Massachusetts Institute of Technology
Instrumentation Laboratory
Cambridge, Massachusetts

TO: AGC4 Distribution
FROM: Hugh Blair-Smith
DATE: September 30, 1965, Revised July 1, 1966
SUBJECT: AGC4 MEMO # 9 - Block II Instructions
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>2</td>
</tr>
<tr>
<td>Memory</td>
<td>3</td>
</tr>
<tr>
<td>Basic Instructions</td>
<td>3</td>
</tr>
<tr>
<td>Extracode Instructions</td>
<td>11</td>
</tr>
<tr>
<td>Implied-Address Codes</td>
<td>15</td>
</tr>
<tr>
<td>Unprogrammed Sequences</td>
<td>21</td>
</tr>
<tr>
<td>Address Constant Formats</td>
<td>25</td>
</tr>
<tr>
<td>Control Pulse Definitions</td>
<td>27</td>
</tr>
<tr>
<td>Condensed List of Programmable Instructions</td>
<td>34</td>
</tr>
<tr>
<td>Pulse Sequences</td>
<td>35</td>
</tr>
</tbody>
</table>
Introduction

This document supercedes all revisions of and appendices to AGC4 Memo # 8, "Block II Instructions, Revised". The format has been changed to include more information for YUL-language programmers and to include the engineering details formerly relegated to appendices. A new descriptive section on unprogrammed sequences has been added.

Some confusion has arisen about the nature of channel numbers or addresses. Channel addresses should be used just like memory addresses in programming, that is, regarding the channels as a third category of memory, distinct from E and F. The fact that the numbers used as channel addresses coincide with some of the numbers used as memory addresses should cause no confusion, because the addresses in In/Out instructions are always channel addresses, and the addresses in other instructions are always memory addresses. In fact, the coincidence is put to good use: the L register is accessible both at memory address 0001 and at channel address 01.

In YUL language, symbols may be equated to channel addresses as well as memory addresses. The only distinction made by the assembler is that addresses of In/Out instructions have a theoretical maximum of 777.
Memory

Block II differs significantly from Block I in register and memory layout and in addressing. The LP register has been renamed L because it is a lower accumulator in every sense. The IN and OUT registers no longer have addresses in memory, but are referenced with 6-bit channel addresses by the seven input/output instructions (code 10). Channel assignments are given in Digital Development Memo #254, Revision A (Sept. 7, 1965). Figures 1 and 2 show the arrangement of addresses. The erasable banks use local addresses 1400-1777. The fixed banks use local addresses 2000-3777. Figure 3 explains the bank-switching and editing registers.

Basic Instructions

Figure 4 shows the relationships among the operation codes, with alternate spelling in brackets. Subscripts are running times, in MCT EXTEND time of 1 MCT is not included in extracode times.

Code 00.

I: TC K  Transfer Control  1 MCT

K ≠ 3, 4, 6

Set c(Q) = TC I + 1;

Take next instruction from K and proceed from there.

Remarks: Alternate spelling is TCR, for Transfer Control setting up Return.
<table>
<thead>
<tr>
<th>OCTAL PSEUDO-ADDRESS</th>
<th>REGISTER NAME</th>
<th>REMARKS</th>
<th>TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000</td>
<td>A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00001</td>
<td>L</td>
<td>(also channel 01)</td>
<td></td>
</tr>
<tr>
<td>00002</td>
<td>Q</td>
<td>(also channel 02)</td>
<td></td>
</tr>
<tr>
<td>00003</td>
<td>EB</td>
<td>Erasable Bank Register</td>
<td>Flip-flop</td>
</tr>
<tr>
<td>00004</td>
<td>FB</td>
<td>Fixed Bank Register</td>
<td>registers</td>
</tr>
<tr>
<td>00005</td>
<td>Z</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00006</td>
<td>BB</td>
<td>Both Bank Registers</td>
<td></td>
</tr>
<tr>
<td>00007</td>
<td>--</td>
<td>Zeros</td>
<td></td>
</tr>
<tr>
<td>00010</td>
<td>ARUPT</td>
<td>xRUPT = Storage for x</td>
<td></td>
</tr>
<tr>
<td>00011</td>
<td>LRUPT</td>
<td>during Interrupt;</td>
<td></td>
</tr>
<tr>
<td>00012</td>
<td>QRUPT</td>
<td>ZRUPT &amp; BRUPT stored</td>
<td></td>
</tr>
<tr>
<td>00013</td>
<td>(spare)</td>
<td>automatically.</td>
<td></td>
</tr>
<tr>
<td>00014</td>
<td>(spare)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00015</td>
<td>ZRUPT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00016</td>
<td>BBRUPT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00017</td>
<td>BRUPT</td>
<td>(RIP)</td>
<td></td>
</tr>
<tr>
<td>00020</td>
<td>CYR</td>
<td>Cycle Right 1 Bit</td>
<td>2040 words</td>
</tr>
<tr>
<td>00021</td>
<td>SR</td>
<td>Shift Right 1 Bit</td>
<td>of Erasable</td>
</tr>
<tr>
<td>00022</td>
<td>CYL</td>
<td>Cycle Left 1 Bit</td>
<td></td>
</tr>
<tr>
<td>00023</td>
<td>EDOP</td>
<td>Edit (Polish) Opcode</td>
<td></td>
</tr>
<tr>
<td>00024-00057</td>
<td>Counters</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00060-01377</td>
<td>Unswitched Erasable</td>
<td>Fixed</td>
<td></td>
</tr>
<tr>
<td>01400-03777</td>
<td>5 Erasable Banks @ 256 words</td>
<td>Fixed</td>
<td>(See Fig. 2)</td>
</tr>
<tr>
<td>04000 -up</td>
<td>Fixed (See Fig. 2)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 1
<table>
<thead>
<tr>
<th>Octal Pseudo-Address</th>
<th>Memory Type</th>
<th>Erasable Bank Reg.</th>
<th>Fixed Bank Reg.</th>
<th>Fixed Extension bit (channel 7)</th>
<th>S-Reg. Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000-01377</td>
<td>(Note 1)</td>
<td>x</td>
<td>xx</td>
<td>x</td>
<td>0000-1377</td>
</tr>
<tr>
<td>00000-00377</td>
<td>(Note 1)</td>
<td>0</td>
<td>xx</td>
<td>x</td>
<td>1400-1777</td>
</tr>
<tr>
<td>00400-00777</td>
<td>Unswitched E</td>
<td>1</td>
<td>xx</td>
<td>x</td>
<td>1400-1777</td>
</tr>
<tr>
<td>01000-01377</td>
<td>Unswitched E</td>
<td>2</td>
<td>xx</td>
<td>x</td>
<td>1400-1777</td>
</tr>
<tr>
<td>01400-01777</td>
<td>Switched E</td>
<td>3</td>
<td>xx</td>
<td>x</td>
<td>1400-1777</td>
</tr>
<tr>
<td>02000-02377</td>
<td>Switched E</td>
<td>4</td>
<td>xx</td>
<td>x</td>
<td>1400-1777</td>
</tr>
<tr>
<td>02400-02777</td>
<td>Switched E</td>
<td>5</td>
<td>xx</td>
<td>x</td>
<td>1400-1777</td>
</tr>
<tr>
<td>03000-03377</td>
<td>Switched E</td>
<td>6</td>
<td>xx</td>
<td>x</td>
<td>1400-1777</td>
</tr>
<tr>
<td>03400-03777</td>
<td>Switched E</td>
<td>7</td>
<td>xx</td>
<td>x</td>
<td>1400-1777</td>
</tr>
<tr>
<td>04000-07777</td>
<td>Fixed-fixed</td>
<td>x</td>
<td>xx</td>
<td>x</td>
<td>4000-7777</td>
</tr>
<tr>
<td>10000-11777</td>
<td>Common fixed</td>
<td>x</td>
<td>00</td>
<td>x</td>
<td>2000-3777</td>
</tr>
<tr>
<td>12000-13777</td>
<td>Common fixed</td>
<td>x</td>
<td>01</td>
<td>x</td>
<td>2000-3777</td>
</tr>
<tr>
<td>04000-05777</td>
<td>Fixed-fixed</td>
<td>x</td>
<td>02</td>
<td>x</td>
<td>2000-3777</td>
</tr>
<tr>
<td>06000-07777</td>
<td>Fixed-fixed</td>
<td>x</td>
<td>03</td>
<td>x</td>
<td>2000-3777</td>
</tr>
<tr>
<td>20000-21777</td>
<td>Common fixed</td>
<td>x</td>
<td>04</td>
<td>x</td>
<td>2000-3777</td>
</tr>
<tr>
<td>22000-23777</td>
<td>Common fixed</td>
<td>x</td>
<td>05</td>
<td>x</td>
<td>2000-3777</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>64000-65777</td>
<td>Common fixed</td>
<td>x</td>
<td>26</td>
<td>x</td>
<td>2000-3777</td>
</tr>
<tr>
<td>66000-67777</td>
<td>Common fixed</td>
<td>x</td>
<td>27</td>
<td>x</td>
<td>2000-3777</td>
</tr>
<tr>
<td>70000-71777</td>
<td>Super-bank 0</td>
<td>x</td>
<td>30</td>
<td>0</td>
<td>2000-3777</td>
</tr>
<tr>
<td>72000-73777</td>
<td>Super-bank 0</td>
<td>x</td>
<td>31</td>
<td>0</td>
<td>2000-3777</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>106000-107777</td>
<td>Super-bank 0</td>
<td>x</td>
<td>37</td>
<td>0</td>
<td>2000-3777</td>
</tr>
<tr>
<td>110000-111777</td>
<td>Super-bank 1</td>
<td>x</td>
<td>30</td>
<td>1</td>
<td>2000-3777</td>
</tr>
<tr>
<td>112000-113777</td>
<td>Super-bank 1</td>
<td>x</td>
<td>31</td>
<td>1</td>
<td>2000-3777</td>
</tr>
<tr>
<td>114000-115777</td>
<td>Super-bank 1</td>
<td>x</td>
<td>32</td>
<td>1</td>
<td>2000-3777</td>
</tr>
<tr>
<td>116000-117777</td>
<td>Super-bank 1</td>
<td>x</td>
<td>33</td>
<td>1</td>
<td>2000-3777</td>
</tr>
</tbody>
</table>

(Note 1) Flip-flop central registers, counters, and unswitched erasable. Central and special-purpose registers will be accessed as E-bank 0 only under exceptional circumstances.
BANK-SWITCHING AND EDITING REGISTERS

<table>
<thead>
<tr>
<th>Octal Address</th>
<th>Register Name</th>
<th>Access to Bank-Switching Circuits</th>
</tr>
</thead>
<tbody>
<tr>
<td>0003</td>
<td>EB</td>
<td>00000EE000000000000</td>
</tr>
<tr>
<td>0004</td>
<td>FB</td>
<td>FFF000000000000</td>
</tr>
<tr>
<td></td>
<td>(Actual Circuits)</td>
<td>SFFFF</td>
</tr>
<tr>
<td>0006</td>
<td>BB</td>
<td>FFF00000000EEEEE</td>
</tr>
<tr>
<td>Chan. 07</td>
<td>FEB</td>
<td>000000000S0000000</td>
</tr>
</tbody>
</table>

A bank number written into EB or FB is automatically available at BB. Information written into BB is automatically available at EB and FB.

EDITING REGISTER TRANSFORMATIONS

(bit positions) 15 14 13 12 11 10 09 08 07 06 05 04 03 02 01
0020             CYR           01 15 14 13 12 11 10 09 08 07 06 05 04 03 02
0021             SR            15 15 14 13 12 11 10 09 08 07 06 05 04 03 02
0022             CYL           14 13 12 11 10 09 08 07 06 05 04 03 02 01 15
0023             EDOP          -- -- -- -- -- -- -- -- --  14 13 12 11 10 09 08

Fig. 3
<table>
<thead>
<tr>
<th>00</th>
<th>01</th>
<th>02</th>
<th>03</th>
<th>04</th>
<th>05</th>
<th>06</th>
<th>07</th>
</tr>
</thead>
<tbody>
<tr>
<td>RELINT(3)</td>
<td>CCS₂</td>
<td>DAS₃</td>
<td></td>
<td></td>
<td>RESUME(17)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INHINT(4)</td>
<td></td>
<td>LXCH₂</td>
<td>CA₂</td>
<td>CS₂</td>
<td></td>
<td>INDEX₂</td>
<td></td>
</tr>
<tr>
<td>EXTEND(6)</td>
<td>TCF₁</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[NDX]</td>
<td></td>
</tr>
<tr>
<td>TC₁ [TCR]</td>
<td>INCR₂</td>
<td>[CAF]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[CAE]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADS₂</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>READ₂</td>
<td>DV₆</td>
<td>MSU₂</td>
<td>DCA₃</td>
<td>DCS₃</td>
<td>INDEX₂</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WRITE₂</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[NDX]</td>
<td></td>
</tr>
<tr>
<td>RAND₂</td>
<td>BZF₁,₂</td>
<td>QXCH₂</td>
<td></td>
<td></td>
<td></td>
<td>BZMF₁,₂</td>
<td></td>
</tr>
<tr>
<td>WAND₂</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ROR₂</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WOR₂</td>
<td>AUG₂</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RXOR₂</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**OPERATION CODES (10-17 are extracodes)**

**Fig. 4**
Code 00. I: TC K  (Special Cases of TC)  1 MCT
K = 3, 4, or 6
Set indicator specified by K;
Take next instruction from I + 1.
Remarks: TC 3 = RELINT (allow interrupt),
TC 4 = INHINT (inhibit interrupt),
TC 6 = EXTEND (set extracode switch).

The extracode switch causes the next instruction to be an extracode. Any
extracode except INDEX resets the switch. Interrupt is inhibited while the
switch is on.

Code 01. I: CCS K  Count, Compare and Skip  2 MCT
QC0
Set c(A) = DABS [b(K)];
Set c(K) = b(K), editing if K is 0020-0023.
Take next instruction from I + 1 if b(K) > + 0;
from I + 2 if b(K) = + 0;
from I + 3 if b(K) < - 0;
from I + 4 if b(K) = - 0.
Remarks: The Diminished Absolute Value of an integer x
is:
\[
DABS(x) = \begin{cases} 
|x| - 1 & \text{if } |x| > 1 \\
+ 0 & \text{if } |x| \leq 1
\end{cases}
\]

Code 01. I: TCF K  Transfer Control to Fixed  1 MCT
QC1-3
Take next instruction from K and proceed from there.
Remarks: QC n denotes Quarter Code n, where n is bits
12 and 11 of the instruction word.

Code 02. I: DAS K  Double Add to Storage  3 MCT
QC 0
Set c(K, K+1) = b(A, L) + b(K, K+1), editing if K or K + 1
is 0020-0023;
If K ≠ 0, Set c(L) = + 0 and set c(A) = net overflow;
Take next instruction from I + 1.
Remarks: If positive (negative) overflow resulted from the
double precision addition as a whole, the net overflow is +1(-1), otherwise
it is +0. Notice that DAS A doubles the contents of the double precision ac-
cumulator — implied address code DD0UBL assembles as DAS A. Since the
hardware must operate on the low-order operands first, consider DAS as the operation code 20001, to which the address K is added to form the instruction.

Code 02. I: LXCH K Exchange L and K  2 MCT  
QC1  Set c(L) = b'(K);  
Set c(K) = b(L), editing if K is 0020-0023;  
Take next instruction from I + 1.  
Remarks: The prime indicates overflow correction.

Code 02. I: INCR K Increment  2 MCT  
QC2  Set c(K) = b(K) + 1, editing if K is 0020-0023;  
Take next instruction from I + 1.  
Remarks: INCR and two other codes, AUG and DIM, are slightly modified counter-increment sequences. Accordingly, if one of this group overflows when addressing a counter for which overflow during involuntary incrementing is supposed to cause an interrupt, the interrupt will happen. This is true also for chain-reaction increments like T2, which is incremented after an overflow of T1. It should be noted that all these three instructions, unlike the increment sequences, always operate in ones complement, even when addressing CDU counters.

Code 02. I: ADS K Add to storage  2 MCT  
QC3  Set c(A), c(K) = b(K) + b(A), editing if K = 0020-0023;  
Take next instruction from I + 1.

Code 03. I: CA K Clear and Add  2 MCT  
Set c(A) = b(K);  
Set c(K) = b(K), editing if K is 0020-0023;  
Take next instruction from I + 1.  
Remarks: Alternate spelling CAF is permitted when referring to fixed memory; alternate spelling CAE is permitted when referring to erasable memory.

Code 04. I: CS K Clear and Subtract  2 MCT  
Set c(A) = -b(K);  
Set c(K) = b(K), editing if K is 0020-0023;  
Take next instruction from I + 1.
Code 05.

I: INDEX  K  Index Next Instruction  2 MCT

QC0
Set c(K) = b(K), editing if K is 0020-0023;
K ≠ 0017
Use (b(K) + c(I+1)) as the next instruction.
Remarks: The prime indicates overflow correction.

Code 05.

I: INDEX 0017 Resume Interrupted Program  2 MCT

QC0
Set c(Z) = c(0017)
K = 0017
Use c(0017) as the next instruction.
Remarks: The implied-address code RESUME assembles as

INDEX 17.

Code 05.

I: DXCH  K  Double Exchange  3 MCT

QC1
Set c(A, L) = b(K, K+1);
Set c(K, K+1) = b(A, L), editing if K or K+1 is 0020-0023;
Take next instruction from I + 1.
Remarks: The final c(L) will be overflow-corrected. The operation code should be treated as 52001 (see DAS, page 8).

The implied-address codes DTCF (DXCH FB) and DTCB (DXCH Z) are recognized. The idea is that a DXCH, by changing both Z and one of the bank registers, can be a "double-precision transfer control" that can jump banks and leave a D. P. return address in A and L.

Code 05.

I: TS  K  Transfer to Storage  2 MCT

QC2
Set c(K) = b(A), editing if K is 0020-0023;
If + overflow in b(A), set c(A) = + 1 and take next instruction from I + 2;
If no overflow in b(A), take next instruction from I + 1.
Remarks: TS A guarantees c(A) = b(A) but skips to I + 2 on overflow. Implied-address code = OVSK.

Code 05.

I: XCH  K  Exchange A and K  2 MCT

QC3
Set c(A) = b(K);
Set c(K) = b(A), editing if K is 0020-0023;
Take next instruction from I + 1.

Code 06.

I: AD  K  ADD  2 MCT

Set c(A) = b(A) + b(K);
Set c(K) = b(K), editing if K is 0020-0023;
Take next instruction from I + 1.
Remarks: The OVCTR of Block I has been dropped.
Code 07. 

I: MASK K Mask A by K 2 MCT
Set c(A) = b(A) \& c(K);
Take next instruction from I + 1.
Remarks: & denotes Boolean AND. Truth table for each bit

position of b(A) and c(K):

<table>
<thead>
<tr>
<th>A</th>
<th>K</th>
<th>A &amp; K</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

MASK very carefully omits to edit an argument from 0020-0023, in order to aid the interpreter and other software.

Extracode Instructions

Code 10. I: READ KC Read Channel KC 2 MCT
PC0 Set c(A) = c(KC), where KC is an in/out channel;
Take next instruction from I + 1.
Remarks: Code 10 is broken down into seven peripheral codes (PC0-PC6). Each uses a 9-bit address to reference an input/output channel KC. The L register is channel 01, to facilitate fancy logic in an arithmetic register. The Q register is channel 02, for the same reason.

Code 10. I: WRITE KC Write Channel KC 2 MCT
PC1 Set c(KC) = c(A);
Take next instruction from I + 1.

Code 10. I: RAND KC Read and Mask 2 MCT
PC2 Set c(A) = b(A) \& c(KC);
Take next instruction from I + 1.
Remarks; & denotes Boolean AND (see MASK).

Code 10. I: WAND KC Write and Mask 2 MCT
PC3 Set c(KC), c(A) = b(A) \& b(KC);
Take next instruction from I + 1.

Code 10. I: ROR KC Read and Superimpose 2 MCT
PC4 Set c(A) = b(A) \lor c(KC);
Take next instruction from I + 1.

Remarks: \( \lor \) denotes Boolean Inclusive OR. Truth table for each bit position of \( b(A) \) and \( c(KC) \):

<table>
<thead>
<tr>
<th>A</th>
<th>KC</th>
<th>( A \lor KC )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Code 10.  I: WOR KC  Write and Superimpose  2 MCT
PC5  Set \( c(KC), c(A) = b(A) \lor b(KC) \);
Take next instruction from I + 1.

Code 10.  I: RXOR KC  Read and Invert  2 MCT
PC6  Set \( c(A) = \overline{b(A)} \lor c(KC) \);
Take next instruction from I + 1.

Remarks: \( \lor \) denotes Boolean Exclusive OR. Truth table for each bit position of \( b(A) \) and \( c(KC) \):

<table>
<thead>
<tr>
<th>A</th>
<th>KC</th>
<th>( A \lor KC )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Code 10.  EDRU PT  3 MCT
PC7  (For machine checkout only)

Code 11.  I: DV K  Divide  6 MCT
QC0  Set \( c(A) = b(A, L) \div c(K) \);
Set \( c(L) = \) remainder;
Take next instruction from I + 1.

Remarks: The signs of the double-length dividend in \( A \) and \( L \) need not agree. The net sign of the dividend is the sign of \( b(A) \) unless \( b(A) = \pm 0 \), in which case it is the sign of \( b(L) \). The remainder bears the net dividend sign, and the quotient sign is determined strictly by the divisor and net dividend signs. DV does not disturb \( c(Q) \), and does not edit an argument from 0020-0023 because there isn't enough time.
Code 11
I: BZF K
Branch Zero to Fixed
1 or 2 MCT
QC 1-3
If a(A) = ± 0, take next instruction from K and proceed from there (1 MCT);
Otherwise, take next instruction from I + 1 (2 MCT).

Code 12.
I: MSU K
Modular Subtract
2 MCT
QC0
Set c(A) = b(A) ⊕ b(K);
Set c(K) = b(K), editing if K is 0020-0023;
Take next instruction from I + 1.
Remarks: ⊕ denotes modular subtraction, which forms a signed one's complement difference of two unsigned (modular, or periodic) two's complement inputs. The method is to form the two's complement difference, to decrement it if it is negative, and to take the overflow-uncorrected sum as the result.

Code 12.
I: QXCH K
Exchange Q and K
2 MCT
QC1
Set c(Q) = b(K);
Set c(K) = b(Q), editing if K is 0020-0023;
Take next instruction from I + 1.

Code 12.
I: AUG K
Augment
2 MCT
QC2
If b(K) ≥ + 0, set c(K) = b(K) + 1, editing if K is 0020-0023;
If b(K) ≤ - 0, set c(K) = b(K) - 1, editing if K is 0020-0023;
Take next instruction from I + 1.

Code 12.
I: DIM K
Diminish
2 MCT
QC3
If b(K) > + 0, set c(K) = b(K) - 1, editing if K is 0020-0023;
If b(K) = ± 0, set c(K) = b(K), editing if K is 0020-0023;
If b(K) < - 0, set c(K) = b(K) + 1, editing if K is 0020-0023;
Take next instruction from I + 1.
Remarks: DIM does not generate output pulses as DINC does.

Code 13.
I: DCA K
Double Clear and Add
3 MCT
Set c(A, L) = b(K, K+1);
Set c(K) = b(K), editing if K is 0020-0023;
Set c(K+1) = b(K+1), editing if K+1 is 0020-0023;
Take next instruction from I + 1.
Remarks: The final c(L) will be overflow-corrected. The operation code should be treated as 30001 (see DAS, page 8).
Code 14.

I: DCS K  Double Clear and Subtract 3 MCT

Set c(A, L) = -b(K, K+1);
Set c(K) = b(K), editing if K is 0020-0023;
Set c(K+1) - b(K+1), editing if K+1 is 0020-0023;
Take next instruction from I + 1.

Remarks: DCS A succeeds in complementing the double precision accumulator — implied-address code: DCOM. The final c(L) will be overflow-corrected. The operation code should be treated as 40001 (see DAS page 8).

Code 15.

I: INDEX K  Index Extracode Instruction 2 MCT
(See INDEX, page 10).
Remarks: This is the only extracode that does not reset the extracode switch. The way to index an extracode (MP, say) is:
EXTEND
INDEX  ADDRWD
MP  0

The extension (extracode switch) will stay in force during any n-level nesting of extracode INDEXes. This INDEX will never act as a RESUME.

Code 16.

I: SU K  Subtract 2 MCT
QC0
Set c(A) = b(A) - b(K);
Set c(K) = b(K), editing if K is 0020-0023;
Take next instruction from I + 1.

Code 16.

I: BZMF K  Branch Zero or Minus to Fixed 1 or 2 MCT
QC 1-3
If c(A) ≤ 0, take next instruction from K and proceed from there (1 MCT);
Otherwise, take next instruction from I + 1 (2 MCT).

Code 17.

I: MP K  Multiply 3 MCT
Set c(A, L) = b(A) × c(K);
Take next instruction from I + 1.

Remarks: The two words of the product agree in sign. A zero result is positive unless b(A) = ± 0 and c(K) is non-zero with the opposite sign.

MP does not edit an argument from 0020-0023 because there isn't enough time.
Implied-Address Codes

Some operations are defined for only one address value, like RESUME; others have unusual results when addressing central registers. For convenience in using these operations, the YUL System assembler recognizes implied-address codes, written without an address, and fills in the address. These codes are shown in Fig. 5 (alphabetically) and Fig. 6 (by actual code). Brief descriptions follow:

Code 00. I: XXALQ Execute Extracode 2 MCT

K = 0000

Using A, L and Q

Assume that b(A) = 000006 and b(L) is an extracode instruction;

Execute the EXTEND in A, the instruction in L, then return to I + 1; leave c(Q) = 000003.

Remarks: This is a marginally useful operation because an extracode instruction built up in L could usually be executed better by the sequence:

EXTEND

INDEX L

0 0

Code 00. I: XLQ Execute using L and Q 2 MCT

K = 0001

Assume that b(L) is a basic instruction.

Execute the instruction in L and, if it is not a successful branch, return to I + 1;

Leave c(Q) = 000003.

Remarks: Like XXALQ, this operation is marginal.

The time (2 MCT) for XXALQ and XLQ includes the TC to A or L and the return TC from Q, but not the time spent in executing c(A) or c(L).

Code 00. I: RETURN Return from Subroutine 2 MCT

K = 0002

Assume that b(Q) = TC K;
Take the next instruction from \( K' \) and proceed from there; Leave \( c(Q) = 00003 \).

**Code 00.**

**I: RELINT**  Release (allow) Interrupt  

\( K = 0003 \)

Allow interrupt after this instruction (subject to the restriction that interrupt cannot occur while there is + overflow in \( A \));

Take next instruction from \( I + 1 \).

**Remarks:** The inhibition set by INHINT and removed by RELINT is entirely independent of the one set by interrupt and removed by RESUME.

**Code 00.**

**I: INHINT**  Inhibit Interrupt  

\( K = 0004 \)

Inhibit interrupt until a subsequent RELINT;

Take next instruction from \( I + 1 \).

**Remarks:** If the next instruction is INDEX (full code 15), the following instruction will be executed as an extracode too.

**Code 00.**

**I: EXTEND**  Extend Next Instruction  

\( K = 0006 \)

Take the next instruction from \( I + 1 \) and execute it as an extracode.

**Remarks:** This is how NOOP is assembled when \( I \) is in fixed memory.

**Code 01.**

**I: NOOP**  No Operation (Fixed)  

\( QC 1 - 3 \)

Take the next instruction from \( I + 1 \).

\( K = I + 1 \)

**Remarks:** If \( b(A) \) contains + overflow, the results are messy; in particular, \( \text{sgn} \left[ c(A) \right] \neq \text{sgn} \left[ b(A) \right] \). If \( \left| b(A) \right| \geq 1/2 \), overflow will be retained in \( c(A) \).
<table>
<thead>
<tr>
<th>Implied-Address Code</th>
<th>Actual Operation Code</th>
<th>Register (If applicable)</th>
<th>Word as assembled</th>
<th>NOTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>COM</td>
<td>CS</td>
<td>A</td>
<td>40000</td>
<td></td>
</tr>
<tr>
<td>DCOM</td>
<td>DCS</td>
<td>A</td>
<td>40001</td>
<td>X</td>
</tr>
<tr>
<td>DDOUBL</td>
<td>DAS</td>
<td>A</td>
<td>20001</td>
<td></td>
</tr>
<tr>
<td>DOUBLE</td>
<td>AD</td>
<td>A</td>
<td>60000</td>
<td></td>
</tr>
<tr>
<td>DTCB</td>
<td>DXCH</td>
<td>Z</td>
<td>52006</td>
<td></td>
</tr>
<tr>
<td>DTCF</td>
<td>DXCH</td>
<td>FB</td>
<td>52005</td>
<td></td>
</tr>
<tr>
<td>EXTEND</td>
<td>TC</td>
<td></td>
<td>00006</td>
<td>S</td>
</tr>
<tr>
<td>INHINT</td>
<td>TC</td>
<td></td>
<td>00004</td>
<td>S</td>
</tr>
<tr>
<td>NOOP</td>
<td>TCF</td>
<td></td>
<td>1 (I+1)</td>
<td>F</td>
</tr>
<tr>
<td>NOOP</td>
<td>CA</td>
<td>A</td>
<td>30000</td>
<td>E</td>
</tr>
<tr>
<td>OVSK</td>
<td>TS</td>
<td>A</td>
<td>54000</td>
<td></td>
</tr>
<tr>
<td>RELINT</td>
<td>TC</td>
<td></td>
<td>00003</td>
<td>S</td>
</tr>
<tr>
<td>RESUME</td>
<td>INDEX</td>
<td>BRUPT</td>
<td>50017</td>
<td>R</td>
</tr>
<tr>
<td>RETURN</td>
<td>TC</td>
<td>Q</td>
<td>00002</td>
<td></td>
</tr>
<tr>
<td>SQUARE</td>
<td>MP</td>
<td>A</td>
<td>70000</td>
<td>X</td>
</tr>
<tr>
<td>TCAA</td>
<td>TS</td>
<td>Z</td>
<td>54005</td>
<td></td>
</tr>
<tr>
<td>XLQ</td>
<td>TC</td>
<td>L</td>
<td>00001</td>
<td></td>
</tr>
<tr>
<td>XXALQ</td>
<td>TC</td>
<td>A</td>
<td>00000</td>
<td></td>
</tr>
<tr>
<td>ZL</td>
<td>LXCH</td>
<td></td>
<td>22007</td>
<td>X</td>
</tr>
<tr>
<td>ZQ</td>
<td>QXCH</td>
<td></td>
<td>22007</td>
<td>X</td>
</tr>
</tbody>
</table>

NOTE EXPLANATION:

E  Applies when I (location of instruction) is in erasable memory.
F  Applies when I is in fixed memory.
R  Special RESUME hardware responds to address 0017.
S  Special Indicator-setting hardware responds to addresses 0003, 0004, and 0006.
X  Extracode instruction.

Fig. 5
### IMPLIED ADDRESS CODES

(By Actual Code)

<table>
<thead>
<tr>
<th>Actual Operation Code</th>
<th>Register (If applicable)</th>
<th>Word as assembled</th>
<th>Implied-Address Code</th>
<th>NOTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>TC</td>
<td>A</td>
<td>00000</td>
<td>XXALQ</td>
<td></td>
</tr>
<tr>
<td>TC</td>
<td>L</td>
<td>00001</td>
<td>XLQ</td>
<td></td>
</tr>
<tr>
<td>TC</td>
<td>Q</td>
<td>00002</td>
<td>RETURN</td>
<td></td>
</tr>
<tr>
<td>TC</td>
<td></td>
<td>00003</td>
<td>RELINT S</td>
<td></td>
</tr>
<tr>
<td>TC</td>
<td></td>
<td>00004</td>
<td>INHINT S</td>
<td></td>
</tr>
<tr>
<td>TC</td>
<td></td>
<td>00006</td>
<td>EXTEND S</td>
<td></td>
</tr>
<tr>
<td>TCF</td>
<td></td>
<td>1 (I+1)</td>
<td>NOOP F</td>
<td></td>
</tr>
<tr>
<td>DAS</td>
<td>A</td>
<td>20001</td>
<td>DDOUBL</td>
<td></td>
</tr>
<tr>
<td>LXCH</td>
<td>A</td>
<td>22007</td>
<td>ZL</td>
<td></td>
</tr>
<tr>
<td>CS</td>
<td>A</td>
<td>30000</td>
<td>NOOP E</td>
<td></td>
</tr>
<tr>
<td>INDEX</td>
<td>BRUPT</td>
<td>50017</td>
<td>RESUME R</td>
<td></td>
</tr>
<tr>
<td>DXCH</td>
<td>FB</td>
<td>52005</td>
<td>DTCF</td>
<td></td>
</tr>
<tr>
<td>DXCH</td>
<td>Z</td>
<td>52006</td>
<td>DTCB</td>
<td></td>
</tr>
<tr>
<td>TS</td>
<td>A</td>
<td>54000</td>
<td>OVKSK</td>
<td></td>
</tr>
<tr>
<td>TS</td>
<td>Z</td>
<td>54005</td>
<td>TCAA</td>
<td></td>
</tr>
<tr>
<td>AD</td>
<td>A</td>
<td>60000</td>
<td>DOUBLE</td>
<td></td>
</tr>
<tr>
<td>QXCH</td>
<td></td>
<td>22007</td>
<td>ZQ X</td>
<td></td>
</tr>
<tr>
<td>DCS</td>
<td>A</td>
<td>40001</td>
<td>DCOM X</td>
<td></td>
</tr>
<tr>
<td>MP</td>
<td>A</td>
<td>70000</td>
<td>SQUARE X</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 6
Code 02. I: ZL Zero L
QC 1 Set c(L) = +0;
K = 0007 Take next instruction from I + 1.
Remarks: This code and its companion ZQ depend on two properties of address 0007: no storage is associated with it, and references to it (in fact, to any of 0000-0007) are not checked for good parity. Address 0007 is therefore a generally usable source of zeros.

Code 03 I: NOOP No Operation (Erasable)
K = 0000 Take next instruction from I + 1.
Remarks: This is how NOOP is assembled when I is in erasable memory.

Code 04. I: COM Complement c(A)
K = 0000 Set c(A) = -b(A);
Take next instruction from I + 1.
Remarks: All 16 bits are complemented.

Code 05. I: RESUME Resume Interrupted Program
QC 0 Set c(Z) = c(0015);
K = 0017 Use c(0017) as the next instruction.

Code 05. I: DTCF Double Transfer Control,
QC 1 Switching F bank
K = 0004 Set c(A,L) = b(FB,Z);
Set c(FB,Z) = b(A,L);
Take next instruction from I + 1.
Remarks: A double-precision address constant format, 2FCADR, is defined for use with DTCF.
Code 05.  I: DTCB  Double Transfer Control  3 MCT
        QC 1  Switching Both Banks

K = 0005  Set c(A, L) = b(Z, BB);
          Set c(Z, BB) = b(A, L);
          Take next instruction from I + 1.

Remarks: A double-precision address constant format, 2 BCADR, is defined for use with DTCB.

Code 05.  I: OVSK  Overflow Skip  2 MCT
        QC 2  Do not change c(A);

K = 0000  If + overflow in c(A), take next instruction from I + 2;
          If no overflow in c(A), take next instruction from I + 1.

Code 05.  I: TCAA  Transfer Control to  2 MCT
        QC 2  Address in A

K = 0005  If + overflow in b(A), set c(A) = +1;
          Take next instruction from the location whose address is
          in bits 12-1 of b(A).

Remarks: The perils associated with TCAA in Mod 3C and Block I AGC do not exist in Block II AGC.

Code 06.  I: DOUBLE  Double c(A)  2 MCT

K = 0000  Set c(A) = b(A) + b(A);
          Take next instruction from I + 1.

Remarks: See remarks on overflow under DDOUBL.

Code 12.  I: ZQ  Zero Q  2 MCT
        QC 1  Set c(Q) = +0;

K = 0007  Take next instruction from I + 1.

Remarks: See under ZL.
K = 0000  Set c(A, L) = - b(A, L);
        Take next instruction from I + 1.
Remarks: All 32 bits of A and L are complemented.

Code 17.  I: SQUARE    Square c(A)      3 MCT
K = 0000  Set c(A, L) = b(A) × b(A);
        Take next instruction from I + 1.
Remarks: Results are messy if b(A) contains + overflow.

Unprogrammed Sequences

Some of the actions performed by the computer are not programmed but occur in response to external events. The categories of these unprogrammed sequences are shown in Fig. 7. Interrupt is inhibited if an interrupt has occurred after the latest RESUME, or an INHINT has occurred after the latest RELINT, or c(A) contains + overflow. Otherwise interrupt may occur before any basic (non-extracode) instruction except RELINT, INHINT, or EXTEND.

RUPT          Interrupt Program      3 MCT
Set c(0015) = b(Z);
Set c(0017) = the postponed instruction;
Take next instruction from the location whose address is permanently associated with the cause of the interrupt, and proceed from there. Inhibit further interrupt until RESUME.
Remarks: See also remarks under INHINT.

Counter increments and decrements, serial-parallel conversion steps, and GSE interface transactions are lumped together under the name of counter interrupts because they perform limited tasks by snatching one or two memory cycles and then let the computer continue. They can occur before any instruction except RELINT, INHINT or EXTEND.
UNPROGRAMMED SEQUENCES

Program Interrupt

Counter Increment/Decrement

Serial-Parallel Conversion (and vice-versa)

Ground Support Interface

Manual Override

RUPT
PINC
PCDU
MINC
MCDU
DINC
SHINC
SHANC
INOTRD
INOTLD
FETCH
STORE
GOJ
TCSAJ

Fig. 7
PINC  Plus Increment  1 MCT

Set c(CTR) = b(CTR) + 1;
If + overflow, set c(CTR) = +0 and set up
an interrupt if CTR = T3, T4 or T5 or set up a PINC for T2
if CTR = T1.

Remarks: This sequence and its priority chain effects
are shared by the instruction INCR.

PCDU  Plus Increment (CDU)  1 MCT

Set c(CDUCTR) = b(CDUCCTR) + 1 in two's
complement modular notation.

Remarks: Incrementing in two's-complement modular
notation transforms 77777 into 00000 and 37777 into 40000, and is other-
wise like one's-complement. INC never acts like PCDU. PCDU and
MCDU replace PINC and MINC for counters 0032-0036.

MINC  Minus Increment  1 MCT

Set c(CTR) = b(CTR) - 1;
If - overflow, set c(CTR) = -0.

MCDU  Minus Increment (CDU)  1 MCT

Set c(CDUCCTR) = c(CDUCCTR) - 1 in two's
complement modular notation.

Remarks: Transforms 40000 into 37777 and 00000
into 77777. See remarks under PCDU.

DINC  Diminishing Increment  1 MCT

If c(CTR) > +0, set c(CTR) = b(CTR) - 1 and
emit signal POUT (Plus Output);
If c(CTR) < -0, set c(CTR) = b(CTR) + 1 and
emit signal MOUT (Minus Output);
If c(CTR) = ±0, leave c(CTR) unchanged and
emit signal ZOUT (Zero Output & turn off DINC request).
Remarks: Used to generate output pulse trains and to count down T6. Values to be counted down by DINC might be developed by the instruction MSU from a desired and an actual CDU angle. This sequence is shared by the instruction DIM, but without POUT, MOUT and ZOUT.

SHINC Shift Increment

Set c(CTR) = b(CTR) + b(CTR);
If + overflow, set the priority chain station
for this counter.

Remarks: SHINC and SHANC are used to convert incoming serial bit streams into words for parallel access, and to convert words to outgoing serial bit streams.

SHANC Shift and Add Increment

Set c(CTR) = b(CTR) + b(CTR) + 1;
If + overflow, set the priority chain station
for this counter.

Remarks: See under SHINC.

INOTRD In/Out Read to GSE

Accept a channel address from the Ground Support Equipment and place the contents of the addressed input/output channel on the GSE data busses.

INOTLD In/Out Load from GSE

Accept a channel address from the Ground Support Equipment and write the contents of the GSE data busses into the addressed input/output channel.

FETCH Fetch from Memory to GSE

Accept from the Ground Support Equipment a setting for either FB or EB and an address for the corresponding memory, and place the contents of the addresses location on the GSE data busses. Do not edit if the address is 0020-0023. Then restore b(BB).
STORE

Store in Memory from GSE

Accept from the Ground Support Equipment a setting for EB and an address in erasable memory, and write the contents of the GSE data busses into the addressed location. Then restore \( b(BB) \), unless the location stored into is BB itself.

The manual override instructions can occur at any time because they are not obliged to preserve the state of the computer.

GOJ

Go Jam

Set \( c(Q) = b(Z) \);

Take next instruction from location 4000 and proceed from there.

TCSAJ

Transfer Control to Specified Address Jam

Take next instruction from the location whose address is on the Ground Support Equipment data busses, and proceed from there.

Address Constant Formats

The address constants available for Block II programming are considerably different than for Block I. A summary of them follows. The EBANK= code is also discussed.

ADRES Address
REMADR Remote Address
GENADR General Address
Each of these codes creates a single precision constant word identical to the instruction word that would have resulted if the opcode had been TC. ADRES requires the location and address values to be in the same F - Bank if both are in F - Banks and to be in the same E - Bank if both are in E - Banks. REMADR requires the location and address values to be in different F - Banks if both are in F - Banks and to be in different E - Banks if both are in E - Banks. GENADR doesn't care.

CADR FCADR (Fixed) Complete Address

These codes are synonymous. The address value must be in an F - Bank. The resulting single precision constant word equals the pseudo-address value minus octal 10000. Bits 15-11 equal the F - Bank number and bits 10 - 1 equal the relative location of the address in that bank.

CADR Erasable Complete Address

The address value must be erasable, 0000-3777, and the resulting single precision word equals the the eleven bit pseudo-address. Bits 15-12 = 0.

EBANK= Erasable Bank Declaration

This code does not generate an AGC word. It informs the assembler of which E-Bank the programmer intends subsequent E-Bank addresses to be in. For basic instructions and interpretive address words, the assembler complains wherever an address is equivalent to a location in a different E-Bank. If the EBANK= code is followed by* a BBCON, 2BCADR or 2CADR code, this EBANK= value is good only for that one subsequent code, and then the previous EBANK= setting is restored. This is called a "one-shot EBANK= declaration".

* "followed by" means with no instructions, interpretive opcode words, or address constants intervening.
BBCON  Both Bank Constant

This code generates a single precision constant word intended as
data to be placed in the BB central register. The address value must
be a fixed memory location or it must be equivalent to a valid F-Bank
number, (range 0-27 now, 0-43 later). Bits 15-11 of the resulting word
equal the address' bank number (fixed - fixed being banks 2 and 3).
Bits 10 - 4 are zeros. Bits 3 - 1 equal the current EBANK= code.

2CADR  2BCADR  Double Complete Address Including a BBCON

These codes are synonymous. This code is intended to be used as
the operand of a DTCB (DXCH Z) instruction. Two constant words are
generated by this code. The first word is formed under the rules for
GENADR. If the address value is in fixed memory, the second word is
formed under the rules for BBCON. For an erasable address the second
word becomes 0000x where x = the address' octal code EBANK number
in the range 0 - 7.

2FCADR  Double Complete Address Including an FCADR

This code's address value must be in fixed memory. The code is
intended as an operand of a DTCF (DXCH FB) instruction. Two constant
words are generated by this code. The first word is formed under the
rules for FCADR, and second under the rules for GENADR. Exception:
both words are GENADRs if address value is in fixed fixed.

Control Pulse Definitions

To understand the control pulses and the pulse sequences, it is necessary
to know the unaddressable central registers:

G  Memory Local Register  Bits 1 - 16

In an MCT in which erasable memory is cycled,
the word from memory appears in G by the 5th microsecond (time 5
of 12 times) of the MCT. If it is left there through time 12, it is

27
restored exactly as it was read out. If a new value is written into G before time 10, that becomes the new value in the memory location. When fixed memory is cycled, the word appears in G by time 7.

WL Write Lines or Busses Bits 1 - 16
These are the normal medium of communication among central registers, although some private lines exist.

B General Buffer Register Bits 1 - 16
The B register always holds the instruction word at the beginning of each instruction.

C Complement Output of B Bits 1 - 16
Not a separate storage. Each bit of C is the opposite of the corresponding bit of B.

Y Primary Adder Input Bits 1 - 16
Has conventional and doubling inputs.

X Secondary Adder Input Bits 1 - 16
Fed by private line from A and from constant generators.

U Adder Output Bits 1 - 16
Exists as a function of c(X) and c(Y) - has no storage of its own.

S Address Selection Register Bits 1 - 12
Holds the address of a fixed memory location from time 8 of the preceding MCT through time 7 of the current MCT, or holds (in bits 1 - 10) the address of an erasable memory location from time 1 through time 7 of an MCT.
SQ  Sequence Selection Register  Bits 10-16

Holds the operation code during execution of each instruction. Bit 15 is the extracode bit. SQ is aided by a three-bit stage counter and two branch flip-flops. A stage counter value of 2 selects the standard fetch-next-instruction subinstruction, regardless of the c(SQ) and the branch bits. Sequence selection by SQ is suppressed during counter interrupts by a signal called INKL.
CONTROL PULSF DEFINITIONS

A2X

COPY A1-16 INTO X1-16 BY PRIVATE LINE.

R15X

SET BIT 15 OF X TO 1.

CI

INSERT CARRY INTO BIT 1 OF THE ADDER.

CLXC

CLEAR X CONDITIONAL ON THE OUTCOME OF TSGU. X IS CLEARED IF BR1 = 0. USED IN DIVIDE.

DVST

CAUSE DIVIDE STAGING BY A SIMPLE RULF. ALSO PERMIT STAGING TO OCCUR AT TIMES OF DIVIDE CYCLES.

EXT

SET THE EXTEND FLIP FLOP.

G2LS

COPY G4-15,16,1 INTO L1-12,16,15.

KRP1

RESET INTERRUPT PRIORITY CELL.

L16

SET BIT 16 OF L TO 1.

COPY L1-14,16 INTO G2-15,16 -- ALSO MCRO INTO G1.

L2GD

MONEY

SET BITS 2-16 OF X TO ONES.

MO1UT

NEGATIVE RATF OUTPUT PULSE.

NEACOF

PERMIT END AROUND CARRY AFTER END OF MP3.

NEACON

INHIBIT END AROUND CARRY UNTIL NEACOF.

NISO

NEXT INSTRUCTION IS TO BE LOADED INTO SQ. ALSO FREES CERTAIN RESTRICTIONS- PERMITS INCREMENTS AND INTERRUPTS.

PIFL

WHEN L15 = 1, BLOCK WRITING INTO Y1 ON A WYD.

PONEX

SET BIT 1 OF X TO 1.

PO1UT

POSITIVE RATF OUTPUT PULSE.

PTWOX

SET BIT 2 OF X TO 1.

R15

PLACE OCTAL 000015 ON WL'S.
**CONTROL PULSF DEFINITIONS**

- **R1C**
  - **PLACE OCTAL 177776 = -1 ON WL'S.**

- **R6**
  - **PLACE OCTAL 000006 ON WRITE LINES.**

- **RA**
  - **READ A1-16 TO WL1-16.**

- **RAD**
  - **READ ADDRESS OF NEXT CYCLE. THIS APPEARS AT THE END OF AN INSTRUCTION AND NORMALLY IS INTERPRETED AS RG. IF THE NEXT INSTRUCTION IS TO BE A PSFUDU CODE (INHINT, RELINT, EXTEND) IT IS INSTEAD INTERPRETED AS RZ ST2.**

- **RB**
  - **READ B1-16 TO WL1-16.**

- **RB1**
  - **PLACE OCTAL 000001 ON THE WL'S.**

- **RB1F**
  - **PLACE OCTAL 000001 ON THE WL'S CONDITIONAL ON THE OUTCOME OF ISGU. RB1F IF RR1=1.**

- **RB2**
  - **PLACE OCTAL 000002 ON THE WL'S.**

- **RB8K**
  - **READ THE BLOCK (BOTH RANK) CONFIGURATION ONTO THE WRITE LINES; I.E., FB 9-11 TO WL 1-3 AND FB 11-14-16 TO WL 11-14-15-16.**

- **RC**
  - **READ THE CONTENT OF B INVERTED: C1-16 TO WL1-16.**

- **RCH**
  - **READ THE CONTENT OF THE INPUT OR OUTPUT CHANNEL SPECIFIED BY THE CURRENT CONTENT OF 5; CHANNEL BITS 1-14 TO WL1-14, AND HIT 16 TO WL15-16. CHANNELS 1 AND 2 READ AS RL AND RQ.**

- **RG**
  - **READ G1-16 TO WL1-16.**

- **RL**
  - **READ L1-14 TO WL1-14 AND L1A TO WL15 AND 16.**

- **RL10RB**
  - **READ LOW 10 BITS OF B TO WL 1-10.**

- **RQ**
  - **READ Q1-16 TO WL1-16.**

- **RRPA**
  - **READ THE ADDRESS OF THE HIGHEST PRIORITY INTERRUPT REQUESTED.**

- **RSC**
  - **READ THE CONTENT OF CENTRAL STORE DEFINED BY THE ADDRESS CURRENTLY IN 5; CENTRAL STORE BITS 1-16 ARE COPIED TO WL1-16.**

- **RSC1**
  - **READ THE ADDRESS OF HIGHEST PRIORITY COUNTER REQUEST.**

- **RST3T**
  - **PLACE OCTAL 004000 = BLOCK 2 START ADDRESS ON WL'S.**

- **RSTSTG**
  - **RESET THE DIVIDE TO 03 STAGING CONDITION.**

- **RU**
  - **READ U1-16 TO WL1-16.**
CONTROL PULSF DEFINITIONS

BUS
READ U1-14 TO WL1-14, AND U15 TO WL15 AND 16.
READ 21-16 TO WL1-16.

RZ

ST1
SET STAGF1 FLIP FLOP NEXT T17.

ST2
SET STAGF2 FLIP FLOP NEXT T17.

STAGF
EXECUTE GRAY-CODED STAGE ADVANCE COMPUTED BY DVST.

TL15
COPY L15 INTO BR1.

TM7
TEST WL1-16 FOR ALL ONES (-0). SET BR2 IF TRUE.

TOV
TEST FOR + OR - OVERFLOW. SET BR1*2 TO 00 IF NO OVERFLOW, 11 IF + OVERFLOW, 10 IF - OVERFLOW.

TPZG
TEST CONTENT OF G FOR PLUS ZERO. IF TRUE SET BR2=1.

TR5H
TEST FOR RFSHIME ADDRESS ON INDEX. ST2 IF (S)=0017.

TSGN
TEST SIGN. COPY WL16 TO BR1.

TSGN2
TEST SIGN. COPY WL16 TO BR2.

TSGU
TEST SIGN OF SUM (U). COPY U16 INTO BR1.

U2BBK
ADDER BITS 1-3 AND 11-14,16 ARE TRANSFERRED INTO ERASABLE AND FIXED BANKS. THIS PULSF MAY BE INHIBITED BY CTS SIGNAL MONBBK.

WA
CLEAR AND WRITE WL1-16 INTO A1-16.

WALS
CLEAR AND WRITE INTO A1-14 FROM WL3-16. CLEAR AND WRITE INTO L13-14 FROM WL1-2. CLEAR AND WRITE INTO A15-16 FROM G16 (IF G1=0) OR FROM WL16 (IF G1=1).

WB
CLEAR AND WRITE WL1-16 INTO P1-16.

WCH
CLEAR AND WRITE WL1-14,16, PARITY INTO CHANNEL B1TS 1-14,16, PARITY. CHANNELS 1 AND 2 WRITE AS WL AND WQ. THE CHANNEL TO BE LOADED IS SPECIFIED BY THE CURRENT CONTENT OF S.

WG
CLEAR AND WRITE WL1-16 INTO G1-16 EXCEPT FOR ADDRESSES OCTAL 20-23, WHICH CAUSE EDITING.

CLEAR AND WRITE WL1-16 INTO L1-16.
CONTROL PULSE DEFINITIONS

WOVR
TEST FOR OVERFLOW DURING COUNTER INCREMENTS AND
PROGRAM INITIATED INCREMENTS (INCR AND AUG). RUP IF
OVERFLOW OCCURS WHEN ADDRESSING CERTAIN COUNTERS.

WO
CLEAR AND WRITE WL1-16 INTO Y1-16.

WS.
CLEAR AND WRITE WL1-12 INTO S1-12.

WSC
CLEAR AND WRITE WL1-16 INTO THE CENTRAL REGISTER
SPECIFIED BY THE CURRENT CONTENT OF S. BITS
1-16 INTO POSITIONS 1-16.

WSQ
CLEAR AND WRITE WL10-14, 16 INTO SQ10-14, 16, AND COPY
THE EXTEND FLIP FLOP INTO SQ15.

WY
CLEAR Y AND X, WRITE WL1-16 INTO Y1-16.

WY12
CLEAR Y AND X, WRITE WL1-12 INTO Y1-12.

WYD
CLEAR Y AND X, WRITE WL1-14 INTO Y2-15,
WRITE WL16 INTO Y16. WRITE WL16 INTO Y1 EXCEPT:
(1) WHEN END-AROUND CARRY IS INHIBITED BY MEACON,
(2) DURING SHRIN SEQUENCE, OR
(3) PIFL IS ACTIVE AND L15 = 1.

WZ
CLEAR AND WRITE WL1-16 INTO Z1-16.

Z15
SET BIT 15 OF Z TO 1.

Z16
SET BIT 16 OF Z TO 1.

ZAP
ALWAYS IMPLIF S RU, GZLS, AND WALS.

ZIP
ALWAYS IMPLIF S A2X AND L2GD. ALSO IF L15, 2, 1 ARE:

<table>
<thead>
<tr>
<th>L15 L2 L1</th>
<th>READ WRITE CARRY REMEMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0</td>
<td>- WY - -</td>
</tr>
<tr>
<td>0 0 1</td>
<td>RB WY - -</td>
</tr>
<tr>
<td>0 1 0</td>
<td>RB WYD - -</td>
</tr>
<tr>
<td>0 1 1</td>
<td>RC WY CI MCRO</td>
</tr>
<tr>
<td>1 0 0</td>
<td>RB WY - -</td>
</tr>
<tr>
<td>1 0 1</td>
<td>RB WYD - -</td>
</tr>
<tr>
<td>1 1 0</td>
<td>RC WY CI MCRO</td>
</tr>
<tr>
<td>1 1 1</td>
<td>- WY - MCRO</td>
</tr>
</tbody>
</table>

ZOUT
NO RATE OUTPUT PULSE. RESET OUTBIT REQUESTING DINC.

* THESE PULSES DO NOT APPEAR IN THE PULSE SEQUENCES.
### Programmable Instructions

<table>
<thead>
<tr>
<th>OP CODE</th>
<th>EXT 5016-14-10</th>
<th>OPERATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>TC</td>
<td>0 000</td>
<td>TRANSFER CONTROL AND PSEUDO-CODES *</td>
</tr>
<tr>
<td>CCS</td>
<td>0 001 00</td>
<td>COUNT, COMPARE, AND SKIP</td>
</tr>
<tr>
<td>TCF</td>
<td>0 001 01</td>
<td>TRANSFER CONTROL TO FIXED</td>
</tr>
<tr>
<td>TCF</td>
<td>0 001 10</td>
<td></td>
</tr>
<tr>
<td>TCF</td>
<td>0 001 11</td>
<td></td>
</tr>
<tr>
<td>DAS</td>
<td>0 010 00</td>
<td>DOUBLE PRECISION ADD TO STORAGE</td>
</tr>
<tr>
<td>LXCH</td>
<td>0 010 01</td>
<td>EXCHANGE WITH MEMORY</td>
</tr>
<tr>
<td>INCR</td>
<td>0 010 10</td>
<td>INCREMENT MEMORY</td>
</tr>
<tr>
<td>ADS</td>
<td>0 010 11</td>
<td>ADD TO STORAGE</td>
</tr>
<tr>
<td>CA</td>
<td>0 011</td>
<td>CLEAR AND ADD</td>
</tr>
<tr>
<td>CS</td>
<td>0 100</td>
<td>CLEAR AND SUBTRACT</td>
</tr>
<tr>
<td>INDEX (NDX)</td>
<td>0 101 00</td>
<td>INDEX NEXT INSTRUCTION (INDEX 17=RESUME)</td>
</tr>
<tr>
<td>DXCH</td>
<td>0 101 01</td>
<td>DOUBLE PRECISION EXCHANGE WITH MEMORY</td>
</tr>
<tr>
<td>TS</td>
<td>0 101 10</td>
<td>TRANSFER TO STORAGE</td>
</tr>
<tr>
<td>XCH</td>
<td>0 101 11</td>
<td>EXCHANGE WITH MEMORY</td>
</tr>
<tr>
<td>AD</td>
<td>0 110</td>
<td>ADD</td>
</tr>
<tr>
<td>MASK (MSK)</td>
<td>0 111</td>
<td>MASK (&quot;AND&quot; TO A)</td>
</tr>
<tr>
<td>READ</td>
<td>1 000 00 0</td>
<td>READ FROM CHANNEL</td>
</tr>
<tr>
<td>WRITF</td>
<td>1 000 00 1</td>
<td>WRITE IN CHANNEL</td>
</tr>
<tr>
<td>RAND</td>
<td>1 000 01 0</td>
<td>READ, &quot;AND&quot; TO A</td>
</tr>
<tr>
<td>WAND</td>
<td>1 000 01 1</td>
<td>WRITE, &quot;AND&quot; TO CHANNEL</td>
</tr>
<tr>
<td>ROR</td>
<td>1 000 10 0</td>
<td>READ, &quot;OR&quot; TO A</td>
</tr>
<tr>
<td>WOR</td>
<td>1 000 10 1</td>
<td>WRITE, &quot;OR&quot; TO CHANNEL</td>
</tr>
<tr>
<td>RXOR</td>
<td>1 000 11 0</td>
<td>READ, EXCLUSIVE &quot;OR&quot; TO A</td>
</tr>
<tr>
<td>EDRUPT</td>
<td>1 000 11 1</td>
<td>ED SMALLY'S OWN RUPT ORDER</td>
</tr>
<tr>
<td>DV</td>
<td>1 001 00</td>
<td>DIVIDE</td>
</tr>
<tr>
<td>BZF</td>
<td>1 001 01</td>
<td>BRANCH ON ZERO TO FIXED</td>
</tr>
<tr>
<td>BZF</td>
<td>1 001 10</td>
<td></td>
</tr>
<tr>
<td>BZF</td>
<td>1 001 11</td>
<td></td>
</tr>
<tr>
<td>MSU</td>
<td>1 010 00</td>
<td>MODULAR SUBTRACT</td>
</tr>
<tr>
<td>OXCH</td>
<td>1 010 01</td>
<td>EXCHANGE WITH MEMORY</td>
</tr>
<tr>
<td>AUG</td>
<td>1 010 10</td>
<td>AUGMENT MEMORY</td>
</tr>
<tr>
<td>DIM</td>
<td>1 010 11</td>
<td>DIMINISH MEMORY</td>
</tr>
<tr>
<td>DCA</td>
<td>1 011</td>
<td>DOUBLE PRECISION CLEAR AND ADD</td>
</tr>
<tr>
<td>DCS</td>
<td>1 100</td>
<td>DOUBLE PRECISION CLEAR AND SUBTRACT</td>
</tr>
<tr>
<td>INDEX (NDX)</td>
<td>1 101</td>
<td>INDEX NEXT EXTRACODE INSTRUCTION</td>
</tr>
<tr>
<td>SU</td>
<td>1 110 00</td>
<td>SUBTRACT</td>
</tr>
<tr>
<td>BZMF</td>
<td>1 110 01</td>
<td>BRANCH ON ZERO OR MINUS TO FIXED</td>
</tr>
<tr>
<td>BZMF</td>
<td>1 110 10</td>
<td></td>
</tr>
<tr>
<td>BZMF</td>
<td>1 110 11</td>
<td></td>
</tr>
<tr>
<td>MP</td>
<td>1 111</td>
<td>MULTIPLY</td>
</tr>
</tbody>
</table>

* PSEUDO-CODES: RELINT = TC 0003, INHINT = TC 0004, EXTEND = TC 0006. THE TC OPERATION CODE IS SHARED BY THE NON-PROGRAMMABLE SEQUENCES GOJ1 (FOLLOWED BY TCO) AND TCSAJ3 (FOLLOWED BY STD2).
PULSE SEQUENCES

RCS
1. RSC WG NISO
2. RZ WQ
3. RU WZ
4. RAD WR WS

GOJ1
1. RSC WG
2. RSTP WS WB

TC5A.73
1. RSC WG
2. WS WZ ST2

CC50
1. RI 10BR WS
2. RSC WG
5. RG WB TSGN TIZ TPZG
7. 00 RZ WY12
7. 01 RZ WY12 PONFX
7. 10 RZ WY12 PTWOX
7. 11 RZ WY12 PONFX PTWOX
8. RU WZ WS
9. RB WG
10. 00 RB WY MONFX CI ST2
10. X1 WY ST2
10. 10 RC WY MONFX CI ST2
11. RI WA

TCF0
1. RB WY12 CI
2. RSC WG NISO
6. RU WZ
8. RAD WR WS
PULSE SEQUENCES

DASO

1. RL 10BB WS WY 12 MONEX CI
2. RSC WG
3. RA WB
4. RL WA
5. RU WL
6. RG WY A2X
7. RR WA
8. RL WB
9. RU WSC WG TOV
10. 00 RA WY ST1
10. 01 RA WY ST1 PONEX
10. 10 RA WY ST1 MONEX
10. 11 RA WY ST1

DAS1

1. RL 10BB WS
2. RSC WG
3. RU WA
5. RG WY A2X
6. RU WG WSC TOV
7. 00 WA
7. 01 WA RB1
7. 10 WA R1C
7. 11 WA
8. RZ W5 ST2
9. RC TMZ
10. X0 WL
11. X1 RU WA

LXCHO

1. RL10BB WS
2. RSC WG
3. RL WB
5. RG WL
7. RR WSC WG
8. RZ W5 ST2

INCRO

1. RL10BB WS
2. RSC WG
5. RG WY TSGN TMZ TPZG
6. PONEX
7. RU WSC WG WOVR
8. RZ W5 ST2

36
### PULSE SEQUENCES

**AD50**

<table>
<thead>
<tr>
<th>Step</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>RL10BB WS</td>
</tr>
<tr>
<td>2</td>
<td>RSC WG</td>
</tr>
<tr>
<td>5</td>
<td>RG WY A2X</td>
</tr>
<tr>
<td>6</td>
<td>RLI WSC WG TOV</td>
</tr>
<tr>
<td>7:00</td>
<td>WA</td>
</tr>
<tr>
<td>7:01</td>
<td>WA RH1</td>
</tr>
<tr>
<td>7:10</td>
<td>WA R1C</td>
</tr>
<tr>
<td>7:11</td>
<td>WA</td>
</tr>
<tr>
<td>8</td>
<td>RZ WS ST2</td>
</tr>
<tr>
<td>9</td>
<td>RC TH2</td>
</tr>
<tr>
<td>11</td>
<td>RLI WA</td>
</tr>
</tbody>
</table>

**CA0**

<table>
<thead>
<tr>
<th>Step</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>RSC WG</td>
</tr>
<tr>
<td>7</td>
<td>RG WB</td>
</tr>
<tr>
<td>8</td>
<td>RZ WS ST2</td>
</tr>
<tr>
<td>9</td>
<td>RB WG</td>
</tr>
<tr>
<td>10</td>
<td>RB WA</td>
</tr>
</tbody>
</table>

**C50**

<table>
<thead>
<tr>
<th>Step</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>RSC WG</td>
</tr>
<tr>
<td>7</td>
<td>RG WB</td>
</tr>
<tr>
<td>8</td>
<td>RZ WS ST2</td>
</tr>
<tr>
<td>9</td>
<td>RB WG</td>
</tr>
<tr>
<td>10</td>
<td>RC WA</td>
</tr>
</tbody>
</table>
PULSF SECUENCES

NDX0
2. RSC WG
5. TRSM
7. RG WB
8. RZ WS
9. RB WG
10. ST1

NDX1
1. RZ WY12 CI
2. RSC WG NISQ
3. RR WZ
4. RA WB
5. RZ WA
6. RU WZ
7. RG WY A2X
8. RU WS
9. RR WA
10. RU WB

RSM3
1. R15 WS
2. RSC WG NISQ
5. RG WZ
6. RR WG
8. RAD WR WS

DXCHO
1. RL10BB WS WY12 MOMEX CI
2. RSC WG
3. RL WB
5. RG WL
7. RB WSC WG
8. RU WS WA
10. ST1

DXCH1
1. RL10BR WS
2. RSC WG
3. RA WB
5. RG WA
7. RR WSC WG
8. RZ WS ST2
### PULSF Sequences

#### TS0

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>RL108R WS</td>
</tr>
<tr>
<td>2</td>
<td>R5C WG</td>
</tr>
<tr>
<td>3</td>
<td>RA WB T0V</td>
</tr>
<tr>
<td>4</td>
<td>RZ WY12</td>
</tr>
<tr>
<td>4</td>
<td>RZ WY12 CI</td>
</tr>
<tr>
<td>4</td>
<td>RZ WY12 CI</td>
</tr>
<tr>
<td>5</td>
<td>R7 WY12</td>
</tr>
<tr>
<td>5</td>
<td>RB1 WA</td>
</tr>
<tr>
<td>5</td>
<td>R1C WA</td>
</tr>
<tr>
<td>6</td>
<td>RU W2</td>
</tr>
<tr>
<td>7</td>
<td>RB WSC WG</td>
</tr>
<tr>
<td>8</td>
<td>RZ WS ST2</td>
</tr>
</tbody>
</table>

#### XCHO

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>RL108R WS</td>
</tr>
<tr>
<td>2</td>
<td>R5C WG</td>
</tr>
<tr>
<td>3</td>
<td>RA WB</td>
</tr>
<tr>
<td>5</td>
<td>RG WA</td>
</tr>
<tr>
<td>7</td>
<td>RR WSC WG</td>
</tr>
<tr>
<td>8</td>
<td>RZ WS ST2</td>
</tr>
</tbody>
</table>

#### ADO

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>R5C WG</td>
</tr>
<tr>
<td>7</td>
<td>RG WB</td>
</tr>
<tr>
<td>8</td>
<td>RZ WS ST2</td>
</tr>
<tr>
<td>9</td>
<td>BB WG</td>
</tr>
<tr>
<td>10</td>
<td>RR WY A2X</td>
</tr>
<tr>
<td>11</td>
<td>RU WA</td>
</tr>
</tbody>
</table>

#### MSKO

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>R5C WG</td>
</tr>
<tr>
<td>3</td>
<td>RA WB</td>
</tr>
<tr>
<td>4</td>
<td>RC WA</td>
</tr>
<tr>
<td>7</td>
<td>RG WB</td>
</tr>
<tr>
<td>8</td>
<td>RZ WS ST2</td>
</tr>
<tr>
<td>9</td>
<td>RC RA WY</td>
</tr>
<tr>
<td>10</td>
<td>RU WB</td>
</tr>
<tr>
<td>11</td>
<td>RC WA</td>
</tr>
</tbody>
</table>

---

39
PULSE SEQUENCES

READO

1. RL10BB WS
2. RA WB
3. WY
4. RCH WR
5. RB WA
6. RA WB
7. RZ WS ST2

WRITEO

1. RL10BB WS
2. RA WB WG
3. WY
4. RCH WR
5. RA WCH
6. RA WB
7. RZ WS ST2

RANDO

1. RL10BB WS
2. RA WB
3. RC WY
4. RCH WB
5. RC RU WA
6. RA WB
7. RC WA
8. RZ WS ST2

WANDO

1. RL10BB WS
2. RA WB
3. RC WY
4. RCH WB
5. RC RU WA
6. RA WB
7. RC WA
8. RZ WS ST2
PULSF SEQUENCES

RORO
1. R100R WS
2. RA WB
3. RR WY
4. RCH WR
5. RB RU WA
6. RA WB
7. RZ WS ST2

WORO
1. R100R WS
2. RA WB
3. RR WY
4. RCH WR
5. RB RU WA WCH
6. RA WB
7. RZ WS ST2

RXORO
1. R100R WS
2. RA WB
3. RC RCH WY
4. RCH WB
5. RA RC WG
6. RG WB
7. RZ WS ST2
8. RC WG
9. RU WB
10. RC RG WA

RUPTO
1. R15 WS
2. RSC WG
3. RZ WG
4. ST1

RUPT1
1. R15 RR2 WS
2. RSC WG
3. RRPA WZ
4. RZ WS ST2
5. RB WG KRPT
PULSE SEQUENCES

DV0
1. RA WB TSGN TIZ
2. 0X RA WA TMZ DVST
2. 1X DVST
3. RU WB STAGE

DV1
4. X0 RL WB
4. X1 RL WB TSGN
5. 0X EE WB B15X
5. 1X RC WY B15X Z16
6. RU WL TOV
7. RG RSC WB TSGN
8. X0 RA WY PONEK
8. X1 RA WY
9. 0X EE WA
9. 1X RC WA Z15
10. RU WB
11. RL WYD
12. RU WL
1. L2GD PB WYD A2X PIFL
2. 0X RG WL TSGU DVST CLXC
2. 1X RG WL TSGU DVST RR1F
3. RU WB STAGE

DV3
4. L2GD RB WYD A2X PIFL
5. 0X RG WL TSGU CLXC
5. 1X RG WL TSGU RB1F
6. RU WB
7. L2GD RB WYD A2X PIFL
8. 0X RG WL TSGU CLXC
8. 1X RG WL TSGU RB1F
9. RU WB
10. L2GD RB WYD A2X PIFL
11. 0X RG WL TSGU CLXC
11. 1X RG WL TSGU RB1F
12. RU WB
1. L2GD RB WYD A2X PIFL
2. 0X RG WL TSGU DVST CLXC
2. 1X RG WL TSGU DVST RR1F
3. RU WB STAGE
PULSE SEQUENCES

DV7

4. L2GD RR WYD A2X P1FL
5. 0X RG WL TSGU CLXC
6. 1X RG WL TSGU RB1F
7. RG WL
8. L2GD RB WYD A2X P1FL
9. RG WL TSGU CLXC
10. RG WL TSGU RB1F
11. RU WB
12. L2GD RB WYD A2X P1FL
13. RG WL TSGU CLXC
14. RG WL TSGU RB1F
15. RU WB
16. L2GD RB WYD A2X P1FL
17. RG WL TSGU CLXC
18. RG WL TSGU RB1F
19. RU WB
20. L2GD RB WYD A2X P1FL
21. RG WL TSGU CLXC
22. RG WL TSGU RB1F
23. RU WB
24. L2GD RB WYD A2X P1FL
25. RG WL TSGU CLXC
26. RG WL TSGU RB1F
27. RU WB
28. L2GD RB WYD A2X P1FL
29. RG WL TSGU CLXC
30. RG WL TSGU RB1F

DV6

4. L2GD RB WYD A2X P1FL
5. 0X RG WL TSGU CLXC
6. 1X RG WL TSGU RB1F
7. RG WL
8. L2GD RR WYD A2X P1FL
9. RG WL TSGU CLXC
10. RG WL
11. L2GD RB WYD A2X P1FL
12. RG WL TSGU CLXC
13. RG WL TSGU RB1F
14. RU WB
15. L2GD RB WYD A2X P1FL
16. RG WL TSGU CLXC
17. RG WL TSGU RB1F
18. RU WB
19. L2GD RB WYD A2X P1FL
20. RG WL TSGU CLXC
21. RG WL TSGU RB1F
22. RU WB
23. L2GD RB WYD A2X P1FL
24. RG WL TSGU CLXC
25. RG WL TSGU RB1F

DV4

3. RU WB STAGE
4. L2GD RB WYD A2X P1FL
5. 0X RG WB WA TSGU CLXC
6. 1X RG WB WA TSGU RB1F
7. RZ TOV
8. RZ WA
9. RZ WS ST2 T5GN R5T5TG
10. RZ WB WL
11. RC WA
12. RC WA
13. RC WL
14. RC WL

43
PULSE SEQUENCES

BZFO

1. RA WG TSGN TMZ
2. TPZG
3. RSC WG
5. X1 RR WY12 CI
6. X1 RL WZ
8. X0 RZ WS ST2
8. X1 RAD WB WS NISO

MSUO

1. RL10BB WS
2. RSC WG
5. RG WB
6. RC WY CI A2X
7. RUS WA TSGN
8. RZ WS ST2
9. RB WG
10. 1X RA WY MONEX
11. RUS WA

OXCHO

1. RL10BR WS
2. RSC WG
3. RO WB
5. RG WQ
7. RP WSC WG
8. RZ WS ST2

AUGO

1. RL10BB WS
2. RSC WG
5. RG WY TSGN TMZ TPZG
6. 0X PONEX
6. 1X MONFX
7. RU WSC WG WOVR
8. RZ WS ST2

DIMO

1. RL10BB WS
2. RSC WG
5. RG WY TSGN TMZ TPZG
6. 00 MONEX
6. 10 PONEX
7. RU WSC WG WOVR
8. RZ WS ST2
PULSF SEQUENCES

DCA0
1. RR WY12 MONEX C1
2. RSC AG
7. RG WB
8. RU WS
9. RR WG
10. RR WL ST1

DCA1
2. RSC AG
7. RG WB
8. RZ WS ST2
9. RR WG
10. RR WA

DC50
1. RR WY12 MONEX C1
2. RSC WG
7. RG WB
8. RU WS
9. RR WG
10. RC WL ST1

DC51
2. RSC WG
7. RG WB
8. RZ WS ST2
9. RR WG
10. RC WA
### PULSE SEQUENCES

**NDxx0**

| 2. | RSC WG |
| 7. | RG WB |
| 8. | RZ WS |
| 9. | RP WG |
| 10. | ST1 |

**NDxx1**

| 1. | RZ WY12 CI |
| 2. | RSC WG NISO |
| 3. | RB WZ |
| 4. | RA WB |
| 5. | RZ WA |
| 6. | RU WZ |
| 7. | RG WY A2X |
| 8. | RU WS |
| 9. | RR WA |
| 10. | RU WB EXT |

**SU0**

| 2. | RSC WG |
| 7. | RG WB |
| 8. | RZ WS ST2 |
| 9. | RP WG |
| 10. | WC WY A2X |
| 11. | RU WA |

**BZMFO**

| 1. | RA WG TSGN TMZ |
| 2. | TPZG |
| 3. | RSC WG |
| 5. | 01 RB WY12 CI |
| 5. | 10 RB WY12 CI |
| 5. | 11 RB WY12 CI |
| 6. | 01 RU WZ |
| 6. | 10 RU WZ |
| 6. | 11 RU WZ |
| 8. | 00 RZ WS ST2 |
| 8. | 01 RAD WP WS NISO |
| 8. | 10 RAD WR WS NISO |
| 8. | 11 RAD WR WS NISO |
**PULSE SEQUENCES**

**MP0**

<table>
<thead>
<tr>
<th>Step</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.</td>
<td>RSC AG</td>
</tr>
<tr>
<td>3.</td>
<td>RA WB TSGM</td>
</tr>
<tr>
<td>4.</td>
<td>0X</td>
</tr>
<tr>
<td>4.</td>
<td>1X</td>
</tr>
<tr>
<td>7.</td>
<td>RG WB TSGM2</td>
</tr>
<tr>
<td>8.</td>
<td>RZ WS</td>
</tr>
<tr>
<td>9.</td>
<td>00</td>
</tr>
<tr>
<td>9.</td>
<td>01</td>
</tr>
<tr>
<td>9.</td>
<td>10</td>
</tr>
<tr>
<td>9.</td>
<td>11</td>
</tr>
<tr>
<td>10.</td>
<td>RU WB TSGM ST1 NEACON</td>
</tr>
<tr>
<td>11.</td>
<td>0X</td>
</tr>
<tr>
<td>11.</td>
<td>1X</td>
</tr>
</tbody>
</table>

**MP1**

<table>
<thead>
<tr>
<th>Step</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>ZIP</td>
</tr>
<tr>
<td>2.</td>
<td>ZAP</td>
</tr>
<tr>
<td>3.</td>
<td>ZIP</td>
</tr>
<tr>
<td>4.</td>
<td>ZAP</td>
</tr>
<tr>
<td>5.</td>
<td>ZIP</td>
</tr>
<tr>
<td>6.</td>
<td>ZAP</td>
</tr>
<tr>
<td>7.</td>
<td>ZIP</td>
</tr>
<tr>
<td>8.</td>
<td>ZAP</td>
</tr>
<tr>
<td>9.</td>
<td>ZIP</td>
</tr>
<tr>
<td>10.</td>
<td>ZAP ST1 ST2</td>
</tr>
<tr>
<td>11.</td>
<td>ZIP</td>
</tr>
</tbody>
</table>

**MP3**

<table>
<thead>
<tr>
<th>Step</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>ZAP</td>
</tr>
<tr>
<td>2.</td>
<td>ZIP NISQ</td>
</tr>
<tr>
<td>3.</td>
<td>ZAP</td>
</tr>
<tr>
<td>4.</td>
<td>RSC WG</td>
</tr>
<tr>
<td>5.</td>
<td>RZ WY12 CI</td>
</tr>
<tr>
<td>6.</td>
<td>RU WZ TL15 NFACOF</td>
</tr>
<tr>
<td>7.</td>
<td>1X</td>
</tr>
<tr>
<td>8.</td>
<td>RAD WB WS</td>
</tr>
<tr>
<td>9.</td>
<td>RA</td>
</tr>
<tr>
<td>10.</td>
<td>RL</td>
</tr>
<tr>
<td>11.</td>
<td>1X</td>
</tr>
</tbody>
</table>

**STD2**

<table>
<thead>
<tr>
<th>Step</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>RZ WY12 CI</td>
</tr>
<tr>
<td>2.</td>
<td>RSC WG NISQ</td>
</tr>
<tr>
<td>6.</td>
<td>RU WZ</td>
</tr>
<tr>
<td>8.</td>
<td>RAD WB WS</td>
</tr>
</tbody>
</table>
### PULSF SEQUENCES

#### PINC

<table>
<thead>
<tr>
<th>Step</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>RSCT WS</td>
</tr>
<tr>
<td>2</td>
<td>RSC WG</td>
</tr>
<tr>
<td>5</td>
<td>RG WY TSGN TMZ TPZG</td>
</tr>
<tr>
<td>6</td>
<td>MONEX</td>
</tr>
<tr>
<td>7</td>
<td>RU WSC WG WOVR</td>
</tr>
<tr>
<td>8</td>
<td>RB WS</td>
</tr>
</tbody>
</table>

#### PCDU

<table>
<thead>
<tr>
<th>Step</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>RSCT WS</td>
</tr>
<tr>
<td>2</td>
<td>RSC WG</td>
</tr>
<tr>
<td>5</td>
<td>RG WY TSGN TMZ TPZG</td>
</tr>
<tr>
<td>6</td>
<td>CI</td>
</tr>
<tr>
<td>7</td>
<td>RU WSC WG WOVR</td>
</tr>
<tr>
<td>8</td>
<td>RB WS</td>
</tr>
</tbody>
</table>

#### MINC

<table>
<thead>
<tr>
<th>Step</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>RSCT WS</td>
</tr>
<tr>
<td>2</td>
<td>RSC WG</td>
</tr>
<tr>
<td>5</td>
<td>RG WY TSGN TMZ TPZG</td>
</tr>
<tr>
<td>6</td>
<td>MONEX</td>
</tr>
<tr>
<td>7</td>
<td>RU WSC WG WOVR</td>
</tr>
<tr>
<td>8</td>
<td>RB WS</td>
</tr>
</tbody>
</table>

#### MCDU

<table>
<thead>
<tr>
<th>Step</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>RSCT WS</td>
</tr>
<tr>
<td>2</td>
<td>RSC WG</td>
</tr>
<tr>
<td>5</td>
<td>RG WY TSGN TMZ TPZG</td>
</tr>
<tr>
<td>6</td>
<td>MONEX CI</td>
</tr>
<tr>
<td>7</td>
<td>RU WSC WG WOVR</td>
</tr>
<tr>
<td>8</td>
<td>RB WS</td>
</tr>
</tbody>
</table>

#### DINC

<table>
<thead>
<tr>
<th>Step</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>RSCT WS</td>
</tr>
<tr>
<td>2</td>
<td>RSC WG</td>
</tr>
<tr>
<td>5</td>
<td>RG WY TSGN TMZ TPZG</td>
</tr>
<tr>
<td>6*</td>
<td>MONEX POUT</td>
</tr>
<tr>
<td>6*</td>
<td>PONFX MOUT</td>
</tr>
<tr>
<td>6*</td>
<td>X1 ZOUT</td>
</tr>
<tr>
<td>7</td>
<td>RU WSC WG WOVR</td>
</tr>
<tr>
<td>8</td>
<td>RP WS</td>
</tr>
</tbody>
</table>
## PULSE SEQUENCES

### SHINC

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>RSCT WS</td>
</tr>
<tr>
<td>2</td>
<td>RSC WG</td>
</tr>
<tr>
<td>5</td>
<td>RG WYD TSGN</td>
</tr>
<tr>
<td>7</td>
<td>RUS WSC WG WOVR</td>
</tr>
<tr>
<td>8</td>
<td>RB WS</td>
</tr>
</tbody>
</table>

### SHANC

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>RSCT WS</td>
</tr>
<tr>
<td>2</td>
<td>RSC WG</td>
</tr>
<tr>
<td>5</td>
<td>RG WYD TSGN CI</td>
</tr>
<tr>
<td>7</td>
<td>RUS WSC WG WOVR</td>
</tr>
<tr>
<td>8</td>
<td>RB WS</td>
</tr>
</tbody>
</table>

### INOTRD

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>WS</td>
</tr>
<tr>
<td>2</td>
<td>RSC WG</td>
</tr>
<tr>
<td>5</td>
<td>RCH</td>
</tr>
<tr>
<td>8</td>
<td>RR WS</td>
</tr>
</tbody>
</table>

### INOTLD

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>WS</td>
</tr>
<tr>
<td>2</td>
<td>RSC WG</td>
</tr>
<tr>
<td>5</td>
<td>RCH</td>
</tr>
<tr>
<td>7</td>
<td>WCH</td>
</tr>
<tr>
<td>8</td>
<td>RR WS</td>
</tr>
</tbody>
</table>
PULSE SEQUENCES

FETCH0
1. RA WS
2. RSC AG WY ST1
4. WSC
8. WS

FETCH1
2. RSC AG
7. RG
8. RR WS U2BBK
10. RBBK

STORE0
1. RA WS
2. RSC AG WY ST1
4. WSC
8. WS

STORE1
2. RSC AG
4. WSC
7. RG
8. RR WS U2BBK
9. WG
10. RBBK
## AGC BLOCK II MEMORY ORGANIZATION AND ADDRESSING

<table>
<thead>
<tr>
<th>Addressing</th>
<th>ERASABLE MEMORY 2000 (OCTAL) WORDS PER BOX</th>
<th>FIXED MEMORY 10000 (OCTAL) WORDS PER BOX</th>
<th>FIXED MEMORY 10000 (OCTAL) WORDS PER BOX</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSEUDO</td>
<td>00 000</td>
<td>00 000</td>
<td>00 000</td>
</tr>
<tr>
<td>ECADR</td>
<td>0 000</td>
<td>0 000</td>
<td>0 000</td>
</tr>
<tr>
<td>EB SREG</td>
<td>0 000</td>
<td>0 000</td>
<td>0 000</td>
</tr>
<tr>
<td>EB SREG*</td>
<td>0 1400</td>
<td>0 1400</td>
<td>0 1400</td>
</tr>
<tr>
<td>PSEUDO</td>
<td>00 400</td>
<td>00 400</td>
<td>00 400</td>
</tr>
<tr>
<td>ECADR</td>
<td>0 400</td>
<td>0 400</td>
<td>0 400</td>
</tr>
<tr>
<td>EB SREG</td>
<td>0 400</td>
<td>0 400</td>
<td>0 400</td>
</tr>
<tr>
<td>EB SREG*</td>
<td>1 1400</td>
<td>1 1400</td>
<td>1 1400</td>
</tr>
<tr>
<td>PSEUDO</td>
<td>01 000</td>
<td>01 000</td>
<td>01 000</td>
</tr>
<tr>
<td>ECADR</td>
<td>1 000</td>
<td>1 000</td>
<td>1 000</td>
</tr>
<tr>
<td>EB SREG</td>
<td>1 000</td>
<td>1 000</td>
<td>1 000</td>
</tr>
<tr>
<td>EB SREG*</td>
<td>2 1400</td>
<td>2 1400</td>
<td>2 1400</td>
</tr>
<tr>
<td>PSEUDO</td>
<td>02 000</td>
<td>02 000</td>
<td>02 000</td>
</tr>
<tr>
<td>ECADR</td>
<td>2 000</td>
<td>2 000</td>
<td>2 000</td>
</tr>
<tr>
<td>EB SREG</td>
<td>4 1400</td>
<td>4 1400</td>
<td>4 1400</td>
</tr>
<tr>
<td>EB SREG*</td>
<td>5 1400</td>
<td>5 1400</td>
<td>5 1400</td>
</tr>
<tr>
<td>PSEUDO</td>
<td>03 000</td>
<td>03 000</td>
<td>03 000</td>
</tr>
<tr>
<td>ECADR</td>
<td>3 000</td>
<td>3 000</td>
<td>3 000</td>
</tr>
<tr>
<td>EB SREG</td>
<td>6 1400</td>
<td>6 1400</td>
<td>6 1400</td>
</tr>
<tr>
<td>EB SREG*</td>
<td>7 1400</td>
<td>7 1400</td>
<td>7 1400</td>
</tr>
</tbody>
</table>

**NOT PREFERRED**

**OCTAL WORD COUNT ADDRESSABLE WITHOUT CHANGING ANY BANK BITS.**