Pub. No. 4200011A

General Purpose
Controller/Processor
GPC/P

USE OF RALU FLAGS

Application Note

June 1972

INTRODUCTION

The use of the RALU status flags in the General Purpose Controller/Processor (GPC/P) is the subject of this application note. It is assumed that the reader is familiar with the Product Description, General Purpose Controller/Processor (GPC/P), MOS/LSI System Kit, Publication Number 4200005A.

USE OF RALU STATUS FLAGS

The RALU is the Register, Arithmetic, and Logic Unit of the GPC/P. A RALU chip has a 4-bit wide portion of the register and ALU functions. Also, four status flags are provided on each RALU chip. These flags are in addition to the external, TTL logic flags. As shown in figure 1 the flags interface to the R- and A-busses in a manner similar to one of the registers. This provides a convenient method of saving the status during interrupts, and for setting the status flags to a pattern specified by a register or a memory word.

RALU Flag Functions

The four flags have the following functions. See figure 2.

LINK - This flag may be loaded from the most-significant bit (MSB) of the S-bus on any left shift and from the CSH3X input on any right shift. If the SELX input is true, the link will be included in the shift (for example: on right shift CSH3X loads the link and the link loads the MSB of the R-bus). If SELX is false, the link is not loaded and it is not included in the shift. This function is only enabled in the most-significant two RALU chips (that is, where SININX is true at T5).

OVERFLOW - The overflow is set true (when enabled) if there is an arithmetic overflow during an add operation. This flag may be programmed (by mask modification) as either accumulative or nonaccumulative. The standard mask provides for a nonaccumulative flag where a set or a reset occurs on every add operation. A custom mask provides an accumulative function where the flag will be set if an overflow occurs, but will not be reset if an overflow does not occur. This allows a single test for overflow at the end of a routine having many add operations.

CARRY - The carry flag is set to the value of the ripple carry out of the MSB on every add operation (if enabled). This flag may also be mask programmed as accumulative.

FLAG - This flag is provided for the programmer's use (for example: for multilevel interrupt-enable flag).

Example of RALU Flags in 16-bit System

Four RALU chips comprise a 16-bit system. The distribution of flags in a 16-bit system is shown in figure 3. There are nine general-purpose flags and three special-purpose flags. Seven of the general-purpose flags are available at an output pin at any given time. The other two flags are available through the use of the SELX input. In addition, there are four flags (marked I) which may be used internally but are not available at a pin.

The overflow and carry functions are only enabled if CSH3X equals 0 at T1. This occurs for only the most-significant RALU chip as commanded by the CROM. In the other three chips these flags may be used for general purpose.

Loading the flags to and from the A- and R-busses is accomplished using the SVRSTX input (figure 2). If this input is high during T5 of one microcycle, then during the next microcycle, the flags will be loaded onto the A-bus, provided that the A-bus address on the CBX lines is zero (that is, no register is put on the bus). Any pull-stack operation during this microcycle is inhibited; this allows the flags to be put on the A-bus and then stored on the stack. The flags are also loaded from the R-bus during this microcycle, independent of whether a nonzero R-bus address is specified. Because the flags are loaded from the R-bus, the functions of overflow, carry, and link are inhibited at this time.

The SELX input selects whether the link will be included in shift operations and also whether CY0V1 will output the carry or the overflow flag. In minimum-cost systems, this pin could be tied high or low to permanently select one or the other function. In more-general-purpose systems, both SELX and SVRSTX would be driven by flags as shown in figure 4. SELX may be driven by a RALU status flag (for convenience in saving it during interrupts) or from an external flag. The output from the overflow and carry flags (and any other flags desired) is connected into the jump condition multiplexer for use in conditional branches.

Microprocessing Operations Involving RALU Flags

Given below are a number of examples of the microinstructions required to implement operations involving RALU flags. (Note that the term $\underline{\mu I}$ stands for microinstruction.)

Examples Involving RALU Flags

1. Clear All Flags

   $\underline{\mu I}$ 1 - Pulse SVRSTX

   $\underline{\mu I}$ 2 - Reg. 1 on A; 0's on B, $A \cdot B \rightarrow R = 0$, (R into flags)

2. Clear a Specific Flag - Others Unaffected

   μI 1 to N - Load mask from memory to Reg. 1 or form mask by shifting

   μI N+1 - Pulse SVRSTX

   μI N+2 - Reg. 1 on B, 0's (Flags) on A, A · B → R, (R → flags)

3. Push Flags onto Stack

   μI 1 - Pulse SVRSTX

   μI 2 - P/P stack = 1, 0's on B, 0's (Flags) on A, A + B → R → Stack & Flag

4. Pull Stack into Flags

   μI 1 - Pulse SVRSTX, pull stack into MDR

   μI 2 - MDR on A, 0's on B, A + B → R → Flags

5. Store flags in memory

   μI 1 - Address on A-bus, pulse LDAR

   μI 2 - Pulse SVRSTX

   μI 3 - 0's (flags) on A → Data Bus, Pulse WRITE

6. Load Flags from Memory

   μI 1 - Pulse SVRSTX

   μI 2 - Memory Address on A-bus, pulse READ, mux → R → Flags

7. Double-word Left Shift Reg. 2 into Reg. 1, Open

   μI 1 - Set SELX (if not already set)

   μI 2 - Left Shift Reg. 2, Open (trailing zeroes)

   μI 3 - Left Shift Reg. 1, Circular

8. Double-word Left Shift Reg. 2 into Reg. 1, Circular

$\underline{\mu I}$ 1 Set SELX

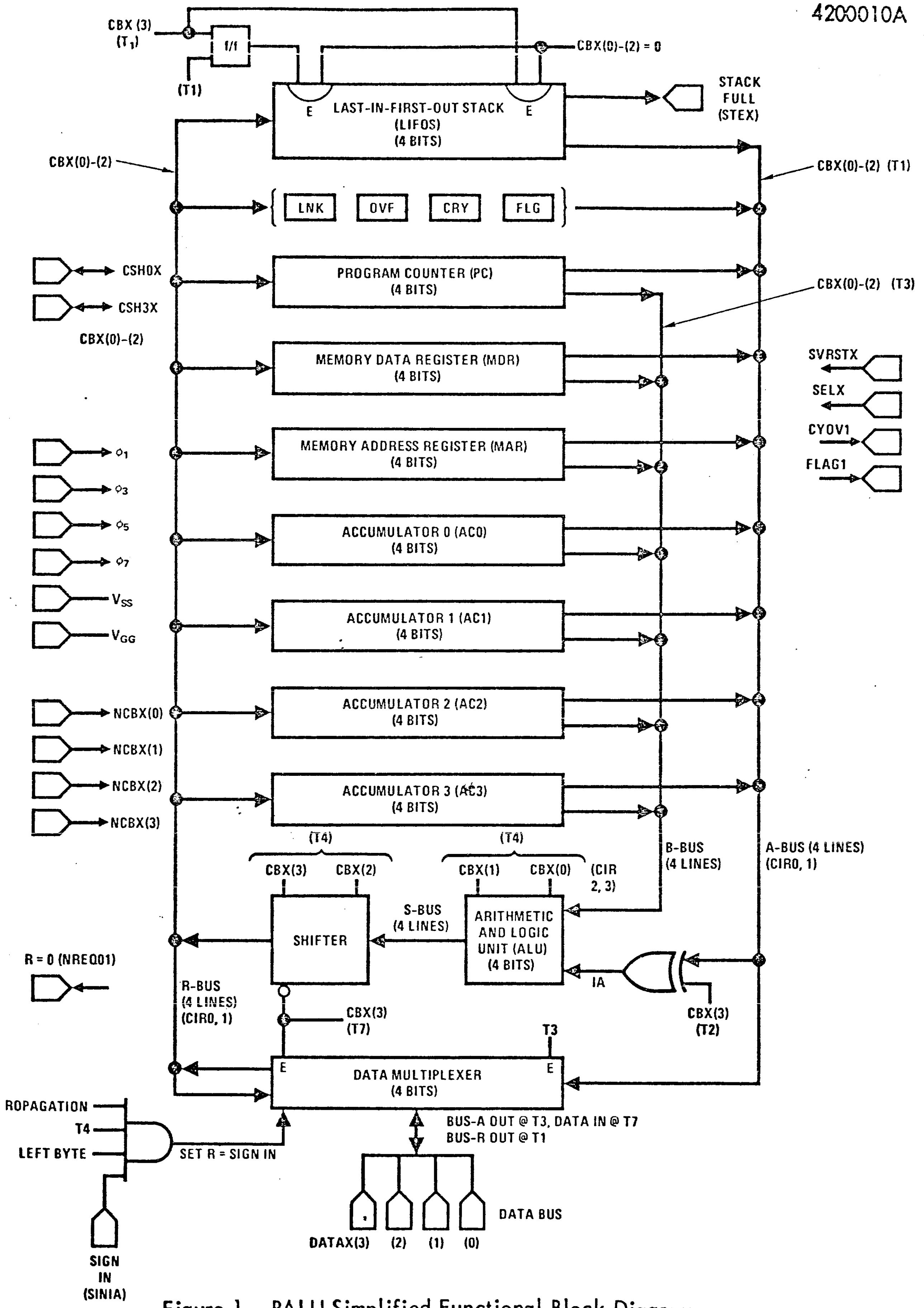
$\underline{\mu I}$ 2 Left Shift Reg. 2, Open but do not store result

$\underline{\mu I}$ 3 Left Shift Reg. 1, Circular

$\underline{\mu I}$ 4 Left Shift Reg. 2, Circular

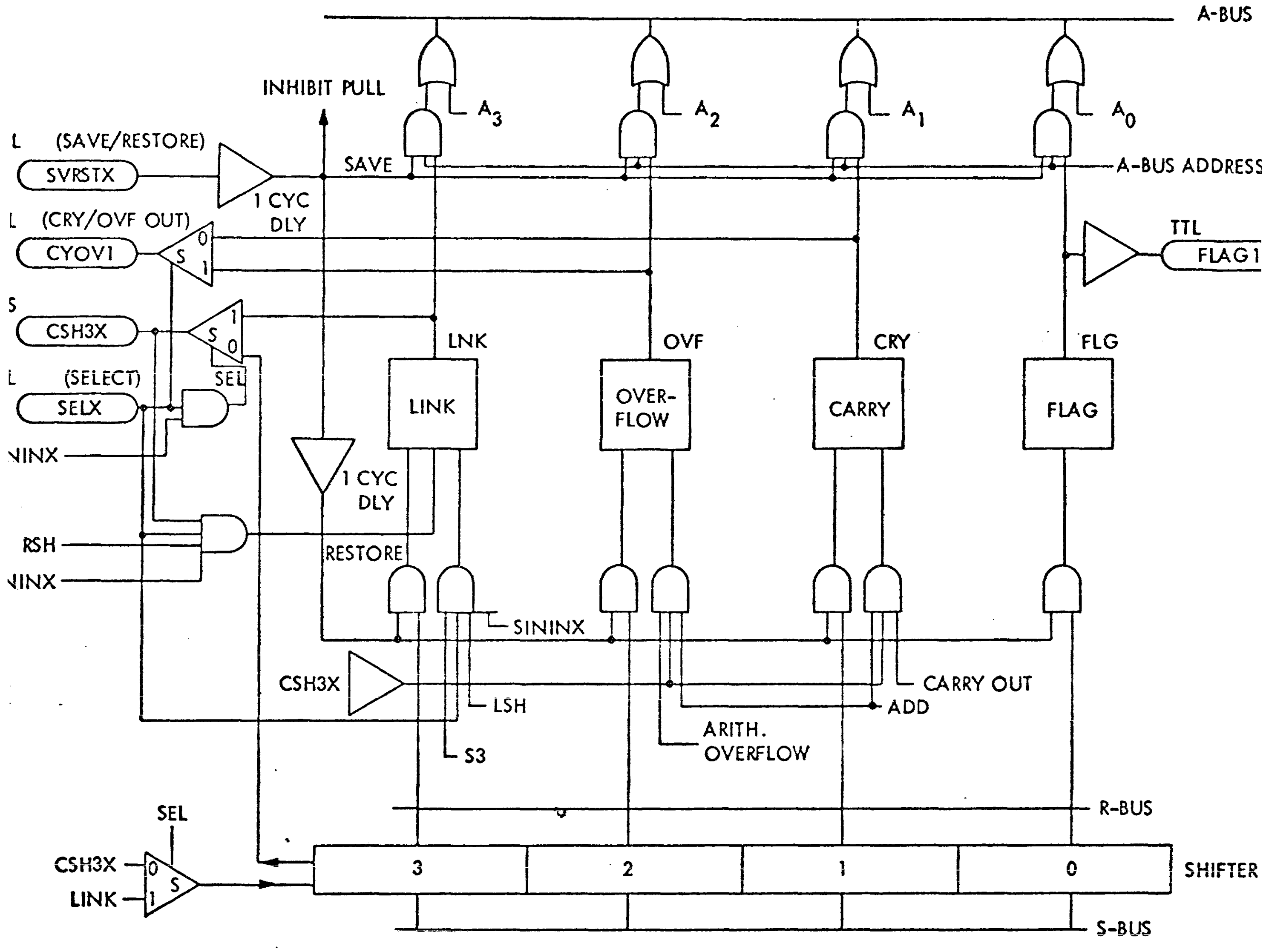
9. Arithmetic Left or Right Shift

Cannot be done conveniently. However, if the LINK is set equal to the sign bit and the SVRSTX input is used to reload the LINK from the sign bit in the same microcycle as the shift occurs, the sign bit will remain unchanged for a right shift.



CBX (3) (T1)
f/f
(T1)
CBX(0)-(2) = 0
E
LAST-IN-FIRST-OUT STACK (LIFOS) (4 BITS)
E
STACK FULL (STEX)
CBX(0)-(2)
CBX(0)-(2) (T1)
LNK
OVF
CRY
FLG
CSH0X
CSH3X
CBX(0)-(2)
PROGRAM COUNTER (PC) (4 BITS)
CBX(0)-(2) (T3)
MEMORY DATA REGISTER (MDR) (4 BITS)
SVRSTX
SELX
CYOV1
FLAG1
MEMORY ADDRESS REGISTER (MAR) (4 BITS)
φ1
φ3
φ5
φ7
VSS
VGG
ACCUMULATOR 0 (AC0) (4 BITS)
ACCUMULATOR 1 (AC1) (4 BITS)
ACCUMULATOR 2 (AC2) (4 BITS)
ACCUMULATOR 3 (AC3) (4 BITS)
NCBX(0)
NCBX(1)
NCBX(2)
NCBX(3)
(T4)
CBX(3)
CBX(2)
(T4)
CBX(1)
CBX(0)
(CIR 2, 3)
B-BUS (4 LINES)
A-BUS (4 LINES) (CIR0, 1)
SHIFTER
S-BUS (4 LINES)
ARITHMETIC AND LOGIC UNIT (ALU) (4 BITS)
R = 0 (NREQ01)
R-BUS (4 LINES) (CIR0, 1)
IA
CBX(3) (T7)
CBX(3) (T2)
T3
E
DATA MULTIPLEXER (4 BITS)
E
ROPAGATION
T4
LEFT BYTE
SET R = SIGN IN
BUS-A OUT @ T3, DATA IN @ T7
BUS-R OUT @ T1
DATA BUS
DATAX(3)
(2)
(1)
(0)
SIGN IN (SINIA)

Figure 1. RALU Simplified Functional Block Diagram



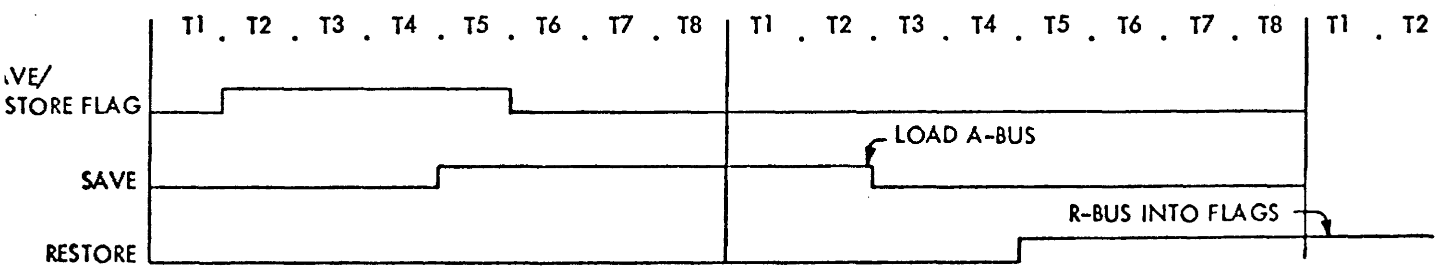


Figure 2. RALU Status Flags

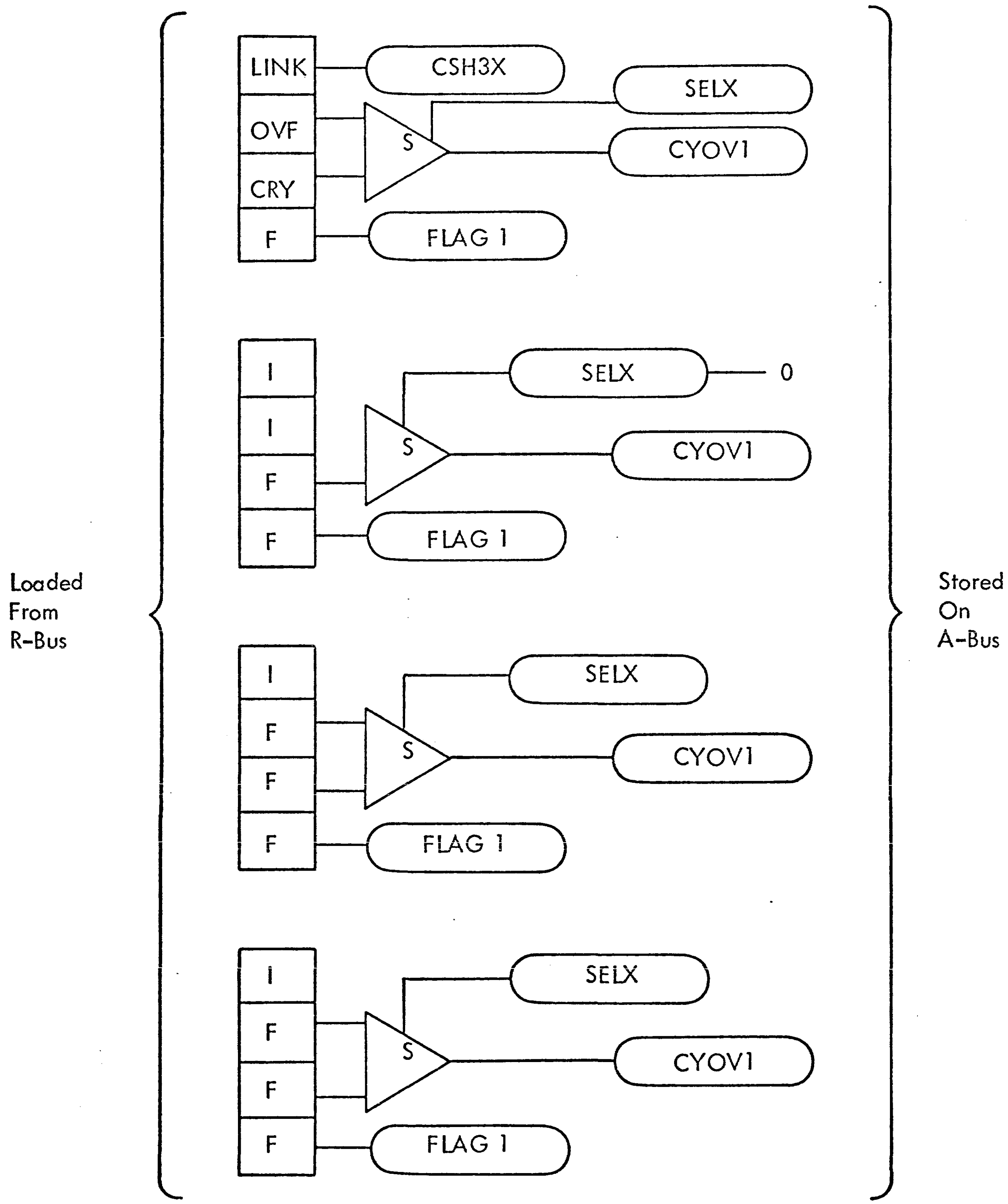


Figure 3. Flags Available in 16-bit Configuration

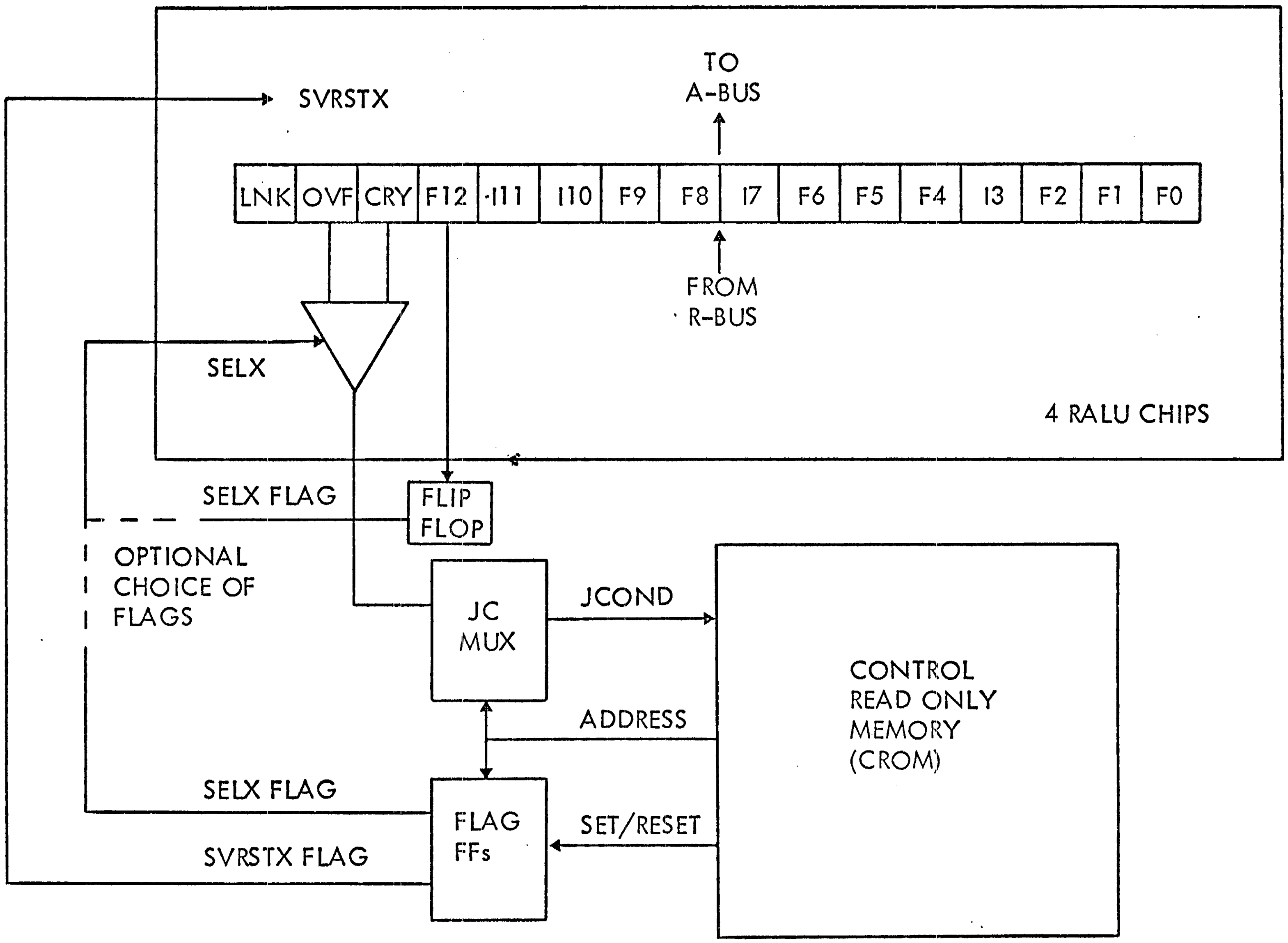


Figure 4. System Diagram, Use of Status Flags