TO: AGC4 Distribution
FROM: Hugh Blair-Smith
DATE: September 30, 1965, Revised June 1, 1967
SUBJECT: AGC4 MEMO # 9 - Block II Instructions
# Table of Contents

- Introduction .............................................. 2
- Memory .................................................... 3
- Basic Instructions ....................................... 3
- Extracode Instructions .................................. 11
- Implied-Address Codes .................................. 15
- Unprogrammed Sequences ................................ 21
- Address Constant Formats .............................. 25
- Control Pulse Definitions .............................. 27
- Condensed List of Programmable Instructions .... 34
- Pulse Sequences ......................................... 35
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 9-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.
## ARRANGEMENT OF ADDRESSES

<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 registers</td>
</tr>
<tr>
<td>00004</td>
<td>FB</td>
<td>Fixed Bank Register</td>
<td></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></td>
</tr>
<tr>
<td>00021</td>
<td>SR</td>
<td>Shift Right 1 Bit</td>
<td></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></td>
<td></td>
</tr>
<tr>
<td>01400-03777</td>
<td>5 Erasable Banks @ 256 words (See Fig. 2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>04000-up</td>
<td>Fixed (See Fig. 2)</td>
<td></td>
<td>Fixed</td>
</tr>
</tbody>
</table>

Fig. 1
### Fixed and Erasable Bank-Switching
*(Fig. 2)*

<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 Bits (7, 6, 5 of 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 F</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></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>
</tr>
<tr>
<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>
</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.

* Any value 0-3 is OK. 3 is conventional.
## 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>0000 EEE0000000000</td>
</tr>
<tr>
<td>0004</td>
<td>FB</td>
<td>FFF000000000000</td>
</tr>
<tr>
<td></td>
<td>(Actual Circuits)</td>
<td>SSS FFF EEE</td>
</tr>
<tr>
<td>0006</td>
<td>BB</td>
<td>FFF0000000EEE</td>
</tr>
<tr>
<td>Chan. 07</td>
<td>FEB</td>
<td>0000000000SSS0000</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

<table>
<thead>
<tr>
<th>(bit positions)</th>
<th>15 14 13 12 11 10 09 08 07 06 05 04 03 02 01</th>
</tr>
</thead>
<tbody>
<tr>
<td>0020</td>
<td>CYR</td>
</tr>
<tr>
<td>0021</td>
<td>SR</td>
</tr>
<tr>
<td>0022</td>
<td>CYL</td>
</tr>
<tr>
<td>0023</td>
<td>EDOP</td>
</tr>
</tbody>
</table>

---

Fig. 3
### Fig. 4

<table>
<thead>
<tr>
<th>Operation Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RELINT(3)</td>
<td></td>
</tr>
<tr>
<td>INHINT(4)</td>
<td></td>
</tr>
<tr>
<td>EXTEND(6)</td>
<td></td>
</tr>
<tr>
<td>TC1 [TCR]</td>
<td></td>
</tr>
<tr>
<td>TCF1</td>
<td></td>
</tr>
<tr>
<td>DXCH2</td>
<td></td>
</tr>
<tr>
<td>CA2</td>
<td></td>
</tr>
<tr>
<td>CS2</td>
<td></td>
</tr>
<tr>
<td>INDEX2</td>
<td></td>
</tr>
<tr>
<td>NDX</td>
<td></td>
</tr>
<tr>
<td>SYSTEM</td>
<td></td>
</tr>
<tr>
<td>READD</td>
<td></td>
</tr>
<tr>
<td>WRITE2</td>
<td></td>
</tr>
<tr>
<td>RAND2</td>
<td></td>
</tr>
<tr>
<td>WAND2</td>
<td></td>
</tr>
<tr>
<td>ROR2</td>
<td></td>
</tr>
<tr>
<td>WOR2</td>
<td></td>
</tr>
<tr>
<td>RXOR2</td>
<td></td>
</tr>
<tr>
<td>DIM2</td>
<td></td>
</tr>
<tr>
<td>AD3</td>
<td></td>
</tr>
<tr>
<td>MSK2 [MSK]</td>
<td></td>
</tr>
<tr>
<td>SF</td>
<td></td>
</tr>
</tbody>
</table>

**Operation Codes (10-17 are extracodes)**
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), re-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), re-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 accumulator — implied address code DDOUBL 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), re-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, re-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), re-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), re-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), re-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), re-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(0015)
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), re-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), re-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), re-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), re-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 specifically omits to re-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) \vee 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 \( c(A) = b(A) \oplus c(KC) \);
Take next instruction from I + 1.
Remarks: \( \oplus \) 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 ( \oplus ) 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.

EDRUPT 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 re-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 \(C(A) = +0\), 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 \(c(A) = b(A) \oplus b(K)\);
Set \(c(K) = b(K)\), re-editing if \(K\) is 0020-0023;
Take next instruction from \(I + 1\).
Remarks: \(\oplus\) 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 \(c(K) = b(Q)\), re-editing if \(K\) is 0020-0023;
Take next instruction from \(I + 1\).

Code 12.
I: AUG K Augment 2 MCT
QC2
If \(b(K) \geq +0\), set \(c(K) = b(K) + 1\), re-editing if \(K\) is 0020-0023;
If \(b(K) \leq -0\), set \(c(K) = b(K) - 1\), re-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 \(c(K) = b(K) - 1\), re-editing if \(K\) is 0020-0023;
If \(b(K) = +0\), set \(c(K) = b(K)\), re-editing if \(K\) is 0020-0023;
If \(b(K) < -0\), set \(c(K) = b(K) + 1\), re-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 \(c(K) = b(K)\), re-editing if \(K\) is 0020-0023;
Set \(c(K+1) = b(K+1)\), re-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).
Set \( c(A, L) = -b(K, K+1) \);
Set \( c(K) = b(K) \), re-editing if \( K \) is 0020-0023;
Set \( c(K+1) - b(K+1) \), re-editing if \( K+1 \) is 0020-0023;
Take next instruction from \( I + 1 \).
Remarks: DCS 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:

\[
\text{EXTEND}
\text{INDEX} \quad \text{ADDRWD}
\text{MP} \quad 0
\]

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

Code 16.  QC 1-3  I: BZMF K  Branch Zero or Minus to Fixed  1 or 2 MCT
If \( c(A) \leq +0 \), 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 \( c(A, L) = b(A) \times c(K) \);
Take next instruction from \( I + 1 \).
Remarks: The two words of the product agree in sign. A zero result is positive unless \( b(A) = \pm 0 \) and \( c(K) \) is non-zero with the opposite sign.

MP does not re-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 operation, 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;

\[ \text{Leave } c(Q) = 000003. \]

**Code 00.**

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

1 MCT

\( K = 0003 \)

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

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

**Code 00.**

**I: INHINT  Inhibit Interrupt**

1 MCT

\( K = 0004 \)

Inhibit interrupt until a subsequent RELINT;

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: EXTEND  Extend Next Instruction**

1 MCT

\( K = 0006 \)

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

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

**Code 01.**

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

1 MCT

\( QC 1 - 3 \)

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

\( K = I + 1 \)

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

**Code 02.**

**I: DDOUBL  Double Precision Double**

3 MCT

\( QC 0 \)

Set \( c(A, L) = b(A, L) + b(A, L) \);

\( K = 0000 \)

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

Remarks: If \( b(A) \) contains \(+\) overflow, the results are messy; in particular, \( \text{sgn } [c(A)] \neq \text{sgn } [b(A)] \). If \( |b(A)| \geq 1/2 \), overflow will be retained in \( c(A) \).
### IMPLIED ADDRESS CODES

<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>1 (I+1)</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>NOOP</td>
<td>CA</td>
<td>A</td>
<td>30000</td>
<td>E</td>
</tr>
<tr>
<td>OVSU</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>XXALLQ</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></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 (See Fig. 5)</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</td>
<td>S</td>
</tr>
<tr>
<td>TC</td>
<td></td>
<td>00004</td>
<td>INHINT</td>
<td>S</td>
</tr>
<tr>
<td>TC</td>
<td></td>
<td>00006</td>
<td>EXTEND</td>
<td>S</td>
</tr>
<tr>
<td>TCF</td>
<td></td>
<td>1 (I+1)</td>
<td>NOOP</td>
<td>F</td>
</tr>
<tr>
<td>DAS</td>
<td>A</td>
<td>20001</td>
<td>DDOUBL</td>
<td></td>
</tr>
<tr>
<td>LXCH</td>
<td></td>
<td>22007</td>
<td>ZL</td>
<td></td>
</tr>
<tr>
<td>CA</td>
<td>A</td>
<td>30000</td>
<td>NOOP</td>
<td>E</td>
</tr>
<tr>
<td>CS</td>
<td>A</td>
<td>40000</td>
<td>COM</td>
<td></td>
</tr>
<tr>
<td>INDEX</td>
<td>BRUPT</td>
<td>50017</td>
<td>RESUME</td>
<td>R</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>OVSK</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</td>
<td>X</td>
</tr>
<tr>
<td>DCS</td>
<td>A</td>
<td>40001</td>
<td>DCOM</td>
<td>X</td>
</tr>
<tr>
<td>MP</td>
<td>A</td>
<td>70000</td>
<td>SQUARE</td>
<td>X</td>
</tr>
</tbody>
</table>

Fig. 6
Code 02. I: ZL Zero L 2 MCT
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) 2 MCT
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) 2 MCT
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 2 MCT
QC 0 Set c(Z) = c(0015);
K = 0017 Use c(0017) as the next instruction.

Code 05. I: DTCF Double Transfer Control, 3 MCT
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 new c(Z)
Remarks: A double-precision address constant
format, 2 FCADR, 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 new c(Z)
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 Address in A  2 MCT
QC 2  
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.
Code 14.

I: DCOM  
Double Complement  3 MCT

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) \times 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, but never inside instructions.
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(CDUCTR) + 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. INCR 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(CDUCTR) = c(CDUCTR) - 1 in twos
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 \( b(BB); \)

2 MCT
STORE Store in Memory from GSE 2 MCT

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 2 MCT

Set c(Q) = b(Z);
Take next instruction from location 4000
and proceed from there.

TCSAJ Transfer Control to Specified Address Jam 2 MCT

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.

**ECADR**

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", and is required before every BBCON and 2BCADR or 2CADR.

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

** Interpretive address EBANK restriction is for BLK2 programs but not for AGC programs.
SBANK= Super Bank Declaration

Similar to EBANK=, but not required before BBCONs and 2CADRs. The address must be in a superbank.

1DNADR 1-Word Downlist Address

Same as ECADR, but used when the word addressed is the left half of a double-precision word for down telemetry.

2DNADR - 6DNADR N-Word Downlist Address, 2 ≤ N ≤ 6

Same as 1DNADR, but with the 4 unused bits of the ECADR format filled in with 0001-0101. Used to point to a list of N double-precision words, stored consecutively, for down telemetry.

DNCHAN Downlist Channel Address

Same as 1DNADR, but with prefix bits 0111. Used to point to a pair of channels for down telemetry.

DNPTR Down Telemetry Sublist Pointer

Same as CAF, but tagged as a constant. Used in a downlist to point to a sublist.
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
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 LINF.

B15X  SET BIT 15 OF X TO 1.

CI  INSERT CARRY INTO BIT 1 OF THE ADDFR.

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 HULF. ALSO PERMIT
      STAGING TO OCCUR AT TIMES OF DIVIDE CYCLES.

EXT  SET THE EXTEND FLIP FLOP.

G2LS  COPY G4-15,16,l INTO L1-12,16,15.

KRPI  RESFT INTERRUPT PRIORITY CELL.

L16  SET BIT 16 OF L TO 1.

L2GD  COPY L1-14,16 INTO G7-15,16 -- ALSO MCU INTO G1.

MONEX  SET BITS 2-16 OF X TO ONES.

MOUT  NEGATIVE R{$\bar{A}$}T $\bar{F}$ OUTPUT PULSE.

NEACOF  PERMIT END AROUND CARRY AFTER END OF MP3.

NEACON  INHIBIT END AROUND CARRY UNTIL NEACOF.

NISQ  NEXT INSTRUCTION IS TO BE LOADED INTO S0. ALSO
      FREES CERTAIN RESTRICTIONS- PERMITS INCRFMENTS AND
      INTERRUPTS.

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

PONEX  SET BIT 1 OF X TO 1.

POUT  POSITIVE R{$\bar{A}$}T $\bar{F}$ OUTPUT PULSE.

PTWOX  SET BIT 2 OF X TO 1.

R15  PLACE OCTAL 000015 ON WL'S.
CONTROL PULSF DEFINITIONS

RIC
PLACE OCTAL 177776 = -1 ON WL'S.

R6
PLACE OCTAL 000006 ON WRITE LINES.

RA
READ A1-16 TO WL1-16.

RAD
READ ADDRESS IF NEXT CYCLE. THIS APPEARS AT THE END OF AN INSTRUCTION AND NORMALLY IS INTERPRETED AS RG. IF THE NEXT INSTRUCTION IS TO BE A PSFUDO 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 T5GU. RB1F IF RB1=1.

RB2
PLACE OCTAL 000002 ON THE WL'S.

RBBK
READ THE BH (BOTH RANK) CONFIGURATION ONTO THE WRITE LINES. I.E.: FB 9-11 TO WL 1-3 AND FB 11-14, 16, 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 S;
CHANNEL BITS 1-14 TO WL1-14, AND BIT 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 L14 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 S;
CENTRAL STORE BITS 1-16 ARE COPIED TO WL1-16.

RSCT
READ THE ADDRESS OF HIGHEST PRIORITY COUNTER REQUEST.

RSTRT
PLACE OCTAL 004000 = BLOCK 2 START ADDRESS ON WL'S.

RSTSTG
RESFT THE DIVIDE TO 03 STAGING CONDITION.

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

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RUS</td>
<td>READ U1-14 TO WL1-14 AND U15 TO AL15 AND 16.</td>
</tr>
<tr>
<td>RZ</td>
<td>READ Z1-16 TO WL1-16.</td>
</tr>
<tr>
<td>ST1</td>
<td>SET STAGF1 FLIP FLOP NEXT T12.</td>
</tr>
<tr>
<td>ST2</td>
<td>SET STAGF2 FLIP FLOP NEXT T12.</td>
</tr>
<tr>
<td>STAGF</td>
<td>EXECUTE GRAY-CODED STAGE ADVANCE COMPUTED BY DVST.</td>
</tr>
<tr>
<td>TL15</td>
<td>COPY L15 INTO HK1.</td>
</tr>
<tr>
<td>TM2</td>
<td>TEST WL1-16 FOR ALL ONES (0). SET BR2 IF TRUE.</td>
</tr>
<tr>
<td>TOV</td>
<td>TEST FOR + OR - OVERFLOW. SET BR1,2 TO 00 IF NO OVERFLOW, 01 IF + OVERFLOW, 10 IF - OVERFLOW.</td>
</tr>
<tr>
<td>TP2G</td>
<td>TEST CONTENT OF G FOR PLUS ZERO. IF TRUE SET BR2=1.</td>
</tr>
<tr>
<td>TRSM</td>
<td>TEST FOR RFSIME ADDRESS ON INDEX. ST2 IF (S)=0017.</td>
</tr>
<tr>
<td>TSGN</td>
<td>TEST SIGN. COPY WL16 TO BR1.</td>
</tr>
<tr>
<td>TSGN2</td>
<td>COPY WL16 TO BR2.</td>
</tr>
<tr>
<td>TSGU</td>
<td>TEST SIGN OF SUM (U). COPY U16 INTO BR1.</td>
</tr>
<tr>
<td>U2BBK</td>
<td>ADDER BITS 1-3 AND 11-14+16 ARE TRANSFERRED INTO ERASABLE AND FIXED BANKS. THIS PULSF MAY BE INHIBITED BY CTS SIGNAL MOTWKB.</td>
</tr>
<tr>
<td>WA</td>
<td>CLFAR AND WRITE WL1-16 INTO A1-16.</td>
</tr>
<tr>
<td>WALS</td>
<td>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).</td>
</tr>
<tr>
<td>WB</td>
<td>CLEAR AND WRITE WL1-16 INTO R1-16.</td>
</tr>
<tr>
<td>WCH</td>
<td>CLFAR AND WRITE WL1-16 INTO R1-16+16+PARITY INTO CHANNEL BITS 1-14+16+PARITY. CHANNELS 1 AND 2 WRITE AS WL AND WO. THE CHANNEL TO BE LOADED IS SPECIFIED BY THE CURRENT CONTENT OF S.</td>
</tr>
<tr>
<td>WG</td>
<td>CLFAR AND WRITE WL1-16 INTO G1-16 EXCEPT FOR ADDRESSES OCTAL 20-23, WHICH CAUSE EDITING.</td>
</tr>
<tr>
<td>WL</td>
<td>CLEAR AND WRITE WL1-16 INTO L1-16.</td>
</tr>
</tbody>
</table>
CONTROL PULSF DEFINITIONS

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

WQ CLRAR AND WRITE WL1-16 INTO Q1-16.

WS CLRAR 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 COP 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 Y11 EXCEPT:
(1) WHEN END-AROUND CARRY IS INHIBITED BY DEACON;
(2) DURING SHINC 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 IMPLEMENTS RU, G2LS, AND WALS.

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

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

ZOUT NO RATE OUTPUT PULSE. RESET OUTBIT REQUESTING DINC.

* THESE PULSES DO NOT APPEAR IN THE PULSE SEQUENCES.
<table>
<thead>
<tr>
<th>OP CODEF</th>
<th>EXT S016:14-10</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>&quot; &quot; &quot; &quot;</td>
</tr>
<tr>
<td>TCF</td>
<td>0 001 11</td>
<td>&quot; &quot; &quot; &quot;</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>L 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>WRITF 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 RUIT OPDFR</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>&quot; &quot; &quot; &quot;</td>
</tr>
<tr>
<td>BZF</td>
<td>1 001 11</td>
<td>&quot; &quot; &quot; &quot;</td>
</tr>
<tr>
<td>MSU</td>
<td>0 010 00</td>
<td>MODULAR SUBTRACT</td>
</tr>
<tr>
<td>QXCH</td>
<td>0 010 01</td>
<td>0 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>&quot; &quot; &quot; &quot;</td>
</tr>
<tr>
<td>BZMF</td>
<td>1 110 11</td>
<td>&quot; &quot; &quot; &quot;</td>
</tr>
<tr>
<td>MP</td>
<td>1 111</td>
<td>MULTIPLY</td>
</tr>
</tbody>
</table>

* PSFUDO-CODES: RELINT = TC 0003, INHINT = TC 0004, EXTEND = TC 0006.  
THE TC OPERATION CODE IS SHARED BY THE NON-PROGRAMMABLE SEQUENCES  
GOJ1 (FOLLOWED BY TC0) AND TCSAJ3 (FOLLOWED BY STD2).
PULSE SEQUENCES

TC0
1. RR WY12 CI
2. RSC WG NISQ
3. RZ WQ
6. RU WZ
8. RAD WB WS

GOJ1
2. RSC WG
8. RSTRT WS WB

TC5A13
2. RSC WG
8. WS WZ ST2

CC50
1. RI 10BR WS
2. RSC WG
5. RG WB T5GM T5M T5G
7. 00 RZ WY12
7. 01 RZ WY12 PONFX
7. 10 RZ WY12 PTW0X
7. 11 RZ WY12 PONFX PTW0X
8. RU WZ WS
9. RR WG
10. 00 RR WY MONFX CI ST?
10. X1 WY ST2
10. 10 RC WY MONFX CI ST?
11. RIJ WA

TCFO
1. RR WY12 CI
2. RSC WG NISQ
6. RU WZ
8. RAD WB WS
PULSE SEQUENCES

DAS0
1. RL10BB WS WY12 MONEX CI
2. RSC WG
3. RA WB
4. RI 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. RL10BB 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 P1C
7. 11 WA
8. RZ WS 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 WS ST2

INCRO
1. RL10BB WS
2. RSC WG
5. RG WY TSZN TMZ TPZG
6. PONEX
7. RU WSC WG WOVR
8. RZ WS ST2
**PULSAR SEQUENCES**

**AD50**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>RL10BR WS</td>
</tr>
<tr>
<td>2</td>
<td>R5C WG</td>
</tr>
<tr>
<td>5</td>
<td>RG WY A2X</td>
</tr>
<tr>
<td>6</td>
<td>RU WSC WG TOV</td>
</tr>
<tr>
<td>7</td>
<td>WA</td>
</tr>
<tr>
<td>7</td>
<td>WA P81</td>
</tr>
<tr>
<td>7</td>
<td>WA R1C</td>
</tr>
<tr>
<td>7</td>
<td>WA</td>
</tr>
<tr>
<td>8</td>
<td>RZ WS ST2</td>
</tr>
<tr>
<td>9</td>
<td>RC TM7</td>
</tr>
<tr>
<td>11</td>
<td>RU WA</td>
</tr>
</tbody>
</table>

**CAO**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></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>RB WG</td>
</tr>
<tr>
<td>10</td>
<td>RB WA</td>
</tr>
</tbody>
</table>

**CSO**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></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>RB WG</td>
</tr>
<tr>
<td>10</td>
<td>RC WA</td>
</tr>
</tbody>
</table>
PULSF SEQUENCES

**NDX0**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>RSC WG</td>
</tr>
<tr>
<td>2</td>
<td>TRSM</td>
</tr>
<tr>
<td>3</td>
<td>RG WB</td>
</tr>
<tr>
<td>4</td>
<td>RZ WS</td>
</tr>
<tr>
<td>5</td>
<td>RB WG</td>
</tr>
<tr>
<td>6</td>
<td>ST1</td>
</tr>
</tbody>
</table>

**NDX1**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>RZ WY12 CI</td>
</tr>
<tr>
<td>2</td>
<td>RSC WG NISO</td>
</tr>
<tr>
<td>3</td>
<td>RR WZ</td>
</tr>
<tr>
<td>4</td>
<td>RA WB</td>
</tr>
<tr>
<td>5</td>
<td>RZ WA</td>
</tr>
<tr>
<td>6</td>
<td>RU WZ</td>
</tr>
<tr>
<td>7</td>
<td>RG WY A2X</td>
</tr>
<tr>
<td>8</td>
<td>RU WS</td>
</tr>
<tr>
<td>9</td>
<td>RR WA</td>
</tr>
<tr>
<td>10</td>
<td>RU WB</td>
</tr>
</tbody>
</table>

**RSM3**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>R15 WS</td>
</tr>
<tr>
<td>2</td>
<td>RSC WG NISO</td>
</tr>
<tr>
<td>3</td>
<td>RG WZ</td>
</tr>
<tr>
<td>4</td>
<td>RR WG</td>
</tr>
<tr>
<td>5</td>
<td>RAD WP WS</td>
</tr>
</tbody>
</table>

**DXCHO**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>RL10BB WS WY12 MOREX CI</td>
</tr>
<tr>
<td>2</td>
<td>RSC WG</td>
</tr>
<tr>
<td>3</td>
<td>RL WB</td>
</tr>
<tr>
<td>4</td>
<td>RG WL</td>
</tr>
<tr>
<td>5</td>
<td>RB WSC WG</td>
</tr>
<tr>
<td>6</td>
<td>RU WS WS</td>
</tr>
<tr>
<td>7</td>
<td>ST1</td>
</tr>
</tbody>
</table>

**DXCH1**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>RL10BR WS</td>
</tr>
<tr>
<td>2</td>
<td>RSC WG</td>
</tr>
<tr>
<td>3</td>
<td>RA WB</td>
</tr>
<tr>
<td>4</td>
<td>RG WA</td>
</tr>
<tr>
<td>5</td>
<td>RB WSC WG</td>
</tr>
<tr>
<td>6</td>
<td>RZ WS ST2</td>
</tr>
</tbody>
</table>
# PULSF SEQUENCES

## TSO

| 1  | RL10BB WS          |
| 2  | RSC WG             |
| 3  | RA WB TOV          |
| 4  | 00 RZ WY12         |
| 4  | 01 RZ WY12 CI      |
| 4  | 10 RZ WY12 CI      |
| 4  | 11 RZ WY12         |
| 5  | 01 RR1 WA          |
| 5  | 10 R1C WA          |
| 6  | RU WZ              |
| 7  | RB WSC WG          |
| 8  | RZ WS ST2          |

## XCHO

| 1  | RL10BB WS          |
| 2  | RSC WG             |
| 3  | RA WB              |
| 5  | RG WA              |
| 7  | RB WSC WG          |
| 8  | RZ WS ST2          |

## ADO

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

## MSKO

| 2  | RSC WG             |
| 3  | RA WB              |
| 4  | RC WA              |
| 7  | RG WB              |
| 8  | RZ WS ST2          |
| 9  | RC RA WY           |
| 10 | RU WB              |
| 11 | RC WA              |
PULSE SEQUENCES

READO
1. RL10BB WS
2. RA WB
3. WY
4. RCH WR
5. RR WA
6. RA WB
8. RZ WS ST2

WRITEO
1. RL10BB WS
2. RA WB WG
3. WY
4. RCH WR
5. RA WCH
6. RA WB
8. 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 WCH
8. RZ WS ST2
PULSF SEQUENCES

RORO
1. R10B R W S
2. R A B W
3. R P W Y
4. R C H W R
5. R R R U W A
6. R A W B
8. R Z W S S T 2

WORO
1. R L10B R W S
2. R A W B
3. R P W Y
4. R C H W R
5. R R R U W A W C H
6. R A W B
8. R Z W S S T 2

RXORO
1. R L10B R W S
2. R A W B
3. R C H W Y
4. R C H W B
5. R A R C W G
7. R G W B
8. R Z W S S T 2
9. R C W G
10. R U W B
11. R C R G W A

RUPTO
1. R 15 W S
2. R S C W G
9. R Z W G
10. S T 1

RUPT1
1. R 15 R R 2 W S
2. R S C W G
3. R R P A W Z
8. R Z W S S T 2
9. R B W G K R P T
**Pulse Sequences**

**DVO**

1. RA WB TSGN TMZ
2. 0X RC WA TMZ DVST
3. 1X DVST
4. RU WB STAGE

**DV1**

4. 0X RI WB TSGN
5. 1X RP WY B15X
6. 1X RC WY B15X Z16
7. RU WL TOV
8. 0X RG RSC WB TSGN
9. 1X RA WY PONEX
10. 1X RA WY
11. 0X RR WA
12. 1X RC WA Z15
13. RU WB
14. RL WYD
15. RU WL
16. L2GD PB WYD A2X PIFL
17. 0X RG WL TSGU DVST CLXC
18. 1X RG WL TSGU DVST RB1F
19. RU WB
20. L2GD PB WYD A2X PIFL
21. 0X RG WL TSGU CLXC
22. 1X RG WL TSGU RB1F
23. RU WB
24. L2GD RR WYD A2X PIFL
25. 0X RG WL TSGU CLXC
26. 1X RG WL TSGU RB1F
27. RU WB
28. L2GD RR WYD A2X PIFL
29. 0X RG WL TSGU DVST CLXC
30. 1X RG WL TSGU DVST RB1F
31. RU WB
32. L2GD RR WYD A2X PIFL
33. 0X RG WL TSGU CLXC
34. 1X RG WL TSGU RB1F
35. RU WB
36. L2GD RR WYD A2X PIFL
37. 0X RG WL TSGU DVST CLXC
38. 1X RG WL TSGU DVST RB1F
39. RU WB
40. L2GD RR WYD A2X PIFL
41. 0X RG WL TSGU CLXC
42. 1X RG WL TSGU RB1F
43. RU WB
44. L2GD RR WYD A2X PIFL
45. 0X RG WL TSGU DVST CLXC
46. 1X RG WL TSGU DVST RB1F
47. RU WB
48. L2GD RR WYD A2X PIFL
49. 0X RG WL TSGU CLXC
50. 1X RG WL TSGU RB1F
51. RU WB
52. L2GD RR WYD A2X PIFL
53. 0X RG WL TSGU DVST CLXC
54. 1X RG WL TSGU DVST RB1F
55. RU WB
56. L2GD RR WYD A2X PIFL
57. 0X RG WL TSGU CLXC
58. 1X RG WL TSGU RB1F
59. RU WB
60. L2GD RR WYD A2X PIFL
61. 0X RG WL TSGU DVST CLXC
62. 1X RG WL TSGU DVST RB1F
63. RU WB
64. L2GD RR WYD A2X PIFL
65. 0X RG WL TSGU CLXC
66. 1X RG WL TSGU RB1F
67. RU WB
68. L2GD RR WYD A2X PIFL
69. 0X RG WL TSGU DVST CLXC
70. 1X RG WL TSGU DVST RB1F
71. RU WB
72. L2GD RR WYD A2X PIFL
73. 0X RG WL TSGU CLXC
74. 1X RG WL TSGU RB1F
75. RU WB
76. L2GD RR WYD A2X PIFL
77. 0X RG WL TSGU DVST CLXC
78. 1X RG WL TSGU DVST RB1F
79. RU WB
80. L2GD RR WYD A2X PIFL
81. 0X RG WL TSGU CLXC
82. 1X RG WL TSGU RB1F
83. RU WB
84. L2GD RR WYD A2X PIFL
85. 0X RG WL TSGU DVST CLXC
86. 1X RG WL TSGU DVST RB1F
87. RU WB
88. L2GD RR WYD A2X PIFL
89. 0X RG WL TSGU CLXC
90. 1X RG WL TSGU RB1F
91. RU WB
92. L2GD RR WYD A2X PIFL
93. 0X RG WL TSGU DVST CLXC
94. 1X RG WL TSGU DVST RB1F
95. RU WB
96. L2GD RR WYD A2X PIFL
97. 0X RG WL TSGU CLXC
98. 1X RG WL TSGU RB1F
99. RU WB
100. L2GD RR WYD A2X PIFL
101. 0X RG WL TSGU DVST CLXC
102. 1X RG WL TSGU DVST RB1F
103. RU WB
104. L2GD RR WYD A2X PIFL
105. 0X RG WL TSGU CLXC
106. 1X RG WL TSGU RB1F
107. RU WB
108. L2GD RR WYD A2X PIFL
109. 0X RG WL TSGU DVST CLXC
110. 1X RG WL TSGU DVST RB1F
111. RU WB
112. L2GD RR WYD A2X PIFL
113. 0X RG WL TSGU CLXC
114. 1X RG WL TSGU RB1F
115. RU WB
116. L2GD RR WYD A2X PIFL
117. 0X RG WL TSGU DVST CLXC
118. 1X RG WL TSGU DVST RB1F
119. RU WB
120. L2GD RR WYD A2X PIFL
121. 0X RG WL TSGU CLXC
122. 1X RG WL TSGU RB1F
123. RU WB
124. L2GD RR WYD A2X PIFL
125. 0X RG WL TSGU DVST CLXC
126. 1X RG WL TSGU DVST RB1F
127. RU WB
128. L2GD RR WYD A2X PIFL
129. 0X RG WL TSGU CLXC
130. 1X RG WL TSGU RB1F
131. RU WB
132. L2GD RR WYD A2X PIFL
133. 0X RG WL TSGU DVST CLXC
134. 1X RG WL TSGU DVST RB1F
135. RU WB
136. L2GD RR WYD A2X PIFL
137. 0X RG WL TSGU CLXC
138. 1X RG WL TSGU RB1F
139. RU WB
140. L2GD RR WYD A2X PIFL
141. 0X RG WL TSGU DVST CLXC
142. 1X RG WL TSGU DVST RB1F
143. RU WB
144. L2GD RR WYD A2X PIFL
145. 0X RG WL TSGU CLXC
146. 1X RG WL TSGU RB1F
147. RU WB
148. L2GD RR WYD A2X PIFL
149. 0X RG WL TSGU DVST CLXC
150. 1X RG WL TSGU DVST RB1F
151. RU WB
152. L2GD RR WYD A2X PIFL
153. 0X RG WL TSGU CLXC
154. 1X RG WL TSGU RB1F
155. RU WB
156. L2GD RR WYD A2X PIFL
157. 0X RG WL TSGU DVST CLXC
158. 1X RG WL TSGU DVST RB1F
159. RU WB
160. L2GD RR WYD A2X PIFL
161. 0X RG WL TSGU CLXC
162. 1X RG WL TSGU RB1F
163. RU WB
164. L2GD RR WYD A2X PIFL
165. 0X RG WL TSGU DVST CLXC
166. 1X RG WL TSGU DVST RB1F
167. RU WB
168. L2GD RR WYD A2X PIFL
169. 0X RG WL TSGU CLXC
170. 1X RG WL TSGU RB1F
171. RU WB
172. L2GD RR WYD A2X PIFL
173. 0X RG WL TSGU DVST CLXC
174. 1X RG WL TSGU DVST RB1F
175. RU WB
176. L2GD RR WYD A2X PIFL
177. 0X RG WL TSGU CLXC
178. 1X RG WL TSGU RB1F
179. RU WB
180. L2GD RR WYD A2X PIFL
181. 0X RG WL TSGU DVST CLXC
182. 1X RG WL TSGU DVST RB1F
183. RU WB
184. L2GD RR WYD A2X PIFL
185. 0X RG WL TSGU CLXC
186. 1X RG WL TSGU RB1F
187. RU WB
188. L2GD RR WYD A2X PIFL
189. 0X RG WL TSGU DVST CLXC
190. 1X RG WL TSGU DVST RB1F
191. RU WB
192. L2GD RR WYD A2X PIFL
193. 0X RG WL TSGU CLXC
194. 1X RG WL TSGU RB1F
195. RU WB
196. L2GD RR WYD A2X PIFL
197. 0X RG WL TSGU DVST CLXC
198. 1X RG WL TSGU DVST RB1F
199. RU WB
200. L2GD RR WYD A2X PIFL
201. 0X RG WL TSGU CLXC
202. 1X RG WL TSGU RB1F
203. RU WB
204. L2GD RR WYD A2X PIFL
205. 0X RG WL TSGU DVST CLXC
206. 1X RG WL TSGU DVST RB1F
207. RU WB
PULSE SEQUENCES

DV7

4. L2GD RB WYD A2X P1FL
5. 0X RG WL TSGU CLXC
6. 1X RG WL TSGU RH1F
7. L2GD RB WYD A2X P1FL
8. 0X RG WL TSGU CLXC
9. 1X RG WL TSGU RH1F
10. L2GD RB WYD A2X P1FL
11. 0X RG WL TSGU CLXC
12. 1X RG WL TSGU RH1F

DV6

4. L2GD RB WYD A2X P1FL
5. 0X RG WL TSGU CLXC
6. 1X RG WL TSGU RH1F
7. L2GD RB WYD A2X P1FL
8. 0X RG WL TSGU CLXC
9. 1X RG WL TSGU RH1F
10. L2GD RB WYD A2X P1FL
11. 0X RG WL TSGU CLXC
12. 1X RG WL TSGU RH1F

DV4

3. RU WB STAGE
4. L2GD RB WYD A2X P1FL
5. 0X RG WB WA TSGU CLXC
6. 1X RG WB WA TSGU RH1F
7. 01 RC WA
8. 1X RC WA
9. RZ WS ST2 T5GN R5TSTG
10. 0X RU WB WL
11. RC WL
PULS E SEQUENCES

BZFO
1. RA WG TSGN TMZ
2. TPZG
3. RSC WG
5. X1 RB WY12 CI
6. X1 RU 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

QXCHO
1. RL10BB WS
2. RSC WG
3. RQ WB
5. RG WQ
7. RB 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. RP WY12 MONEX CI
2. RSC WG
7. RG WB
8. RU WS
9. RP WG
10. RP WL ST1

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

DCS0
1. RP WY12 MONEX CI
2. RSC WG
7. RG WB
8. RU WS
9. RB WG
10. RC WL ST1

DCS1
2. RSC WG
7. RG WB
8. RZ WS ST2
9. RR WG
10. RC WA
### PULSE SEQUENCES

**NDXXD**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></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</td>
</tr>
<tr>
<td>9</td>
<td>RB WG</td>
</tr>
<tr>
<td>10</td>
<td>ST1</td>
</tr>
</tbody>
</table>

**NDXX1**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>RZ WY12 CI</td>
</tr>
<tr>
<td>2</td>
<td>RSC WG NISO</td>
</tr>
<tr>
<td>3</td>
<td>WB WZ</td>
</tr>
<tr>
<td>4</td>
<td>RA WB</td>
</tr>
<tr>
<td>5</td>
<td>RZ WA</td>
</tr>
<tr>
<td>6</td>
<td>RU WZ</td>
</tr>
<tr>
<td>7</td>
<td>RG WY A2X</td>
</tr>
<tr>
<td>8</td>
<td>RU WS</td>
</tr>
<tr>
<td>9</td>
<td>RB WA</td>
</tr>
<tr>
<td>10</td>
<td>RU WB EXT</td>
</tr>
</tbody>
</table>

**SUO**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></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>RP WG</td>
</tr>
<tr>
<td>10</td>
<td>RC WY A2X</td>
</tr>
<tr>
<td>11</td>
<td>RU WA</td>
</tr>
</tbody>
</table>

**BZMFD**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>RA WG TSGN TMZ</td>
</tr>
<tr>
<td>2</td>
<td>TPZG</td>
</tr>
<tr>
<td>3</td>
<td>RSC WG</td>
</tr>
<tr>
<td>5 01</td>
<td>RP WY12 CI</td>
</tr>
<tr>
<td>5 10</td>
<td>RP WY12 CI</td>
</tr>
<tr>
<td>5 11</td>
<td>RP WY12 CI</td>
</tr>
<tr>
<td>6 01</td>
<td>RU WZ</td>
</tr>
<tr>
<td>6 10</td>
<td>RU WZ</td>
</tr>
<tr>
<td>6 11</td>
<td>RU WZ</td>
</tr>
<tr>
<td>8 00</td>
<td>RZ WS ST2</td>
</tr>
<tr>
<td>8 01</td>
<td>RAD WR WS NISO</td>
</tr>
<tr>
<td>8 10</td>
<td>RAD WR WS NISO</td>
</tr>
<tr>
<td>8 11</td>
<td>RAD WR WS NISO</td>
</tr>
</tbody>
</table>
# PULSE SEQUENCES

## MP0

<table>
<thead>
<tr>
<th>Number</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>RSC WG</td>
</tr>
<tr>
<td>3</td>
<td>RA WB TSGN</td>
</tr>
<tr>
<td>4</td>
<td>0X RR WL</td>
</tr>
<tr>
<td>4</td>
<td>1X RC WL</td>
</tr>
<tr>
<td>7</td>
<td>RG WA TSGN2</td>
</tr>
<tr>
<td>8</td>
<td>RZ WS</td>
</tr>
<tr>
<td>9</td>
<td>00 RP WY</td>
</tr>
<tr>
<td>9</td>
<td>01 RR WY CI</td>
</tr>
<tr>
<td>9</td>
<td>10 RC WY CI</td>
</tr>
<tr>
<td>9</td>
<td>11 RC WY</td>
</tr>
<tr>
<td>10</td>
<td>RU WB TSGN ST1 NACON</td>
</tr>
<tr>
<td>11</td>
<td>0X WA</td>
</tr>
<tr>
<td>11</td>
<td>1X WA RB1 R1C 16</td>
</tr>
</tbody>
</table>

## MP1

<table>
<thead>
<tr>
<th>Number</th>
<th>Instruction</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>ZAP</td>
</tr>
</tbody>
</table>

## MP3

<table>
<thead>
<tr>
<th>Number</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ZAP NISQ</td>
</tr>
<tr>
<td>2</td>
<td>ZAP</td>
</tr>
<tr>
<td>3</td>
<td>ZIP RSC WG</td>
</tr>
<tr>
<td>4</td>
<td>RZ WY12 CI</td>
</tr>
<tr>
<td>5</td>
<td>RU WZ TL15 NFACOF</td>
</tr>
<tr>
<td>6</td>
<td>1X RR WY A2X</td>
</tr>
<tr>
<td>7</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 RU WA</td>
</tr>
</tbody>
</table>

## STD2

<table>
<thead>
<tr>
<th>Number</th>
<th>Instruction</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
1. RSCT WS
2. RSC WG
5. RG WY TSGN TMZ TPZG
6. PONFX
7. RU WSC WG WOVR
8. RR WS

PCDU
1. RSCT WS
2. RSC WG
5. RG WY TSGN TMZ TPZG
6. CI
7. RU WSC WG WOVR
8. RR WS

MINC
1. RSCT WS
2. RSC WG
5. RG WY TSGN TMZ TPZG
6. MONFX
7. RU WSC WG WOVR
8. RR WS

MCDU
1. RSCT WS
2. RSC WG
5. RG WY TSGN TMZ TPZG
6. MONFX CI
7. RU WSC WG WOVR
8. RR WS

DINC
1. RSCT WS
2. RSC WG
5. RG WY TSGN TMZ TPZG
6. 00 MONFX POUT
7. RU WSC WG WOVR
8. RR WS

48
PULSF SEQUENCES

SHINC
1. RSCT WS
2. RSC WG
5. RG WYD TSGN
7. RUS WSC WG WOVR
8. RR WS

SHNCH
1. RSCT WS
2. RSC WG
5. RG WYD TSGN CI
7. RUS WSC WG WOVR
8. RR WS

INOTRD
1. WS
2. RSC WG
5. RCH
8. RR WS

INOTLD
1. WS
2. RSC WG
5. RCH
7. WCH
8. RR WS
PULSE SEQUENCES

FETCH0

1. RA WS
2. RSC WG WY ST1
4. WSC
8. WS

FETCH1

2. RSC WG
7. RG
8. RP WS U2BBK
10. RRBK

STOREF0

1. RA WS
2. RSC WG WY ST1
4. WSC
8. WS

STOREF1

2. RSC WG
4. WSC
7. RG
8. RP WS U2BBK
9. WG
10. RRBK
### AGC BLOCK II MEMORY ORGANIZATION AND ADDRESSING

#### ERASABLE MEMORY

<table>
<thead>
<tr>
<th>Words Per Box</th>
<th>Pseudo</th>
<th>ECADR</th>
<th>EB SREG</th>
<th>EB SREG*</th>
</tr>
</thead>
<tbody>
<tr>
<td>400 (OCTAL)</td>
<td>00 000</td>
<td>0 000</td>
<td>X 0 000</td>
<td>0 1 400</td>
</tr>
<tr>
<td></td>
<td>00 400</td>
<td>0 400</td>
<td>X 0 400</td>
<td>1 1 400</td>
</tr>
<tr>
<td></td>
<td>01 000</td>
<td>1 000</td>
<td>X 1 000</td>
<td>2 1 400</td>
</tr>
<tr>
<td></td>
<td>02 400</td>
<td>2 400</td>
<td>5 1 400</td>
<td>EB SREG</td>
</tr>
<tr>
<td></td>
<td>03 000</td>
<td>3 000</td>
<td>6 1 400</td>
<td>EB SREG</td>
</tr>
<tr>
<td></td>
<td>04 000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EXT FB SREG</td>
<td>EXT FB SREG*X</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>00 2 000</td>
<td>01 2 000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>02 2 000</td>
<td>03 2 000</td>
</tr>
<tr>
<td></td>
<td>06 000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10 000</td>
<td>12 000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 000 Words**</td>
<td>Pseudo</td>
<td>ECADR</td>
<td>EB SREG</td>
<td>EB SREG*</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EXT FB SREG*X XX</td>
<td>EXT FB SREG*X</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4 000</td>
<td>2 000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>00 2 000</td>
<td>01 2 000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>02 2 000</td>
<td>03 2 000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>06 000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>10 000</td>
<td>12 000</td>
</tr>
<tr>
<td>10 000 Words**</td>
<td></td>
<td></td>
<td>EXT FB SREG</td>
<td>EXT FB SREG*</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>X XX 6 000</td>
<td>X XX 03 2 000</td>
</tr>
<tr>
<td>6 000 Words**</td>
<td>Pseudo</td>
<td>ECADR</td>
<td>EB SREG</td>
<td>EB SREG*</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EXT FB SREG</td>
<td>EXT FB SREG*</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>14 000-17 777</td>
<td>27 2 000</td>
</tr>
</tbody>
</table>

#### FIXED MEMORY

<table>
<thead>
<tr>
<th>Words Per Box</th>
<th>Pseudo</th>
<th>FCADR</th>
<th>EXT FB SREG</th>
<th>EXT FB SREG*</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 000 (OCTAL)</td>
<td>00 000</td>
<td>0 000</td>
<td>X 0 2 000</td>
<td>0 3 2 000</td>
</tr>
<tr>
<td></td>
<td>04 000</td>
<td></td>
<td>EXT FB SREG</td>
<td>EXT FB SREG*</td>
</tr>
<tr>
<td></td>
<td>10 000</td>
<td>14 000</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10 000</td>
<td>14 000</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10 000</td>
<td>14 000</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10 000</td>
<td>14 000</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10 000</td>
<td>14 000</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* NOT PREFERRED
** OCTAL WORD COUNT ADDRESSABLE WITHOUT CHANGING ANY BANK BITS.

**SUPER BANK 3**

**SUPER BANK 4**

---

M.I.T. INSTRUMENTATION LABORATORY——TP1745-3-1 —- 12/65