ANL-7149

ANL-7149

MASTER

Argonne National Laboratory

GYRO II,
A MACRO-DEFINED SYSTEM FOR LIST PROCESSING

by

Daniel F. Carson and
George A. Robinson

RELEASED FOR ANNOUNCEMENT

IN NUCLEAR SCIENCE ABSTRACTS

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency Thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

LEGAL NOTICE

This report was prepared as an account of Government sponsored work. Neither the United States, nor the Commission, nor any person acting on behalf of the Commission:

A. Makes any warranty or representation, expressed or implied, with respect to the accuracy, completeness, or usefulness of the information contained in this report, or that the use of any information, apparatus, method, or process disclosed in this report may not infringe privately owned rights; or

B. Assumes any liabilities with respect to the use of, or for damages resulting from the use of any information, apparatus, method, or process disclosed in this report.

As used in the above, "person acting on behalf of the Commission" includes any employee or contractor of the Commission, or employee of such contractor, to the extent that such employee or contractor of the Commission, or employee of such contractor prepares, disseminates, or provides access to, any information pursuant to his employment or contract with the Commission, or his employment with such contractor.

ARGONNE NATIONAL LABORATORY
9700 South Cass Avenue
Argonne, Illinois 60439

ERRATA for ANL-7149,
GYRO II, A MACRO-DEFINED SYSTEM
FOR LIST PROCESSING

by Daniel F. Carson
and George A. Robinson

In the List of References on page 39, Reference 3 should be 4, and
Reference 4 should be 3.

Alvin Glassner
Technical Publications Department

December 8, 1966

ANL-7149
Mathematics and
Computers (TID-4500)
AEC Research and
Development Report

ARGONNE NATIONAL LABORATORY
9700 South Cass Avenue
Argonne, Illinois 60439

CFSH PRICES

H.C. \$7.00; MN .50

GYRO II,
A MACRO-DEFINED SYSTEM FOR LIST PROCESSING

by

Daniel F. Carson and
George A. Robinson

Applied Mathematics Division

May 1966

LEGAL NOTICE

This report was prepared as an account of Government sponsored work. Neither the United States, nor the Commission, nor any person acting on behalf of the Commission:

A. Makes any warranty or representation, expressed or implied, with respect to the accuracy, completeness, or usefulness of the information contained in this report, or that the use of any information, apparatus, method, or process disclosed in this report may not infringe privately owned rights; or

B. Assumes any liabilities with respect to the use of, or for damages resulting from the use of any information, apparatus, method, or process disclosed in this report.

As used in the above, "person acting on behalf of the Commission" includes any employee or contractor of the Commission, or employee of such contractor, to the extent that such employee or contractor of the Commission, or employee of such contractor prepares, disseminates, or provides access to, any information pursuant to his employment or contract with the Commission, or his employment with such contractor.

Operated by The University of Chicago
under
Contract W-31-109-eng-38
with the
U. S. Atomic Energy Commission

THIS PAGE
WAS INTENTIONALLY
LEFT BLANK

TABLE OF CONTENTS

	<u>Page</u>
CHAPTER I. INTRODUCTION.	5
CHAPTER II. NOMENCLATURE.	6
A. Lists	6
1. Linked Memory Cells	6
2. Lists	6
B. Available Memory	6
C. Down-push and Up-pop	6
D. Indirect Addressing--Upper Address	6
CHAPTER III. THE GYRO II SYSTEM	7
A. The GYRO II Source Language	7
B. Requirements for Implementation of GYRO II	7
1. Hardware	7
2. Software	7
C. The GYRO II System COMMON	7
D. Bounds Interrupt Routine	8
E. Automatic Check for End of List	8
F. Recursive Return Jumps	8
CHAPTER IV. THE GYRO II MACRO-INSTRUCTIONS.	9
A. Notation	9
1. Indirect Addressing	9
2. Conditional Jumps	9
3. Implied Parameter Values	9
B. GYRO II System Initialization	9
C. List Manipulation	10
1. Available Memory	10
2. Down-push and Up-pop	10
3. Insert and Attach	11
4. Remove	11
5. Load	11
6. Terminal Members	12
7. Clear	12
8. Multiple Register Save and Restore	12

TABLE OF CONTENTS

	<u>Page</u>
D. Input-Output	13
1. Basic Input-Output	13
2. GYRO II Subroutine Calls	13
E. Miscellaneous Macro-instructions	14
1. Jump	14
2. Bit Toggle and Test	15
3. Register Operations	15
3.1 Enter	15
3.2 Transfer	15
3.3 Store	16
3.4 Load	16
4. Store and Place	17
5. Clear	17
6. Subroutine Call	17
CHAPTER V. IMPLEMENTATION OF GYRO II FOR THE CDC-3600	18
A. Language-implementation Correspondences . .	18
1. Registers	18
2. Memory Cells	18
3. Indirect Addressing	18
4. Assembly Language	18
5. Abnormal Termination	18
6. Auxiliary Subroutines	18
7. Memory Bounds and Interrupts	19
B. Registers Altered by Execution of GYRO-II Macros	19
C. Expansions of GYRO II Macros for the CDC- 3600	19
REFERENCES	39

GYRO II, A MARCO-DEFINED SYSTEM FOR LIST PROCESSING

by

Daniel F. Carson and
George A. Robinson

CHAPTER I INTRODUCTION

List-processing languages such as IPL-V (Ref. 1) and LISP (Ref. 2) provide the user with a certain degree of programming ease. The running times and restrictions in data representation in such languages may be avoided by a user willing and competent to write in an assembly language. As pointed out in Ref. 3, even direct hardware implementation of IPL-V did not decrease running times sufficiently to compete favorably with a language more directly oriented to the structure of present-day machines. Given that one is familiar with an assembly language, two obvious objections to a straightforward coding of a list-processing program (or indeed any program) are the time necessary to produce a working code and the possible lack of machine independence. These difficulties however, can be reduced appreciably by the development of a set of macro-instructions which apply to a large class of problems and reduce the coding time per problem.

GYRO II, unlike some other macro-defined list-processing languages (e.g., see Ref. 4) has imposed no restrictions on the internal machine representation of the nonlink portion of list elements. The user can therefore develop, by means of GYRO II and the assembly language it includes, a system of whatever degree of sophistication his problem may require.

Chapters II, III, and IV of this report give a formal description of the GYRO II language. Chapter V discusses the implementation of GYRO II for the Control Data 3600 at Argonne National Laboratory.

CHAPTER II

NOMENCLATURE

A. Lists

1. Linked Memory Cells

Each memory cell consists of a link portion and a nonlink portion. Memory cell A links to memory cell B, and B is linked by A when the link portion of A specifies the location of B in memory. The integer represented in the link portion is called the link.

2. Lists

A list L is a finite sequence $\{B_i\}$ of n memory cells such that B_i links to B_{i+1} for $1 \leq i \leq n-1$ and the link of B_n is zero. The B_i are called the members of the list L. B_1 is called the head of L, and B_n is called the terminal member of L.

B. Available Memory

Those macro-instructions which add or delete members of lists rely on the existence of a pool of unused memory cells. This pool, called the available memory, consists initially of a list of 20,000 memory cells in consecutive memory locations in the array GYROSAV which is in the system COMMON. To return a cell to available memory is to make it a member of this list. To get a cell from available memory is to remove a member from this list.

C. Down-push and Up-pop

To down-push a list L is to get a cell C from the available memory, to transfer the contents of the head of L to C, and to cause the head of L to link to C.

To up-pop a list L is to transfer the contents of the cell linked by the head H of L to H.

D. Indirect Addressing--Upper Address

For GYRO II macro-instructions, indirect addressing means that the location LOC specified in the parameter list points to the address A to be used in execution of the instruction. GYRO II allows for two different methods of indirect addressing, indirect upper and indirect lower. For indirect-lower addressing the link portion of LOC specifies A; for indirect-upper the nonlink portion specifies A. The address in the nonlink portion is called the upper address.

CHAPTER III

THE GYRO II SYSTEM

A. The GYRO II Source Language

The source language of GYRO II consists of the macros defined in Chapter IV together with the assembly language for the machine for which GYRO II is being implemented. The restrictions which apply to the particular assembly language also apply to GYRO II for that particular implementation.

B. Requirements for Implementation of GYRO II

1. Hardware

Full implementation of GYRO II requires at least 4 program-accessible registers, a word size large enough to hold 2 machine addresses, and some method of program-controllable memory protection. The mnemonics used for the registers are A, B, D, and Q, of which A and Q must be full-word registers. Memory protection is used only for the automatic check for up-popping a list whose head is also its terminal member.

2. Software

Full implementation of GYRO II also requires the assembler to have macro- and COMMON-defining capabilities.

C. The GYRO II System COMMON

Available memory, the interrupt routine, and other special-purpose memory cells are made available to the subroutines written in GYRO II through a common storage area (COMMON). The names of these cells and their use are:

AVAILSPA	The head of available memory.
NOAVAILS	Contains an unconditional bank jump to the location to which control is to be transferred after a bounds interrupt has been processed. If no program request has been given, control will be transferred to a location which will cause abnormal termination.
PROPLIST (64)	Reserves 64 locations.
TEMP	A temporary storage cell.

APDL	The head of the system down-push list for the A register.
BPDL	The head of the system down-push list for the B register.
QPD L	The head of the system down-push list for the Q register.
DPDL	The head of the system down-push list for the D register.
ZERO	Contains positive zero.
TEN	Contains the integer 10.
LISTBIT	Has the sign bit set; all other bits are cleared.
GBIA (I)	Reserves I locations for the bounds interrupt routine.
GYROSAV (K)	Reserves K memory cells to form the initial available memory list.
G2RECUR (N)	Reserves N locations for the recursive jump routine.
RCPDL	The head of the system down-push list used by the recursive jump routine.

D. Bounds Interrupt Routine

This routine, which is transmitted to the systems COMMON by the INITIAL macro, performs the following functions: (1) plants the address where the interrupt occurred in the link portion of TEMP; (2) clears the interrupt condition; (3) jumps to NOAVAILS.

E. Automatic Check for End of List

The INITIAL macro-instruction sets the bounds limit so that any attempt to up-pop a list past its terminal member will result in a bounds interrupt. Control will be transferred on interrupt to the last location specified by the EOL macro (see Chapter IV, Section B) or to a location which will cause abnormal termination.

F. Recursive Return Jumps

After the initial call of a subroutine by a return jump instruction (JR), it is sometimes desirable to call the subroutine from within itself or from subroutines called by the given subroutine. The macro JRR saves the return addresses in the down-push list headed by RCPDL. These addresses are then restored with each normal exit from the subroutine. The macro REXIT causes an exit to the return address for the initial call without exiting through all the return addresses in RCPDL.

CHAPTER IV

THE GYRO II MACRO-INSTRUCTIONS

A. Notation

1. Indirect Addressing

Indirect addressing, where allowed, is indicated by modification of the macro-instruction name by suffixing either a J for indirect upper or a K for indirect lower. The macro-instructions which allow this modification will have the modified form below the base form in the descriptions.

2. Conditional Jumps

In some cases it is possible to compress a basic macro-instruction followed by a conditional jump into a single instruction. This is done by adding the parameter list for the conditional jump to the parameter list for the basic operation. Those macro-instructions which permit conditional-jump criteria as part of their parameters are denoted in the descriptions as having the jump criteria JAD, C1, C2, C3 as the last arguments on their parameter list. The jump criteria must be one of the forms listed under the description of the jump macro. If the conditional jump is not desired, the jump criteria should be left blank.

3. Implied Parameter Values

Certain of the formal parameter's in the macro-instruction description are underlined. If the user omits such a parameter when employing the macro-instruction, B is assumed to contain the appropriate pointer.

B. GYRO II System Initialization

INITIAL	Links the available memory array and sets the bounds limits. The GYRO II bounds-interrupt and recursive-jump routines are set up in COMMON. It is necessary to use this instruction only once for the entire program.
START EP	Defines EP as an entry point, reserves a memory cell at EP, and defines the COMMON needed by GYRO II. EP must appear in the location field of this macro-instruction. This instruction must be used once and only once per subroutine.

EOL LOC Arms the GYRO II bounds-interrupt apparatus so that control is transferred to LOC after a bounds interrupt and execution of the GYRO II interrupt routine. If this instruction has not been executed before a bounds interrupt, the job will terminate abnormally.

C. List Manipulation

1. Available Memory

GETAV Up-pops the list of available memory. After execution, B points to the freed cell.

GETAV D Same as GETAV except D points to the freed cell.

RETAV Causes the cell pointed to by B to be a member of the available memory list.

RELINK Causes available memory to consist of the array of the linked memory cells which initially constitutes available memory.

2. Down-push and Up-pop

D S,JAD,C1,C2,C3 Down-pushes the list S. After execution,
DJ B will point to the cell containing a copy of
DK the former contents of the head of S.

DS S,JAD,C1,C2,C3 Down-pushes the list S and stores the
DSJ A register into the head of S. After execu-
DSK tion, B will point to the cell containing a
copy of the former contents of the head of S.

DP S,JAD,C1,C2,C3 Down-pushes the list S and replaces the
nonlink portion of the head of S with the
nonlink portion of A. After execution, B will
point to the cell containing a copy of the
former contents of the head of S.

DF S,JAD,C1,C2,C3 Down-pushes the list S and stores the
DFJ A register into a cell C obtained from
DFK available memory. The upper address of
the head of S will point to C.

U S,JAD,C1,C2,C3 Up-pops the list S and returns the freed cell
 UJ to available memory. After execution,
 UK B points to the freed cell.

UF S,JAD,C1,C2,C3 Loads the A register with the contents of
 UFJ the cell C pointed to by the upper address
 UFK of the head of S, returns C to available
 memory, and up-pops S.

3. Insert and Attach

I S,JAD,C1,C2,C3 Insert the cell C pointed to by B into the
 IJ list S by replacing the contents of C with
 IK the contents of the head of S and replacing
 the link of the head with the contents of B.

IS S,JAD,C1,C2,C3 Performs the operations of the I macro and
 ISJ then stores the A register into the head
 of S.

IP S,JAD,C1,C2,C3 Performs the operations of the I macro
 IPJ and then replaces the nonlink portion of the
 IPK head of S with the nonlink portion of A.

A S,JAD,C1,C2,C3 Attaches the cell C pointed to by B to the
 AJ list S by causing C to link to the second
 AK member of S and then causing the head of
 S to link to C. After execution A contains
 the initial contents of the head of S.

4. Remove

R S,JAD,C1,C2,C3 Removes the head of the list S by replacing
 RJ the contents of the head by the contents of
 RK the second member C of S. After execution,
 B will point to C.

5. Load

L S,JAD,C1,C2,C3 Loads the A register with the contents of
 LJ the memory cell S. When indirect addressing
 LK is not used, S may be of the form (X,Y),
 where X represents a memory location and
 Y an index register.

LQ S,JAD,C1,C2,C3 Same as L macro except the Q register
 LQJ is loaded and if S is blank D is assumed
 LQK to point to the desired location.

LR S,JAD,C1,C2,C3 Loads the A register with the contents of
 LRJ the head of the list S and then performs the
 LRK macro R S,JAD,C1,C2,C3. Also alters
 the contents of Q.

LU S,JAD,C1,C2,C3 Performs the LR macro and then returns
 LUJ the cell pointed to by B to available memory.
 LUK

6. Terminal Members

LAST S Causes D to point to the terminal member of the list S.

TERM S Causes the link portion of S to be zero. If S = D, then D
 points to the desired memory cell.

RLM S Removes the terminal member T of the list S by causing
 the cell T' which links to T to be the terminal member.
 After execution, B will point to T' and D to T.

7. Clear

ZPD S Down-pushes the list S and then clears the nonlink
 portion of the head of S to zero.

8. Multiple Register Save and Restore

SAV A,B,Q,D Saves the specified registers by placing their con-
 tents on the appropriate system push-down lists.
 If a proper subset is to be saved, the order of the
 registers in the parameter list must be as indicated.
 Thus to save registers B and D would require
 SAV B,D, not SAV D,B. If the parameter list is
 empty, all the possible registers will be saved.

REST A,B,Q,D Restores the specified registers from their
 appropriate push-down lists. The same parameter
 restrictions as with SAV apply.

B Sets B to the last value entered into BPDFL.
 BPDFL is unchanged.

D. Input-Output

1. Basic Input-Output

- RITE CWA,LU,INT** Outputs information onto the specified logical unit LU under control of the control word at CWA. The reject address is assumed to be the initial address of the output request. If LU is blank, the standard output unit is assumed. If INT is blank, no interrupt address is used.
- REED CWA,LU,INT** Inputs information from the specified logical unit LU under control of the control word at CWA. The reject address is assumed to be the initial address of the input request. If LU is blank, standard input unit is assumed. If INT is blank, no interrupt address is used.
- WAIT LU,EF** Delays execution of the next instruction until logical unit LU is available. If EF is non-blank, a jump to EF is executed if an end of file has been encountered on LU.
- INC CWA,N** Increases the lower address of the word at CWA by N. If N is blank, 10 is assumed.

2. GYRO II Subroutine Calls

The subroutines which are called by the following macros wait until the standard output unit is available and then initiate the request. The subroutines do not wait until the operation is complete before returning to the calling program.

- PAGE** Writes a record on the standard output unit telling the printer to set the paper to the top of a new page.
- LINE I** Writes a record on the standard output unit which will produce I blank lines when printed. I can assume only the values of 1 or 2. If I is blank, 1 is assumed.
- NOW** Writes the date and time on the standard output unit with the following format:
- DATE MM/DD/YY TIME HHMM SS.

PRINT ((MESSAGE)) Writes the string of characters MESSAGE on the standard output unit. MESSAGE may contain up to 15 characters. If there are no blanks in MESSAGE, the parentheses may be omitted. MESSAGE may not contain a left or right parenthesis.

HELP ((MESSAGE)) Outputs MESSAGE as in PRINT and then outputs the contents of program accessible registers.

E. Miscellaneous Macro-instructions

1. Jump

J JAD Executes an unconditional jump to JAD.

JR JAD Executes a return jump to JAD.

J JAD,X,TERM Executes a jump to JAD if register X has a link of zero. The possible registers are A,Q, B,D.

J JAD,TERM Same as J JAD,A,TERM.

J JAD,X,ATOM Executes a jump to JAD if, for X = A or Q, the X register is positive; or for X = B or D, X points to a cell whose contents are positive; or for X = memory location, the contents of X are positive.

J JAD,ATOM Same as J JAD,A,ATOM.

**J JAD,X,EQ
NE** Executes a jump to JAD if the contents of the A register and the contents of X are equal (not equal). X may be either Q (i.e., the Q register) or a memory location.

J JAD,X,Y Executes a jump to JAD if register X is in state Y. X may be either A or Q; Y may be PL (plus), MI (minus), ZR (zero), or NZ (nonzero).

J JAD,X,Y,Z Executes a jump to JAD if register X and the integer denoted by Z bear the relation Y. The possible values for X are any program-accessible registers. The mnemonics for Y are EQ,NE,GT,GE,LT,LE, with the usual meanings.

JRR JAD Executes a recursive return jump to JAD. See Chapter III, Section F.

REXIT Exists without going through recursive return address. See Chapter III, Section F.

2. Bit Toggle and Test

BIT X,N,T Performs the specified operation T on the N'th bit of register X. X may be any register, and T may be CL (clear), ST (set), or CM (complement).

BITJ JAD,X,N,S,T Jumps to JAD if the N'th bit of register X is in the state S and then performs the specified operation T on that bit. X may be any register. S may be either CL or ST, and T may have the values as above. If T is blank, no operation is performed on the bit.

3. Register Operations

3.1 Enter

LM S Enters the address of the specified location S into the A register.

LMJ S Enters the address of the specified location S into the upper address of the A register.

LMK S Enters the address of the specified location S into the lower address of the A register.

E X,Y Enters the quantity Y into the X register. X may be A,B,D,Q, or any of the index registers.

3.2 Transfer

T X,Y Transfers register X to register Y. X and Y may be A,B,D,Q, or any of the other registers allowed.

BDT Transfers the contents of B to D.

BJT	Transfers the contents of B to the upper address of A.
BKT	Transfers the contents of B to the link-portion of A.
DBT	Transfers the contents of D to B.
DJT	Transfers contents of D to the upper address of A.
DKT	Transfers contents of D to the link-portion of A.
JBT	Transfers contents of the upper address of A to B.
JDT	Transfers contents of the upper address of A to D.
JKT	Transfers contents of the upper address of A to the link-portion of A.
KBT	Transfers contents of the link-portion of A to B.
KDT	Transfers contents of the link-portion of A to D.
KJT	Transfers contents of the link-portion of A to the upper address of A.

3.3 Store

BJS	<u>S</u>	Stores B into the upper address at S.
BKS	<u>S</u>	Stores B into the lower address at S.
DJS	<u>S</u>	Stores D into the upper address at S.
DKS	<u>S</u>	Stores D into the lower address at S.
JJS	<u>S</u>	Stores upper address of A into the upper address at S.
JKS	<u>S</u>	Stores upper address of A into the lower address at S.
KJS	<u>S</u>	Stores link portion of A into the upper address at S.
KKS	<u>S</u>	Stores link portion of A into the lower address at S.

3.4 Load

JBL	<u>S</u>	Loads B with the upper address at S.
KBL	<u>S</u>	Loads B with the link portion at S.

JDL S Loads D with the upper address at S.

KDL S Loads D with the link portion at S.

JJL S Loads the upper address of A with the upper address at S.

JKL S Loads the link portion of A with the upper address at S.

KJL S Loads the upper address of A with the lower address at S.

KKL S Loads the link portion of A with the lower address at S.

4. Store and Place

S S,JAD,C1,C2,C3 Stores the entire contents of the A register
 SJ into the memory location S. If indirect
 SK addressing is not used, S may be of the form
 (X,Y), where X represents a memory location
 and Y an index register.

SQ S,JAD,C1,C2,C3 Same as the S macro except the Q register
 SQJ is stored and if S is blank, D is assumed to
 SQK point to the desired location.

P S,JAD,C1,C2,C3 Places the nonlink portion of the A register
 PJ into the memory location S.
 PK

5. Clear

ZS S Sets the contents of S to positive zero.

ZP S,Q Clears the nonlink portion of S by means of the Q register
 unless the parameter Q is blank, in which case the A
 register is used.

ZPROP Sets the elements of the COMMON array PROPLIST to
 positive zero.

6. Subroutine Call

CALL SUB For SUB≠DUMP, transfers control to SUB and
 defines SUB as external to the current routine.
 For SUB=DUMP, generates an instruction which
 will cause abnormal termination to occur.

CHAPTER V

IMPLEMENTATION OF GYRO II FOR THE CDC-3600

The reader is assumed to be familiar with the Control Data 3600 computer system and its assembly language COMPASS.

A. Language-implementation Correspondences

1. Registers

- A - The arithmetic register A
- B - The index register B6
- Q - The auxiliary register Q
- D - The index register B5

2. Memory Cells

- Link Portion - Bits 0-14 or the lower address
- Nonlink Portion - Bits 15-47
- Sign Bit - Bit 47

3. Indirect Addressing

- Upper - The upper address, bits 24-38, is used
- Lower - The lower address, bits 0-14, is used

4. Assembly Language

The assembly language is COMPASS Version 5.1.

5. Abnormal Termination

Abnormal termination, due to not having specified a location to which control is to be transferred in the event of a bounds interrupt when no interrupt address has been specified by the EOL macro or due to execution of the CALL DUMP macro, will be caused by an illegal instruction. In the first case the instruction will be at TEN in COMMON; in the second case the illegal instruction will be at the location of the CALL macro.

6. Auxiliary Subroutines

The output macros PAGE, LINE, and NOW generate calls to the subroutine OUTCONT, which resides in bank O. The macros PRINT and HELP generate calls to the subroutine SPOTCHK, which also resides in bank O.

7. Memory Bounds and Interrupts

The upper and lower bounds limits are (1)77001₈ and (1)1, respectively. Thus, in particular, any attempt to store into location 0 in bank 1 will cause a bounds interrupt. When a bounds interrupt occurs, TEMP in the system COMMON will contain the contents of (0)0 after the interrupt. Thus TEMP will contain an unconditional bank jump to the address where the interrupt was detected.

B. Registers Altered by Execution of GYRO-II Macros

The following table notes which registers are altered in the CDC-3600 implementation of GYRO II when the corresponding macros are executed. If a macro does not appear in the table, it does not alter any registers.

MACRO	REGISTERS ALTERED	MACRO	REGISTERS ALTERED
A	A	AJ, AK	A, B5
B	B6	BDT	B5
BIT X,N,T	X	BITJ X,N,T	X if T is nonblank
BJT	AU	BKT	AL
D	B6, Q	DJ,DK	B6, Q, B5
DBT	B6		
DF	B4,Q,B5	DFJ, DFK	B4, B5, Q
DJT	AU	DKT	AL
DP	B6, Q	DPJ, DPK	B5, B6, Q,AL
DS	B6, Q	DSJ, DSK	B5, B6, Q
E X,Y	X		
EOL	B4		
GETAV	B6, Q	GETAV D	B5, Q
I	Q	IJ, IK	B5, B6, Q
INC	B4		
INITIAL	All except B5 and B6		
IP	Q, AL	IPJ, IPK	Q, AL, B5
IS	Q	ISJ	Q, AL
J JAD,X, EQ	Q		
NE			

MACRO	REGISTERS ALTERED	MACRO	REGISTERS ALTERED
JBL	B6	JDL	B5
JBT	B6	JDT	B5
JJL	B4, AU	JJS	B4
JKL	B4, AU	JKS	B4
JKT	AL		
JRR	B4		
KBT	B6	KDT	B5
KJT	AU		
KBL	B6	KDL	B5
KJL	B4, AU	KKL	B4, AL
L	A	LJ, LK	A, B4
LAST	B4, B5		
LQ	Q	LQJ, LQK	Q, B4
LM	A	LMJ, LMK	B4, AU or AL
LR	B6, A, Q	LRJ, LRK	B5, B6, A, Q
LU	B6, A, Q	LUJ, LUK	B5, B6, A, Q
P	Q, AL	PJ, PK	B5, Q, AL
R	B6, Q	RJ, RK	B5, B6, Q
RELINK	B5, A		
REST A	B4, B5, A, Q	REST Q	B4, B5, A, Q
REST B	B4, B6, Q	REST D	B4, B5, Q

(For other parameter combinations the union of the sets of registers indicated above will be altered.)

RETAV	Q		
RLM	B5, B6, Q		
SJ, SK	B4	SQJ, SQK	B4
SAV	Same as REST		
T X, Y	Y		
U	B6, Q	UJ, UK	B5, B6, Q
UF	B4, B5, A	UFJ, UFK	B4, B5, B6, A

MACRO	REGISTERS ALTERED	MACRO	REGISTERS ALTERED
WAIT	A, Q		
ZP S	A	ZP S,Q	Q
ZPD	B4, Q		
ZPROP	B1,B2,B3,B4,B5		
ZS	Q		
(blank address)			

C. Expansions of GYRO II Macros for the CDC-3600

In the following macro expansions, it will sometimes be convenient to define one macro using another. The appearance of one macro inside another does not necessarily mean that the actual macro was coded that way. The actual sequence of machine instructions is, however, independent of whether macros or actual assembly-language instructions were used.

The indirect-upper and indirect-lower form of a macro-instruction when written together at the head of a definition will have the same expansions, except for the terms flagged with a †. These operands are given for indirect-upper and should be replaced by the lower address form of the instruction. For instance, if LIU is flagged it should be replaced with LIL for indirect-lower addressing.

A	<u>A</u>	<u>S</u>	<u>AJ,AK</u>	<u>S</u>	
	LDA	S	LIU†	S,5	
	SAL	0,6	LDA	0,5	
	SIL	S,6	SAL	0,6	
			SIL	0,6,5	
B	<u>B</u>		<u>BDT</u>		
	LIU	B6PDL,6	RXT	B6,B5	
BIT	<u>BIT</u>	<u>X,N,T</u>	<u>BITJ</u>	<u>JAD,X,N,S,T</u>	
	ZBJP,T	X,N,*+ 1	ZBJP,T	X,N,JAD	for S = CL
			or NBJP,T	X,N,JAD	for S = ST
BJS	<u>BJS</u>		<u>BKS</u>		
	SIU	S,6	SIL	S,6	

	<u>BJT</u>			<u>BKT</u>	
	RXT	B6,AU		RXT	B6,AL
CALL	<u>CALL</u>	<u>SUB</u>		<u>CALL</u>	<u>DUMP</u>
	BRTJ	(\$)SUB,,*		ZRO	0
	EXT	SUB			
D	<u>D</u>	<u>S</u>		<u>DJ,DK</u>	<u>S</u>
	GETAV			LIU [†]	S,5
+	LDQ	S		GETAV	
	STQ	0,6	+	LDQ	0,5
	SIL	S,6		STQ	0,6
				SIL	0,6,5
DBT	<u>DBT</u>			<u>DJT</u>	
	RXT	B5,B6		RXT	B5,AU
	<u>DKT</u>				
	RXT	B5,AL			
DF	<u>DF</u>	<u>S</u>		<u>DFJ,</u>	<u>S</u>
	LIL	AVAILSPA,4		<u>DFK</u>	
	LDQ	0,4		LIU [†]	S,6(or 0,6,6 if S is blank)
	RXT	QL,B5		DF	
	LDQ	0,5			
	STQ	AVAILSPA			
	LDQ	S(or 0,6 if S is blank)			
	STQ	0,4			
	LDQ	LISTBIT			
	RXT	B4,QL			
	RXT	B5,QU			
	STQ	S(or 0,6 if S is blank)			
	STA	0,5			
DJS	<u>DJS</u>	<u>S</u>		<u>DKS</u>	<u>S</u>
	SIU	S,5		SIL	S,5

	<u>DP</u>	<u>S</u>		<u>DPJ,DPK</u>	<u>S</u>
	GETAV			LIU [†]	S,5
	LDQ	S		GETAV	
	STQ	0,6	+	LDQ	0,5
	STA	S		STQ	0,6
	SIL	S,6		RXT	B6,AL
				STA	0,5
DS	<u>DS</u>	<u>S</u>		<u>DSJ,DSK</u>	<u>S</u>
	GETAV			LIU [†]	S,5
	LDQ	S		GETAV	
	STQ	0,6		LDQ	0,5
	STA	S		STQ	0,6
				STA	0,5
E	<u>E</u>	<u>X,Y</u>		<u>E</u>	<u>X,Y</u>
	ENX	Y for X = A,Q		ENI	X,Y for X = 1-6
EOL	<u>EOL</u>	<u>S</u>			
	ENI	S,4			
	SIL	NOAVAILS,4			
GETAV	<u>GETAV</u>			GETAV	D
	LIL	AVAILSPA,6		LIL	AVAILSPA,5
	LDQ	0,6		LDQ	0,5
	STQ	AVAILSPA		STQ	AVAILSPA
HELP	<u>HELP</u>	<u>((MESSAGE))</u>			
	UNBOUND				
	BRTJ	(\$)SPOTCHK,,*			
	SLJ	*+4			
	OO	(*)*+2			
	OCT	0			
	VFD	06/60,H90/MESSAGE			
	BOUND	(((1)1), ((1)77001B),*,GBIA)			

<u>I</u>	<u>I</u>	<u>S</u>	<u>IJ,IK</u>	<u>S</u>
	LDQ	S	LIU†	S,5
	STQ	0,6	LDQ	0,5
	SIL	S,6	STQ	0,6
			RXT	B6,QL
			STQ	0,5

<u>INC</u>	<u>INC</u>	<u>S</u>	<u>INC</u>	<u>S,N</u>
	LIL	S,4	LIL	S,4
	INI	10,4	INI	N,4
	SIL	S,4	SIL	S,4

<u>INITIAL</u>	<u>INITIAL</u>	
	ENA	0
	STA	ZERO
	BOUND	((1)1),((1)77001B),*,GBIA)
	LDA	NONE
	STA	NOAVAILS
	ENI	5,1
	ENI	0,2
	ENI	1,3
	ENI	0,4
	ENI	1,5
	XMIT,AUG	(*)*+2,GBIA
	SLJ	P1
	UBJP	** INTERRUPT ROUTINE
	XMIT	(0)0,TEMP
	XMIT	(*)*+2,(0)0
	SLJ	GBIA END OF INTERRUPT ROUTINE
	UBJP	(\$)NOAVAILS,, \$

P2	NOP	** RECURSIVE RETURN JUMP ROUTINE
	LDA	**
	SUB	G2RECUR+24
	RGJP,EQ	AU,0,G2RECUR+11
	LIL	AVAILSPA,4

```
LDA      0,4
STA      AVAILSPA
LDA      RCPDL
STA      0,4
LDA      G2RECUR,7
STA      RCPDL
SIL      RCPDL,4
LIL      AVAILSPA,4
LDA      0,4
STA      AVAILSPA
LDA      RCPDL
STA      0,4
RXT,CR   B4,AL
STA      RCPDL
LIL      AVAILSPA,4
LDA      0,4
STA      AVAILSPA
LDA      RCPDL
STA      0,4
LDA      G2RECUR
STA      RCPDL
SIL      RCPDL,4
LDA      TEMP
RTJ      G2RECUR,7
STA      TEMP
LDA      RCPDL
RXT,CR   AU,AL
SAL      G2RECUR+23
LIL      RCPDL,4
LDA      0,4
```

```

STA      RCPDL
LDA      G2RECUR+23
RGJP,EQ  AL,0,G2RECUR+18
LDA      TEMP
SLJ      **
ZRO      G2RECUR+17
ZRO      0
+ STA      TEMP
+ LIU      RCPDL,4
RGJP,EQ  B4,0,G2RECUR+32
LIL      RCPDL,4
LDA      0,4
STA      RCPDL
LDA      AVAILSPA
STA      0,4
SIL      AVAILSPA,4
SLJ      G2RECUR+26
+ LIL      RCPDL,4
LDA      0,4
STA      RCPDL
RXT,CR   AU,AL
SAU      G2RECUR+39
LDA      AVAILSPA
STA      0,4
SIL      AVAILSPA,4
LIL      RCPDL,4
LDA      0,4
STA      RCPDL
LDA      AVAILSPA
STA      0,4
SIL      AVAILSPA,4
SLJ      **   END OF RECURSIVE JUMP ROUTINE

```

NONE	UBJP	(\$)TEN,, \$		
	OCT	4000000000000000		
P1	ENA	10		
	STA	TEN		
	LDA	NONE+1		
	ENI	40,1		
	ENI	0,4		
	ENI	0,2		
	XMIT,AUG	(*)P2, (*)G2RECUR		
	STA	LISTBIT		
	ENA	GYROSAV		
	ENI	0,5		
	SAL	AVAILSPA		
+	INA	1		
	STA	GYROSAV,5		
	ISK	19998,5		
	SLJ	*-1		
	SIL	GYROSAV+19999		
	SIL	APDL		
	SIL	QPDL		
	SIL	B6PDL		
	SIL	B5PDL		
IP	<u>IP</u>	<u>S</u>	<u>IPJ,IPK</u>	<u>S</u>
	LDQ	S	LIU [†]	S,5
	STQ	0,6	LDQ	0,5
	RXT	B6,AL	STQ	0,6
	STA	S	RXT	B6,AL
			STA	0,5
IS	<u>IS</u>	<u>S</u>	<u>ISJ</u>	<u>S</u>
	LDQ	S	LIU	S,5
	STQ	0,6	LDQ	0,5

	STA	S	STQ	0,6
			STA	0,5
J	<u>J</u>	<u>JAD</u>	<u>J</u>	<u>JAD,X,TERM</u>
	SLJ	JAD	RGJP,EQ	XL,0,JAD
			or RGJP,EQ	for X = A,Q
			or RGJP,EQ	B6,0,JAD
			or RGJP,EQ	for X = B
				B5,0,JAD
				for X = D
	<u>J</u>	<u>JAD,X,ATOM</u>	<u>J</u>	<u>JAD,X,Y</u>
	XJP,PL	JAD for X = A,Q	XJP,Y	JAD
				X = A,Q
or +	SSK	0,6 (5) for X = B(D)		
or +	SSK	X		
		for X = memory loc.		
	<u>J</u>	<u>JAD,X,EQ</u>	<u>J</u>	<u>JAD,X,NE</u>
	LDQ	X (Only for X = Q)	LDQ	X
				(Only for X = Q)
	ROP,XOR	A,Q,Q	ROP,XOR	A,Q,Q
	QJP,ZR	JAD	QJP,NZ	JAD
	<u>J</u>	<u>JAD,X,Y,Z</u>		
	RGJP,Y	X,Z,JAD		
JBL	<u>JBL</u>	<u>S</u>	<u>JDL</u>	<u>S</u>
	LIU	S,6	LIU	S,5
or	LIU	0,6,6(if S is blank)	LIU	0,5,6
				(if S is blank)
JBT	<u>JBT</u>		<u>JDT</u>	
	RXT	AU,B6	RXT	AU,B5

JKT	<u>JKT</u>		
	RXT	AU,AL	
JJL	<u>JJL</u>	<u>S</u>	<u>JJL</u>
	LIU	S,4	LIU 0,4,6
	RXT	B4,AU	RXT B4,AU
JJS	<u>JJS</u>	<u>S</u>	<u>JJS</u>
	RXT	AU,B4	RXT AU,B4
	SIU	S,4	SIU 0,4,6
JKL	<u>JKL</u>	<u>S</u>	<u>JKL</u>
	LIU	S,4	LIU 0,4,6
	RXT	B4,AL	RXT B4,AL
JKS	<u>JKS</u>	<u>S</u>	<u>JKS</u>
	RXT	AU,B4	RXT AU,B4
	SIL	S,4	SIL 0,4,6
JR	<u>JR</u>	<u>JAD</u>	
	RTJ	JAD	
	REM	FORCE NEXT INSTRUCTION UPPER	
JRR	<u>JRR</u>	<u>JAD</u>	
	ENI	JAD,4	
	SIL	G2RECUR,4	
	STA	TEMP	
	RTJ	G2RECUR	
	REM	FORCE NEXT INSTRUCTION UPPER	
KBT	<u>KBT</u>		<u>KDT</u>
	RXT	AL,B6	RXT AL,B5
KJT	<u>KJT</u>		
	RXT	AL,AU	

KBL	<u>KBL</u>	<u>S</u>	KDL	<u>S</u>
	LIL	S,6	LIL	S,6
or	LIL	0,6,6 if S is blank.	or LIL	0,6,6 if S is blank.
KJL	<u>KJL</u>	<u>S</u>	<u>KJL</u>	
	LIL	S,4	LIL	0,6,6
	RXT	B4,AU	RXT	B4,AU
KJS	<u>KJS</u>	<u>S</u>	<u>KKS</u>	<u>S</u>
	SAU	S	SAL	S
or	SAU	0,6 if S is blank.	or SAL	0,6 if S is blank.
KKL	<u>KKL</u>	<u>S</u>	<u>KKL</u>	
	LIL	S,4	LIL	0,4,6
	RXT	B4,AL	RXT	B4,AL
L	<u>L</u>	<u>S</u>	<u>LJ,LK</u>	<u>S</u>
	LDA	S	LIU [†]	S,4
or	LDA	0,6 if S is blank.	LDA	0,4
	<u>LJ,LK</u>			
	LIU [†]	0,4,6		
	LDA	0,4		
LAST	<u>LAST</u>	<u>S</u>	<u>LAST</u>	
	ENI	S,5	RXT	B6,B5
P1	LIL	0,4,5	P1 LIL	0,4,5
	RGJP,EQ	B4,0,P2	RGJP,EQ	B4,0,P2
	RXT	B4,B5	RXT	B4,B5
	SLJ	P1	SLJ	P1
P2	BSS	0	P2 BSS	0
LQ	<u>LQ</u>	<u>S</u>	<u>LQJ,LQK</u>	<u>S</u>
	LDQ	S	LIU [†]	S,4
or	LDQ	0,5 if S is blank.	LDQ	0,4

	<u>LQJ, LQK</u>			
	LIU [†]	0,4,5		
	LDQ	0,4		
LINE	<u>LINE</u>	<u>I</u>		
	UNBOUND			
	SIU	*+5,6		
	ENI	2,6 if I = 2		
or	ENI	0,6 if I ≠ 2		
	BRTJ	(\$)OUTCONT,,*		
	BOUND	((1)1),((1)77001B),*,GBIA)		
	ENI	**6		
LM	<u>LM</u>	<u>S</u>	<u>LMJ</u>	<u>S</u>
	ENA	S	ENI	S,4
			RXT	B4,AU
	<u>LMK</u>	<u>S</u>		
	ENI	S,4		
	RXT	B4,AL		
LR	<u>LR</u>	<u>S</u>	<u>LRJ, LRK</u>	<u>S</u>
	LDA	S	LIU [†]	S,5
	RXT	AL,B6	LDA	0,5
	LDQ	0,6	RXT	AL,B6
	STQ	S	LDQ	0,6
			STQ	0,5
LU	<u>LU</u>	<u>S</u>	<u>LUJ, LUK</u>	<u>S</u>
	LR	S	LRJ [†]	S
	RETAV		RETAV	
NOW	<u>NOW</u>			
	UNBOUND			
	SIU	*+5,6		
	ENI	3,6		
	BRTJ	(\$)OUTCONT,,*		

	BOUND	(((1)1),((1)77001B),*,GBIA)		
	ENI	** ,6		
P	<u>P</u>	<u>S</u>	<u>PJ,PK</u>	<u>S</u>
	LDQ	S	LIU [†]	S,5
	RXT	QL,AL	LDQ	0,5
	STA	S	RXT	QL,AL
			STA	0,5
PAGE	<u>PAGE</u>			
	UNBOUND			
	SIU	*+5,6		
	ENI	1,6		
	BRTJ	(\$)OUTCONT,,*		
	BOUND	(((1)1),((1)77001B),*,GBIA)		
	ENI	** ,6		
PRINT	PRINT	((MESSAGE))		
	UNBOUND			
	BRTJ	(\$)SPOTCHK,,*		
	SLJ	*+3		
	OO	(*)*+1		
	VFD	06/60,H90/MESSAGE		
	BOUND	(((1)1),((1)77001B),*,GBIA)		
R	<u>R</u>	<u>S</u>	<u>RJ,RK</u>	<u>S</u>
	LIL	S,6	LIU [†]	S,5
	LDQ	0,6	LIL	0,6,5
	STQ	S	LDQ	0,6
			STQ	0,5
REED	<u>REED</u>	<u>CWA</u>	<u>REED</u>	<u>CWA,LU</u>
	READ	INP,CWA,*	READ	LU,CWA,*
	<u>REED</u>	<u>CWA,LU,INT</u>	<u>REED</u>	<u>CWA,INT</u>
	READ	LU,CWA,*,INT	READ	INP,CWA,*,INT

RELINKRELINK

ENA GYROSAV
 ENI 0,5
 SAL AVAILSPA
 + INA 1
 SAL GYROSAV,5
 ISK 19998,5
 SLJ *-1
 SIL GYROSAV+19999
 SIU APDL
 SIU B6PDL
 SIU QPDL
 SIU B5PDL

RESTRESTARESTQ

UF APDL

UF QPDL

RXT A,Q

RESTBRESTD

LIL B6PDL,4

LIL B5PDL,4

LIU B6PDL,6

LIU B5PDL,5

LDQ 0,4

LDQ 0,4

STQ B6PDL

STQ B5PDL

LDQ AVAILSPA

LDQ AVAILSPA

STQ 0,4

STQ 0,4

SIL AVAILSPA,4

SIL AVAILSPA,4

RESTRESTA,B,Q,D

UF QPDL

Same as REST

STA TEMP

UF APDL

REST D

REST B

LDQ TEMP

<u>REST</u>	<u>A,B,Q</u>	<u>REST</u>	<u>A,Q,D</u>
UF	QPDL	UF	QPDL
STA	TEMP	STA	TEMP
UF	APDL	UF	APDL
REST	B	REST	D
LDQ	TEMP	LDQ	TEMP
<u>REST</u>	<u>A,B</u>	<u>REST</u>	<u>A,D</u>
UF	APDL	UF	QPDL
REST	B	REST	D
<u>REST</u>	<u>A,Q</u>	<u>REST</u>	<u>B,Q,D</u>
UF	QPDL	UF	QPDL
STA	TEMP	REST	D
UF	APDL	REST	B
LDQ	TEMP	RXT	A,Q
<u>REST</u>	<u>B,Q</u>	<u>REST</u>	<u>Q,D</u>
UF	QPDL	UF	QPDL
REST	B	REST	D
RXT	A,Q	RXT	A,Q
<u>REST</u>	<u>A,B,D</u>	<u>REST</u>	<u>B,D</u>
UF	APDL	REST	D
REST	D	REST	B
REST	B		

RETAV

RETAV

LDQ

AVAILSPA

STQ

0,6

SIL

AVAILSPA,6

REXIT

REXIT

SLJ

G2RECUR+25

RITE	<u>RITE</u>	<u>CWA</u>	RITE	<u>CWA, LU</u>
	WRITE	OUT, CWA, *	WRITE	LU, CWA, *
	<u>RITE</u>	<u>CWA, LU, INT</u>	<u>RITE</u>	<u>CWA, , INT</u>
	WRITE	LU, CWA, *, INT	WRITE	OUT, CWA, *, INT
RLM	<u>RLM</u>	<u>S</u>		
	ENI	S, 6 (Only if S is nonblank)		
	LDA	0, 6		
	RGJP, EQ	AL, 0, LAST+1		
P1	RXT	AL, B5		
	LDA	0, 5		
	RGJP, EQ	AL, 0, LAST		
	RXT	B5, B6		
	SLJ	P1		
LAST	SIL	0, 0, 6		
S	<u>S</u>	<u>S</u>	<u>SJ, SK</u>	<u>S</u>
	STA	S	LIU [†]	S, 4
or	STA	0, 6 if S is blank	STA	0, 4
	<u>SJ, SK</u>	<u>S</u>		
	LIU [†]	0, 4, 6		
	STA	0, 4		
SAV	<u>SAV</u>	<u>A</u>	<u>SAV</u>	<u>Q</u>
	DF	APDL	RXT	Q, A
			DF	QPDL
	<u>SAV</u>	<u>B</u>	<u>SAV</u>	<u>D</u>
	LIL	AVAILSPA, 4	LIL	AVAILSPA, 4
	LDQ	0, 4	LDQ	0, 4
	STQ	AVAILSPA	STQ	AVAILSPA
	LDQ	B6PDL	LDQ	B5PDL
	STQ	0, 4	STQ	0, 4
	SIU	B6PDL, 6	STQ	B5PDL, 5

	SIL	B6PDL,4	SIL	B5PDL,4
	<u>SAV</u>		<u>SAV</u>	<u>A,B,Q,D</u>
	STQ	TEMP	Same as SAV	
	SAV	B		
	DF	APDL		
	SAV	D		
	LDA	TEMP		
	DF	QPDL		
SQ	<u>SQ</u>	<u>S</u>	<u>SQJ,SQK</u>	<u>S</u>
	STQ	S	LIU [†]	S,4
or	STQ	0,5 if S is blank	STQ	0,4
	<u>SQJ,SQK</u>			
	LIU [†]	0,4,5		
	STQ	0,4		
	<u>SAV</u>	<u>A,B,Q</u>	<u>SAV</u>	<u>A,Q,D</u>
	STQ	TEMP	STQ	TEMP
	SAV	B	SAV	D
	DF	APDL	DF	APDL
	LDA	TEMP	LDA	TEMP
	DF	QPDL	DF	QPDL
	<u>SAV</u>	<u>A,B</u>	<u>SAV</u>	<u>A,D</u>
	SAV	B	DF	APDL
	DF	APDL	SAV	D
	<u>SAV</u>	<u>A,Q</u>	<u>SAV</u>	<u>B,Q,D</u>
	STQ	TEMP	RXT	Q,A
	DF	APDL	SAV	D
	LDA	TEMP	SAV	DB
	DF	QPDL	DF	QPDL
	<u>SAV</u>	<u>B,Q</u>	<u>SAV</u>	<u>Q,D</u>
	RXT	Q,A	RXT	Q,A
	SAV	B	DF	QPDL

	DF	QPDL	SAV	D
	<u>SAV</u>	<u>A,B,D</u>	<u>SAV</u>	<u>B,D</u>
	SAV	B	SAV	B
	DF	APDL	SAV	D
	SAV	D		
START	<u>START</u>	<u>EP</u>		
	UBJP	**		
	ENTRY	EP		
	BLOCK			
	COMMON	AVAILSPA,NOAVAILS,PROPLIST(64),TEMP,APDL, B6PDL,QPDL		
	COMMON	B5PDL,ZERO,TEN,LISTBIT,GBIA(5), GYROSAV(20000)		
	COMMON	G2RECUR (40), RCPDL		
	EXT	SENTRY		
T	<u>T</u>	<u>X,Y</u>		
	RXT	X,Y		
TERM	<u>TERM</u>	<u>S</u>	<u>TERM</u>	<u>D</u>
	SIL	S,0	SIL	0,0,5
or	SIL	0,0,6		
		if S is blank.		
U	<u>U</u>	<u>S</u>	<u>UJ,UK</u>	<u>S</u>
	LIL	S,6	LIU [†]	S,5
	LDQ	0,6	LIL	0,6,5
-	STQ	S	LDQ	0,6
	RETAV		- STQ	0,5
			RETAV	
UF	<u>UF</u>	<u>S</u>	<u>UFJ,UFK</u>	<u>S</u>
	LDA	S	LIU [†]	S,6
		(or 0,6 if S is blank)		(or 0,6,6 if S is blank)
		UF		
	RXT	AU,5		
	RXT	AL,4		

	LDA	0,4		
	STA	S (or 0,6 if S is blank)		
	LDA	AVAILSPA		
	STA	0,4		
	SIL	AVAILSPA,5		
	LDA	0,5		
	SIL	0,4,5		
WAIT	<u>WAIT</u>	<u>LU</u>	<u>WAIT</u>	<u>LU,EF</u>
P1	STATUS	LU	P1 STATUS	LU
	QJP,MI	P1	QJP,MI	P1
			NBJP	Q,35,EF
ZP	<u>ZP</u>	<u>S</u>	<u>ZP</u>	<u>S,Q</u>
	LDA	S	LDQ	S
	RXT,CR	AL,AL	RXT,CR	QL,QL
	STA	S	STQ	S
	<u>ZP</u>		<u>ZP</u>	<u>(,Q)</u>
	LDA	0,6	LDQ	0,6
	RXT,CR	AL,AL	RXT,CR	QL,QL
	STA	0,6	STQ	0,6
ZPD	<u>ZPD</u>	<u>S</u>	<u>ZPD</u>	
	LIL	AVAILSPA,4	LIL	AVAILSPA,4
	LDQ	0,4	LDQ	0,4
	STQ	AVAILSPA	STQ	AVAILSPA
	LDQ	S	LDQ	0,6
	STQ	0,4	STQ	0,4
	RXT,CR	B4,QL	RXT,CR	B4,QL
	STQ	S	STQ	0,6
ZPROP	<u>ZPROP</u>			
	ENI	64,1		
	ENI	0,2		

	ENI	0,3		
	ENI	0,4		
	ENI	1,5		
	XMIT,AUG	ZERO,PROPLIST		
ZS	<u>ZS</u>	<u>S</u>	<u>ZS</u>	
	XMIT	ZERO,S	ENQ	0
			STQ	0,6

REFERENCES

1. *IPL-V Manual*, Second Edition, Prentice Hall, Englewood Cliffs, N.J. (Jan 1964).
2. *LISP 1.5 PROGRAMMER'S MANUAL*, MIT Press, Cambridge (1963).
3. *The NU-SPEAK System*, NYO-1480-9 (Nov 1964).
4. Robinson, George, *GYRO I Macro-programmed List Processing System*, AMD Tech. Memo. No. 74 (Sept 1964). (Internal publication, Argonne National Laboratory.)