) in the "R" register.

SUMMARY OF OPERATION

The contents of the repeat-count field are transferred to the R register.

7.65 SKIP BIT DESTINATION (SBDL)

PURPOSE

To skip "n" number of bits in the destination string as specified by the repeat field.

SUMMARY OF OPERATION

If the repeat field is zero upon entry, the operation terminates. If the repeat field is not zero, the bit pointer ("V" register) is counted up while the repeat field is counted down. If the bit pointer is counted through a complete character, the character pointer ("K" register) is counted up one.

If the character pointer is counted through a complete word, the word is stored and the stack address incremented. This action continues until the repeat field equals zero at which time the operation terminates.

7.66 SKIP BIT SOURCE (SBSL)

PURPOSE

To skip "n" number of bits in the source string as specified by the repeat field.

SUMMARY OF OPERATION

If the repeat field is zero upon entry, the operation is terminated. If the repeat field is not zero, the bit pointer ("H" register) is counted up while the repeat field is counted down. If the bit pointer is counted through a complete character, the character pointer ("G" register) is counted up one.

If the character pointer is counted through a complete word, count the "M" register up one. This action continues until the repeat field equals zero at which time the operation terminates.

7.67 SKIP FORWARD DESTINATION (FSDL)

PURPOSE

To skip forward over a number of characters in the destination string as specified by the repeat field.

SUMMARY OF OPERATION

Upon entry, if the bit pointer ("V" register) is not pointing at a full character, increase the character pointer ("K" register) by one. If the "B" register is aligned, it must be shifted one character position to the left. Should the character pointer be counted to seven during this shift, increase the "S" register by one and store the "B" register.

If the repeat field is greater than eight, increase the "S" register by one and decrease the "T" register by eight.

If the repeat field is less than eight, increase the character pointer by one and decrease the "T" register by one.

This action continues until the repeat field equals zero at which time the operator is terminated.

7.68 SKIP FORWARD SOURCE (FSSL) - SKIP REVERSE SOURCE (RSSL)

PURPOSE

To skip forward or backward over a number of characters is the source string as specified by the repeat field.

SUMMARY OF OPERATION

If the repeat field is zero upon entry, exit the operator. If the bit pointer ("H" register) is not equal to zero, increase the character pointer ("G" register) by one. If the character is counted to seven, increase the "M" register by one, and mark the "A" register unoccupied.

If a Skip Forward operator is being executed and the repeat field is greater than eight, increase the "M" register by one and decrease the "T" register by eight. If the repeat field is less than eight, decrease the "T" register by one and increase the "G" register by one.

If a Skip Reverse operator is being executed and the repeat field is greater than eight, decrease the "M" register by one and decrease the "T" register by eight. If the repeat field is less than eight, decrease the "T" register by one and decrease the "G" register by one.

This action continues until the repeat field is equal to zero at which time the operator is terminated.

7.69 SKIP REVERSE DESTINATION (RSDL)

PURPOSE

To skip over a number of characters in the destination string as specified by the repeat field.

SUMMARY OF OPERATION

If the destination string is not pointing at a full character, the bit pointer ("V" register) is cleared and the character pointer ("K" register) is increased by one. If the "K" register is counted to seven, the "S" register is counted up one. If the "B" register is aligned, the count of "K" will misalign it by one shift, therefore, it is given a shift to re-align it.

If the repeat field is zero, exit the operator. If the repeat field is greater than eight, decrement the destination address ("S" register) by one and decrement the field by eight.

If the repeat field is less than eight, decrement the "K" register by one and the repeat field by one. This action continues until the repeat field equals zero at which time the operator terminates.

7.70 STORE CONTROL ADDRESS (STPL)

PURPOSE

To store the contents of the "C" and "L" registers in an address formed by reducing the Return Control address by the amount of the repeat field.

SUMMARY OF OPERATION

Reduce the Return Control Word address, presently contained in the "F" register, by the value of the repeat field. Transfer the "C" and "L" registers to the "B" register and store "B" at this address.

7.71 STORE DESTINATION ADDRESS (STDL)

PURPOSE

To store the contents of the "S" and "K" registers in an address formed by reducing the Return Control Word address by the repeat field.

SUMMARY OF OPERATION

If the bit pointer does not equal zero shift the "B" register right and adjust the "K" register accordingly, until "K" equals "N". If the character pointer is addressing the last character (KE7L), either upon entry or during the shift of the "B" register, store "B" in the destination string.

Reduce the Return Control Word address, presently stored in the "F" register, by the value of the repeat field. Transfer the "S" and "K" registers to the "B" register and store "B" at this address.

7.72 STORE SOURCE ADDRESS (STSL)

PURPOSE

To store the contents of the "M" and "G" registers in an address formed by reducing the Return Control address by the amount of the repeat field.

SUMMARY OF OPERATION

Reduce the Return Control Word address, presently contained in the "F" register, by the value of the repeat field. Transfer the "M" and "G" registers to the "B" register and store "B" at this address.

7.73 STORE TALLY (STAL)

PURPOSE

This operator stores the Value in "R" Register at the address formed by reducing the address of the Return Control Word by the amount in the repeat field.

SUMMARY OF OPERATION

Upon entry into the operator the "B" register is restored to its original state and stored in the destination string. The Return Control Word address, in the "F" register, is temporarily stored in the "A" register. The tally contained in the "R" register is transferred to the "B" register and stored in an address formed by decreasing the Return Control Word address by the amount of the repeat field. The Return Control Word address temporarily stored in "A" is restored to the "F" register. The "B" register is marked as empty.

7.74 TEST BIT (TEBL)

PURPOSE

To test one bit in the source string at the position specified by the "M", "G", and "H" registers against the low-order bit of the repeat field. If they are equal, the true/false indicator is set to true, otherwise the true/false indicator is set to false.

SUMMARY OF OPERATION

Upon entry into the operator the true/false flip-flop is set to zero. The "A" register is loaded from the source string if it is empty.

When the "A" register is loaded, or if loaded on entry, the character addressed by the "G" register is transferred to the "Y" register. A comparison is now made between the bit in "Y", pointed to by the "H" register, and the least significant bit of the repeat field. If the bits compare equal, the true/false flip-flop is set to indicate this condition and the operation is terminated.

7.75 TEST FOR ALPHANUMERIC (TANL)

PURPOSE

To compare a character in the source string with a character in the repeat field. If the character is alphanumeric, the True/False Flip-flop is turned on. If it is a special character, the True/False Flip-flop is turned off.

SUMMARY OF OPERATION

Load the "A" register, if required. If the bit pointer ("H" register) is not pointing at a full character, count the character pointer ("G" register) up one.

To test for alphanumeric, the repeat field should contain the letter "A". This character in the repeat field is transferred to the "Z" register. The character in the "A" register, being pointed to by "G", is transferred to the "Y" register.

If the source string character is greater than or equal to the repeat field character, a further comparision is made to determine if the source character is either a "multiply" or "not equal" character. If the results of this compare indicate that the source character is not a "multiply" or "not equal" character and is greater than or equal to the repeat field character, the True/False Flip-flop is set to one.

The True/False Flip-flop is turned off if the source character is less than the repeat field character.

7.76 TEST FOR EQUAL (TEQL)

TEST FOR EQUAL OR LESS (TLEL)

TEST FOR GREATER (TGTL)

TEST FOR GREATER OR EQUAL (TGEL)

TEST FOR LESS (TLTL)

TEST FOR NOT EQUAL (TNEL)

PURPOSE

To compare a character in the source string with a character contained in the repeat field. If the test is met, the True/False Flip-flop is turned on, otherwise it is turned off. The characters are tested in accordance with the collating sequence.

SUMMARY OF OPERATION

Load the "A" register, if required. If the bit pointer ("H" register) is not equal to zero, count the character pointer ("G" register) up one.

The character in the repeat field is transferred to the "Z" register and the source character, being pointed to by the "G" register, is transferred to the "Y" register for comparison.

The results of the compare determine whether the source character is:

1. Greater or less than the repeat field character.
2. Equal or not equal to the repeat field character.

These two conditions, along with the decoded output of the operator matrix, will determine the state of the True/False Flip-flop.

7.77 TRANSFER DESTINATION ADDRESS (SDAL)

PURPOSE

To transfer three characters from the destination string, as specified by the "S" and "K" registers, to the "S" and "K" registers. The three most significant bits are transferred to the "K" register and the remaining 15 low order bits are transferred to the "S" register.

SUMMARY OF OPERATION

Upon entry, if the "N" register is not equal to zero, the "B" register is restored to its original configuration and stored in the destination string.

If the bit pointer ("V" register) is not equal to zero, increase the character pointer ("K" register) by one. If the "K" register was increased to seven at this time, mark the "B" register empty and load the "A" register from the destination string.

The contents of registers that will be used for data manipulation are temporarily transferred to registers that are not affected by this operator.

The "N" register is set to two and will be used to count off the three characters to be transferred. The actual transfer path is from the "A" to the "B" register via the "Y" register. If the character pointer ("G" register) is counted to seven during this transfer, the "A" register is loaded from the destination string.

The transfer of characters is completed when the "N" register is counted to zero. At this time the 15 low order bits of "B" are transferred to the "S" register and the remaining three bits are transferred to the "K" register.

Restore all registers, mark the "A" and "B" registers unoccupied and exit the operator.

7.78 TRANSFER NUMERICS (TNDL)

PURPOSE

To transfer the numeric portion of a character in the source string to the numeric portion of a character in the destination string. The number of characters transferred is determined by the value of the repeat field.

SUMMARY OF OPERATION

Load the "A" and "B" registers, if required. If the Bit pointer ("V" register) is not equal to zero, increase the character pointer ("K" register) by one. The "B" register is shifted until a proper state of alignment is achieved (KENL). If the character pointer is counted to seven during this alignment process, load the "B" register from the destination string.

After the "B" register is in alignment, transfer B16 and B15 (output alignment station) to the "Z" register and the character in "A", as specified by the "G" register, to "Y".

The numeric portion of the character contained in the "Y" register is transferred to the low order bit positions of the "B" register. This action continues until the repeat field equals zero at which time the operator is terminated.

7.79 TRANSFER BLANKS FOR NON-NUMERICS (TBZL)

PURPOSE

To test a character being pointed to by the "S" and "K" registers. If the character is non-numeric it is replaced in the string with a blank. If the character is numeric it is returned to the string and the operator is terminated. The number of characters to be tested is specified by the repeat field.

SUMMARY OF OPERATION

Load the "B" register, if required. If the bit pointer ("V" register) is not equal to zero, increase the character pointer ("K" register) by one.

With the "B" register aligned, transfer B15 and B16 (output alignment station) to the "Z" register. The character is tested to determine whether it is numeric or non-numeric.

If the character is less than or equal to zero, it is replaced by a blank and the next character in sequence is tested.

If the character is greater than zero, it is returned to the "B" register and Q03F is turned on to flag the character as numeric and the operator is terminated.

If no numeric character is encountered, the testing continues until the repeat field equals zero at which time the operation is terminated.

7.80 TRANSFER PROGRAM CHARACTERS (TDPL)

PURPOSE

To transfer characters from the program string, as specified by the "C" and "L" registers, to the destination string, as specified by the "S" and "K" registers. The number of characters to be transferred is specified by the repeat field.

SUMMARY OF OPERATION

If the repeat field is not equal to zero, and the bit pointer ("V" register) is equal to zero, load the "B" register as addressed by "S". If the bit pointer is not equal to zero, increase the character pointer ("K" register) by one.

The "B" register is shifted left or right until an alignment has been achieved (KENL). At this time B15 and B16 (output alignment station) are transferred to the "Z" register. The character in the "P" register, pointed to by "L" plus T7, is transferred to the "Y" register. The "K" register is increased by one to point at the next character of the destination string and the repeat field is decreased by one.

This action continues until the repeat field equals zero at which time the operator is terminated.

7.81 TRANSFER SOURCE ADDRESS (SSAL)

PURPOSE

This operator will replace the contents of the "G" and "M" registers with eighteen bits of three characters in the Source String. This field in the Source String is pointed at by the "G" and "M" registers.

The first three bits in the first character (positions A, B, and 8) are placed in the "G" register. The remaining fifteen bits are placed in the "M" register.

SUMMARY OF OPERATION

The bit pointer (H) is checked for zero and cleared. If it is not zero, the character pointer (G) is incremented. If the character pointer overflows, the word address is incremented and "A" is loaded.

The "G" character in "A" is placed into "Y". The "B" register is shifted left and "Y" is placed into the low order character position of "B".

The "T" register repeat count field is loaded to tally three load and shift operations.

The eighteen bits that were pointed at by the "G" and "M" registers are placed into the "G" and "M" registers.

The "A" register is marked empty and the operator is terminated.

7.82 TRANSFER SOURCE CHARACTERS (TSDL)

PURPOSE

The Transfer Source Character Operator is used to transfer "n" number of characters from the source string to the destination string. The number of characters to be transferred is found in the repeat field of the operator and is limited to a maximum of 63.

SUMMARY OF OPERATION

The operator will exit if the repeat-count field of the "T" register equals zero (TEZL) and the bit pointer for the Destination String equals zero (VEZL).

The "B" register is aligned and the B15 and B16 octades are transferred to the "Z" register. This shift into "Z" is a redundent function in this operator.

The "A" register character specified by the "G" register is shifted into the "Y" register. The "G" and "K" registers are counted up by one to point at the next character in the "A" and "B" registers.

The character from the Source String in "Y" is shifted into the B1 and B2 octades.

The repeat field of "T" is counted down by one to tally the transfer of one character.

This process continues until the repeat field of "T" is equal to zero (TEZL). When TEZL occurs the operator will exit.

7.83 TRANSFER WORDS (TWDL)

PURPOSE

To transfer words from the source string, starting at the position specified by the "M" register, to the destination string, starting at the position specified by the "S" register. The number of words transferred is specified by the repeat field.

SUMMARY OF OPERATION

Upon entry into the operator, the "B" register is restored and then placed in the destination string, if it is occupied. The character address and bit pointer for the source string are cleared. If they had not been zero, the source word address is increased by one and the "A" register is marked as empty.

If the repeat field is not equal to zero ($\overline{\text{TEZL}}$), the process of transferring words begins by loading the "A" register. With the repeat field counted down to zero (or if zero on entry into this operator) the operation terminates.

The character address and bit pointer for the destination string are cleared, and if they had not been zero, the destination word address is increased by one. The "A" register is stored at the address specified by the destination word address.

The source word address register is increased by one and "A" is reloaded. The repeat field is counted down for each store. The operation continues, loading "A" from the source string and storing the contents of "A" in the destination string. This process continues until the repeat field is counted to zero. With the repeat field equal to zero, the operation terminates with the "M" and "S" registers each addressing the next word in sequence.

7.84 TRANSFER SOURCE ZONE (TZDL)

PURPOSE

The Transfer Source Zone Operator is used to transfer the zone portion of characters in the source string to the zone portion of characters in the destination string without disturbing the numeric portion of the characters in the destination string.

SUMMARY OF OPERATION

The repeat-count field of the "T" Register specifies the number of zone transfers to be effected. If the repeat-count field of the "T" Register equals zero (TEZL) and the Bit Pointer for the destination string equals zero (VEZL), exist at the beginning of the execution cycle, exit; otherwise, these conditions will only exist after the zone bits of the desired number of characters have been transferred.

If the "A" Register is unoccupied, the "B" Register is occupied, and there is no Memory Cycle in process, the "A" Register must be loaded. If the "B" Register is unoccupied, it is loaded.

If the "V" Register is not equal to 0 ($\overline{\text{VEZL}}$), "K" is counted up by one so that it can point to the next character in sequence. If the bit pointer for "B" is not at zero ($\overline{\text{VEZL}}$) and KENL • BROF exist, the "B" Register must be octally shifted to the left. Since "K" was counted up by 1 as a result of $\overline{\text{VEZL}}$, K will no longer equal "N".

The "N" Register is counted up by 1 if VEZL is off and KENL and BROF are on. This tallies the octal left shift.

If VEZL was off, "K" will be counted up by 1 to point to the next character. If the count-up of "K" places "K" at a count of 7 (KE7L), the next character in sequence is the first character of the next word. BROF is turned off to indicate that the information in "B" is no longer valid.

If KE7L and NEZL are true in addition to $\overline{\text{VEZL}}$, the "S" Register is counted up by one to address the next word in the destination string.

The "B" Register will be in the proper state of alignment only when "K" equals "N". The 4-bit of "K" is checked. If KO4F is off, it is more expedient to shift left to achieve the desired equality. "N" is incremented with each shift and the alignment process halts when "K" equals "N".

If "K" does not equal "N" ($\overline{\text{KENL}}$) and KO4F is on, "B" will be shifted right until "K" equals "N". If, when "K" equals "N", NO1F is on, there was an odd octade shift. Hence, one more shift is needed to get the complete character into alignment.

B16 and B15 (output alignment station) are shifted to the "Z" Register. The "A" Register character pointed at by the "G" Register is shifted to the "Y" Register.

The bit pointers for the "A" and "B" registers are unconditionally cleared. The A and B bits of "Y" are shifted to B05F and B06F, respectively. "Y" contained a character from the source string. Hence, the 5 and 6 bits of "B" (zone bits of the first character position) now contain new zone bits.

"N" is counted up by one to tally the octal shift. The repeat-count field of "T" is counted down by one to tally the transfer of the zone portion of one character. This continues until the repeat-count field of the "T" register has been reduced to zero (TEZL).

If TEZL is true (repeat-count field of "T" reduced to zero) and EEZL is true (no Memory Cycle in process), exit this operator.

7.85 COMMUNICATE OPERATOR (COML)

PURPOSE

To allow the Processor to communicate with the MCP where a special sub-routine or action by the MCP is desired or needed by the program in process.

SUMMARY OF OPERATION

If the Processor is in the Control State, exit the operation. If the Processor is in the Normal State, check for the location of the top word in the stack. If it is not in the "A" or "B" registers, load it into "B" from Memory. If it is in either "A" or "B" or when placed there, initiate a Store Memory Access and store the word into the Program Reference Table. The particular location in the PRT is addressed by transferring the contents of the "R" register to the "M" register and increasing it by a value of nine. The Communication Interrupt bit in the Interrupt Register is set.

7.86 CONDITIONAL HALT (CHPL)

PURPOSE

To stop the Processor with a combination of manual intervention and a Conditional Halt operator.

SUMMARY OF OPERATION

If the Stop On Operator switch is on, and a Conditional Halt operator is executed, stop the Processor Clock. If the Stop On Operator switch is not on, continue in sequence (No-OP).

7.87 OPERAND CALL (OCSL), DESCRIPTOR CALL (DCSL)

PURPOSE

To locate an Operand or Descriptor using the ten high order bits of the syllable as a relative address. This relative address is combined with either the "R", "C" or "F" register to locate the word sought or to locate a word with the address of the word sought. Based on the word found, terminate the operator or set up for sub-routine entry and then terminate the operator.

SUMMARY OF OPERATION

The "R" register is added to the relative address when operation is not in sub-level and T12F is off. These conditions define that the address of Operand or Descriptor being called is relative to the Base Address of the Program Reference Table (PRT) contained in the "R" register. With T12F off, the area is limited to 512 cells above the Base Address.

Any operation which requires more than 512 words of PRT space when in Sub Program Level may obtain the space if a Set Variant Operator precedes the syllable which indexes it. This action resets the Sub-Level Flip-flop and sets the Variant Flip-flop to remember this fact. The operator may then index as if in Program Level as described above.

If the system is in sub-level, the three high order bit configuration of the syllable and the state of the Mark Stack Flip-flop are used to decide whether the "F", "C" or "R+7" address is used.

The "C" register is added to the relative address if operation is in sub-level, T12F and T11F are on and T10F is off. This is accomplished by transferring the contents of the "C" register to the "M" register and the address increment portion of the syllable (T10F thru T07F) to the low order bit positions of the "A" register. Add the "A" register to the "M" register and retain the sum in "M". The area addressed is limited to 128 words above the "C" register address.

The "F" register is used for indexing if the system is in sub-level with T12F on and the Mark Stack Flip-flop off with either T10F on or T11F off. With these conditions, transfer the contents of the "F" register to the "M" register. The Mark-Stack Flip-flop being off indicates that there is no previous Mark Stack Word in the stack and the address contained in the "F" register is the entry stack location.

The "R+7" address is used when the Mark Stack Flip-flop is on with T10F on or T11F off. The Mark Stack Flip-flop being on indicates that there is a previous Mark Stack Word in the stack and the required "F" register address of the entry stack location is not in the "F" register. It is in the "F" register field of the Mark Stack Control Word which is stored in the 7th cell of the Program Reference Table. Therefore, this address (R+7) is transferred to the "M" register and a special Memory Access is initiated by setting the "E" register to 6. When this word is read out, the "F" register address field of the word is transferred to the "M" register.

The "A" register is loaded with the word found at the address derived from this address indexing process. The following is a description of actions which occur depending on the type of word accessed.

If an Operand Call is being executed and the word found is an Operand, the operation is terminated. The "A" Register is marked OCCUPIED.

When a Descriptor Call is being executed and the word found is an Operand or a Control Word, it is made into a Descriptor. Essentially, if a Descriptor Call Syllable references an Operand or a Control Word, we change it to a Data Descriptor referencing a single word array which is presently pointing at the Address from which the Operand or Control Word was obtained.

If an Operand or Descriptor Call is being executed and either a Program Descriptor or a Data Descriptor is found with the Presence Bit (46th) OFF, then the Presence Interrupt is set and the operation terminated.

When a Descriptor Call is being executed, a Data Descriptor is referenced and this Descriptor references only one word; then mark the "A" Register as OCCUPIED (AROF = 1) and terminate.

If an Operand Call is being executed and a Data Descriptor is referenced with the Presence bit ON referencing only one word; then transfer the Address portion of the Data Descriptor in the "A" Register to the "M" Register and initiate a LOAD of the "A" Register using this Address.

Essentially, an Operand Call Syllable which referenced a Data Descriptor was being executed. The Data Descriptor in turn, referred to a single word which was present at the Address specified in the Data Descriptor. This word should be an Operand. If this new word is NOT an Operand, an Interrupt condition is set and the operation terminates. If it is an Operand, the operation is complete.

If a Descriptor or an Operand Call is being executed and a Data Descriptor is referenced, and the number of words referenced in both cases is not a single word (that is, the Data Descriptor is referencing or describing a group or array of words), then the word in the top of the stack is brought into the "B" Register. Depending upon its value, it is used to index the Base Address if the Data Descriptor is to develop the specific Address. On a Descriptor Call, the operation is terminated. On an Operand Call, this Address is then used to place the Operand into the "A" Register and the operation is terminated.

The Invalid Index Interrupt is set if the value of the indexing field in the "B" Register by which the Address of the Array is to be incremented is equal to or greater than the size of the array described by the Descriptor; or, if it is equal to or less than -1. If the word in the "B" Register is normalized and the exponent is greater than zero, then the Integer Overflow Interrupt is set. When the "B" Register contains an exponent less than zero and can be scaled to make the Mantissa zero, the Address is incremented by zero.

The operation is terminated if a Program Descriptor is referenced on either an Operand or Descriptor Call where arguments are required (A43F) and not supplied ($\overline{\text{MSFF}}$); or, where arguments are not required ($\overline{\text{A43F}}$) and the entry is to Character Mode (A44F).

Effectively, on encountering a Program Descriptor, a routine is to be entered. If the values needed for the sub-routine are not present, this Operator is terminated.

A spontaneous entry is when a Program Descriptor is referenced, arguments are not required ($\overline{A43F}$), and the sub-routine is to remain in Word Mode ($\overline{A44F}$). This entry requires the construction of a Mark Stack Control Word, a Return Control Word, and a branch to the Address specified by the Program Descriptor.

If the arguments (parameters) are required (A43F) and are present (MSSF), then the parameters desired are already in the stack. Therefore, a Mark Stack Control Word is already constructed and the construction of another Mark Stack Control word is unnecessary. At this time, a Return Control Word is stored and the Program is branched to the Character Word Mode sub-routine called for by the Program Descriptor.

7.88 EXIT (REWL)

PURPOSE

To provide a means of exit from sub-routine to either program or sub-program level.

SUMMARY OF OPERATION

Obtain a Return Control Word using the present "F" register contents as an address. If the Flag Bit of the word accessed is off, mark the "B" register occupied, set the Flag Bit interrupt if in Normal State and exit the operator. If the word obtained is a Return Control Word (B48F on), initiate an access of the Mark Stack Control Word, distribute the Return Control Word to the appropriate registers, temporarily store the Return Control Word in the "X" register and initiate a fetch of the program syllable following the one which initiated sub routine entry.

With the Mark Stack Control Word in the "B" register, distribute the Mark Stack Control Word to the appropriate registers. If the Mark Stack bit (B32F) and the Sub-Level bit (B31F) are on, initiate an access of the next lower Mark Stack Control Word. This process continues until the first Mark Stack Control Word is located. Control Flip-flop Q06F is set to indicate that more than one Mark Stack Control Word exists before the next Return Control Word.

When Exit is made to sub program level and more than one Mark Stack Control Word exists, store the bottom Mark Stack Control Word in R+7. This provides reference to the address of the Return Control Word of the sub routine to which Exit is made.

If Exit is made to program level, or when the first Mark Stack Control Word is found, transfer the address of the top Mark Stack Control, which was temporarily stored in the "X" register, to the "S" register and exit the operator.

7.89 SYLLABLE INTERFACE, SECL/FETCH

PURPOSE

The Syllable Execute Complete Level (SECL) is not an operator. It accomplishes several functions which are required for termination of one operator and for the establishment of the next operator. Fetch is similar since it serves to obtain operator syllables in proper time and sequence to maintain an uninterruputed flow of operators to the "T" register.

SUMMARY OF OPERATION

Syllable Execute Complete Level (SECL) is built by the logic of each operator finish pulse. At that time, control is given to SECL/FETCH to perform housekeeping, obtain the next syllable or any other special function necessary to proper operation.

Included in SECL housekeeping are logics to clear the "Y", "J", "Z" and "Q" registers. If exit is made from a Word Mode Operator, clear the "N" and "X" registers also.

Operand look ahead is implemented by the levels MRAL, MRBL and MRCL. The former looks at operators in the "P" register as they are transferred to the "T" register (PTTL), the latter two look at operators in MIR as they are transferred to the "T" register (MTTL). The purpose of this look ahead logic is to detect operators whose function may require address modification. If the operator is of the address modification type, transfer the "R" register to the high order bit positions of the "M" register and turn on Q09F in preparation for address addition.

If none of the address modification type operators are detected, clear Q09F. If operation is in Word Mode with no Interrupts, or the Controlling Processor is not in Control State, clear the "M" register.

The "T" register is cleared, during a normal Fetch, in preparation for a single ended transfer from MIR (MTTL), even though a "P" to "T" register transfer is double ended.

The "T" register is marked as occupied by an MIR to "T" register transfer or when a Fetch request has been accomplished. The "P" register is marked unoccupied when an operator sets E16 to one or after the transfer of the third program syllable.

A Store For Interrupt Operator is set in the "T" register if an interrupt has occurred, the Processor is in Normal State, the "T" register is unoccupied and J = 0. With these conditions, turn TROF back on, turn Q07F on to remember this as a hardware generated syllable and reset Q09F to eliminate address adder functions.

If this is the end of a MTR (Q04T) operation, set a Store For Test Operator in the "T" register and turn on TROF.

Reset TROF and Idle the Processor if a program word is being fetched during SECL or an Interrupt has occurred while in operating in Normal State.

The CCCF Flip-flop is strictly a MTR function. The three logical expressions in which it appears are used to terminate test of operator "X". If during test of operator "X" no memory operations were generated, the expression EEZL • SENL is used to terminate the test. If a memory write or a memory read operation was generated during the test of operator "X", the expressions E08F • MTOL and $\overline{\text{EEZL}}$ • MWOF are used, respectively.

The latter two prevent a reset of TROF until after the appropriate memory cycles have been initiated, since register values may change due to this action. The set of E16F is done under MTR conditions to prevent automatic PTTL counts of the "L" register while waiting for memory cycles.

During termination of operator "X", clear the "J" register but store its contents in TM Flip-flops (1 through 4), clear CCCF and store the NCSF Flip-flop in TM5 for later use in building a Test Initiate Control Word. During the set-up for test of operator "X", the conditions for placing the contents of TM8F in MROF, TM7F in MWOF and the clearing of the TM register will be true. If the actions of operator "X" will involve a memory cycle ($\overline{\text{SENL}}$), set VARF to allow "R" register addressing only.

The ACFL, under control of TM6F, is a special count function of MTR when entering with CCCF off. See not 5 on Initiate For Test flow (3.07.0) which cross-references this function. Count the "L" register up one for any transfer of syllables to the "T" register.

Count the "C" register up one during the Fetch of any new program word, whether the Fetch is initiated by sequential operations (ACFL), or a Fetch occurring within an operator ($\overline{\text{ICFL}}$) or a Fetch required for Interrupt handling (IO4L), provided that no branch operators are involved ($\overline{\text{E16F}}$). An additional condition for counting "C" up one is allowed when the Processor is in Normal State, and Interrupt has occurred (IO4L) with the "T" and "P" registers both empty.

Set E17F to one if a new program word is required (E16F), there is no request for memory (SENL), this is not MTR test, and there are no memory read or write cycles in progress or this is memory time zero (B 5261 High Speed Memory Units only).

With E17F set, mark the word in the "P" register as invalid. When Memory Read Access has been obtained for the new program word (MRAF), transfer MIR to the "P" register, set PROF to mark the word valid and reset E16F and E17F.

When an invalid address has been obtained without Fetch, clear the "E" register (8 through 1) to withdraw the request for memory. Similarly, if an invalid address is obtained during Fetch, clear Fetch request and set PROF.

When testing operator "X", if E16F gets set or if E16F was set by the TROF ← 0 box on this flow, clear this request when ending operator "X".

If a Memory Parity or invalid Fetch as described above occurs, enable the level to stop the Master Clock.

The Processor is considered busy (UBSS) if the "P" register is occupied (PROF), or the "T" register is occupied (TROF), or a Fetch is in progress or the Processor is in the Normal State.

7.90 HALT P2 (HP2L)

PURPOSE

To provide a means whereby Processor No. 2 can be halted.

SUMMARY OF OPERATION

If Processor No. 2 is already in a halted condition or Processor No. 1 is in the Normal State, exit the operator. If Processor No. 1 is in the Control State and Processor No. 2 is not in a halt condition, turn on the Halt P2 Flip-flop in Central Control.

7.91 INITIATE I/O (UCMIL) - INITIATE P2 (PTOL)

PURPOSE

The Initiate P_2 or Initiate I/O operators are used to activate Processor No. 2 or an Input/Output Channel, respectively.

SUMMARY OF OPERATION

If Processor No. 1 is operating in the Normal State, exit the operator. If Processor No. 1 is operating in the Control State, the top of the stack contains an Initiate Control Word. The "M" Register is set to 8 so that "M" can address word location eight of the stack. With both AROF and BROF off, the word addressed by "S" is read into "A".

If the term $\overline{\text{AROF}} \cdot \text{BROF}$ is true, "B" contains the top word of the stack. The word in "B" is stored in the address indicated by "M". If AROF is true, "A" will contain the Initiate Control Word. The word in "A" is stored in the address indicated by "M".

The Commerce Timing Level (CMTL) is combined with UCMIL. UCMIL is a decoded Initiate I/O Operator Level. Either an Initiate Processor No. 2 signal or an Initiate Input/Output signal is sent to Central Control when CMIL is true. CMTF (Commence Timing Flip-flop) is turned on and the operator is terminated.

7.92 INITIATE P1(INIL), INITIAL LOAD, INITIATED P2

PURPOSE

The Initiate P1 operator will restore the various Processor registers and control Flip-flops to their original states prior to the last interrupt.

Initial Load will initiate a fetch from cell 16.

Initiated P2 action will occur if P1 has executed an Initiate P2 operator and the resulting Initiate Level is available at this time ($\overline{PKIL}$•INITIATE).

SUMMARY OF OPERATION

The Initiate P1 operator will access and distribute, in reverse order, the Control Words that were formed during the Store For Interrupt operator. The Initiate Control Word is loaded into the "B" register first and distributed to the appropriate registers. The Initiate Return Control Word and the Interrupt Control follow.

If the Processor was placed in Word Mode by the Initiate Control Word, place the Processor in Normal State and exit. If the Processor was placed in Character Mode by the Initiate Control Word, the Interrupt Loop Control Word is accessed and distributed, the Processor placed in Normal State and the operator terminated.

Initial Load will set a 16 in the "C" register, and initiate a fetch of a Program Word located at that address.

The term Initiated P2 indicates that P1 has executed an Initiate P2 operator and the Initiate Level is available at this time. The process of restoring registers and flip-flops to their original states prior to the P2 interrupt will follow. The "B" register of P2 will be loaded with an Initiate Control Word and the distribution of Control Words will be identical to the action described for Initiate P1.

7.93 INITIATE FOR TEST (IFTL)

PURPOSE

This operator is used by the Maintenance Test Routine Program. Its purpose is to set registers and control flip-flops to a predetermined configuration so that a test can be made of enable and inhibit logic of a particular operator under test. The results of the test are then checked by the Evaluation Program section of the Maintenance Test Routine.

SUMMARY OF OPERATION

The stack will be loaded by the Maintenance Test Routine Set Up Section with a sequence of Control Words. These Control Words will contain the register and flip-flop configuration desired prior to execution of the operator being tested.

This operator produces the same results as the Initiate P1 operator with the exception of distributing sections of the Control Word which are active only during an Initiate For Test operator (T1OF on), and an exit at J = 8 which resets TROF. This exit also establishes the "J" count from which the operator under test will be initiated and the Clock Count Control (CCCF) which will determine if the operator under test will be truncated or allowed a normal SECL.

7.94 INTERROGATE INTERRUPT (IINL)

PURPOSE

If an Interrupt exists, transfer control to an Interrupt handling routine as defined by the contents of the Interrupt Address register.

SUMMARY OF OPERATION

If the Processor is in Normal State or no Interrupt exists (I03L), exit the operator. If the Processor is in the Control State and an Interrupt (I02L) does exist, transfer the contents of the Interrupt Address register, in Central Control, to the low order bit positions of the "C" register. Initiate a fetch of a program word to handle this particular interrupt condition.

7.95 I/O RELEASE (IORL)

PURPOSE

To access an I/O descriptor from the Program Reference Table, set the Presence Bit, and return the I/O descriptor to its cell within the Program Reference Table.

SUMMARY OF OPERATION

Stack adjustment will occur to bring the top word of the stack into the "A" register. If the word in the "A" register is a descriptor that is marked not present, exit the operator.

If the word in the "A" register is a descriptor that is present, use the 15 low order bits of the word to access an I/O descriptor. With the I/O descriptor in the "A" register, set the Presence Bit on and return the I/O descriptor to the cell it was accessed from.

If the word in the "A" register is an operand, the 10 low order bits of this operand will be used as a relative address to access an I/O descriptor. Set the Presence Bit of the I/O descriptor and return it to the cell from which it was accessed.

7.96 LITERAL SYLLABLE (LTSL)

PURPOSE

To transfer the literal portion of the syllable to the least significant bit positions of the "A" register.

SUMMARY OF OPERATION

Accomplish a stack push down, if required. Transfer the literal portion of the syllable to the low order bit positions of the "A" register, mark the "A" register occupied and exit the operator.

7.97 PROGRAM RELEASE (PREL)

PURPOSE

To access a descriptor from the Program Reference Table, reset the Presence Bit, and return the descriptor to its cell within the Program Reference Table.

SUMMARY OF OPERATION

Stack adjustment will occur to bring the top word of the stack into the "A" register. If the word in the "A" register is a descriptor that is marked as not present, exit the operator. If the Processor is in the Normal State, set the Presence Bit Interrupt.

If the word in the "A" register is a descriptor that is marked present, use the 15 low order bits of the word to access the descriptor that is to be acted upon. With this descriptor in the "A" register, reset the Presence Bit, set a Continuity Interrupt Bit if in Normal State and A28F is on, or a Program Release Interrupt Bit if in Control State and A28F is off. Store the descriptor in the cell it was originally accessed from. The address used to store the descriptor is also stored in R+9.

If the word in the "A" register is an operand, the 10 low order bits of this operand are used as a relative address to access the descriptor to be acted upon. With this descriptor in the "A" register, reset the Presence Bit, set a Continuity Interrupt Bit if in Normal State and A28F is on or a Program Release Interrupt Bit if A28F is off. Store the descriptor in the cell it was originally accessed from. The address used to store the descriptor is also stored in R+9.

7.98 READ TIMER (RDTL)

PURPOSE

To transfer the contents of the Real Time flip-flops in Central Control to the Processor "A" register.

SUMMARY OF OPERATION

If the Processor is in the Normal State, exit the operator. If the Processor is in the Control State, adjust the stack as required, and transfer the TM registers in Central Control, including CCI03F, to "A" register flip-flops A01F thru A07F respectively.

7.99 STORE FOR INTERRUPT (SFIL)

PURPOSE

To store the necessary registers and flip-flops so that a program may be interrupted and later resumed. It also provides for interrogation of the cause of the interrupt.

SUMMARY OF OPERATION

If Q07F is on (set by SECL), place the Processor in the Control State. The Processor may be in Character or Word Mode upon entry to this operator. The sequence of operations resulting from entry while in Character Mode will be described first.

The destination address contained in the "S" register is interchanged with "F" register field of the Loop Control Word or Return Control which is presently in the "X" register. Store the "A" and "B" registers, if required, and adjust the "S" register accordingly. Build an Interrupt Loop Control Word in the "A" register and store this Control Word as addressed by the "S" register.

Build an Interrupt Control Word and an Interrupt Return Control Word in the "B" register. Store these Control Words and adjust the "S" register accordingly. Transfer the address of the Return Control Word, presently contained in the "F" register, to the "S" register and load the "B" register with the Return Control Word.

Bit positions B(30 through 16) of the Return Control Word contain an address of a Mark Stack Control Word. Transfer this field to the "S" register and load the "B" register with a Mark Stack Control Word. Transfer bit positions B(42 through 34) of the Mark Stack Control Word to the "R" register. Build an Initiate Control Word in the "B" register, add eight to the present "R" register setting and place this sum in the "M" register. Store the Initiate Control Word as addressed by the "M" register which will be R+8.

The following is a sequence of operations resulting from entry with the Processor in Word Mode. The Interrupt Loop Control Word is not built with the Processor in Word Mode, hence the sequence of Control Words will start with an Interrupt Control Word followed by an Interrupt Return Control Word. Do not access a Return Control Word and a Mark Stack Control Word as was the case in Character Mode, but instead build an Interrupt Control Word and store this Control Word in R+8.

From this point on the operation is identical for Word Mode or Character Mode. If Processor No. 1 is being interrupted, an Interrogate Interrupt Operator is placed in the "T" register. If Processor No. 2 is being interrupted, clear the "T" register and idle. Since Processor No. 2 does not operate in the Control State, it must wait until Processor No. 1 can Store For Interrupt, Interrogate Interrupt and take appropriate action.

7.100 STORE FOR TEST (SFTL)

PURPOSE

This operator is used to store registers and control flip-flops in a manner usable by other Maintenance Test Routine Operators for testing and evaluating hardware functions.

SUMMARY OF OPERATION

This operator produces the same results as executing Store For Interrupt Operator in Character Mode, except at J09L time. Instead of the normal Interrogate Interrupt operation, special logic will enable a fetch from cell zero which will contain an Evaluation Program Syllable.