LINUXPAK

(Do original UPAK, de Bill Milner)

ADAC – 2

CHIL - 10

SCAN - 43

LEMO - 52

DAMM - 72

ADAC - DATA ACQUISITION WITH THE EVENT-HANDLER

SECTION CONTENTS

010 HOW TO ASSEMBLE A PROGRAM
020 GENERAL FEATURES OF THE SYSTEM
025 READOUT & CONTROL OF UP TO 4 CRATES WITH ONE EVENT HANDLER
030 BASIC INSTRUCTION LIST AND DEFINITIONS
040 MACRO INSTRUCTION LIST AND DEFINITIONS
050 DIRECTIVES TO THE ASSEMBLER ("EQUATE", LOOP, ENDLOOP)
060 ASSEMBLY-TIME VARIABLES
065 STATEMENT LABELS
070 EXPRESSIONS
080 SYNTAX RULES
090 PROGRAMMING EXAMPLES
110 HOW TO SCREW UP WITHOUT REALLY TRYING
120 INSTRUCTION SET (BIT-PATTERN)

250.030 BASIC INSTRUCTION LIST AND DEFINITIONS

```
INSTRUCTION          ;MEANING OR DESCRIPTION
NOP                 ;NO-OP
SETB                ;SET BUSY - front panel output
CLRB                ;CLR BUSY - front panel output

NAF   N,A,F          ;Do NAF  - data to or from CA1 implied
NAF  (P),N,A,F       ;Do NAF  - (proceed)
NAF  (S),N,A,F       ;Do NAF  - (short   CAMAC cycle - No S2)
NAF  (N),N,A,F       ;Do NAF  - (shorter CAMAC cycle - No S1 or S2)
NAF  (PS),N,A,F      ;Do NAF  - (short   CAMAC cycle & proceed)
NAF  (PN),N,A,F      ;Do NAF  - (shorter CAMAC cycle & proceed)

MOV  CAX,PAT         ;MOV contents of CAX to Pattern  Reg (PAT)
MOV  PAT,CAX         ;MOV contents of PAT to CAX (16 bits)
MOV  CAX,TXR         ;MOV contents of CAX to Transfer Reg (TXR)
MOV  UCAX,CAX        ;MOV HI 8-bits to LO 8-bits (zero bits 9 - 16)
MOV  #,TXR          ;MOV # to TXR (16 bits)
MOV  #,CAX          ;MOV # to CAX (16 bits)

OUT  CAX         ;Output CAX (MOV to TXR & XMIT to FIFO)(24 bits)
OUT  #           ;Output #   (MOV to TXR & XMIT to FIFO)(16 bits)
OUT  UCAX       ;MOV HI 8-bits to LO 8-bits of TXR & XMIT; (HI 8-bits of TXR set to
zero)
OUT  PAT         ;MOV LO 16-bits of PAT to TXR & XMIT
OUT  SPEC       ;Special OUT - not yet supported.

MERG #        ;MOV bits 13-16 of # to bits 13-16 of TXR & XMIT;    (bits 1-12
unchanged)

INCX             ;Increment CAMAC register CAX
DECX             ;Decrement CAMAC register CAX
```

```
SPLX              ;"Split"   CAMAC register CAX
STOX AD             ;Store CAX in AUX memory at address AD
RECX AD             ;Recall AUX memory location AD & store in CAX
-------------------------------------------------------------------
CAX  denotes CA  (implies CA1),  CA1,  CA2,  CA3  or CA4
INCX denotes INC (implies INC1), INC1, INC2, INC3 or INC4
DECX denotes DEC (implies DEC1), DEC1, DEC2, DEC3 or DEC4
SPLX denotes SPL (implies SPL1), SPL1, SPL2, SPL3 or SPL4
```

250.030 BASIC INSTRUCTION LIST (CONTINUED)

```
SKIP EXX.ANY.#      ;Skip next INS if any set bits in EXX  & # match
SKIP CAX.ANY.#      ;Skip next INS if any set bits in CAX  & # match
SKIP PAT.ANY.#     ;Skip next INS if any set bits in PAT  & # match
SKIP UPAT.ANY.#     ;Skip next INS if any set bits in UPAT & # match

SKIP EXX.NONE.#     ;Skip next INS if no  set bits in EXX  & # match
SKIP CAX.NONE.#     ;Skip next INS if no  set bits in CAX  & # match
SKIP PAT.NONE.#    ;Skip next INS if no  set bits in PAT  & # match
SKIP UPAT.NONE.#    ;Skip next INS if no  set bits in UPAT & # match

SKIP CAX.LT.#       ;Skip next INS if contents of CAX < # (16 bits)
SKIP PAT.LT.#      ;Skip next INS if contents of PAT < # (16 bits)
SKIP CAX.GT.#       ;Skip next INS if contents of CAX > # (16 bits)
SKIP PAT.GT.#      ;Skip next INS if contents of PAT > # (16 bits)

BRU   NAME         ;Branch to Destination NAME

SPB   DEST         ;SPB = STORE POSITION and BRANCH - Similar to
                     ;CALL in Fortran. DEST same form as for BRU.

BRUR              ;Return from subroutine

DLAY  #            ;Wait for #*(0.1 micro-sec) up to 409.6 micro-sec

-------------------------------------------------------------------
EXX denotes EX (implies EX1), EX1, EX2, EX3 or EX4
-------------------------------------------------------------------
```

250.040 MACRO INSTRUCTION LIST AND DEFINITIONS

```
IF(EXX.ANY.#)ADDR     ;If set bits in EXX  match any  in #, GO TO ADDR
IF(CAX.ANY.#)ADDR     ;If set bits in CAX  match any  in #, GO TO ADDR
IF(PAT.ANY.#)ADDR     ;If set bits in PAT  match any  in #, GO TO ADDR
IF(UPAT.ANY.#)ADDR    ;If set bits in UPAT match any  in #, GO TO ADDR

IF(EXX.NONE.#)ADDR    ;If set bits in EXX  match none in #, GO TO ADDR
IF(CAX.NONE.#)ADDR    ;If set bits in CAX  match none in #, GO TO ADDR
IF(PAT.NONE.#)ADDR    ;If set bits in PAT  match none in #, GO TO ADDR
IF(UPAT.NONE.#)ADDR   ;If set bits in UPAT match none in #, GO TO ADDR

IF(CAX.LT.#)ADDR      ;If contents of CAX < # (16 bits) GO TO ADDR
IF(PAT.LT.#)ADDR      ;If contents of PAT < # (16 bits) GO TO ADDR

IF(CAX.GT.#)ADDR      ;If contents of CAX > # (16 bits) GO TO ADDR
```

```
IF(PAT.GT.#)ADDR      ;If contents of PAT > # (16 bits) GO TO ADDR

SPAT  N,A,F           ;Read 24-bit WD and save in Pattern Reg (PAT)
CSPAT C,N,A,F         ;Read 24-bit WD and save in Pattern Reg (PAT)

ROUT  N,A,F           ;Read data and XMIT to FIFO
CROUT C,N,A,F         ;Read data and XMIT to FIFO

RSTO  N,A,F,AD        ;Read into CA1 & store LO-16 in AUX memory at AD
CRSTO C,N,A,F,AD      ;Read into CAC & store LO-16 in AUX memory at AD
INCX  AD              ;Increment AUX memory AD using CAMAC reg CAX
DECX  AD              ;Decrement AUX memory AD using CAMAC reg CAX
-------------------------------------------------------------------
EXX  denotes EX  (implies EX1),  EX1,  EX2,  EX3  or EX4
CAX  denotes CA  (implies CA1),  CA1,  CA2,  CA3  or CA4
INCX denotes INC (implies INC1), INC1, INC2, INC3 or INC4
DECX denotes DEC (implies DEC1), DEC1, DEC2, DEC3 or DEC4

All NAF & CNAF generating instructions support an optional first-field
(S), (P), (N), (PS), or (PN). (See first page of SEC# 250.030)

All NAF & CNAF generating instructions support an optional last-field
T which denotes a Totalizer Module register no. (T = 0,1,2,3)
-------------------------------------------------------------------
```

250.040 MACRO INSTRUCTION LIST (EXPANSIONS)

Each macro-instruction generates two or more basic instructions as indicated below.

```
MACRO-INSTRUCTION      BASIC INSTRUCTIONS GENERATED BY ASSEMBLER
-----------------      -----------------------------------------
SPAT   N,A,F     NAF   N,A,F
                   MOV   CA1,PAT


CSPAT  C,N,A,F         CNAF  C,N,A,F       (C must be 1,2,3 or 4)
                 MOV   CAX,PAT       (CAX=CA1,CA2,CA3 or CA4)

ROUT   N,A,F           NAF   N,A,F
                 OUT   CA1

CROUT  C,N,A,F         CNAF  C,N,A,F       (C must be 1,2,3 or 4)
                 OUT   CAX           (CAX=CA1,CA2,CA3 or CA4)

RSTO   N,A,F,AD        NAF   N,A,F
                 STO   AD

CRSTO  C,N,A,F,AD      CNAF  C,N,A,F       (C must be 1,2,3 or 4)
                 STOX  AD            (STOX=STO1,STO2,STO3 or STO4)

INCX   AD              RECX  AD            (uses CAMAC reg CAX)
                 INCX                (incremented value in CAX)
                 STOX  AD

DECX   AD              RECX  AD            (uses CAMAC reg CAX)
```

```
                DECX            (decremented value in CAX)
                STOX  AD

IF(REG.OP.MSK)ADDR     SKIP  REG.IOP.MSK   (for OP = ANY ,NONE)
                BRU   ADDR         (    IOP= NONE,ANY )

IF(REG.LT.VAL)ADDR     SKIP  REG.GT.VAL-1  (REG=CA1,CA2,CA3,CA4 or PAT)
                BRU   ADDR

IF(REG.GT.VAL)ADDR     SKIP  REG.LT.VAL+1  (REG=CA1,CA2,CA3,CA4 or PAT)
                BRU   ADDR
```

INCX denotes INC (implies INC1), INC1, INC2, INC3 or INC4
DECX denotes DEC (implies DEC1), DEC1, DEC2, DEC3 or DEC4

All NAF & CNAF generating instructions support an optional first-field (S), (P), (N), (PS), or (PN). (See first page of SEC# 250.030)

250.050 DIRECTIVES TO THE ASSEMBLER ("EQUATE", LOOP, ENDLOOP)

SYM=EXPRESSION Directs assembler to evaluate EXPRESSION and assign the resulting value to SYM.

LABL LOOP NLOOP Directs the assembler to "replicate" the set of statements bounded by the LOOP- and ENDLOOP-directives. Do it NLOOP times.

ENDL ENDLOOP Defines End-of-Loop.

250.060 ASSEMBLY-TIME VARIABLES

Symbols defined by the "Equate Directive" (SYM=EXPRESSION) are called assembly time variables or just assembly variables. Such variables may be re-defined without restriction. Note: These are not run-time variables! - THERE ARE NO RUN-TIME VARIABLES! Assembly variables must always be defined in terms of numbers and/or previously defined assembly variables. Defining expressions may include "Bit-Lists".

250.065 STATEMENT LABELS

The label on the ENDLOOP directive as well as any other labels Defined between the LOOP and ENDLOOP directives are undefined outside of the loop and may be used in other loops. All other statement labels, including those on the LOOP directive Are defined to all parts of the program and, therefore, must be unique.

* * * * LABELING OF THE "EQUATE" DIRECTIVE IS NOT ALLOWED * * * *

250.070 EXPRESSIONS

The assembler supports simple expressions which are evaluated left to right. REPEAT!! EVALUATED! LEFT! TO! RIGHT! Let "V" represent a single value (number or previously defined symbol). Let "S" represent an algebraic sum of "V's". Expressions of the following type are legal:

A=V
A=S
A=S*V+S Means: A=(S)*V+S
A=S/V+S Means: A=(S)/V+S
A=S/V*V+S Means: A=((S)/V)*V+S
A=MOD(S,S) No additional terms allowed
(same argument definition as in fortran)
A=S+[S,S,..] Where [ ] encloses a list of "bits"

NUMERICAL EXAMPLES:

ASSIGNMENT RESULT

A=10 A=10
B=A+4 B=14
C=A+B-9 C=15
D=100/B D=7
E=A+B+C/6 E=6
F=A+B-C*7 F=63
G=A+B-C/6*A G=10
H=A+B-C/6*A+C+10 H=35
I=MOD(A+B,7) I=3
J=MOD(A+B,E-1) J=4
K=[A+6,3] K=8004 (HEX)
L=[A+6,3]+40H L=8044 (HEX)
M=[2,1] M=3
N=[A+6,M] N=8004 (HEX)

* * * PARENTHESES ARE NOT ALLOWED EXCEPT IN THE "MOD STATEMENT" * * *

250.080 SYNTAX RULES

(1)....All statement labels must start in col-1 and be no greater than 8 characters in length.

(2)....All instructions (Basic and Macro) must start after col-1. At least one blank must separate any statement label and the instruction field.

(3)....At least one blank must separate the instruction and operand fields.

(4)....Imbedded blanks are allowed in (but removed from) the IF-MACRO, expressions and operand fields of all types. I.e. you can type it any way you wish but I will re-format it to be the way I like it.

(5)....The symbol being defined by an EQUATE may start in any column and be up to 8 characters in length.

(6)....All comment fields must be preceded by a semicolon.

(7)....A completely blank line will show up as a blank line in the assembly listing and otherwise be ignored.

(8)....All numbers are decimal by default.

(9)....Hex numbers are specified by a trailing "H" (8000H for example)
The first character must be a decimal integer (0 to 9)

(10)..."Bit Expressions" are lists of bit-numbers enclosed in [ ].
Example: [16,1,2] produces 8003H.

(11)...Bit-Lists are not allowed in elements of a N,A,F or C,N,A,F

(12)...Expressions enclosed by < > within a comment-field are evaluated and replaced by a 3-digit decimal integer in the assembly listing. this is for labeling purposes only: any errors will be ignored.

* * * EXPRESSIONS INCLOSED BY < > MAY CONTAIN NO IMBEDDED BLANKS * * *

250.090 PROGRAMMING EXAMPLES

Loops are initiated by the LOOP directive and terminated by the ENDLOOP (equivalent to CONTINUE in fortran). An example is shown below:

```
    SPAT 1,1,0          ;READ & STORE PATTERN REGISTER
    B=0                ;INIT PATTERN REG BIT COUNTER
    AA=-1               ;INIT ADC SUB-ADDR CNTR
    AT=-1              ;INIT TDC SUB-ADDR CNTR
    DET=0               ;INIT DETECTOR COUNTER
    ID=-2              ;INIT PARM ID #

    NADC=10             ;SLOT FOR ADC (ADC IN 10 & 11)
    NTDC1=12            ;SLOT FOR TDC1 (TDC1 IN 12 & 13)
    NTDC2=7             ;SLOT FOR TDC2 (TDC2 IN 7  &  9)
    DSA=1              ;SLOT INC FOR ADC
    DST1=1              ;SLOT INC FOR TDC1
    DST2=2              ;SLOT INC FOR TDC2
    CA=-1              ;CNTR FOR ADC  SLOT SWITCHING
    CT1=-1              ;CNTR FOR TDC1 SLOT SWITCHING
    CT2=-1              ;CNTR FOR TDC2 SLOT SWITCHING

LU1   LOOP 16
    B=B+1
    AA=MOD(AA+1,12)
    AT=MOD(AT+1,8)
    DET=DET+1
    ID=ID+3
    CA=CA+1
    CT1=CT1+1
    CT2=CT2+1
    NA=CA/12*DSA+NADC
    NT1=CT1/8*DST1+NTDC1
    NT2=CT2/8*DST2+NTDC2
    IF(PAT.NONE.[B])LEND    ;TEST PATTERN REG BIT <B>
    OUT 8000H+ID          ;OUTPUT PARAMETER ID <ID>
    ROUT NA,AA,0          ;N,A  = <NA>,<AA>
    ROUT NT1,AT,0         ;N,A = <NT1>,<AT>
```

```
     ROUT NT2,AT,0          ;N,A = <NT2>,<AT>
LEND  ENDLOOP               ;END OF LOOP

         * * * * NESTED LOOPS ARE NOT SUPPORTED * * * *
```

250.090 PROGRAMMING EXAMPLES (continued)

```
TITLE LINE MUST BE PRESENT
CSYS=2    ;SYSTEM (BRANCH) CRATE-#
CAUX=08   ;CRATE-# CONTAINING EVENT HANDLER
NFIF=18   ;SLOT-# FOR FIFO
RECL=8192 ;MAG TAPE RECORD LENGTH (BYTES)
NPAR=34   ;# OF PARAMETERS (FOR YOUR INFO ONLY)
NAUX=22   ;SLOT-# FOR EVENT-HANDLER AUX
NSUC=19   ;SLOT-# FOR SUCK-CONTROLLER
BPI=800   ;TAPE DENSITY (BITS PER INCH)
FORM=L002 ;L002 FORMAT (FOR INFO ONLY)
COM$
EXAMPLE EVENT-HANDLER PROGRAM
PGM$

     S=12                ;STOP BIT = 12
     W=11                ;WAIT BIT = 11
     E=10                ;EVENT BIT = 10
     M=1                 ;MASTER BIT IN GATED LATCH
     NAD1=5              ;ADC1 IN SLOT 5
     NLAT=2              ;GATED LATCH IN SLOT 2
     SETB                ;SET BUSY TO INIT CAMAC
INIT  IF(EX.ANY.[S,W])INIT    ;WAIT FOR EXT HANGUPS TO END
CLR   NAF NAD1,12,11          ;CLEAR ADC
     NAF NLAT,0,9           ;CLEAR LATCH
WAIT  IF(EX.ANY.[S,W])WAIT    ;WAIT FOR EXT STOPS AND WAIT STATES
     CLRB                ; * GO *
IDLE  IF(EX.NONE.[S,W,E])IDLE ;LOOP UNTIL SOMETHING HAPPENS
     IF(EX.ANY.[S,W])STOP     ;STOP OR WAIT
     IF(EX.NONE.[E])IDLE      ;? - WELL , GO BACK AND TRY AGAIN
     SETB                ;PREPARE TO READ OUT
     DLAY 1000           ;WAIT FOR CONVERSIONS
     SPAT NLAT,0,0       ;GET GATED LATCH SIGNAL
     IF(PAT.NONE.[M])CLR      ;IF MASTER COUNTER DID NOT FIRE,IGNORE
     OUT  8001H          ;EVENT HEADER
     OUT  CA             ;LATCH TO FIFO
     A=-1                ;SET UP ADC READ LOOP
     LOOP 8
     A=A+1               ;INC ADDRESS
     ROUT NAD1,A,0       ;ADC TO FIFO
     ENDLOOP
     OUT 0FFFFH          ;END OF EVENT
     BRU CLR             ;GO AND RESET ADCS ETC
STOP  SETB                ;STOP - SET BUSY
     BRU WAIT            ;AND GO INTO WAIT STATE
     END
```

250.120 BASIC INSTRUCTION LIST (BIT PATTERN)

```
          !2222!2111!1111!1110!0000!0000!
```

```
                !4321!0987!6543!2109!8765!4321! HEX REPRESENTATION
-----------------!----!----!----!----!----!----!--------------------
NOP               !0000!0000!    !    !    !    ! 0
                !   !   !   !   !   !   !
NAF     N,A,F<,T> !0001!00TT!00NN!NNNA!AAAF!FFFF! 100000+N,A,F,T
                !   !   !   !   !   !   !
NAF  (P)N,A,F<,T> !0001!10TT!00NN!NNNA!AAAF!FFFF! 180000+N,A,F,T
                !   !   !   !   !   !   !
NAF  (S)N,A,F<,T> !0001!00TT!01NN!NNNA!AAAF!FFFF! 104000+N,A,F,T
                !   !   !   !   !   !   !
NAF  (N)N,A,F<,T> !0001!00TT!10NN!NNNA!AAAF!FFFF! 108000+N,A,F,T
                !   !   !   !   !   !   !
NAF (PS)N,A,F<,T> !0001!10TT!01NN!NNNA!AAAF!FFFF! 184000+N,A,F,T
                !   !   !   !   !   !   !
NAF (PN)N,A,F<,T> !0001!10TT!10NN!NNNA!AAAF!FFFF! 188000+N,A,F,T
                !   !   !   !   !   !   !
BRU   A           !0010!0101!0   !0AAA!AAAA!AAAA! 250000+A
                !   !   !   !   !   !   !
BRUR              !0010!0000!0   !    !    !    ! 200000
                !   !   !   !   !   !   !
SPB   A           !0010!1101!0   !0AAA!AAAA!AAAA! 2D0000+A
                !   !   !   !   !   !   !
SPBR              !0010!1000!0   !    !    !    ! 280000
                !   !   !   !   !   !   !
INTE  A           !0010!1111!0   !0AAA!AAAA!AAAA! 2F0000+A
                !   !   !   !   !   !   !
INTR              !0010!1010!    !    !    !    ! 2A0000
                !   !   !   !   !   !   !
SKIP  PAT.ANY.#   !0011!0000!####!####!####!####! 300000+#
                !   !   !   !   !   !   !
SKIP UPAT.ANY.#   !0011!0001!0000!0000!####!####! 310000+#
                !   !   !   !   !   !   !
SKIP  EXX.ANY.#   !0011!0010!####!####!####!####! 320000+#
                !   !   !   !   !   !   !
SKIP  CAX.ANY.#   !0011!0011!####!####!####!####! 330000+#
                !   !   !   !   !   !   !
SKIP  PAT.NONE.#  !0011!0100!####!####!####!####! 340000+#
                !   !   !   !   !   !   !
SKIP UPAT.NONE.#  !0011!0101!0000!0000!####!####! 350000+#
                !   !   !   !   !   !   !
SKIP  EXX.NONE.#  !0011!0110!####!####!####!####! 360000+#
                !   !   !   !   !   !   !
SKIP  CAX.NONE.#  !0011!0111!####!####!####!####! 370000+#
                !   !   !   !   !   !   !
SKIP  CAX.LT.#    !0011!1011!####!####!####!####! 3B0000+#
                !   !   !   !   !   !   !
SKIP  CAX.GT.#    !0011!1111!####!####!####!####! 3F0000+#
                !   !   !   !   !   !   !
SKIP  PAT.LT.#    !0011!1000!####!####!####!####! 380000+#
                !   !   !   !   !   !   !
SKIP  PAT.GT.#    !0011!1100!####!####!####!####! 3C0000+#
-------------------------------------------------------------------
```

The optional T-field in NAF & CNAF is supported by NEW-DEAL AUX only

Comprehensive HIstogramming Language - CHIL

SECTION CONTENTS

010 HOW TO COMPILE A CHIL PROGRAM
020 INTRODUCTION
030 CHIL SYNTAX - LIST
040 CHIL SYNTAX - GENERAL RULES
050 PARAMETER LENGTHS
060 PARAMETER NAMES
070 GATES - SIMPLE   (LO-LIMIT, HI-LIMIT PAIRS)
080 GATES - SIMPLE   (MAPPED)
090 GATES - FREE-FORM (BANANAS)
100 IF-STATEMENTS AND COMPUTED GO TO'S
110 BIT-TESTS
120 HISTOGRAM - BITS/CHANNEL
130 HISTOGRAM - ID NUMBERS
140 HISTOGRAM - TITLES
150 HISTOGRAM - STATEMENTS
160 LOOPS
170 SYMBOLS & EXPRESSIONS
180 PRE-SCANNING - CONSIDERATIONS
190 USER-SUPPLIED SUBROUTINES
200 HOW TO CREATE CUSTOMIZED CHIL-BASED TASKS
210 DIRECTORY FILE - STRUCTURE
220 EXAMPLES
230 COMMENTS AND WARNINGS

320.010 HOW TO COMPILE A CHIL PROGRAM

TO COMPILE A CHIL PROGRAM ON THE CONCURRENT SYSTEM, TYPE:

```
CHIL FILE                ;For compilation but no listing
CHIL FILE,PR:              ;For list of Source & Table on printer
CHIL FILE,FILDEV           ;For list of Source & Table on FILDEV
CHIL FILE,FILDEV,OP OP..    ;For list on FILDEV  wtih options
CHIL FILE,PR:,OP OP OP      ;For list on printer with options
```

TO COMPILE A CHIL PROGRAM ON THE VAX, TYPE:

```
CHIL FILE                ;Compiles program with standard options
CHIL FILE "OP OP .."       ;Compiles program with specific options
                      ;Listing is always directed to FILE.PRT
```

Where, FILE denotes the filename prefix of a source-file whose name-extension must be .CHL (you don't type the .CHL part) and:

OP = NOT says, no HIS-Table listing
OP = NOS says, no Source listing
OP = NOC Says, no Condition listing
OP = NOTR Says, no Condition Trace accumulated

Note: If you run CHIL and get one of the following diagnostics:
CONDITION BUFFER OVERFLOW or CONDITION LIBRARY OVERFLOW
You must use the NOTR option in order to CHIL successfully.

320.020 INTRODUCTION

CHIL is a histogramming language used for writing data processing programs for both on-line (for data acquisition) and off-line (for tape scanning) analysis. Some features of CHIL-based processing programs are listed below.

(1) WCS routines for many basic operations.
(2) Both simple and free-form gating accomodated.
(3) Input of simple gate lists from files (created graphically).
(4) IF-statements of the form - IF (CONDITION) DEST-LABEL
(5) Computed-GOTO's of the form - IF (COND-SET) L1,L2,L3,L4..
(6) Bit testing and branching.
(7) Loops (no nesting).
(8) Histograms of dimensionality as large as 4 may be specified.
(9) Histogram dimensions not restricted to a power of 2.
(10) Any mixture of legal histogram dimensions may be specified.
(11) Total size of all histograms may not exceed 33,554,432 half-wds.
(12) Unrestricted switching between 16- and 32-bits/channel.
(13) Use of Parm-Names as well as Parm-Numbers in CHIL programming.
(14) Multiple USERSUBS (up to 3) called from CHIL at will.
(15) Up to 3 output data-streams from LEMO-based prescan programs.
(16) Specification of TITLES for individual histograms supported.
(17) Specification of ID's for individual histograms supported.

320.030 CHIL SYNTAX - DEFINITIONS AND ASSIGNMENTS

```
$LSTL = LENG                    ;Specify Tape Record Length (bytes)

$NPR = NPAR                     ;NPAR =Max# Parms (RAW+GENERATED)

$LPR IPA TO IPB,IS  = LENG       ;Specify Parm Lengths (loop)
$LPR IPA TO IPB     = LENG       ;Specify Parm Lengths (loop)
$LPR IP1,IP2,IP3,.. = LENG       ;Specify Parm Lengths (list)
```

```
$BAN (LENG) ID1,ID2,ID3 ..... ;Specify BAN-ID's to get from file
$BAN (LENG) IDA,IDB,IDC ..... ;Determines stacking order
$BAN (LENG) ..... ;LENG must match that from file

$BAF FILENAME.BAN/ACT            ;Specify BAN-file to process
                          ;Must follow $BAN specification
-------------------------------------------------------------------
$GAT (ISN,NG) (ISN,NG) (ISN,NG)   ;Specify (Set#, # of gates)
$GAT (ISN,NG) (ISN,NG) (ISN,NG)   ;   ''     ''      ''       ''
$GAT .....  ;As many lines as required

$MAPF (LENG) ISA,ISB,ISC .....  ;Map Sets ISA,ISB,ISC .....
                          ;with length = LENG
                          ;must follow $GAT-definition

$GAF FILENAME.GAF/ACT            ;Specify Gate-file to process
                          ;Must follow $MAPF spec if any
-------------------------------------------------------------------
$MAPL  LENG,ISN (LO,HI) (LO,HI).. ;Map specific Gate-List with
&(LO,HI) (LO,HI) ..... ;SET# = ISN (must be unique)
& ..... ;(any # of continuation lines)

$GLST  LENG,ISN (LO,HI) (LO,HI).. ;Specific unmapped Gate-List with
&(LO,HI) (LO,HI) ..... ;SET# = ISN (must be unique)
& ..... ;(any # of continuation lines)
-------------------------------------------------------------------
$DIP SYMA(IV),SYMB(JV)...        ;Define Parameter names & dimensions

$ASS SYM(ISA TO ISB,INC)=JLO,JNC  ;Assign values to Parameter names
$ASS SYM(ISA TO ISB)    =JLO,JNC  ;Assign values to Parameter names
$ASS SYM(IS1,IS2...)    =JLO      ;Assign value  to Parameter name
-------------------------------------------------------------------
$DIM SYMA(IV),SYMB(JV)...        ;Define name & dimension of symbols

$DAT SYM(ISA TO ISB)    =V1,V2... ;Assign values to symbol names
$DAT SYM(IS)            =V1       ;Assign value  to symbol name
-------------------------------------------------------------------
XX=EXPRESSION                    ;Define or re-define scaler XX
SYMB(I)=EXPRESSION               ;Set SYMB(I) equal to EXPRESSION
-------------------------------------------------------------------
$HWD ICOD,ICOD,ICOD .....        ;Insert Half-WD code into MIL
$FWD ICOD,ICOD,ICOD .....        ;Insert Full-WD code into MIL
$H16                        ;Specify 16-bits/channel
$H32                        ;Specify 32-bits/channel
$HID NUID                      ;Specify next H-ID to use
$TEX TEXT                      ;Up to 76 bytes of text
```

```
$TIT TITLE                      ;Specify title (40 bytes max)
---------------------------------------------------------------------

320.030  CHIL SYNTAX - ASP, IF & CALL STATEMENTS

ASP(IP,IV)                      ;Assign to Parm-IP the value IV

IFU(COND)LABEL                  ;IF COND  unsatisfied, GO TO LABEL
IFS(COND)LABEL                  ;IF COND    satisfied, GO TO LABEL
                        ;otherwise, drop thru (IFU & IFS)

IFC(CONDITION-SET)   L1,L2,L3.... ;Defines a "Computed GO TO"
IFC(GS(P,IS,NA,NB))  L1,L2,L3.... ;IF COND(1) satisfied, GO TO L1
IFC(B(PX,PY,IDA,IDB))L1,L2,L3.... ;IF COND(2) satisfied, GO TO L2
                        ;IF COND(J) satisfied, GO TO LJ
                        ;otherwise, drop through
                        ;Limited use inside loop: ref Note-1

IFX(PARM)LABEL                  ;IF PARM       exists, GO TO LABEL
IFN(PARM)LABEL                  ;IF PARM    non-exist, GO TO LABEL
                        ;otherwise, drop thru (IFX & IFN)

IFP(PARM)L1,L2,L3,,LJ,.....  ;IF PARM = 0         , GO TO L1
                        ;IF PARM = 1         , GO TO L2
                        ;IF PARM = J-1       , GO TO LJ
                        ;otherwise, drop through
                        ;Cannot be used inside loops!

BTAB(PARM,MASK)LALL,LSOME,LNONE   ;GO TO LALL,LSOME,LNONE IF
                        ;       ALL ,SOME, NONE bits match
                        ;Goes to LNONE if Parm non-exist
                        ;no drop thru for BTAB

CALL USERSUB1                   ;CALL USERSUB1
CALL USERSUB2                   ;CALL USERSUB2
CALL USERSUB3                   ;CALL USERSUB3

CALL REPACK1   PA,PB             ;Send Parms PA-PB to Out-Stream-1
CALL REPACK2   PA,PB             ;Send Parms PA-PB to Out-Stream-2
CALL REPACK3   PA,PB             ;Send Parms PA-PB to Out-Stream-3


Important Comment on Banana-Gate Specifications - B(PX,PY IDA,IDB)

At a given X-coordinate - Banana IDA+1 must lie above Banana IDA
```

- Banana IDA+2 must lie above Banana IDA+1

- Banana IDA+N must lie above Banana IDA+N-1

Note-1: IFC- and IFP-statements may be used inside loops ONLY on the condition that all associated branches are to labels outside the loop.

320.030 CHIL SYNTAX - HISTOGRAM STATEMENTS

THE HISTOGRAM SPECIFICATION STATEMENT TAKES THE GENERAL FORM:

H(I,J..) L(LI,LJ..) "CONDITIONS"

or

H(I,J..) L(LI,LJ..) R(LOI,HII LOJ,HIJ..) "CONDITIONS"

Where:

```
H(I,J..)              ;Says histogram Parms I,J..

OH(I,J.)              ;Says histogram Parms I,J..
                      ;in previously defined "space"

L(LI,LJ,LK..)         ;Defines histogram "Lengths" (Pwr of 2)
                      ;Length of Parm-I = LI
                      ;Length of Parm-J = LJ
                      ;Length of Parm-K = LK

R(LOI,HII LOJ,HIJ ..)  ;Says select Hist-Parm Range (after
                      ;shifting) for
                      ;Parm-I to be LOI thru HII
                      ;Parm-J to be LOJ thru HIJ
-------------------------------------------------------------------
G(P L,H L,H ..)       ;Explicit Gate-List (P# LO,HI LO,HI ...)
                      ;(not mapped)

GS(P,IS,NA,NB)        ;Gate-List specified by Set# (IS), Gate#'s
                      ;(NA,NB) from $GAT, $GLST- or $MAPL-spec.
                      ;Mapped if included in $MAPF-spec or given
                      ;in $MAPL-spec and if (NA,NB) specifies
                      ;full Gate-Set: Otherwise, not mapped.

B(PX,PY,IDA,IDB)      ;BAN-List from BAN-file (X-Parm, Y-Parm,
                      ;list of adjacent Bananas)
```

Important Comment on Banana-Gate Specifications - B(PX,PY IDA,IDB)

At a given X-coordinate - Banana IDA+1 must lie above Banana IDA

- Banana IDA+2 must lie above Banana IDA+1

- Banana IDA+N must lie above Banana IDA+N-1

320.040 CHIL SYNTAX - GENERAL RULES

RULES WHICH APPLY TO ALL STATEMENTS

(1)....All 80 cols of a line may be used.

(2)....A blank line shows up as a blank line on the listing.

(3)....All left and right parenthesis must match.

(4)....A semicolon ";" is used to introduce a "comment field".

RULES FOR THE $-DIRECTIVE AND EQUATE STATEMENTS

(1)....All $-directives ($NPR, $DIM, $ASS, etc) must start in col-1.

(2)....No continuation of $-directives are allowed except for the
$MAPL- and $GLST-statements.

(3)....Equate-directives may start anywhere but shall not be labeled.

RULES FOR ALL OTHER STATEMENTS

(1)....Statement labels must be contained within cols 1 thru 5 and be
no more than 4 characters in length.

(2)....Statement labels may be composed of any combination of numeric
and alphabetic characters.

(3)....All statements must start in col-7 or beyond.

(4)....Only the H-statement may be "continued" - any char in col-6.
the max number of continuation lines = 19.

SYNTAX IS RIGID - FOLLOW DOCUMENTATION & EXAMPLES CAREFULLY

COMMENTS

Meaning of the Assign directive - $ASS SYM(I TO J)=JLO,JNC

This means assign symbols SYM(I), SYM(I+1) ... SYM(J) the values JLO, JLO+JNC, JLO+2*JNC ....

Difference between $DIP and $DIM

Values assigned to symbols dimensioned via the $DIP specification must be unique and in the range 1 through NPAR (the max # Parms given by $NPR). In addition, only those parameters defined by the $DIP specification will be represented by their "names" on the histogram summary. Values may not be assigned to $DIP-symbols via an "Equate". None of these rules apply to $DIM-symbols.

320.050 PARAMETER LENGTHS

The "Parameter Length" defines the maximum value (always a power of 2) to be associated with a given parameter. Parameter Length specifications must follow the $NPR-statement. The following specification forms are supported:

```
$LPR IPA TO IPB,IS  = LENG    ;Sets length of parameters (IPA to IPB
                              ;in steps of IS) to be LENG.

$LPR IPA TO IPB     = LENG    ;Sets length of parameters (IPA to IPB
                              ;in steps of 1) to be LENG.

$LPR IP1,IP2,IP3,.. = LENG    ;Sets length of parameters (IP1,IP2,
                              ;IP3,...) to be LENG.
```

Lengths may be re-specified as many times as convenient but all lengths must be given at least once before another statement type is encountered. The following example illustrates the procedure:

```
$NPR = 240

$LPR   1 TO 240    = 2048

$LPR 220 TO 222    = 1024

$LPR 229 TO 239,2  = 8192

$LPR 219           = 64
```

320.060 PARAMETER NAMES

Names (beginning with a letter and up to 8 characters in length) may be associated with some or all parameter numbers and can be used to reference any parameter, so named, in the CHIL program. Two steps are required.

(1)....First, the name must appear in a $DIP-statement (dimension parameter statement) even if it's dimension is to be only unity.

(2)....Values must then be assigned to the names by means of the $ASS-statement (assign statement).

The $DIP-statement takes the form:

```
$DIP NAME1(IV),NAME2(JV),..     ;Where, NAME1, NAME2, .. denotes the
                                ;parameter names being defined and
```

```
                    ;IV, JV, .. denotes the associated
                    ;dimension.
```

The $ASS-statement takes the form:

```
$ASS NAME1(IA TO IB,IS)=JLO,JNC ;Where, NAME1 denotes the name to
                    ;which values are being assigned and
                    ;(IA TO IB,IS) defines a loop on the
                    ;name index.

                    ;JLO is the first value to assign.
                    ;JNC is the value-increment to be
                    ;added for successive assignments.
                    ;In this case we have:

                    ;NAME1(IA)      = JLO
                    ;NAME1(IA+IS)   = JLO+JNC
                    ;NAME1(IA+2*IS) = JLO+2*JNC

$ASS NAME2(IA TO IB)  =JLO,JNC  ;Assigns values with index increment
                    ;equal to unity.

$ASS NAME2(IC)        =JLO      ;Assigns the value JLO to NAME2(IC)
```

THE FOLLOWING EXAMPLE ILLUSTRATES:

```
$DIP NAI(72),TAC(72),GELI(6)    ;Define parameter names & dimensions
$DIP TOTK(1)

$ASS NAI(1 TO 72) = 1,3         ;Assign values 1,4,7,10,13 .........
$ASS TAC(1 TO 72) = 2,3         ;Assign values 2,5,8,11,14 .........
$ASS TOTK(1)      = 219         ;Assign the value 219 to TOTK(1)
```

320.070 GATES - SIMPLE (LO-LIMIT, HI-LIMIT PAIRS)

Simple gates (i.e. Lo-Limit, Hi-Limit pairs to be imposed on single parameters) can be entered in three different ways.

METHOD - 1 ----------

A single gate of the form: G(PARM LO,HI) may be defined in an IF-statement (IFS or IFU) or an H-statement. The H-statement also allows the form: G(PARM LO,HI LO,HI LO,HI ..) - i.e multiple gates on a single parameter. Here, PARM denotes the test-parameter number or name and LO,HI denotes the pair of limits to be imposed. Gates entered in this way must be given in terms of the raw test-parameter length.

METHOD - 2 ----------

Multiple gates may also be entered by means of the $GLST-statement which takes the form:

$GLST LENG,ISN LO,HI LO,HI LO,HI .....

Where, LENG represents the Parameter-Length on which the limits are based and ISN the Set-Number by which this particular Gate-Set is to be referenced. In this case, limits do not have to be entered on the basis of the raw test-parameter length.

METHOD - 3 ----------

Gate-Sets may also be read from a disk file generated by program GATLIN which allows one to define gates interactively. Input of gates from such a file requires two types of statements:

$GAT (ISN,NG) (ISN,NG) ....
and
$GAF FILENAME.GAF ;The .GAF-extension is required

The $GAT-statement specifies which Gate-Set ID's (denoted by ISN) are to be read from the file and the maximum number of gates (denoted by NG) to be read from each Gate-Set. Note: that the file may actually contain more or less gates per Set than is specified. If it contains less, the "excess" gates are set to "impossible". If it contains more, only the first NG gates are read in. The length basis on which the gates are defined is provided automatically by GATLIN. The $GAF-statement specifies the name of the file to be processed

IMPORTANT COMMENTS

Gates entered via METHOD-2 or -3 may only be used in H-statements and Computed GOTO's (IFC-statements) and are referenced by Set# (ISN) and a Gate# range (NA thu NB) as follows.

GS(PARM,ISN NA,NB)

Gates must be entered in increasing order and cannot overlap.

320.080 GATES - SIMPLE (MAPPED)

If a large number (5 or more) gates are to be imposed on a given parameter (in an H-statement), processing will proceed faster if the gates are

"mapped". This may be accomplished in two ways:

METHOD - 1 ----------

One may specify the gates to be mapped by means of the $MAPL-statement as follows:

$MAPL LENG,ISN LO,HI LO,HI LO,HI .....
& ..... as many continuation lines as required .....

Where, LENG represents the Parameter-Length on which the limits are based and ISN the Set-Number by which this particular Gate-Set is to be referenced. In this case, limits do not have to be entered on the basis of the raw Test-Parameter length.

METHOD - 2 ----------

Gate-Sets may also be read from a disk file generated by program GATLIN which allows one to define gates interactively. Mapping of gates from such a file requires three types of statements given in the order shown below:

$GAT (ISN,NG) (ISN,NG) .....
$MAPF LENG ISA,ISB,ISC .....
$GAF FILENAME.GAF ;The .GAF-extension is required

The $GAT-statement specifies which Gate-Set ID's (denoted by ISN) are to be read from the file and the maximum number of gates (denoted by NG) to be read from each Gate-Set. Note: that the file may actually contain more or less gates per set than is specified. If it contains less, the "excess" gates are set to "impossible". If it contains more, only the first NG gates are read in.

The $MAPF-statement specifies which of these Gate-Sets are to be mapped (ISA, ISB .. etc) and gives the length basis LENG on which mapping is to be done. Note: This can be different from the basis on which gates were entered.

The $GAF-statement specifies the name of the file to be processed

IMPORTANT COMMENT

Gates entered as described above may only be used in H-statements and Computed GOTO's (IFC-statements) and are referenced by Set# (ISN) and a Gate# range (NA thu NB) as follows.

GS(PARM,ISN NA,NB)

For gate mapping to be used, the full Gate-Set specified by the $GAT-statement or by the $MAPL-statement must be requested by (NA,NB).

320.090 GATES - FREE-FORM (BANANAS)

Free-Form-Gates must always be created "outside" of the CHIL program using the interactive display program RIP and stored on a special file (BAN-file). Incorporation of such Free-Form-Gates into a CHIL program requires one or more statements of the following form:

$BAN LENG IDA,IDB,IDC ....

The $BAN-statement specifies which Banana ID-numbers (IDA, IDB ...) are to be retrieved from the BAN-file and gives the X-length basis LENG on which they were constructed or will be constructed. At this time space is set aside to accommodate Free-Form-Gates which are to be processed.

The BAN-file to be processed may be specified in the CHIL program or this specification may be deferred until run-time. Specification in the CHIL program takes the following form:

$BAF FILENAME.BAN ;The .BAN-extension is required

If the $BAF-statement is not included in the CHIL program or if it desired to process a new or modified BAN-file at run-time, the following command may be issued to the monitor task or scan program.

NUBAN FILENAME.BAN

At such time, all Free-Form-Gates are reset to "impossible" and the given file is processed to provide a new set.

COMMENTS

(1)....The length given in the $BAN-statement must match that contained in the BAN-file.

(2)....Any ID-numbers specified in the $BAN-statement but not found in the BAN-file will be left set to "impossible" and a "not found" diagnostic message will be issued when such file is processed.

(3)....If no $BAF-statement is entered at CHIL-time, no comment will be made - you will find out sooner or later, I guess.

320.100 IF-STATEMENTS AND COMPUTED GOTO'S

CHIL supports IF-statements of the following forms: ----------

```
IFS(CON)LABEL          ;IF condition CON   satisfied, GO TO LABEL
IFU(CON)LABEL          ;IF condition CON unsatisfied, GO TO LABEL
IFX(PAR)LABEL          ;IF parameter PAR      exists, GO TO LABEL
IFN(PAR)LABEL          ;IF parameter PAR  non-exists, GO TO LABEL
```

Here, CON denotes a single Simple-Gate or a single Free-Form-Gate and PAR denotes a parameter number or name. Note examples below:

```
IFS(G(1,50,100))100     ;IF gate on Parm-1 (50 thru 100) is
                        ;satisfied, GO TO statement LABEL 100.

IFU(G(NAI(7),50,100))20 ;IF gate on Parm-NAI(7) (50 thru 100)
                        ;is not satisfied, GO TO 20.

IFU(B(E(1),DE(1) 1))ZIP ;IF Free-Form-Gate test on X-Parm-E(1) and
                        ;Y-Parm-DE(1), using Banana-ID# 1, is not
                        ;satisfied, GO TO the statement labeled, ZIP.

IFN(23) NEXT            ;IF Parm-23 does not exist, GO TO NEXT.
```

CHIL supports computed GOTO's of the following forms: ----------

```
IFC(COND-SET)L1,L2,L3 ..;COND-SET denotes a Condition-Set
                        ;If 1st member is satisfied, GO TO L1
                        ;If 2nd member is satisfied, GO TO L2
                        ;... etc ..., otherwise, drop through.

IFP(PARM)    L1,L2,L3 ..;PARM denotes a parameter number or name
                        ;IF PARM = 0,   GO TO L1
                        ;IF PARM = 1,   GO TO L2
                        ;IF PARM = J-1, GO TO LJ
                        ;... etc ..., otherwise, drop through.
```

The "Condition Set" refered to here, denotes an adjacent set of Free-Form-Gates (ordered such that at a given X-coordinate, Banana-N lies below Banana-(N+1) which are all of the same "length" and whose ID-numbers are consecutive or a list of Simple Gates which have been specified by the $GLST-statement or defined by the $GAT-statement and read in from a disk file by the $GAF-statement or the NUGAT command. A couple of examples are given below:

```
IFC(GS(ETOT(1),ISN 1,3)) 10,20,30  ;Test-Parm=ETOT(1), Gate-Set#=ISN
```

;gates 1 thru 3 are to be tested.

IFC(B(10,11 5,8)) 100,200,300,400 ;X-Parm=10, Y-Parm=11, Banana-ID's
;5 thru 8 are to be tested.

Note: # of gates or Banana-ID's implied must match number of labels!

320.110 BIT-TESTS

CHIL supports a bit-test and three-way branch statement of the following form:

BTAB(PARM,MASK)LALL,LSOME,LNONE

where,

PARM denotes the parameter number or name to be tested and MASK defines the bits of interest. MASK can be, most conveniently, specified as hexadecimal number whose first character must be a decimal integer and whose last character must be an "H".

The statement means:

GOTO...the statement labeled LALL, if all of the bits which are set in the MASK are also set in the Test-Parameter.

GOTO...the statement labeled LSOME, if some (but not all) of the bits which are set in the MASK are also set in the Test-Parameter.

GOTO...the statement labeled LNONE, if none of the bits which are set in the MASK are set in the Test-Parameter.

GOTO...the statement labeled LNONE if the Test-Parameter does not exist (i.e. has the hexidecimal value FFFF).

There is no "drop through" condition for this test.

320.120 HISTOGRAM - BITS/CHANNEL

The number of bits per channel used for histogram storage may be set to be either 16 (half word) or 32 (full word). This is accomplished in the CHIL program by the following two direstives:

$H16   ;Sets 16-bits/channel - Max count = 65,535 ------ the default

$H32   ;Sets 32-bits/channel - Max count = 4,294,967,303

There is no restriction on how many such directives may be entered but the user should be aware of the fact that each such entry results in the histogram space required being rounded up to the next integer multiple of 32768 16-bit channels (i.e. next multiple of 64 kb). I will say it again.

Each $H16 or $H32 entry rounds up memory required to a multiple of
64 kilobytes

320.130 HISTOGRAM - ID NUMBERS

If you do nothing, histogram ID-numbers will start at 1 and run consecutively, however, the beginning of the ID-number sequence may be redefined by the following statement:

$HID ID

Where, ID (a decimal integer of up to 8 digits) denotes the ID-number to be assigned to the next histogram processed. Histogram ID-sequences may be defined any number of times in the CHIL program so long as one sequence does not overlap another.

320.140 HISTOGRAM - TITLES

A title of up to 40 characters may be associated with each histogram by using the following statement.

$TIT TITLE

Where TITLE denotes the 40 ASCII characters mentioned above. Once a title is defined, it becomes associated (stored in the directory entry) of all subsequent histograms until re-defined.

Program HEDDO may be used to display and list directory entries as well modify (or add) titles contained therein.

320.150 HISTOGRAM - STATEMENTS

The Histogram statement (H-statement) specifies which parameters are to be histogrammed, the length of each histogram parameter (implies the number of bits to shift the associated raw parameter), optionally - the range to be selected after shifting and any conditions (Simple Gates or Free-Form Gates) to be imposed. The H-statement must start in col-7 or beyond and has the following general form:

```
 H(I,J..) L(LI,LJ..) R(LOI,HII LOJ,HIJ..)  "CONDITIONS"
                  !                !  !         !
                  !---- optional ----!  ! optional !
```
or

```
OH(I,J..) L(LI,LJ..) R(LOI,HII LOJ,HIJ..)  "CONDITIONS"
                  !                !  !         !
                  !---- optional ----!  ! optional !
```
Where:

```
H(I,J..)              ;Means histogram parameters I,J .. etc

OH(I,J.)               ;Means histogram parameters I,J .. etc
                      ;in the same "space" defined by the most
                      ;recent H-statement.

L(LI,LJ,LK..)          ;Defines histogram parameter "lengths".
                      ;always a power-of-2
                      ;Implies # of bits to shift raw parameter.
                      ;Specifies histogram dimensions unless the
                      ;range is specified explicitly by the
                      ;R-specification below.
                      ;Length of histogram-parameter-I = LI
                      ;Length of histogram-parameter-J = LJ
                      ;Length of histogram-parameter-K = LK

R(LOI,HII LOJ,HIJ ..)    ;Means select histogram-parameter Range
                      ;(after shifting - i.e. on the basis of the
                      ;length defined above) for:
                      ;Parameter-I  to be LOI thru HII
                      ;Parameter-J  to be LOJ thru HIJ
```

I will try to say it in a different way - just to fill out the page if nothing else. The L-specification L(LI,LJ ..) tells CHIL how many bits to shift the raw parameters prior to histogramming. The R-specification R(LOI,HII LOJ,HIJ ..), which is optional, can be used to select a limited portion of the histogram dimensions given by the L-specification. See the last section for some examples.

320.150 HISTOGRAM - STATEMENTS (CONTINUED)

Histogram "CONDITIONS" consist of one or more Condition Lists (OR-lists) of the following types:

```
G(P LO,HI LO,HI ..)      ;Explicit list of non-overlapping simple
                      ;gates
```

```
                    ;P - denotes test parameter number or name
                    ;LO,HI - denotes gate-limits
                    ;not mapped

GS(P,IS,NA,NB)      ;Gate-List specified by Set# (IS), Gate#'s
                    ;(NA,NB) from $GAT-, $GLST-, or
                    ;$MAPL-statement.
                    ;mapped if included in a $MAPF-statement or
                    ;given in a $MAPL-statement and if (NA,NB)
                    ;specifies the full Gate-Set:
                    ;Otherwise, not mapped.

B(PX,PY IDA,IDB)    ;Free-Form Gate-List from BAN-file
                    ;PX = X-test-parameter number or name
                    ;PY = Y-test-parameter number or name
                    ;IDA,IDB - gives first and last ID-numbers
                    ;of a list of "adjacent" Free-Form-Gates.
                    ;That is, at a given X-coordinate, BAN-IDA
                    ;must lie below BAN-(IDA+1).
                    ;All members of list must be of equal
                    ;length and have consecutive ID-numbers.
```

WHAT IS THE IMPLICATION OF MULTIPLE GATE-LISTS?

Definition: OR-list = Simple Gate-List or Free-Form Gate-List.

You will get a count "somewhere" if and only if some member of each OR-list is satisfied. If you have N OR-lists in the H-statement and NG(I) represents the number of gates in the Ith list, then the number of histograms (NH) actually implied by the H-statement is given by:

NH=NG(1)*NG(2)*------NG(N-1)*NG(N)

320.160 LOOPS

Loops are implemented so as to "look like FORTRAN". However, one must keep in mind that "loops" are actually expanded by the CHIL complier and that all "loop parameters" are constants. See SEC# 320.170 for more discussion of symbols and expressions.

Two examples which make use of loops are given below.

EXAMPLE-1: ----------

Produce 72 1-D histograms of parameters 1,4,7...214 with a gate (500,1000) set on associated parameters 2,5,8...215.

```
      DO 100 I=1,214,3
      J=I+1
      H(I) L(512) G(J,500,1000)
  100 CONTINUE
```

EXAMPLE-2: ----------

Produce 72 1-D histograms of parameters 1,4,7...214 with the requirement that a parameter named ETOT (parameter # 250) be in the range 1000 to 2000. Define parameter names and associated values as follows:

```
$DIP NAI(72),ETOT(1)              ;Define parameter names
$ASS NAI(1 TO 72)=1,3               ;Assign values to NAI
$ASS ETOT(1)=250                   ;Assign value  to ETOT

      IFU(G(ETOT(1),1000,2000)) 200  ;Skip loop if gate un-satisfied

      DO 100 I=1,72                ;Loop on 72 NAI detectors
      H(NAI(I)) L(256)             ;Histogram NAI(I)
  100 CONTINUE

  200 CONTINUE
```

320.170 SYMBOLS & EXPRESSIONS

Symbols defined by the "Equate Directive" (SYM=EXPRESSION) are called compile-time variables. Such variables may be re-defined without restriction. These are not run-time variables! - THERE ARE NO RUN-TIME VARIABLES IN CHIL!! Compile-time variables must be defined in terms of numbers and/or previously defined compile-time variables. At a given place in a CHIL program, the value associated with a given symbol is always the same - NO MATTER HOW YOU GET THERE!!

The CHIL compiler supports simple expressions which are evaluated left to right. REPEAT!! EVALUATED! LEFT! TO! RIGHT! Let V represent a single value (number or previously defined symbol). Let S represent an algebraic sum of V's. Expressions of the following type are legal:

```
A=V
A=S
A=S*V+S              Means: A=(S)*V+S
A=S/V+S              Means: A=(S)/V+S
A=S/V*V+S            Means: A=((S)/V)*V+S
A=MOD(S,S)           No additional terms allowed
                  (same argument definition as in FORTRAN)
```

A=S+[S,S,..] Where [ ] encloses a list of bit-numbers
(Lo-order bit-number = 1)

NUMERICAL EXAMPLES:

ASSIGNMENT	RESULT
A=10	A=10
B=A+4	B=14
C=A+B-9	C=15
D=100/B	D=7
E=A+B+C/6	E=6
F=A+B-C*7	F=63
G=A+B-C/6*A	G=10
H=A+B-C/6*A+C+10	H=35
I=MOD(A+B,7)	I=3
J=MOD(A+B,E-1)	J=4
K=[A+6,3]	K=8004 (HEX)
L=[A+6,3]+40H	L=8044 (HEX)
M=[2,1]	M=3
N=[A+6,M]	N=8004 (HEX)

* * * PARENTHESES ARE NOT ALLOWED EXCEPT IN THE "MOD STATEMENT" * * *

320.180 PRE-SCANNING - CONSIDERATIONS

Pre-scanning (as interpreted here) involves the processing of an input data stream (usually from Mag Tape) to produce an output data stream (usually to Mag Tape). The "processing" may include selection of certain events, selection of certain parameters, modification of input parameters or creation of new parameters or any combination of the preceding.

A CHIL-based prescan task involves the use of LEMO (or a customized version thereof) to control the process combined with a CHIL program which aides in the selection of events and/or parameters. Modification or creation of parameters will normally require one or more user-supplied subroutines USERSUBS.

The CHIL-based prescan program may include any legal CHIL statement except for the H-statement (i.e. concurrent histogramming is not supported). The CALL REPACK1 (not legal in histogramming programs) is used to initiate the "saving of an event" into the output data stream. The following simple example illustrates:

Prescan program which selects only those events for which parameters 18

and 16 satisfy Free-Form-Gate number-1 from file DEC2.BAN. If gate is satisfied, all parameters of event are saved.

```
$LSTL       = 8192       ;Specify tape record length in bytes

$NPR 18                  ;Specify number of parameters

$LPR 1 TO  2 = 64        ;Specify length of parameters 1 & 2
$LPR 3 TO 18 = 2048      ;Specify length of parameters 3 to 18

$BAN (256) 1             ;Request Free-Form-Gate from BAN-file
                         ;(X-length = 256, ID-number = 1)

$BAF DCE2.BAN            ;Give name of BAN-file

     IFU(B(18,16 1)) 100  ;Test X,Y-parameters (18,16) against gate
                         ;Skip it if gate not satisfied

     CALL REPACK1 1,18    ;Otherwise, save event in output stream
                         ; 1 is lowest  parameter-# to save
                         ;18 is highest parameter-# to save

  100 CONTINUE
```

320.190 USER-SUPPLIED SUBROUTINES

The user is able (by means of from 1 to 3 user-supplied subroutines named USERSUB1, USERSUB2, USERSUB3) to intercept and modify the Event-by-Event data stream that is being processed by a CHIL-based tape scan, prescan, or on-line monitor task. (i.e. standard programs designed to process HHIRF format L002). The interception occurrs after the event is unpacked, always, and when the CALL USERSUB1, etc is executed in the CHIL program. Thus, new or modified parameters may be ,subsequently, tested and histogrammed in the same way as any others. The number of parameters specified by the $NPR-statement must be increased to include any additional parameters which are generated.

Stock histogramming and prescan tasks include a dummy USERSUB: Customized packages are created by linking a new task in which this dummy routine is replaced by the user's routines. A customized task usually includes a user command processor USERCMP and possibly other support routines in addition to the USERSUBS. A skeleton USERSUB which just sets parameter-50 = parameter-1 is shown below:

```
     SUBROUTINE USERSUB1(IBUF)
```

```
      INTEGER*2 IBUF(512)
      IBUF(50)=IBUF(1)
      RETURN
      END
```

Your routine will receive the "Event" in expanded form. That is, all parameters will be in their "proper place" and any parameters which were not present in the raw event will be set to X'FFFF'.

Suppose that the maximum # of parameters in the original event is 20 and USERSUB1 is to generate up to 10 more. You would specify $NPR = 30 in your CHIL program. When your subroutine is entered, IBUF(I),I=1,20 would be set to the parameter value or to X'FFFF' for any missing parameters. IBUF(I),I=21,30 would be set to X'FFFF'.

Parameters which you generate should be in the range 0 - 16383 or less. Any missing parameters should be left set to X'FFFF'. In the interest of speed, USERSUBS should avoid calling other routines on an Event-by-Event basis. Subroutine and function calls take at least 10 microseconds and the linkage time increases as the number of subprogram arguments increase.

USER COMMAND PROCESSOR - USERCMP

The function of the user command processor is to process user-defined commands for set-up purposes etc. When the main program receives a command UCOM, it strips off the UCOM as well as blanks between UCOM and the next non-blank character and calls USERCMP with the remaining buffer as an argument. A skeleton user command processor is shown on the next page.

The skeleton user command processor given below illustrates some of the basic functions normally required. Note: the use of routines GREAD (reformats input line), FINAME (picks up file name), FILMAN (creates, opens & closes files), MILV (pause proof decoder of integer & floating number fields), and UMESSO (message sender).

```
      SUBROUTINE USERCMP(IWD)
      INTEGER*4 IWD(20),LWD(2,40),ITYP(40),NAMFIL(6),MESBUF(13,3)
      COMMON/MYCOM/ IV(50),NI
      EQUIVALENCE (KMD,LWD(1,1))
      DATA MESBUF/
     1'UNRECOGNIZED UCOM COMMAND - IGNORED               ',
     2'SYNTAX ERROR IN UCOM COMMAND - IGNORED            ',
     3'ERROR DECODING LIST OF INTEGER NUMBERS            '/
C
```

```
      CALL GREAD(IWD,LWD,ITYP,NF,1,80,NTER)    ;RE-FORMAT INPUT LINE
C
      IF(KMD.EQ.'FILE') GO TO 100              ;TEST COMMAND TYPE
      IF(KMD.EQ.'ICON') GO TO 200               ;  "     "     "
      GO TO 510                               ;ERROR IF NOT FOUND
C
C     PICK UP FILE-NAME AND OPEN A FILE ****************************
C
  100 CALL FINAME(IWD,5,80,NAMFIL,IERR)        ;GET FILE NAME
      IF(IERR.NE.0) RETURN                     ;TST FOR ERROR
      LU=1                                    ;SPECIFY LOGICAL UNIT
      CALL FILMAN(2,NAMFIL,LU,0,0,0,0,0,ISTAT) ;OPEN THE FILE
      CALL OPERR(ISTAT)                       ;REPORT ANY ERROR
      IF(ISTAT.NE.0) RETURN                    ;TEST FOR ERROR
C
C     DO WHATEVER  -  AND RETURN ----------------------------------
C
      RETURN
C
C     DECODE A LIST OF INTEGER VALUES ******************************
C
  200 IF(NTER.NE.0) GO TO 520                  ;TEST FOR GREAD ERROR
      NI=0                                    ;INIT # OF VALUES
      DO 210 J=2,NF                           ;LOOP ON NF-1 FIELDS
      CALL MILV(LWD(1,J),IV(J-1),XX,KIND,IERR) ;DECODE INTO IV(J-1)
      IF(IERR.NE.0.OR.KIND.NE.1) GO TO 530     ;TEST FOR ERROR
  210 CONTINUE
      NI=NF-1                                 ;SET # OF VALUES
      RETURN
C
C     SEND ERROR MESSAGES ******************************************
C
  510 CALL UMESSO(1,MESBUF(1,1))
      RETURN
  520 CALL UMESSO(1,MESBUF(1,2))
      RETURN
  530 CALL UMESSO(1,MESBUF(1,3))
      RETURN
      END
```

320.190 USER-SUPPLIED SUBROUTINES - MESSAGES FROM

In the data acquisition enviornment, the user does not have direct access to the CRT terminal or the BATMAMA.LOG file. however, information may be sent to the terminal and/or the log-file by means of routine UMESSO which is included in the main package. the procedure is this: Set the text that

you wish to be transmitted into an INTEGER*4 array of dimension 13 and call UMESSO to send it. The following code illustrates the different options:

```
      INTEGER*4 MESBUF(13)
C
      DATA MESBUF/'THIS IS A MESSAGE   ',8*4H    /
C
      CALL UMESSO(1,MESBUF)   ;Send to terminal only
C
      CALL UMESSO(2,MESBUF)   ;Send to log-file only
C
      CALL UMESSO(3,MESBUF)   ;Send to terminal and log-file
C
      CALL UMESSO(-1,MESBUF)  ;Spontaneous message to terminal
C
      CALL UMESSO(-2,MESBUF)  ;Spontaneous message to log-file
C
      CALL UMESSO(-3,MESBUF)  ;Spontaneous message to terminal & log
```

A spontaneous message is one that does not arise as a result a UCOM command to the main program. The negative log flag is required in order that the "prompt" be re-issued in such cases. That is already too much explanation - just do it that way.

A UMESSO routine is also included in all CHIL-based main programs so that the same user routines may be used for customizing. For SCAN- and PRESCAN-tasks, the log-file becomes the print-file.

No distinction is made between plus and minus log-flags in SCAN and PRESCAN programs. If you are not going to use your routines to customize a data acquisition MONITOR program such as MO1, there is no need to worry about the "spontaneous message" business.

320.200 HOW TO CREATE CUSTOMIZED CHIL-BASED TASKS

(1)....Use the editor to create the desired USERSUBS, USERCMP and other required routines and save all such routines on one file (MYRUTS.FTN for example).

(2)....Compile these routines to obtain an object file (MYRUTS.OBJ for example) by typing:

F7D MYRUTS ;For a non-optimized compilation while de-bugging
or
F7O MYRUTS ;For an optimized compilation after de-bugging

(3)....Link your routines with the "main package" by typing:

SCANLNK MYRUTS,MYSCAN ;To produce a tape-scan task

LEMOLNK MYRUTS,MYLEMO ;To produce a prescan task

MOLNK1 MYRUTS,MYMO1 ;To produce a 1024k in-core monitor task

Where, MYSCAN, MYMO1 & MYLEMO denote any legal filename prefixes (up to 8 characters in length) for the task file to be produced.

You may wish to produce your own CSS for executing scan- and prescan-tasks. To do this, use the editor to get SCAN.CSS/S or LEMO.CSS/S, change the line which loads the task (L SCAN,@2 or L LEMO) to load the task which you have created and save the modified CSS in your account under the desired FILENAME.CSS.

LOGICAL UNITS AND COMMON BLOCKS

LOGICAL UINTS -	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
CHIL-based SCAN tasks use -	X	X		X		X		X		X		X		X	
CHIL-based PRESCAN tasks use -	X	X		X	X		X		X		X		X		X
CHIL-based MONITOR tasks use -	X	X			X		X		X		X		X		

USER-SUPPLIED routines may safely use Logical Units 0, 1, 2, 3, 4, 7

All CHIL-based tasks use COMMON BLOCK labels /AAA/ through /ZZZ/
CHIL-based MONITOR tasks use /HIS1/, /MAMACOM/, /ACQCOM/ & /DIR/
as well.

USER-SUPPLIED routines should not use these COMMON BLOCK labels

---

320.210 DIRECTORY FILE - STRUCTURE

STRUCTURE OF .DIR-FILE - FIRST RECORD (128 BYTES) **********

JDIRF(1-3) - 'HHIRFDIR0001'
JDIRF(4) - # of histograms on .HIS-file
JDIRF(5) - # of half-words on .HIS-file

JDIRF(7-12) - YR,MO,DA,HR,MN,SC (date, time of CHIL run)
JDIRF(13-32) - TEXT (entered in CHIL via $TEX command)

STRUCTURE OF .DIR-FILE - DIRECTORY ENTRY (128 BYTES) **********

IDIRH(1)   - Histogram dimensionality (max = 4)
IDIRH(2)   - Number of half-words per channel (1 or 2)
IDIRH(3-6)   - Histogram Parm#'s (up to 4 parameters)
IDIRH(7-10) - Length of raw   parameters (pwr of 2)
IDIRH(11-14) - Length of scaled parameters (pwr of 2)
IDIRH(15-18) - MIN channel# list
IDIRH(19-22) - MAX channel# list
IDIRF(12)   - Disk offset in half-words (1st word# minus 1)
IDIRF(13-15) - X-Parm label
IDIRF(16-18) - Y-Parm label
XDIRF(19-22) - Calibration constants (up to 4 FP numbers)
IDIRF(23-32) - Sub-title (40 bytes) (entered via $TIT cmd)

STRUCTURE OF .DIR-FILE - ID-LIST (32 ID'S/RECORD) **********

IDLST(1)   - ID number of 1st histogram defined
IDLST(2)   - ID number of 2nd histogram defined
           -
DEFINITIONS OF COMMON/DIR/ **********

 COMMON/DIR/KLOC(6),JHSP(4),LENG(4),ND,NHW,LENH,LENT,IOF,LDF,
&NHIS,LEND(4),LENS(4),MINC(4),MAXC(4),CONS(4),ITEX(20),
&ITIT(10),LABX(3),LABY(3),MSER(10),KFILT

KLOC(I),I=1,6 = YR,MO,DAY,HR,MIN,SEC
JHSP(I),I=1,4 = Histogram parameters
LENG(I),I=1,4 = HIST "Lengths" - (MIXC(I)-MINC(I)+1)
LEND(I),I=1,4 = Raw   -Data "Lengths" in channels
LENS(I),I=1,4 = Scaled-Data "Lengths" in channels
MINC(I),I=1,4 = MIN channel# list
MAXC(I),I=1,4 = MAX channel# list
CONS(I),I=1,4 = CAL constants
ITEX(I),I=1,20 = TEXT from $TEX CHIL-entry
ITIT(I),I=1,10 = TITLE from $TIT CHIL-entry
LABX(I),I=1,3 = X-Parm label if any
LABY(I),I=1,3 = Y-Parm label if any
ND   = Dimensionality of histogram (# parms and # lengths)
NHW = # half-words/channel
LENH = # of half-words in this histogram
LENT = Not defined
IOF = Disk "offset" of 1st WD of this histogram (1st WD # -1)

(in half-words)
LDF = Length of disk file "USER.HIS" in half-words
NHIS = Total # of histograms on file "USER.HIS"

320.220 EXAMPLES

EXAMPLE-1 - SIMPLE PROGRAM DEFINING SYMBOLS & USING LOOPS & GATES

```
$TEX XFER EXP 150Nd+154Sm with 2-PPAC 3-GE 4-NAI (P2G3N4)
$LSTL = 8192                ;Tape record length (bytes)
$NPR  = 21                  ;Number of parameters/event (max)
$LPR 1 TO 5,2  = 8192       ;Length of parameters 1,3,5 - GE Energy
$LPR 2 TO 6,2  = 2048       ;Length of parameters 2,4,6 - GE Time
$LPR 7 TO 21,1 = 2048       ;Length of all other parameters
$DIP GE(3),GT(3)            ;Define names - GE  Energy & GE  Time
$DIP NAE(4),NAT(4)          ;Define names - NAI Energy & NAI Time
$DIP LX(1),LY(1)            ;Define names - PPAC Left-X & Left-Y
$DIP RX(1),RY(1)            ;Define names - PPAC Right-X & Right-Y
$DIP DT(1)                  ;Define name  - Delta Time
$DIP H(1)                   ;Define name  - NAI Total Energy
$DIP K(1)                   ;Define name  - NAI Multiplicity
$ASS  GE(1 TO 3) = 1,2      ;Assign Parameter# to GE  Energy
$ASS  GT(1 TO 3) = 2,2      ;Assign Parameter# to GE  Time
$ASS NAE(1 TO 4) = 7,2      ;Assign Parameter# to NAI Energy
$ASS NAT(1 TO 4) = 8,2      ;Assign Parameter# to NAI Time
$ASS H(1)  = 15             ;Assign Parameter# to NAI Total Energy
$ASS K(1)  = 16             ;Assign Parameter# to NAI Multiplicity
$ASS LX(1) = 17             ;Assign Parameter# to PPAC Left-X
$ASS LY(1) = 18             ;Assign Parameter# to PPAC Left-Y
$ASS RX(1) = 19             ;Assign Parameter# to PPAC Right-X
$ASS RY(1) = 20             ;Assign Parameter# to PPAC Right-Y
$ASS DT(1) = 21             ;Assign Parameter# to Delta Time
$H32                        ;Specify 32-bits/channel histogramming

$HID 1
$TIT PPAC's
     H(RX(1),LX(1)) L(256,256)          ;PPAC L-X vs R-X    ID = 1
     H(RY(1),LY(1)) L(256,256)          ;PPAC L-Y vs R-Y    ID = 2
     H(DT(1),LX(1)) L(256,256)          ;PPAC L-X vs D-T    ID = 3
$TIT K vs H
     H(K(1),H(1))   L(64,128)           ;K vs H             ID = 4
$TIT GE RAW
$HID 11
     DO GEE I=1,3                       ;GE Energy   ID = 11 to 13
     H(GE(I)) L(4096)  G(GE(I) 0,8191)
GEE   CONTINUE
```

```
$HID 21
     DO GET I=1,3                    ;GE Time     ID = 21 to 23
     H(GT(I)) L(256)   G(GT(I) 0,2047)
GET   CONTINUE
$TIT NAI RAW
$HID 31
     DO NAIE I=1,4                   ;NAI Energy  ID = 31 to 34
     H(NAE(I)) L(1024) G(NAE(I) 0,2047)
NAIE  CONTINUE
$HID 41
     DO NAIT I=1,4                   ;NAI Time    ID = 41 to 44
     H(NAT(I)) L(256)  G(NAT(I) 0,2047)
NAIT  CONTINUE
```

EXAMPLE-2 - PRESCAN EXAMPLE SELECTING EVENTS SATISFYING 1 CONDITION --

```
$NPR 18                 ;SPECIFY NUMBER OF PARAMETERS
$LPR 1 TO  2 = 64         ;SPECIFY LENGTH OF PARAMETERS 1 & 2
$LPR 3 TO 18 = 2048       ;SPECIFY LENGTH OF PARAMETERS 3 TO 18
$LSTL       = 8192       ;SPECIFY TAPE RECORD LENGTH IN BYTES
$BAN (256) 1             ;REQUEST FREE-FORM-GATE FROM BAN-FILE
                    ;(X-LENGTH = 256, ID-NUMBER = 1)
$BAF DCE2.BAN            ;GIVE NAME OF BAN-FILE

     IFU(B(18,16 1)) 100  ;TEST X,Y-PARAMETERS (18,16) AGAINST GATE
                    ;GO TO 100 IF GATE NOT SATISFIED

     CALL REPACK1 1,18    ;OTHERWISE, SAVE EVENT IN OUTPUT STREAM

  100 CONTINUE
```

EXAMPLE-4 - PROGRAM USING IF-STATEMENTS, COMPUTED GOTO'S & LOOPS

```
$LSTL=8192                        ;TAPE RECORD LENGTH (BYTES)
$NPR =240                        ;SPECIFY # OF PARAMETERS
$LPR   1 TO 240,1=2048               ;ASSIGN PARAMETER LENGTHS
$LPR 219        =64                ;   "     "       "
$LPR 220 TO 222,1=1024               ;   "     "       "
$LPR 223 TO 228,1=256                ;   "     "       "
$LPR 229 TO 239,2=8192               ;   "     "       "
$DIP NAI(72),GE(6)                  ;DEFINE PARAMETER NAMES
$DIP TOTH(1),TOTK(1)                 ;   "     "       "
$DIP LAMD(3),PHI(1)                  ;   "     "       "
$ASS NAI(1 TO 72)=1,3                ;ASSIGN VALUES TO NAMES
$ASS GE(1 TO 6)  =229,2              ;   "     "   "   "
$ASS TOTH(1)     =218                ;   "     "   "   "
```

```
$ASS TOTK(1)     =219                ;   "      "    "   "
$ASS LAMD(1 TO 3)=220,1              ;   "      "    "   "
$ASS PHI(1)      =224                ;   "      "    "   "
$GLST  64,1 1,15 16,18 19,21 22,24 25,60 ;SPECIFY SIMPLE GATE-LIST
$H16                                 ;SPECIFY 16-BITS/CHANNEL
     IFS(G(TOTH(1),320,2000))10      ;TST TOTH VS SIMPLE GATE
     IFU(G(TOTK(1),1,8))     1000    ;TST TOTK VS SIMPLE GATE
   10 DO 15 I=1,6                    ;LOOP ON 6 GELI'S
     IFS(G(GE(I),1431,1440)) 20      ;TST VS GATE (1431,1440)
   15 CONTINUE
     GO TO 1000                      ;RETURN IF NONE HIT

                                     ;TST TOTK VS GATE-LIST IN
   20 IFC(GS(TOTK(1),1,1,5))100,200,300,400,500
                                     ;COMPUTED GOTO
     GO TO 1000                      ;RETURN IF NONE HIT

  100 H(NAI(2))  L(1024)             ;FOR TOTK = 1 TO 15
     DO 110 I=3,72                   ;HISTOGRAM NAI(2)-NAI(72)
     OH(NAI(I)) L(1024)              ;ALL IN SAME SPACE
  110 CONTINUE
     GO TO 1000
  200 H(NAI(2))  L(1024)             ;FOR TOTK = 16 TO 18
     DO 210 I=3,72
     OH(NAI(I)) L(1024)
  210 CONTINUE
     GO TO 1000
  300 H(NAI(2))  L(1024)             ;FOR TOTK = 19 TO 21
     DO 310 I=3,72
     OH(NAI(I)) L(1024)
  310 CONTINUE
     GO TO 1000
  400 H(NAI(2))  L(1024)             ;FOR TOTK = 22 TO 24
     DO 410 I=3,72
     OH(NAI(I)) L(1024)
  410 CONTINUE
     GO TO 1000
  500 H(NAI(2))  L(1024)             ;FOR TOTK = 25 TO 60
     DO 510 I=3,72
     OH(NAI(I)) L(1024)
  510 CONTINUE
 1000 CONTINUE
```

EXAMPLE-5 - PROG USING FREE-FORM-GATES, USERSUB1, RANGES, LOOPS, ETC

```
$TEX PRE-SCANNED GMR DATA WITH NN SUM (3 LEVEL)
```

```
$LSTL = 8192                        ;TAPE RECORD LENGTH (BYTES)

$NPR  = 248                         ;NUMBER OF PARAMETERS

$LPR   1 TO 248,1 = 2048            ;ASSIGN PARAMETER LENGTHS
$LPR   2 TO 215,3 = 1024            ;   "      "       "
$LPR 218 TO 234,1 = 8192            ;   "      "       "
$LPR 220 TO 235,3 = 2048            ;   "      "       "
$LPR 217          = 4096            ;   "      "       "
$LPR 239          = 1024            ;   "      "       "
$LPR 236          = 4096            ;   "      "       "

$DIP NAE(72),NAT(72),NAS(72)        ;DEFINE PARAMETER NAMES
$DIP PE(6),PDE(6),PT(6)             ;   "      "       "
$DIP SUMH(1),FOLD(1),EMAX(1),PART(1)    ;   "      "       "
$DIP VECP(1),COST(1),COSR(1),PHIR(1)    ;   "      "       "
$DIP COSE(1),FFG(1)                 ;   "      "       "

$ASS NAE(1 TO 72) = 1,3             ;ASSIGN VALUES TO NAMES
$ASS NAT(1 TO 72) = 2,3             ;   "      "    "   "
$ASS NAS(1 TO 72) = 3,3             ;   "      "    "   "
$ASS PE( 1 TO  6) = 218,3           ;   "      "    "   "
$ASS PDE(1 TO  6) = 219,3           ;   "      "    "   "
$ASS PT( 1 TO  6) = 220,3           ;   "      "    "   "
$ASS SUMH(1) =236                   ;   "      "    "   "
$ASS FOLD(1) =237                   ;   "      "    "   "
$ASS EMAX(1) =238                   ;   "      "    "   "
$ASS PART(1) =239                   ;   "      "    "   "
$ASS VECP(1) =240                   ;   "      "    "   "
$ASS COST(1) =241                   ;   "      "    "   "
$ASS COSR(1) =242                   ;   "      "    "   "
$ASS PHIR(1) =243                   ;   "      "    "   "
$ASS COSE(1) =244                   ;   "      "    "   "
$ASS FFG(1)  =248                   ;   "      "    "   "

$BAN (1024) 1                       ;REQUEST FREE-FORM GATE
$BAF GPOST.BAN/85                   ;GIVE NAME OF BAN-FILE

$H32                                ;SPECIFY 32-BITS/CHANNEL

    H(SUMH(1)) L(512)                   ;SUMH SINGLES
    H(PART(1)) L(512)                   ;PART SINGLES

    H(PART(1),SUMH(1)) L(256,256) R(70,255 0,255) ;SUMH VS PART
```

```
   IFU(B(PART(1),SUMH(1),1)) ZIP              ;TST BANANA-GATE

   CALL USERSUB1                               ;CALL USERSUB1
                                     ;TO DO THE MAGIC

   H(NAS(2)) L(512)                            ;OVERLAY
   DO 50 J=3,72                               ;NAS(2 THRU 72)
   OH(NAS(J)) L(512)                           ;BANANA-GATED
50 CONTINUE

   H(NAE(2))  L(512)                           ;OVERLAY
   DO 60 J=3,72                               ;NAE(2 THRU 72)
   OH(NAE(J)) L(512)                           ;BANANA-GATED
60 CONTINUE

   H(PART(1),SUMH(1)) L(256,256) R(70,255 0,255) ;SUMH VS PART
                                     ;BANANA-GATED

   H(NAS(2) ,SUMH(1)) L(256,256) R(0,255 0,127)  ;OVERLAY (2-72)
   DO 100 J=3,72                               ;SUMH VS NAS
   OH(NAS(J),SUMH(1)) L(256,256) R(0,255 0,127)  ;BANANA-GATED
100 CONTINUE

   H(NAE(2) ,SUMH(1)) L(256,256) R(0,255 0,127)  ;OVERLAY (2-72)
   DO 200 J=3,72                               ;SUMH VS NAE
   OH(NAE(J),SUMH(1)) L(256,256) R(0,255 0,127)  ;BANANA-GATED
200 CONTINUE

   IFU(G(VECP(1),950,1023)) ZIP               ;1ST GATE ON VECP
                                     ;IF SATISFIED,
   H(NAS(2) ,SUMH(1)) L(256,256) R(0,255 0,127)  ;OVERLAY (2-72)
   DO 300 J=3,72                               ;SUMH VS NAS
   OH(NAS(J),SUMH(1)) L(256,256) R(0,255 0,127)  ;FOR 1ST VECP-GATE
300 CONTINUE                                   ;PLUS BANANA-GATE

   IFU(G(VECP(1),980,1020)) ZIP               ;2ND GATE ON VECP
                                     ;IF SATISFIED,
   H(NAS(2) ,SUMH(1)) L(256,256) R(0,255 0,127)  ;OVERLAY (2-72)
   DO 400 J=3,72                               ;SUMH VS NAS
   OH(NAS(J),SUMH(1)) L(256,256) R(0,255 0,127)  ;FOR 2ND VECP-GATE
400 CONTINUE                                   ;PLUS BANANA-GATE

   H(SUMH(1),COSR(1)) L(256,256) R(0,127 0,255)  ;COSR VS SUMH
   H(SUMH(1),COST(1)) L(256,256) R(0,127 0,255)  ;COST VS SUMH
   H(SUMH(1),COSE(1)) L(256,256) R(0,127 0,255)  ;COSE VS SUMH
   H(SUMH(1),PHIR(1)) L(256,256) R(0,127 0,255)  ;PHIR VS SUMH
```

;FOR 2ND VECP-GATE
;PLUS BANANA-GATE

ZIP CONTINUE

320.230 COMMENTS AND WARNINGS

Warning on NPAR

Don't set NPAR (the maximum number of parameters per event) to be less than it actually is in an attempt to save on unpack time or whatever. Any event found to contain more than NPAR parameters is trashed!

Comment & Warning on Bit-numbering

CHIL defines bit-numbers such that the lo-order bit is number-1 and the hi-order bit (of a 16-bit word) is number-16. See SEC# 320.170 for some examples if you are still uncertain.

Warning on Expressions

CHIL evaluates EXPRESSIONS left-to-right - NOT like FORTRAN. See SEC# 320.170 for some examples.

Warning on Banana-Gate Specifications - B(PX,PY IDA,IDB)

At a given X-coordinate - Banana IDA+1 must lie above Banana IDA

- Banana IDA+2 must lie above Banana IDA+1

- Banana IDA+N must lie above Banana IDA+N-1

This rule is made in the interest of speed. If you can't live by it, you will have to do your Banana gating one at a time.

Final Comment

CHIL hasn't really changed since the 1985 release. The documentation has been improved (at least modified) a bit and that is about all.

Of course, there are a number of improvements that could be made but we may all be obsolete before I have time to do it. We'll see.

250.120 BASIC INSTRUCTION LIST (BIT PATTERN) (CONTINUED)

```
            !2222!2111!1111!1110!0000!0000!
            !4321!0987!6543!2109!8765!4321! HEX REPRESENTATION
---------------  !----!----!----!----!----!----!----------------------
SSET  #         !0100!0011!1111!1111!####!####! 43FF00+#
            !   !   !   !   !   !   !
LOAD  #         !0100!0011!####!####!####!####! 430000+257*#
            !   !   !   !   !   !   !
SCLR  #         !0100!0011!$$$$!$$$$!0000!0000! 430000+256*$
            !   !   !   !   !   !   !
SCMP  #         !0100!0011!$$$$!$$$$!####!####! 430000+256*$+#
            !   !   !   !   !   !   !
            !   !   !   !   !   !   !
MOV    #,CAX    !0100!0000!####!####!####!####! 400000+#
            !   !   !   !   !   !   !
MOV    #,TXR    !0101!0000!####!####!####!####! 500000+#
            !   !   !   !   !   !   !
MOV   CAX,PAT    !0100!0001!   !   !   !   ! 410000
            !   !   !   !   !   !   !
MOV   PAT,CAX    !0100!0010!0  !   !   !   ! 420000
            !   !   !   !   !   !   !
MOV   CAX,TXR    !0101!0001!   !   !   !   ! 510000
            !   !   !   !   !   !   !
MOV  UCAX,CAX    !0100!0010!1000!1  !   !   ! 428000
            !   !   !   !   !   !   !
            !   !   !   !   !   !   !
MERG  #         !0101!1100!####!   !   !   ! 5C0000+4096*#
            !   !   !   !   !   !   !
OUT   #         !0101!1000!####!####!####!####! 580000+#
            !   !   !   !   !   !   !
OUT   CAX        !0101!1001!   !   !   !   ! 590000
            !   !   !   !   !   !   !
OUT   PAT        !0101!1010!   !   !   !   ! 5A0000
            !   !   !   !   !   !   !
OUT   UCAX       !0101!1010!1  !   !   !   ! 5A8000
            !   !   !   !   !   !   !
            !   !   !   !   !   !   !
DLAY  #         !0110!0000!   !@@@@!@@@@!@@@@! 60F000+@
            !   !   !   !   !   !   !
CLRB           !0111!0000!   !   !   !   ! 700000
            !   !   !   !   !   !   !
SETB           !0111!0001!   !   !   !   ! 710000
            !   !   !   !   !   !   !
BNK0           !0111!0010!   !   !   !   ! 720000
            !   !   !   !   !   !   !
BNK1           !0111!0011!   !   !   !   ! 730000
-----------------------------------------------------------------------
(1) $ Denotes compliment of #
(2) CAX denotes: CA, CA1, CA2, CA3, CA4 (CA is same as CA1)
(3) EXX Denotes: EX, EX1, EX2, EX3, EX4 (EX is same as EX1)
   (EX2,EX3,EX4 used to check "Q" or "NAF BUSY" for CAMAC AUX-2,3,4)
(4) CNAF takes same forms as NAF except that C must be given.
(5) EX2,CA2,EX4,CA4 and C=2 or C=4 result in bit-24 being set
(6) @@@@@ Denotes 2's compliment of ##### (i.e. negative)
-----------------------------------------------------------------------
```

250.120 BASIC INSTRUCTION LIST (BIT PATTERN) (NEW-DEAL AUX ONLY)

```
             !2222!2111!1111!1110!0000!0000!
             !4321!0987!6543!2109!8765!4321! HEX REPRESENTATION
-----------------!----!----!----!----!----!----!--------------------
SPLX           !0001!0000!0000!0000!0000!0000! 100000
             !   !   !   !   !   !   !
INCX           !0001!0000!0011!1100!0000!0000! 103C00
             !   !   !   !   !   !   !
DECX           !0001!0000!0011!1110!0000!0000! 103E00
             !   !   !   !   !   !   !
STOX M         !0001!0000!0011!100M!MMMM!MMMM! 103800+M
             !   !   !   !   !   !   !
RECX M         !0001!0000!0011!101M!MMMM!MMMM! 103A00+M
             !   !   !   !   !   !   !
-------------------------------------------------------------------------
```

SPLX,INCX,DECX,STOX,RECX denotes SPL, INC, DEC, STO, REC (uses CA1)

or SPL1,INC1,DEC1,STO1,REC1 (uses CA1)

or SPL2,INC2,DEC2,STO2,REC2 (uses CA2)

or SPL3,INC3,DEC3,STO3,REC3 (uses CA3)

or SPL4,INC4,DEC4,STO4,REC4 (uses CA4)

Use of CA2 or CA4 results in bit-24 being set.

SCAN: CHIL-BASED TAPE SCANNING

SECTION CONTENTS

010 INTRODUCTION
020 USING SHARED HISTOGRAM MEMORY SEGMENTS
030 MAXIMUM HISTOGRAM SIZE
040 OUTLINE OF PROCEDURE
050 SCAN LOG FILE
060 HOW TO CREATE A CUSTOMIZED TAPE SCAN PROGRAM
070 LOGICAL UNITS AND COMMON BLOCKS
080 CONTROL OF THE TAPE SCANNING PROCESS
090 USE OF COMMAND FILES
100 PROCESSING NON-STANDARD TAPES, BYTE-SWAPPING
110 SOMETHING IMPORTANT * * *

U330.010 INTRODUCTION

This document describes the use (operation only) of standard or customized CHIL-based tape scan programs with HHIRF standard (L002 format) list-data tapes as well as some non-standard data formats (see SEC# U320.100).

See CHIL (SEC# U350.) for the following information:

(1) General features of the CHIL system.
(2) How to write a CHIL program - examples included.
(3) How to write USERSUB'S and User-command-processors (USERCMP'S).

To compile a CHIL program, Type:

```
chil cfile              ;For list of source & his-table on cfile.prt
chil cfile OP,OP...     ;For list on cfile.prt with OPTIONS
```

Where, cfile denotes the filename of a source-file whose name-extension must be .chl (You don't type the ".chl" part) and:

OP = NOT says, no HIS-table listing

OP = NOS says, no SOURCE listing
OP = MIL says, list the mil-code (for software developers only)

To run the standard CHIL process, type:

```
scan cfil              ;if scan is defined in your .login
                       ;or .cshrc files, Otherwise:

/usr/hhirf/scan cfile  ;if using a HHIRF DECstation
                       ;or
/home/upak/scan cfile  ;if using a SPARCstation
```

U330.020 USING SHARED HISTOGRAM MEMORY SEGMENTS

Program scan can be requested to generate histograms in either a shared or a local memory segment as indicated below:

```
scan name        ;Starts scan using a shared memory segment (default)

scan name local  ;Starts scan using a local  memory segment
```

The advantage of using the shared segment is that damm can access histograms (in memory) as they are being generated without waiting for an end or a hup.

Problems With Orphan Shared Segments

Once upon a time there was this problem with shared memory segments. It seems that when a user program goes belly up, any shared memory segments it was using remain allocated. In time no one is able to use shared memory since all available memory is allocated to these orphan segments. The existence of this problem was documented and instructions for removing these shared memory segments have been provided (see Note: on shm_fixup below). Strangely enough, users do not alway remove their orphan segments!

In the current implementation, a subroutine (shared_wipe) is called at the startup of scan which deletes orpham shared memory segments belonging to the current user. An orphan shared memory segment is one which meets ALL of the following:

(1)....The CREATOR is the current user

(2)....The OWNER is the current user

(3)....The number of processes attached to this segment is zero (i.e. nobody is using this one).

(4)....The process which created this segment is no longer existent.

If all of these conditions are satisfied, the shared memory segment is removed.

WARNING: This may be machine dependent. It depends on the format of the output from the system command 'ipcs -mcop'. So far it has worked on DECstations, Alphas and SUNs but we can't make any guarantees for other platforms or future operating systems on these platforms.

THE FOLLOWING PROCEDURE SHOULD NO LONGER BE REQUIRED BUT I WILL RETAIN

THE DOCUMENTATION FOR NOW. This only applies DECstations and Alphas.

WARNING!! if scan should terminate abnormally (core dump), the shared segment will not be released as it normally would. You will need to perform the cleanup operation shown below, otherwise the system will eventually be eaten up with "abandoned memory segments".

Type: /usr/hhirf/shm_fixup name

Where, name is the his-file name prefix that you used in starting scan.

IMPORTANT!!

If you do not run shm_fixup at the time of the "abnormal termination" and the machine becomes almost inoperable due to the memory tied up with abandoned segments, do the following: Find these abandoned segments by displaying all files with the .shm name extension. If you find a file name.shm and are not currently running scanu or scan with name.his, then you have found an abandoned segment and should run shm_fixup as described above.

Under certain conditions, abandoned segments may not have an associated shm-file. To find and remove these do the following.

Type: ipcs ;To display shared mmemory segments & IDs

Type: ipcrm -m ID ;To remove shared memory segment ID

U330.030 MAXIMUM HISTOGRAM SIZE

The UNIX version of SCAN generates all histograms in memory. SCAN requests an allocation of memory at run time, therefore, the maximum useful size is limited by the amount of real memory that is available or the amount that it will let you use. If you try to exceed this, your system may allocate "virtual memory" to your process - well, this is no good! Things will be slow as hell!

U330.040 OUTLINE OF PROCEDURE

(1)....Use the editor to create a chil source file (cfile.chl for example) where, cfile denotes any legal filename. Then type:

chil cfile ;To compile the CHIL source program
;See page-1 for options

If there are no errors, you will be informed as to how large the associated his-file must be (number of bytes). You don't have to worry about creating the his-file. If it doesn't exist it will be created automacially at run-time (see (2) below). Two other files will be created automatically at this time. These are:

cfile.drr ;To contain the his-file directory

cfile.mil ;To contain CHIL object code

(2)....Now you should be ready to go, Type:

scan cfile ;cfile.mil must exist
;if cfile.his does not exist, you will
;be prompted for permission to create it

(3)....It will type:

SCAN-> ;and you are off and running
;Don't forget to ZERO if first time thru

(4)....If you wish to interrupt the SCAN, type:

Ctrl/C

(5)....It will type:

SCAN->

(6)....You may now type the desired command.

U330.050 SCAN LOG FILE

All messages (commands) typed on the VDT, read from command files or generated by the scan program are output to a LOG-file (LU-7) (cfile.log for example). See CHIL (SEC# U350.190) for a discussion of messages generated by user-supplied routines - how to produce, display and log them. Of course you don't have to do it this way - You may write directly to the VDT (LU-6) and/or the LOG-file (LU-7).

U330.060 HOW TO CREATE A CUSTOMIZED TAPE SCAN PROGRAM

(1)....Use the editor to create the desired USERSUBS, USERCMP and any

other required routines and save all such routines on one or more files (sub1.f, sub2.f ... for example).

(2)....Copy /usr/hhirf/scan.make (or /home/upak/scan.make) into your directory and use it as a template to construct your own customized make.file. This file is listed below. The portions of this file that you may need to replace are shown in bold faced type.

DECstation users at HHIRF

```
DIRA= /usr/hhirf/
DIRB= /usr/users/milner/Dscan/
OBJS= $(DIRA)scan.o $(DIRB)dummysubs.o
LIBS= $(DIRA)scanlib.a $(DIRA)orphlib.a
scan: $(OBJS) $(LIBS)
      f77 -O2 $(OBJS) $(LIBS) -o scan
```

SUNPAK users

```
DIRA= /home/upak/
DIRB= /home/upak/milner/Dscan/
OBJS= $(DIRA)scan.o $(DIRB)dummysubs.o
LIBS= $(DIRA)scanlib.a $(DIRA)orphlib.a
scan: $(OBJS) $(LIBS)
      f77 -O2 $(OBJS) $(LIBS) -o scan
```

U330.070 LOGICAL UNITS AND COMMON BLOCKS

CHIL-based SCAN programs use - LOGICAL UNITS 4,5,6,7,10,14

CHIL-based PRESCAN progs use - LOGICAL UNITS 5,6,7,8,10,14

User supplied routines should NOT attempt to use these LOGICAL UNITS.

All CHIL-based tasks use COMMON BLOCK labels /AAA/ through /ZZZ/

User supplied routines should not use these COMMON BLOCK labels .

U330.080 CONTROL OF THE TAPE SCANNING PROCESS

SCAN is controlled via a set of commands from the keyboard or a command file. When command input is needed, the SCAN-> prompt will be displayed. You have the following list of commands with which to respond (commands may be typed in upper or lower case).

COMMAND	MEANING OR ACTION TO BE TAKEN
TAPE rxxx	Assigns Tape unit rxxx (rmt0, rst0, rst1, etc) for input
CLOT	Closes tape unit but does not unload
CLUN	Closes and unloads tape unit
UCOM TEXT	Sends TEXT to USERCMP (user's command processor)
ZERO	Set the HIS-file to zero and reset all "pointers"
ZBUC	Zero the buffer counter (record counter)
GO	Start or continue processing (stops on EOF, EOM or error)
GO N	Process until N EOF's encountered (skips "bad records") (lists number of records skipped)
GO N,M	Process N-files or M-records - whichever comes first
GOEN	Start or continue processing - unloads tape and ends properly on ANY!! ABNORMAL!! TAPE!! READ!! STATUS!!
GOEN N	Start or continue processing - unloads tape and ends properly when N-files, DBL-EOF or EOM encountered
GOEN N,M	Process N-files or M-records - whichever comes first
HUP	Updates histogram on disk but does not terminate program
END	END "gracefully" - finish sorting (update HIS-file etc)
KILL	STOP the program immediately (do not finish sorting etc)

REW Rewind the assigned tape unit
BR N Backspace N-records on the assigned tape unit
FR N Skip forward N-records on the assigned tape unit
BF N Backspace N-files on the assigned tape unit
FF N Skip forward N-files on the assigned tape unit
FIND ID Find HEADER-# ID

STX Display/log Exabyte status (MB-used, MB-left, Errors/MB)
(currently for DECstations only)

SWAB Request byte-swap of input data records (see 100)
SWOF Request no byte-swap of input data records (default)

L001 NSKIP,NPPE - Specifies non-standard input tape format (see 100)
L002 - Specifies standard input tape format (default)

To interrupt the SCAN process, type: Ctrl/C

U330.090 USE OF COMMAND FILES

The following commands apply only when using command files for executing the scan process.

COMMAND ;MEANING OR ACTION TO BE TAKEN

CMDF fil.cmd ;Assign fil.cmd as command file
;(no command is read from the file at this point)

CCMD ;Continue - reading instructions from command file
;(this is how you switch to command-file control)

CLCM ;Continue - with last command read from command file
;(backspaces cmd-file and reads next command)

CCON ;Continue - reading instructions from terminal (VDT)
;(this is how you return control to terminal (VDT)

MSG MESSAGE ;Display MESSAGE (44 bytes) on VDT and on LOG-file.

CONDITIONS UNDER WHICH CONTROL IS SWITCHED FROM cmd-file TO TERMINAL

(1) A CCON command is encountered

(2) An ILLEGAL command is encountered

(3) A Ctrl/C is typed on the VDT (terminal)

(4) The command-file is read to the end

(5) A "read error" occurs (command file not assigned, for example)

U330.100 PROCESSING NON-STANDARD TAPES, BYTE SWAPPING

The standard CHIL-based tape scan program normally expects the input tape structure to be in HHIRF standard (L002) format (see 1987 Handbook, SEC# 260., HHIRF TAPES). However, provisions are made for the conversion of other tape formats to L002 format if the following conditions are met.

(1)....The length of event-by-event data records to be processed must be different from all other records which will be encountered.

(2)....All data records must be written in 16-bit mode. NOTE: If the high-order bit of a data word in the input stream is set it will be masked off (lost) in the converted data stream.

(3)....Each data record must contain a fixed (integer) number of events. i.e. events may not be split across record boundries.

(4)....Each event must contain a fixed number of parameters.

(5)....Bytes may be swapped if requested.

(6)....A specified number of data words (record header words etc) may be skipped at the beginning of each data record.

You request processing of a non-standard (L001) format by entering the command:

L001 NSKIP,NPPE

Where, NSKIP denotes the number of record header words to skip before reading events and NPPE the number of parameters per event.

BYTE-SWAPPING

If bytes are to be swapped, enter the command:

SWAB

U330.110 SOMETHING IMPORTANT * * *

You must terminate the SCAN with HUP, END or GOEN command in order to get all of your data processed onto the HIS-file.

LEMO (List-tape Examine, Modify, Output)

SECTION CONTENTS

010 INTRODUCTION
020 LIST OF COMMANDS
030 DISCUSSION OF CERTAIN COMMANDS
040 DISCUSSION OF COMMANDS - RELATED TO MODIFY-COPY (PRESCAN)
050 MULTI-TAPE PROCESSING - MOCO - CONCURRENT ONLY
060 HOW TO EXAMINE A TAPE
070 HOW TO RESTORE EVENT HANDLER PROGRAMS TO DISK
080 COPYING ASCII FILES TO AND FROM TAPE
090 HOW TO CONVERT A NON-HHIRF TAPE TO HHIRF STANDARD (L002)
100 BYTE-SWAPPING
110 PRESCAN (MODIFY - COPY)
120 CHIL PROGRAMMING FOR PRESCAN - EXAMPLE
130 PRESCAN IN BRIEF
140 HOW TO CREATE A CUSTOMIZED PRESCAN PROGRAM
150 OPERATIONS WHICH CAN MODIFY THE DATA STREAM
155 BUFFERED TAPE COPY OPERATIONS
160 USER PROCESSING OF RAW EVENTS - VIA REBUF
170 USER PROCESSING OF RAW DATA BUFFERS - VIA USERMOC
180 COPYING EVENT-DATA FILES TO TAPE AND ADDING HEADERS
190 LOGICAL UNITS AND COMMON BLOCKS

Type: LEMO ;To start on CONCURRENT

Type: @U1:[MILNER]LEMO ;To start on VAX with no LOGIN.COM entry
Type: LEMO ;To start on VAX with LOGIN.COM entry

Type: lemo ;To start on DECstation with .login entry

Type: HELP ;For directory to commands

U310.010 INTRODUCTION

LEMO is a tape processing utility program which can be customized (via USERSUB, USERCMP and other support routines) to produce a prescan program. The stock version of LEMO can do a number of useful things: Some of these are:

(1)....Any tape containing records no longer than 32768 bytes, may be examined (records read and displayed) and/or copied.

(2)....Input data records may be byte-swapped for VAX compatibility.

(3)....List data tapes recorded in HHIRF L002-format may be examined by reading records and displaying or printing portions thereof in "event format" (see SEC# U310.030).

(4)....The text records from L002-format tapes (which normally contain the Event Handler program) can be restored to disk. It is then ready for assembly - if it ever was.

(5)....A limited amount of prescan selection can be carried out with stock LEMO (see SEC# U310.110 for discussion & U310.120 for an example).

(6)....Certain non-HHIRF tapes may be converted to HHIRF L002 format (see SEC# U310.090)

(7)....The usual tape control functions (forward and backward spacing of records and files, rewind etc.) are provided.

(8)....Up to three output data streams can be produced in the MODIFY COPY mode.

See SEC# U350 - CHIL, for information on CHIL programming, USERSUBS and user COMMAND PROCESSORS for customized prescan tasks.

See SEC# U310.150 through SEC# U310.170 for a discussion of other non-CHIL user-supplied customizing routines, REBUF and USERMOC.

U310.020 LIST OF LEMO COMMANDS

Commands for Assigning Input and Output Tapes

IN MXXX: - Specifies tape (MXXX:) for INPUT
OUX MYYY: - Specifies tape (MYYY:) for OUTPUT-X (X=1,3)
MXXX:,MYYY: Denote MTL1:,MSA0:,MUB0:,rmt0:,rmt1:, etc

Commands for Tape Control Operations

RDI N - Read N records from INPUT
RDOX N - Read N records from OUTPUT-X (X=1,3)

FRI N   - Forward N records on   INPUT
FROX N   - Forward N records on   OUTPUT-X (X=1,3)
BRI N   - Backup N records on   INPUT
BROX N   - Backup N records on   OUTPUT-X (X=1,3)
FFI N   - Forward N files   on   INPUT
FFOX N   - Forward N files   on   OUTPUT-X (X=1,3)
BFI N   - Backup N files   on   INPUT
BFOX N   - Backup N files   on   OUTPUT-X (X=1,3)
RWI   - Rewind INPUT
RWOX   - Rewind   OUTPUT-X (X=1,3)
BTI   - Go to BOTTOM of INPUT (to DBL EOF, Backup 1 F)
BTOX   - Go to BOTTOM of OUTPUT (to DBL EOF, Backup 1 F)
CLI   - Close INPUT
CLOX   - Close   OUTPUT-X (X=1,3)
ULI   - Unload and Close INPUT tape
ULOX   - Unload and Close   OUTPUT-X (X=1,3)

Commands for Data Display

PEV IA,IB   - Print 16-bit word IA thru IB in EVENT   Format
DEV IA,IB   - Disp 16-bit word IA thru IB in EVENT   Format
PZ   IA,IB   - Print 16-bit word IA thru IB in HEX   Format
DZ   IA,IB   - Disp 16-bit word IA thru IB in HEX   Format
PA   IA,IB   - Print 16-bit word IA thru IB in ASCII   Format
DA   IA,IB   - Disp 16-bit word IA thru IB in ASCII   Format
PI   IA,IB   - Print 16-bit word IA thru IB in INTEGER Format
DI   IA,IB   - Disp 16-bit word IA thru IB in INTEGER Format
PIF IA,IB   - Print 32-bit word IA thru IB in INTEGER Format
DIF IA,IB   - Disp 32-bit word IA thru IB in INTEGER Format

Commands for Finding and Displaying Headers (Titles)

FIND N   - Find HEADER # N (does 1 BACKSPACE if found)
DTIT   - Displays next TITLE & HEADER # & BACKSPACES

Commands for Interrupting the Process

SEND STOP   - Interrupts READ or COPY process (CONCURRENT)
Ctrl/C   - Interrupts READ or COPY process (VAX)
END   - Terminates program

U310.020 LIST OF LEMO COMMANDS (continued)

Commands Related to Command-file Operations

CMDF FIL.CMD - Assign FIL.CMD as CMD-file (not read yet)
CCMD - Continue reading CMDS from CMD-file
CLCM - Continue with last CMD from file (backspaces)
CCON - Continue reading CMDS from VDT (Terminal)
MSG TEXT - Display TEXT (44 bytes) on VDT

Commands Related to Modify_Copy

MILF FIL.MIL - Read & process MIL-file (required for MOC)
UCOM TEXT - Send TEXT to USERCMP
RECO LBYT - Output-Recl-Bytes (DFLT=8192)(MOC mode only)

SWAB - Request byte-swap of input buffers (SEC# U310.100)
SWOF - Request no byte-swap (default)
SHON - Says byte-swap headers once more than data
SHOF - Says byte-swap headers & data the same way (default)

MOC N,M - MODIFY-COPY (N-files/M-recs - 1st to occur)
MOCO N,M - Same as MOC but waits on next INPUT or OUTPUT tapes
- (Not supported in VAX version)
MOCE N,M - MODIFY-COPY (END on request complete)
INIT - Resets MODIFY-COPY INPUT & OUTPUT buffers
ZBUC - Zero total INPUT & OUTPUT buffer counters

Commands Related to Simple Copy Operations

COPY N - Copy N files from INPUT to OUTPUT-1
CREC N - Copy N records from INPUT to OUTPUT-1
CC - Continue COPY - saves file- or record-count
EOF - Write EOF on OUTPUT-1 (not normally needed)

FCOP FIL.EXT - Copy FIL.EXT to OUTPUT-TAPE-1
TCOP FIL.EXT - Copy 1 file from IN-TAPE to FIL.EXT (new file created)
(To be used for 80 byte ASCII files only)

Commands Related to Event-File-to-Tape Copy & Header Creation

INFI filname - Specify input file for exam (RDI) & copy-to-tape
HTIT TITLE - TITLE contains title for next tape header
HNUM HN - HN specifies next tape header number to use
HOUT - Outputs tape header and increments HN

SHON - Says byte-swap headers once more than data
SHOF - Says byte-swap headers & data the same way

U310.020 LIST OF LEMO COMMANDS (continued)

Miscellaneous and Non-standard Commands

STEX FIL.EXT   - Store text blks on FIL.EXT
ASLU LU,FIL.EXT - Asn LU to FIL.EXT (LU=0,1,2,3,4) (CONCURRENT only)
CLOU LU   - Close LU   (CONCURRENT only)

L001 NSKIP,NPPE - Specify non-HHIRF   input tape (see SEC# U310.090)
L002   - Specify HHIRF-format input tape (default)

Commands Related to User Processing of Raw Buffers & Events

UPON NPRAW,RECL - Turn User-processing ON (see SEC# U310.170)
                - NPRAW = Max # of raw parameters (for user only)
                - RECL = Input data record length in bytes
UPON   - Turn User-processing ON (NPRAW=0, RECL=8192 bytes)
UPON NPARU   - Turn User-processing ON (RECL=8192 bytes)
UPOF   - Turn User-processing OFF (default)

RBON   - Enable REBUF calls (see SEC# U310.160)
RBOF   - Disable REBUF calls (default)

U310.030 DISCUSSION OF CERTAIN COMMANDS

FIND.....N attempts to find HEADER # N by searching forward on the INPUT tape (TITLES and HEADER numbers are displayed along the way): If found, LEMO backs up one record. If not found, LEMO will read past the first Double-EOF, and back up one File Mark.

BTOX.....Advances OUTPUT-X (X=1,3) past the first Double-EOF and backs up one File Mark (i.e. positions properly for appending). Header titles and numbers are displayed along the way.

BTI......Does the same thing for the INPUT tape.

DEV......PEV, DA, PA, DZ, PZ, DI, PI, DIF & PIF are all commands which display on the terminal or list on the printer some portion of the last record read from tape (either the INPUT or OUTPUT). When you do a read (RIN or ROU), LEMO tells you how many bytes were read but you must specify the portion of the buffer to be displayed in half-words (16-bit words).

In EVENT FORMAT (DEV or PEV), LEMO looks for a hex FFFF before starting to accumulate the first EVENT to be displayed.

Therefore, the first EVENT in any record is usually not displayed by DEV or PEV (you can see it via DZ or PZ, however).

COPY.....N says COPY N-files from INPUT to OUTPUT. All that is required is that records be no longer than 32768 bytes. File-marks are copied and the OUTPUT tape is always positioned between Double File-Marks on normal completion of a COPY request. The OUTPUT tape is positioned ahead of Double File-Marks on abnormal completion (via SEND STOP, input error, etc) of a COPY request. The COPY is terminated (normally) if a Double-EOF or an End-Of-Medium is encountered on the INPUT.

CREC.....N says copy N-records from INPUT to OUTPUT. The same rules apply as for COPY except that the OUTPUT tape is always left positioned ahead of a Double File-Mark.

CC.....Says continue previous COPY or CREC. It remembers how many files or records have already been copied and continues the count.

U310.040 DISCUSSION OF COMMANDS - RELATED TO MODIFY-COPY (PRESCAN)

INIT.....Resets both the EVENT- and OUTPUT-buffers. Do this if you have been doing some tests but are now ready to prescan for real or any time you wish to make a clean start.

MOC.....N,M Starts the MODIFY-COPY process, where: N is the number of files to process and M is the number of records to process. The processing terminates on either N or M - the first to be satisfied. The default values of N and M are 1 and 100,000,000 respectvely.

MOCE.....N,M Differs from the MOC-command described above only in that upon completion of the request, both INPUT and OUTPUT tapes are unloaded and the program is terminated.

The following table summarizes the different ways in which processing may be terminated and the state of the OUTPUT-tape, OUTPUT-buffer and EVENT-buffer for each:

TERMINATION BY--- RESULTS IN ----------

Requested # Files Flush OUT-BUF, 2-EOF, 1-BKFIL, Scrub-EV
EOM on INPUT Flush OUT-BUF, 2-EOF, 1-BKFIL, Scrub-EV
Requested # Recs Flush OUT-BUF, 2-EOF, 2-BKFIL, Save-EV
SEND STOP 2-EOF, 2-BKFIL, Save-EV
Input Error Flush OUT-BUF, 2-EOF, 2-BKFIL, Scrub-EV

MOCE Flush OUT-BUF, 2-EOF, Unload Tapes, EXIT

Where, Scrub-EV indicates that any partial-event which has not been fully processed will be deleted from the EVENT-buffer and Save-EV means that any partial-event will be retained for subsequent MOC requests.

NOTE: Processing is always terminated with two File-Marks being written on all OUTPUT tapes.

U310.050 MULTI-TAPE PROCESSING - MOCO - CONCURRENT ONLY

The MOCO command is intended to make it easier for a novice (or at least someone not familiar with your process) to change tapes for you. MOCO means MODIFY-COPY & CONTINUE (waits for input & output tapes to be mounted).

THIS IS HOW IT WORKS

When intervention is required, by the requested number of files being read from the input (or DBL-EOF or EOM encountered) or EOM detected on the output, the program will:

(1) Unload the tape which must be replaced (TELEX drives only),

(2) Type: WAITING FOR INPUT (or OUTPUT) TAPE,

(3) Try to read/write input/output tapes (trying every 2 sec),

(4) and flash the light on top of the drive.

As soon as another tape is mounted and made ready, the program will:

(1) Read/write one record from/on the new tape,

(2) Rewind the new tape,

(3) Turn off the flasher,

(4) and read the next command - normally from a command file.

If the next command read from the command file is another MOCO, MOCE, etc, it will process the tape without anything being typed on the VDT. If the command file contains a long list of MOCO's, the process will continue until you run out of tapes or the tape drive screws up.

A reasonable command file might look like:

```
MOCO 100       ;Says process tape to end & wait for next
MOCO 100
 -
 -
MOCE 100       ;Process last tape and end
```

U310.050 MULTI-TAPE PROCESSING - MOCO - CONCURRENT ONLY (continued)

SUGGESTIONS

(1)....Set up a command file to process a certain number of tapes and then end gracefully.

(2)....Turn the FLIP-SIGN on top of the tape drive to the card which says .... NEXT TAPE WILL BE PROCESSED IN 10 SECONDS ....

(3)....Fill out and attach one of the FLIP-SIGN labels provided.

(4)....Carefully stack tapes to be processed in front of the appropriate tape drive in the space labeled TAPES TO DO.

(5)....Finished tapes should be stacked in the TAPES DONE space.

(5)....Start processing with LEMO.

When a flasher is observed to be active (blinking), do the following:

(1)....Remove the processed tape and place it on the TAPES DONE stack.

(2)....Load next tape from the TAPES TO DO stack and set it ON LINE.

WHAT IF'S

If....the drive drops ready on rewind but the light is blinking, things should proceed normally - once you get the old tape unloaded & the next tape mounted and on line.

If....the tape runs off the end but the light is blinking, things should proceed normally when you finally get the next tape loaded.

If....the drive becomes unavailable (for any reason) and the light is NOT blinking, VDT intervention will be required.

!!! CAUTIONS !!!

Don't try to use a BLINKING drive for anything except to process the next tape from the TAPES TO DO stack.

If you have to unload the tape manually (because of drive malfunction, etc), be carefull NOT to put the old tape ON LINE because the program may start to process it AGAIN! before you have time to press the RESET button.

U310.060 HOW TO EXAMINE A TAPE

(1) Place the tape of interest on some tape drive.
(2) Type: LEMO
(3) Type: HELP

You can probably figure out what to do.

U310.070 HOW TO RESTORE EVENT HANDLER PROGRAMS TO DISK

Position the INPUT tape such that the Header containing the program of interest is the next one to be encountered (via FIND for example). If the desired program is contained in the first Header on the tape, type:

```
RWI                    ;Rewinds the INPUT tape

STEX FILNAM.EVS        ;The extension .EVS is required by ADAC
```

Where, FILNAM.EVS is the file on which the program is to be stored. FILNAM.EVS must not already exist. If you wish to restore the program contained in Header # N, Type:

```
FIND N                 ;Finds Header # N
```

When (and if) the requested Header is found, Type:

```
STEX FILNAM.EVS
```

U310.080 COPYING ASCII FILES TO AND FROM TAPE

LEMO may be used to write and read ASCII tapes for transfer to and from other computers. The following commands are used:

```
FCOP FILENAME  ;Copies FILENAME to OUTPUT-TAPE-1 (previously opened)
               ;Variable length records from FILNAME are de-tabbed and
               ;written as fixed length (80 byte) records on tape.
```

```
            ;(DEC's rules for FORTRAN tabs are used in de-tabbing)

TCOP FILENAME  ;Copies 1 file from INPUT-TAPE to FILENAME (created)
            ;Fixed length (80 byte) records from tape are
            ;written as variable length records on FILENAME.
```

U310.090 HOW TO CONVERT A NON-HHIRF TAPE TO HHIRF STANDARD (L002)

Certain non-standard tapes may be converted to L002 format under the MODIFY - COPY mode if the following conditions are met:

(1)....The length of event-by-event data records to be processed must be different from all other records which will be encountered.

(2)....All data records must be written in 16-bit mode.

(3)....If the high-order bit of a data word in the input stream is set it will be masked off (lost) in the converted data stream.

(4)....Each data record must contain a fixed (integer) number of events. i.e. events may not be split across record boundries.

(5)....Each event must contain a fixed number of parameters.

(6)....Bytes may be swapped if requested.

(7)....A specified number of data words (record header words etc) may be skipped at the beginning of each data record.

Request processing of a non-standard (L001) format by typing:

```
L001 NSKIP,NPPE
```

Where, NSKIP denotes the number of record header words to skip before reading events and NPPE the number of parameters per event. If bytes are to be swapped and/or an output record length other than 8192 bytes is desired, use the following commands:

```
SWAB          ;Requests byte-swap of input records
RECO  NBYTS   ;Specifies that the output record length is to be NBYTS
```

A simple CHIL program of the following form will be required:

```
$LSTL = RECL              ;Specify tape record length (bytes)
$NPR  = NPAR              ;Specify # parameters per EVENT (NPAR)
                    ;Must include any calculated parameters
```

```
CALL USERSUB1        ;Use only if parameters are to be modified
                     ;or created
CALL REPACK1 1,NPAR  ;Request that all parameters be saved
```

(See SEC# U310.130 for additional information on MODIFY-COPY operation)

U310.100 BYTE-SWAPPING

The command SWAB causes the following byte-swapping actions:

(1)....For L002 input, swaps bytes for all input records (appropriately) in both COPY and MOC modes.

(2)....For L001 input, swaps bytes of all input records (assuming 16-bit integers) in the COPY mode.

(3)....For L001 input, swaps bytes of data records only in MOC mode.

Under some circumstances, you may need to byte-swap the data and not the headers or vice versa. In such cases, the command SHON turns on an additional byte-swap for the header only. SHOF turns it off and is the default.

U310.110 PRESCAN (MODIFY - COPY)

Pre-Scanning (as interpreted here) involves the processing of an INPUT data stream (from Mag Tape) to produce an OUTPUT data stream (to Mag Tape). The processing may include selection of certain events, selection of certain parameters, modification of input parameters or creation of new parameters or any combination of the preceding.

A CHIL-based prescan task involves the use of LEMO (or a customized version thereof) to control the process combined with a CHIL program which aides in the selection of events and/or parameters. Modification or creation of parameters will normally require one or more user-supplied subroutines USERSUBS.

The CHIL-based prescan program may include any legal CHIL statement except for the H-statement (i.e. concurrent histogramming is not supported). The CALL REPACK1 (not legal in histogramming programs) is used to initiate the "saving of an event" into the output data stream.

In the MODIFY-COPY mode, records are read from the input tape, events are expanded and passed (one at a time) to the USERSUBS.

Input records which are not of the length specified in the CHIL program

($LSTL-Statement) are simply copied to all output tapes, thus, Headers are copied automatically. If the record length and/or the Max # of Parms for the output tape is specified differently from the input, the primary Header is modified appropriately.

If an End-of-Medium is encountered on an output tape and the recording is continued on a new tape, only the primary header (not the text blocks containing the Event Handler program etc) will be reproduced on the continuation tape.

U310.120 CHIL PROGRAMMING FOR PRESCAN - EXAMPLE

Prescan program which selects only those EVENTS for which parameters 18 and 16 satisfy Free-Form-Gate number-1 from file DEC2.BAN. If gate is satisfied, all parameters of event are saved.

```
$LSTL       = 8192       ;Specify tape record length in bytes

$NPR 18                  ;Specify number of parameters

$LPR 1 TO  2 = 64        ;Specify length of parameters 1 & 2
$LPR 3 TO 18 = 2048      ;Specify length of parameters 3 to 18

$BAN (256) 1             ;Request Free-Form-Gate from BAN-file
                         ;(X-length = 256, ID-number = 1)

$BAF DCE2.BAN            ;Give name of BAN-file

    IFU(B(18,16 1)) 100  ;Test X,Y-parameters (18,16) against gate
                         ;Skip it if gate not satisfied

    CALL REPACK1 1,18    ;otherwise, save EVENT in output stream
                         ; 1 is lowest  Parameter-# to save
                         ;18 is highest Parameter-# to save

 100 CONTINUE
```

U310.130 PRESCAN IN BRIEF

Create your CHIL program and save in file PRESCAN.CHL, for example.

Type: CHIL PRESCAN

Type: DEL PRESCAN.DIR ;For CONCURRENT

Type: DEL PRESCAN.DRR;* ;For VAX

Don't create a HIS-file

Type: LEMO

Type: IN MXXX: ;MXXX: denotes Input Tape Unit

Type: OU1 MYYY: ;MYYY: denotes Output Tape Unit

Type: MILF PRESCAN.MIL ;Reads & processes MIL-file

Type: MOC ;Starts processing

End-of-File or End-of-Medium on Input Tape

When an EOF or EOM is encountered on the input tape, LEMO writes any partially filled output buffer and two File Marks onto all output tapes and backs up one File Mark. To continue with the next input file, type the command MOC. To continue with the next input tape, type RWI, mount next input tape and type MOC.

End-of-Medium on Output Tape

When an EOM is encountered on the output tape, LEMO backs up one record, writes a File Mark, rewinds the output, instructs you to mount a new tape, and pauses. When you type: CONTINUE, LEMO writes the primary header (not the text blocks) and the last output record (the one it was writing when it hit EOM) onto the new tape, and continues processing.

See SEC# U310.040 for more information on termination of processing.

U310.140 HOW TO CREATE A CUSTOMIZED PRESCAN PROGRAM

For CONCURRENT System

(1)....Use the editor to create the desired USERSUB, USERCMP and other required routines and save all such routines on one file (MYRUTS.FTN for example).

(2)....Compile these routines to obtain an object file (MYRUTS.OBJ for example) by typing:

F7D MYRUTS ;For a non-optimize compilation while de-bugging

or
F7O MYRUTS ;For an optimized compilation after de-bugging

(3)....Link your routines with the "main package" by typing:

LEMOLNK MYRUTS,MYLEMO ;To produce a customized task

Where, MYLEMO denotes any legal filename prefix (up to 8 characters) for the task file to be produced. The following CSS procedure loads and starts MYLEMO.

```
L MYLEMO
AS 5,CON: ; AS 6,PR: ; AS 9,CON:
AS 8,LEMO.HEP/S
ST
$EXIT
```

For the VAX

(1)....Use the editor to create the desired USERSUBS, USERCMP and any other required routines and save all such routines on one or more files (F1.FOR, F2.FOR .. for example).

(2)....Compile these routines to obtain associated object files by typing:

```
FOR F1                ;To compile F1
FOR F2                ;To compile F2 etc.
```

(3)....LINK your routines with the MAIN PACKAGE by typing:

```
LEMOLNK "F1,F2,.." MYLEMO     ;To produce EXE-file MYLEMO
```

(4)....Copy the COM-file LEMO.COM to MYLEMO.COM and replace the line:

RUN LEMO with RUN MYLEMO

U310.140 HOW TO CREATE A CUSTOMIZED PRESCAN PROGRAM (continued)

Assume that your username is userdoe and your customizing routines are in your directory Dlemosubs and there are two files to be included, subsa and subsb and you wish to produce a program mylemo. Copy the template file into your directory and create a make file mylemo.make by changing the template file as/where indicated by the bold face type. The generic result is shown below.

```
DIRA= /usr/hhirf/
DIRB= /usr/users/userdoe/Dlemosubs/
OBJS= $(DIRA)lemo.o $(DIRB)subsa.o $(DIRB)subsb.o
LIBS= $(DIRA)lemolib.a $(DIRA)milib.a $(DIRA)jblibf1.a $(DIRA)jblibc1.a
mylemo: $(OBJS) $(LIBS)
        f77 -O2 $(OBJS) $(LIBS) -o mylemo
```

U310.150 OPERATIONS WHICH CAN MODIFY THE DATA STREAM

When running in the modify-copy (MOC) mode, one or more of the following operations may be initiated which can result in the modification of the data stream. Operations are listed in the order that they will be carried out if requested. The column headed CMD gives the run-time LEMO command which requests the associated operation.

CMD OPERATION----------

SWAB...Swap bytes in data buffers (also HHIRF headers) (see SEC# U310.100).

L001...Convert data buffers from L001 to L002 format (see SEC# U310.090).

RBON...Call user-supplied routine BUFMAN event-by-event (see SEC# U310.160).

UPOF...Process & output buffers via CHIL, USERSUBs (see SEC# U310.120). This is the default.

UPON...Process buffers via user-supplied routine USERMOC (see SEC# U310.170). After the new output buffer is constructed, USERMOC must do one of the following:

(1)...Output buffers directly by calling routine UPVALL or:

(2)...Process and output buffers via CHIL, USERSUBs, etc. by calling routine PVALL.

Note: A CHIL program is not required unless PVALL is called by USERMOC. Thus, in the UPON, UPVALL mode, the user can do anything he feels like with the data. Output buffers do not have to be L002 or anything in particular.

U310.155 BUFFERED TAPE COPY OPERATIONS

The simple tape copy executed via the COPY or CREC commands is not buffered (i.e. input and output does not proceed simultaneously). However, you can use the MOC command with UPON and the default USERMOC routine (which just copies buffers) to achieve a double buffered copy operation. This should run at approximately twice the speed of the simple COPY.

U310.160 USER PROCESSING OF RAW EVENTS - VIA REBUF

LEMO provides (via a user-supplied routine REBUF) the user with the means to access and modify the raw event-data prior to any subsequent CHIL or user-processing (USERMOC) action. If REBUF is enabled, the following operations occur.

(1)....Buffers (records) are read from the input tape.

(2)....Input buffers are scanned for events (number sequences ending in hex-FFFF) and one raw event per call is passed to REBUF. Events which are split across records are taken care of.

(3)....REBUF produces a new output data stream by storing new packed events (including the FFFF) into an INTEGER*2 array NULIST.

(4)....The new data stream (in NULIST) is then passed to the usual unpacking and CHIL processing, including USERSUB calls etc.

CHIL only sees the data and number of parameters in NULIST - it doesn't know that there was a REBUF operation. $LSTL (in the CHIL program) is used by the tape reading routine and is unchanged.

Type: RBON ;At run time to enable REBUF calls
Type: RBOF ;At run time to disable REBUF calls (default)

```
C     ---------------------------------------------------------------------
C     A do-nothing REBUF routine which just copies PBUF to NULIST
C
C     PBUF   - contains raw packed event (the FFFF is not included)
C     NP     = number of words in PBUF
C     NULIST - is the new output buffer which you load
C     NN     = number of words loaded into output buffer
C              LEMO resets NN to zero whenever a buffer is processed
C     ---------------------------------------------------------------------
      SUBROUTINE REBUF(PBUF,NP,NULIST,NN)
      INTEGER*2 NULIST(16384),PBUF(2000)
      DO 10 I=1,NP
      NN=NN+1
      NULIST(NN)=PBUF(I)
```

```
  10 CONTINUE
    NN=NN+1
    NULIST(NN)=X'FFFF'        !You must supply the end-of-event FFFF
    RETURN
    END
```

Important Comment

If you think that you may significantly increase the abount of data (in going from input to output), I may need to know about it. I have shown the dimensions of PBUF and NULIST to be that defined in the main program. NULIST is dimensioned to 4 times the usual input record size of 8192 bytes. LEMO does not check the fullness of NULIST on an event-by-event basis but processes NULIST after each input buffer is massaged into it. As things stand, you would be able to expand the input data by almost a factor of 4 (for 8192 byte input records) - beyond that, you blow the program!!

U310.170 USER PROCESSING OF RAW DATA BUFFERS - VIA USERMOC

VAX and DECstation versions of LEMO provide users with the means (via a user-supplied routine USERMOC) to do the following:

(1)....Process raw input buffers into new buffers and subsequently call UPVALL which can generates (1 to 3) output data streams to tape with no CHIL involvment.

(2)....Process raw input buffers into new buffers and subsequently call the standard CHIL processor, PVALL, which then imposes any CHIL conditions and generates output data streams to tape.

At run time, calls to USERMOC are enabled/disabled as follows:

```
UPON            ;Enable calling of USERMOC (NPRAW=0, RECL=8192 bytes)
UPON NPRAW      ;Enable calling of USERMOC (         RECL=8192 bytes)
UPON NPRAW,RECL  ;Enable calling of USERMOC
             ;NPRAW = Max # raw parms (for USERMOC if needed)
             ;RECL  = Input record length (bytes) if not 8192

UPOF            ;Disables USERMOC calls (enables PVALL calls - default)
```

The following USERMOC illustrations do nothing except to copy the input buffer INBUF into an output buffer OUBUF with no modification. The first illustration calls UPVALL to output into stream-1 while second calls PVALL for subsequent CHIL processing and output.

```
    SUBROUTINE USERMOC(INBUF,NWDS)   ;Illustration-1
```

```
C                               ;default USERMOC
      INTEGER*2 INBUF(*),OUBUF(16384)  ;NWDS=#words in INBUF
C
      DO 10 I=1,NWDS                ;Loop on # words in input buffer
      OUBUF(I)=INBUF(I)             ;Set output = input
   10 CONTINUE
      ISTREAM=1                    ;Specify output stream-1
C
      CALL UPVALL(OUBUF,NWDS,ISTREAM)  ;Call UPVALL to handle output
C                               ;NWDS = # of words in OUBUF
      RETURN                        ;and return
      END
-------------------------------------------------------------------------
      SUBROUTINE USERMOC(INBUF,NWDS)   ;Illustration-2
C                               ;NWDS = # of words in INBUF
      INTEGER*2 INBUF(*),OUBUF(16384)  ;Buffer dimensions
C
      DO 10 I=1,NWDS                ;Loop on # words in input buffer
      OUBUF(I)=INBUF(I)             ;Set output = input
   10 CONTINUE
C
      CALL PVALL(OUBUF,NWDS)          ;Call PVALL for CHIL processing &
C                               ;output (CHIL specifies stream-#)
      RETURN                        ;NWDS = # of words in OUBUF
      END
```

U310.180 COPYING EVENT-DATA FILES TO TAPE AND ADDING HEADERS

The DECstation version of lemo accomodates the examining and copying of disk-files containing event-list data. This feature is intended to be an aid to those doing simulations on the DECstations. The following commands are available:

```
INFI  filname   - Specify input file for exam (RDI, DEV) & copy-to-tape
HTIT  TITLE     - TITLE contains title for next tape header
HNUM  HN        - HN specifies next tape header number to use
HOUT           - Outputs tape header and increments HN
```

In addition, an example program found in /usr/users/milner/Develx/evelx.f contains routines for opening an event-file (EVELOPEN) and for writing data to it (EVELOUT). The "simulator" might (or might not) wish to include these routines in his simulation source code. Feel free to copy evelx.f to your directory for examination etc. The routines are internally documented.

The idea is this:

(1)....You do your simulations and write the generated events to a disk file.

(2)....You may then use lemo to examine this file (read, display in event-format, etc).

(2)....You may also specify a tape header number and title, output the header to tape and finally copy the entire file to tape via the normal copy command.

An typical file-to-tape copy session might look as follows:

```
lemo>infi  eventfile.dat          ;open event-file for input
lemo>ou    rmt1:                  ;open rmt1: for output
lemo>htit  simulation-3 no-gates   ;title for tape header
lemo>hnum  3                      ;next header number on tape
lemo>hout                         ;output the header to tape
lemo>copy  1                      ;copy 1-file input-to-output
lemo>end                          ;end program
```

NOTE: You can also use the header setup and output feature (hnum, htit & hout commands) to add headers while copying one tape to another. There is no provision, however, to delete or modify existing headers.

U310.190 LOGICAL UNITS AND COMMON BLOCKS

CHIL-based PRESCAN programs may use - LOGICAL UNITS 5 thru 14
User supplied routines should NOT attempt to use these LOGICAL UNITS.
All CHIL-based tasks may use COMMON BLOCK labels /AAA/ through /ZZZ/
LEMO also uses /INNARDS/ and /ULOCA/
User supplied routines should not use these COMMON BLOCK labels .

DAMM – Display and Manipulation Programm

Section Contents

010 GENERAL - Introduction and General Features
020 GENERAL - Getting Started
030 GENERAL - Assigning Input/Output Files
040 GENERAL - Loop-Execution & Symbol-Definition
050 GENERAL - Log File - damm.log
060 GENERAL - Comments on Hard Copy
070 GENERAL - File ID-directories and Count-Sums
090 GENERAL - Cursor Tracking Problems With Xterminals
100 GENERAL - Changes in Cursor-Mode Commands
110 GENERAL - Mouse Button Customizing
120 GENERAL - Screen Setup and Color Mapping
130 GENERAL - Display Delays (hangup problems)
150 DISPLAY - 1-D Display
155 DISPLAY - 1-D Display (Peak Finding/Logging)
160 DISPLAY - 2-D Display
200 MANIPULATION - Syntax Definitions
210 MANIPULATION - Setup
220 MANIPULATION - I/O 1-D Histograms
230 MANIPULATION - Gating 2-D Histograms
240 MANIPULATION - General 2-D Projections
250 MANIPULATION - Operations on Buffers-1 & -2
260 MANIPULATION - Show Data, Count-Sums etc (from Bufs-1 & -2)
270 MANIPULATION - Modify Buffer Contents
280 MANIPULATION - Printer Plots
290 MANIPULATION - Gain-Shifts & Compressions
300 MANIPULATION - of HIS-files (Copy, Add, Gain-Shift, etc)
350 BANANAS - Definition, Rules, Construction & Display
360 BANANAS - Projections
400 FITTING - Introduction & General Features
410 FITTING - 1-Key (Cursor Mode) Commands
420 FITTING - Setup Commands
430 FITTING - Display of Data, Fits & Printer-Plots
440 FITTING - FIT Execution Commands
450 FITTING - FIT Parameters - Saving, Setting, Default
460 FITTING - FIT Specification Details
470 FITTING - Peak Shapes
480 FITTING - Gradient-Search Method (FIT request)
490 FITTING - Gaussian Method (GFIT request)
500 FITTING - Estimated Uncertainties in Peak Areas
510 FITTING - Reading Fit Results from damm.log

520 FITTING - Common Problems
530 FITTING - Peak Shape vs Asymmetry (Log Plot)
550 CUSTOMIZING - Screen Configuration
560 CUSTOMIZING - Graphic Screen Color Mapping
570 CUSTOMIZING - Graphic Screen Black & White Mapping
600 Implementation
700 Gating 2D Matrix With the Cursor (RVR)

U300.010 Introduction and General Features

DAMM is a Display, Analysis and Manipulation Module which is configured to be used on VAXstations running VMS and DECwindows. DAMM provides the features to be found in VAXPAK programs DAM, SAM, TDX and XAM. Some general features are listed below.

Display features .....

(1)....Works with VAXstations and DECstations running DECwindows.

(2)....Displays 1-D data from HIS- or SPK-files.

(3)....Displays 2-D data from HIS-files.

(4)....Hardcopy available via screen-copy to LN03, LN03 ScriptPrinter

(5)....A dialog record may be saved on a Log-file.

(6)....Supports Free-Form (Banana) gate construction.

(7)....Provides for total number of counts within a Banana.

(8)....Supports X- & Y-projections of Bananas (saved on DAMQ8Q.SPK).

(9)....Provides for peak sum, centroid and fwhm.

(10)...Provides for spectrum analysis (fitting - see SEC# U300.400).

General features .....

(1)....Reads 1-D histograms from either HIS-, SPK-files.

(2)....Extracts GATES (on parameters 1 or 2) from 2-D histograms.

(3)....Supports general projections of Bananas on arbitrary axis.

(4)....Forms linear combinations, gain-shifts etc. of 1-D histograms.

(5)....Forms linear combinations, gain-shifts etc. of 2-D histograms.

(6)....Does linear gain and intercept transformations by rebinning.

(7)....Does crunches (sums a specified number of channels together).

(8)....Lists and plots 1-D histograms on the line printer.

(9)....Shows directory (ID'S) contained in HIS-, SPK- & BAN-files.

(10)...Shows count-sums for all ID's in SPK- or HIS-files.

Program operation .....

The program is controlled by a set of commands (alphabetic directives) and associated data-lists (numbers): I call these command-lists. Input is free-form. Command and list-element delimiters are BLANK , ( ) /

U300.020 GETTING STARTED

The steps given below outline how I would do it. Of course, you can do it any way that you choose or not at all.

(1)....Log onto DECstation, VAXstation, Xterminal, etc. in the usual manner.

(2)....Open a DECterm window and move it to the lower left corner of screen.

(3)....Type: @U1:[MILNER.XWIN]DAMM ;to start program on a VMS host
or: DAMM ;if defined in LOGIN.COM

Type: /usr/hhirf/damm ;to start program on an ULTRIX host
or: damm ;if defined in your .login

(4)....Type: H ;for HELP directory
(5)....Type: H ITEM ;for help on directory ITEM
(6)....Type: H FIG ;for screen configurations

(7)....Try a few FIG commands to get a feel for how they work.

(8)....Note the fact that the display required for entering fitting information (like peak positions, etc) is via the DS & DSX commands

rather than the general display commands D & DX.

(9)....Learn to use the HELP facility. That will usually be more up-to-date than this document.

(10)...Filenames have been made case-sensitive for the ULTRIX version. Where default extensions apply, upper case is assumed for VMS and lower case is assumed for ULTRIX. Acceptable standard extensions now include .spk, .SPK, .his, .HIS, .ban, .BAN, .cmd & .CMD. Note: If the his-file extension is lower/upper case then the drr-file extension must be lower/upper case. Also note:

/usr/users/directory/subdirectory/filename ;is an acceptable form but

../directory/subdirectory/filename ;is NOT! (at least for now)

U300.030 Commands for Assigning Input/Output Files

```
IN  FIL.EXT    - Open N-file (EXT = SPK or HIS)
OU  FIL.EXT    - Open O-file (EXT = SPK or HIS) - OUTPUT for SPK only
OU  FIL.SPK,NEW- Create and open O-file (SPK-file for output)
QF  FIL.EXT    - Open Q-file (EXT = SPK or HIS) - for display only
RF  FIL.EXT    - Open R-file (EXT = SPK or HIS) - for display only
SF  FIL.EXT    - Open S-file (EXT = SPK or HIS) - for display only
BAN FIL        - Open FIL.BAN for store, recall, proj, etc
BAN FIL,NEW    - Create & open FIL.BAN for store, recall, etc
             - (See below for how to specify variable FILENAMES)
CLO F          - Closes F-file (where F = N, O, P, Q, R, S or BAN)
DFIL           - Displays data files currently open

CMD FIL        - Open and process commands from FIL.CMD
CMD FIL.EXT    - Open and process commands from FIL.EXT

LON/LOF        - Turn Log-output (to LU7) ON/OFF (default = OFF)
```

Explanation of variables in FILENAMES .....

One symbol (integer variable) may be incorporated in a FILENAME specification as the following examples illustrate:

Example-1 .....

```
SYM=3
OU FIL"SYM".SPK  ;Opens FIL3.SPK
```

Example-2 .....

```
I=0
```

```
LOOP 3
I=I+1
IN FIL"I".SPK    ;Opens (in succession) FIL1.SPK, FIL2.SPK, FIL3.SPK
.
ENDLOOP
```

U300.040 Commands Related to Loop-Execution & Symbol-Definition

```
SYM = EXPRESSION - Define symbol (SYM) up to 100 symbols supported
              - symbols: UIND CIND ULOC CLOC FIX NONE FITS ALL and
              - COLR GREY DOTS LIVE BAN  M N O P Q R S are reserved
              - expression syntax is same as in CHIL
              - no imbedded blanks are allowed in expressions
              - symbols may contain up to 4 characters (5-8 ignored)
DSYM            - Displays list of currently defined sumbols & values
LOOP N          - Starts LOOP (executed N-times) N=SYM or CONST
CMD  ....       - Nesting supported
CMD  ....       - # lines between 1st LOOP & matching ENDL = 100
ENDL            - Defines end-of-loop
              - KILL (entered before END) kills LOOP
              - Ctrl/C    - aborts loop-in-progress
              - opening of CMD-file within a LOOP not allowed
```

LOOP suspension - the WO command ----------

A command WO [means the same thing as WOA or WHOA - i.e the opposite of GIDDUP (my preferred spellings)] has been implemented to work within LOOPs. Whenever the WO command is encountered (within a LOOP only, otherwise it's illegal), the message:

```
Type [RETURN] to CONTINUE--->
```

will appear on the screen. This gives you an opportunity to look at the display etc. before it gets wiped out. When you are finished looking, press the [RETURN] key and it will continue.

U300.050 Log File - damm.log

The VMS version of DAMM always creates a new version of DAMM.LOG while the ULTRIX version creates a new damm.log or appends to an old version of damm.log if it exists. If you enter the command LON, almost all dialog to and from the host will be logged, otherwise, only certain "print commands" will produce output to the log-file or device. You may turn the log ON/OFF by entering LON/LOF.

U300.060 Comments on Hard Copy

As I have defined the default the color mapping, it is best to set up the Workstation to Print Screen in the negative image mode. If you are printing on something like a LN03 ScriptPrinter, anything you chose to print will be scaled to fit on one page. If you are using an LN03, it may come out on multiple pages (and you may miss some) unless you choose a "Portion of Screen" that it likes.

U300.070 File ID-directories and Count-Sums

DIR KF Displays a list of all ID's in file-KF, where
KF left blank says input-file
KF = N denotes input-file
KF = O denotes output-file
KF = Q denotes Q-file
KF = R denotes R-file
KF = S denotes S-file
KF = BAN denotes BAN-file

LDIR KF Logs a list of all ID's in file-KF on DAMM.LOG

DDIR KF Displays ID's & # of non-zero channels for SPK-files
DDIR KF Displays ID-directory in detail for HIS-files
Also logs on DAMM.LOG if LON

DSUM KF Displays count-sums of all ID's on file-KF
Also logs on DAMM.LOG if LON

U300.090 Cursor Tracking Problems With Xterminals

The software-generated full-window cursor displayed by DAMM when in the "1-key cursor mode" requires a lot of real-time response from the host computer for live tracking of the mouse. This works fine when the host is a local workstation but does not work well for Xterminals hosted by a busy VAX. The following commands are intended to alleviate this problem. Type:

CURT LIVE ;for full-window cursor which tracks mouse "live" (default)

CURT X ;for new cursor display only for mouse-click or key-press
;(works better for Xterminals hosted by busy VAX, etc.)

Execute the desired CURT command and then FIG to make it take effect.

U300.100 Changes in Cursor-Mode Commands

I have eliminated the destinction between upper and lower case in all

cursor-mode commands. The Shift- and Caps-lock keys have no effect. In order to do this and retain meaningful command names, it was neccessary to use two keys for certain commands. These commands (UP, UX, UW, UL, UH, UO, and UB) are defined below:

P/UP Add/Delete peak to Library (pos specified by cursor)
X/UX Fix/Free peak position (for displayed peak nearest to cursor)
W/UW Fix/Free peak width (for displayed peak nearest to cursor)
L/UL Fix/Free Lo-Side ASYM (for displayed peak nearest to cursor)
H/UH Fix/Free Hi-Side ASYM (for displayed peak nearest to cursor)
O/UO Turn peak ON/OFF (for displayed peak nearest to cursor)
B/UB Add/Delete background point at cursor position

Read UP as Unset Peak, for example. As usual, no carriage return is used.
There are also a few other changes in commands. These are listed below:
LF-ARROW Set expand-region lo-limit
RT-ARROW Set expand-region hi-limit
DN-ARROW Pan DOWN - move picture so cursor-chan at right-screen
UP-ARROW Pan UP - move picture so cursor-chan at left-screen
/ Display XCUR, YCUR, channel# & energy

U300.110 Mouse Button Customizing

The Mouse Buttons can be used in place of some key-strokes while in the 1-key cursor-mode. Different button definitions are provided for the three different types of displays (namely: the displays resulting from the commands D, DD & DS). The following customizing commands are supported:
BUD L,M,R ;Defines Left,Middle,Right buttons for cursor in D-window
BUDD L,M,R ;Defines Left,Middle,Right buttons for cursor in DD-window
BUDS L,M,R ;defines Left,Middle,Right buttons for cursor in DS-window
BUD L,H,S ;Example (cursor in 1-D) L-butt sets sum-reg lo-limit,
;M-butt sets sum-reg hi-limit, R-butt requests S-sum
BUDD A,T,Z ;Example (cursor in 2-D) L-butt adds banana points,
;M-butt totalizes enclosed counts, R-butt zots banana
Buttons can't be set to / or ; - set to ? or : instead

U300.120 Screen Setup and Color Mapping

Screen configuration (placement of graphics windows on the screen) and color (or black & white) mapping is discussed in more detail in SEC# U300.550, U300.560 and U300.570 (if you have a B&W monitor, you will probably want to change the color mapping). Here we give the usual list of commands and a brief description of each.

COMMANDS RELATED TO SCREEN SETUP & COLOR MAPPING .....

```
FIGI          ;Set  screen configuration library to default
FIGF FILNAM    ;Read screen configuration library from FILNAM
FIG  N         ;Set current screen configuration to type-N

WIN  ID        ;Set subsequent displays to be in window-ID (dflt=1)

AXON ID        ;Enable  the drawing of axis for  window-ID (dflt)
AXOF ID        ;Disable the drawing of axis for  window-ID

CMAP           ;Set color map to default  ("takes" after next FIG)
CMAP FILNAM    ;Set color map from FILNAM ("takes" after next FIG)
REVV           ;Reverse all color specs   ("takes" after next FIG)

DLNS N         ;Set # disp-lines = N - for HELP, DDIR & DSUM

CURT LIVE      ;Set full-wind cursor to track mouse LIVE (default)
CURT X         ;New full-wind cursor generated via mouse-click or key
            ;The CURT command takes effect only after next FIG

SSI            ;Set screen to initial - erase all graphic windows
```

Program damm has, by default, 16 pre-defined screen connfigurations available. Each is referenced (via the FIG command) by an ID-number. A list of the ID-numbers along with the associated configuration is given below. Try a few FIGs and observe the results or if you really want to get serious, see SEC# U300.550. In particular, if you have an Xterminal with less than 1024 x 860 pixels, you will probably need to modify the configuration table as described in SEC# U300.550.

```
 1-[]     2-[][]   3-[][]   4-[][]   5-[][]   6-[][]   7-[][]   8-[][]
                []       [][]     []       [][]     []       [][]
                                  []       [][]     []       [][]
                                                    []       [][]

 9-[][]  10-[][]  11-[--]  12-[--]  13-[--]  14-[--]  15-[2D]  16-[2d]
   []       [][]            [--]     [--]     [--]
   []       [][]                     [--]     [--]
   []       [][]                              [--]
   []       [][]
```

U300.130 Display Delay (hangup problems)

Some display devices (X-terminals for example) may require a delay between successive displays. If your device "hangs up" or produces "incomplete displays" when executing LOOPS, you might try increasing the appropriate

delay. A range of 10 to 20000 miliseconds is accepted.

```
DLAF MS        ;Set FIG-delay     to MS   milliseconds (default=1000)
DLAF           ;Set FIG-delay     to 1000 milliseconds
DLAD MS        ;Set Display-delay to MS   milliseconds (default=500)
DLAD           ;Set Display-delay to 500  milliseconds
```

U300.150 Commands Related to 1-D Display

```
FIG  NF        ;Choose screen config-NF. See SEC# U300.120 & U300.550
               ;for screen configuration and color mapping commands
WIN  NW        ;Set subsequent displays to be in window-NW

LIN/LOG        ;Set display to linear/log  (default is linear)

PLON/PLOF      ;Turn peak logging ON/OFF (dflt OFF) see SEC# U300.155

ST /OV         ;Set to disp mult hist stacked/overlayed (dflt = OV)

CAL  A,B,C     ;Define energy calibration (E=A+B*Chan+C*Chan**2)

COL  I,J,K...  ;Defines color sequence for display
               ;For I,J.. = 1     2   3     4    5    6       7
               ;       COL= white,red,green,blue,yellow,magenta,cyan
GWID WID       ;Define cursor-mode sum-region width (channels)

DNOR LO,HI     ;Normalize displayed data to count-sum of chans LO,HI

DL   LO,HI     ;Set display limits (min,max chan#)

DMM  LO,HI     ;Set display limits (min,max counts)
               ;LO or HI = X  says use MIN or MAX data value

D    IDLST     ;Display histogram ID's contained in IDLST

DX   IDLST     ;Display IDLST (range defined by expand-region)
               ;IDLST format is   KF C ID, C ID.. KF C ID, C ID..
               ;C is an OPTIONAL floating-point norm-coeff (DFLT=1.0)
               ;KF = M,N,O,P,Q,R,S (default is N) and denotes:
               ;MEM-BUF, IN-FIL, OU-FIL, PROJ-FIL, Q-FIL, R-FIL, S-FIL
               ;If IDLST omitted, uses previously defined IDLST

SUML LO,HI     ;Define sum region for SUM command below
SUM  IDLST     ;Sum counts (LO,HI) (IDLST same as D except C illegal)

C              ;Enter cursor-mode
```

SSI ;Set screen to initial - erase graphic windows

(continued on next page)

U300.150 Commands Related to 1-D Display (continued)

1-KEY CURSOR COMMANDS FOR 1-D DISPLAY .....

<-- (LF-ARROW) ;Set expand-region lo-limit
--> (RT-ARROW) ;Set expand-region hi-limit

V ;Make marker display visible/invisible (toggles)
E ;Expand display

M ;Turn marker display ON
K ;Turn marker display OFF (K is for kill)

UP-ARROW ;Pan up - move picture so cursor-chan at left-screen
DN-ARROW ;Pan down - move picture so cursor-chan at right-screen
;(you must be expanded to execute pan)

L ;Set sum-region lo-limit
H ;Set sum-region hi-limit
G ;Set sum-region limits (LO=XCUR, HI=XCUR+WID-1)

S ;Display sum, centroid, fwhm (2.354*sigma) of sum region
;DATA(LO),DATA(HI) defines BGD for NETS

A ;Display sum, centroid, fwhm (2.354*sigma) of sum region
;YCUR(LO),YCUR(HI) defines BGD for NETS

C ;Draw peak-marker and display chan# at cursor pos
P ;Draw peak-marker and display energy at cursor pos

/ or ? ;Display XCUR, YCUR, channel# & energy
; ;Same as ? except forces logging (see SEC# U300.155)

Q ;Quit cursor-mode (return to normal-mode)

See SEC# U300.110 or Type: h mous for use of Mouse Buttons.

U300.155 Peak Finding/Logging

The following commands control peak finding.

FIND BIAS,IFWHM ;Turn peak-find ON (see definitions below)

```
FIND            ;Turn peak-find ON  (with defaults - see below)
NOFI            ;Turn peak-find OFF (default is OFF)
```

BIAS...is the number of standard deviations above background that a peak channel must be in order to be considered as part of a peak. Useful values of BIAS are in the range 3 to 10. The default value is 5.0.

IFWHM..is the approximate full-width at half-max (in channels) of peaks in the region of interest. This value is not very critical but should be within a factor of 2 or so of the correct value. The default value is 5.

All peaks found within the display region will be marked & labeled with the associated energy-calibration value (def_ault is the same as channel number). Peak labels are integers (no decimals - to minimize screen space used) so if you want labels to be in units of keV, for example, you must enter CAL such that E(keV) is a whole number.

See SEC# U300.430 for how peak-finding is used in fitting operations.

Logging "found" and "marked" peaks on damm.log

Peaks which either found via the FIND command above or marked via the 1-key command (/ or ? or ;) may be logged on damm.log. The following commands (independent of LON/LOF) turns said logging ON and OFF.

```
PLON          ;Turns peak logging (to damm.log) ON
PLOF          ;Turns peak logging (to damm.log) OFF (default)
```

For found peaks, damm.log may be read (skipping prog, date & time) as:

```
      READ(LU,10)IFLG,ID,CH,HEFT,(FILNAM(I),I=1,16)
   10 (29X,A4,I10,2F10.0,2X,16A4)
```

Where: IFLG = 'PEAK' for found peak log entry
ID = Spectrum ID number
CH = Peak location in channels
HEFT = Peak heftiness
FILNAM = First 64 characters of spk- or his-filename

For marked peaks, damm.log may be read (skipping prog, date & time) as:

```
      READ(LU,10)IFLG,ID,CH,ENER,(FILNAM(I),I=1,16)
   10 (29X,A4,I10,2F10.0,2X,16A4)
```

Where: IFLG = 'MARK' for marked peak log entry
ID = Spectrum ID number
CH = Peak location in channels

ENER = Peak "energy" from calibration constants
FILNAM = First 64 characters of spk- or his-filename

U300.160 Commands Related to 2-D Display

```
FIG   NF       ;Choose screen config-NF. See SEC# U300.120 & U300.550
               ;for screen configuration and color mapping commands

WIN   NW       ;Set subsequent displays to be in window-NW

ZLEV  N        ;Set # of color/grey-scale intensity levels to N
ZLEV           ;Set # of color/grey-scale intensity levels to 10 (dflt)

GRAS  I,J,K..  ;Set grey-scale values (range 0-100) & ZLEV (# entries)
               ;Must do after first FIG  ("takes" after next FIG)

ZINT  COLR     ;Set 2-D intensity mapping to Color (default)
ZINT  GREY     ;Set 2-D intensity mapping to Grey-scale
ZINT  DOTS     ;Set 2-D intensity mapping to Dot-matrix (3x3 or 5x5)

ZSON/ZSOF      ;Z-scale ON/OFF - displays cnts vs colors legend (dflt)

ZLIN/ZLOG      ;Set 2-D display to linear/log (default is log)

XC    LO,HI    ;Set min & max X-channel numbers for display
YC    LO,HI    ;Set min & max Y-channel numbers for display

ZMM   LO,HI    ;Set min & max counts/channel    for display & count-sum
ZMM   LO       ;Sets min & searches for max (semi-autoscale)
ZMM            ;Searches for min & max      (full-autoscale)

DD       ID    ;Display 2-D histogram ID from IN-file
DD    KF,ID    ;Display 2-D histogram ID from KF-file

DDX      ID    ;Display 2-D histogram ID from IN-file (expand region)
DDX   KF,ID    ;Display 2-D histogram ID from KF-file (expand region)
               ;Where KF = N,O,Q,R,S (default is N) and denotes:
               ;IN-FIL, OU-FIL, Q-FIL, R-FIL, S-FIL
               ;ID (and KF) omitted says use previously defined spec

DDID           ;Shows ID-number & Filename for current 2-D display

ZBL            ;Zero in-core BAN-library

LBL            ;List in-core BAN-ID numbers
```

C ;Enter cursor-mode

SSI ;Set screen to initial - erase graphic windows

U300.160 Commands Related to 2-D Display (continued)

1-KEY CURSOR COMMANDS FOR 2-D DISPLAY .....

<-- (LF-ARROW) ;Set expand-region lo-left- limit
--> (RT-ARROW) ;Set expand-region hi-right-limit

V ;Make expand markers visible/invisible (toggles)

E ;Expand

1 ;Move display such that cursor is at lo-left
2 ;Move display such that cursor is at hi-left
3 ;Move display such that cursor is at hi-right
4 ;Move display such that cursor is at lo-right

Z ;Zero (open) active X,Y-list
A ;Add point to active X,Y-list
D ;Dele nearest point in active X,Y-list
M ;Move nearest point in active X,Y-list to cursor pos
I ;Insert a point in active X,Y-list at cursor position

L ;List active X,Y-list (on VDT)

B ;Draw active BAN & BAN's in in-core library

G ;Prompt for ID & read into in-core BAN-library

O ;Open nearest in-core BAN for modification

S ;Prompt for ID & store in in-core library & on disk

R ;Store nearest BAN with original ID

F ;Remove nearest BAN from in-core library & erase

K ;Delete nearest BAN from in-core library and disk

T ;Totalize counts in nearest BAN (active or not)

P ;Totalize and & save X- & Y-projections on DAMQ8Q.SPK

/ or ? ;Display X,Y-coordinates of cursor

Q ;Quit cursor-mode (return to normal-mode)

See SEC# U300.110 or Type: h mous for use of Mouse Buttons.

U300.200 Command Syntax - General Definitions

B1 - Memory Buffer-1
B2 - Memory Buffer-2
ID - The ID-number of histogram to be read
NUID - Next ID-number to be assigned to output histogram
LO - A first-channel-number (usually of a Gate)
HI - A last- channel-number (usually of a Gate)
FAC - A multiplication factor

Meaning of the individual command-characters .....

I - Input or read
O - Output or write
A - Add or accumulate
S - Shift (gain shift)
GX - Gate on X-parameter (i.e. parameter # 1)
GY - Gate on Y-parameter (i.e. parameter # 2)
1 - Buffer-1
2 - Buffer-2
M - Multiply
C - Crunch
D - Divide

U300.210 Commands for Setup (no immediate action)

NUID IV - Set next ID to be used to IV

IDST N - Set ID-step to be used in implied I/O loops
(remains active until changed - default=1)

CRUN IVAL - Sets standard crunch value to IVAL

GASP XI1,XI2,XF1,XF2,NCF - Standard gain shift specification

SIDA - Says treat 16-bit HIS-file data as signed
USDA - Says treat 16-bit HIS-file data as un-signed (default)

U300.220 Commands for Input/Output of 1-D Histograms

I ID Input to B1
IS ID Input to B1, gain shift B1
IA ID,FAC Input to B1, B2=B2+FAC*B1
ISA ID,FAC Input to B1, shift B1, B2=B2+FAC*B1
IO ID Input to B1, output B1
ISO ID Input to B1, shift B1, output B1
IO IDA,IDB Input to B1, output B1 (for ID=IDA,IDB)
ISO IDA,IDB Input to B1, shift, output (for ID=IDA,IDB)
O1 Output B1
O2 Output B2

U300.230 Commands for Gating 2-D Histograms

GY ID,LO,HI Y-gate to B1
GYS ID,LO,HI Y-gate to B1, shift B1
GYO ID,LO,HI Y-gate to B1, output B1
GYO IDA,IDB,LO,HI Y-gate to B1, output B1 (for ID=IDA,IDB)
GYSO ID,LO,HI Y-gate to B1, shift & output B1
GYSO IDA,IDB,LO,HI Y-gate to B1, shift & output B1 (for ID=IDA,IDB)
GYA ID,LO,HI,FAC Y-gate to B1, B2=B2+FAC*B1
GYSA ID,LO,HI,FAC Y-gate to B1, shift B1, B2=B2+FAC*B1
GX ID,LO,HI X-gate to B1
GXS ID,LO,HI X-gate to B1, shift B1
GXO ID,LO,HI X-gate to B1, output B1
GXO IDA,IDB,LO,HI X-gate to B1, output B1 (for ID=IDA,IDB)
GXSO ID,LO,HI X-gate to B1, shift & output B1
GXSO IDA,IDB,LO,HI X-gate to B1, shift & output B1 (for ID=IDA,IDB)
GXA ID,LO,HI,FAC X-gate to B1, B2=B2+FAC*B1
GXSA ID,LO,HI,FAC X-gate to B1, shift B1, B2=B2+FAC*B1
O1 Output B1
O2 Output B2

U300.240 Commands for General 2-D Projections

PJ ID,BID,DEGR PROJ TO B1
PJS ID,BID,DEGR Proj to B1, shift B1
PJO ID,BID,DEGR Proj to B1, output B1
PJO IDA,IDB,BIDA,BIDB,DEGR - Proj to B1, output B1
(outer loop on BID, inner loop on ID)
PJSO ID,BID,DEGR Proj to B1, shift & output B1
PJSO IDA,IDB,BIDA,BIDB,DEGR - Proj, shift, output
(outer loop on BID, inner loop on ID)
PJA ID,BID,DEGR,FAC Proj to B1, B2=B2+FAC*B1
PJSA ID,BID,DEGR,FAC Proj to B1, shift B1, B2=B2+FAC*B1

PJAL Project all bananas in currently open
BAN-file for HIS-files, ID's & DEGR'S
contained therin
O1 Output B1
O2 Output B2
ID denotes histogram ID, BID denotes Banana ID
(DEGR = Projection-axis angle in degrees)

U300.250 Commands for Operations on Buffer-1 & Buffer-2

M1 XM Multiply B1 by XM
M2 XM Multiply B2 by XM
C1 ICRUN Crunch B1 by ICRUN (standard crunch unchanged)
C2 ICRUN Crunch B2 by ICRUN (standard crunch unchanged)
S1 Shift B1 by standard GASP
S2 Shift B2 by standard GASP
S1 XI1,XI2,XF1,XF2,NCF - Shift B1 as specified
(standard GASP unchanged)
S2 XI1,XI2,XF1,XF2,NCF - Shift B2 as specified
(standard GASP unchanged)
Z1 Zero B1
Z2 Zero B2
A12 FAC B2=B2+FAC*B1
A21 FAC B1=B1+FAC*B2
SWAP Swap B1 & B2
M2D1 FAC B2=(FAC*B2)/B1
O1 Output B1
O2 Output B2

U300.260 Commands which Show Data, Count-Sums etc (from Bufs-1 & -2)

PR1 Print Buffer-1
PR2 Print Buffer-2

D1 LO,HI Display Buffer-1 (channels LO thru HI)
D2 LO,HI Display Buffer-2 (channels LO thru HI)

SUM1 LO,HI Display sum of counts LO-thru-HI of B1
SUM2 LO,HI Display sum of counts LO-thru-HI of B2

COMP NCH Compare first NCH-channels of B1 & B2
(gives # counts and # mis-matches)

GEN ID,KO,KX,NCH Generate test spectrum in B1 (NCH channels)
Channel contents = KO+KX*(channel#+1)

U300.270 Commands which Modify Buffer Contents

SET1 ICN,YV Set channel ICN of B1 to YV
SET2 ICN,YV Set channel ICN of B2 to YV
SET1 LO,HI,YV Set channels LO-thru-HI of B1 to YV
SET2 LO,HI,YV Set channels LO-thru-HI of B2 to YV
SET1 LO,HI,YA,YB Set channels LO-thru-HI of B1 to YA-thru-YB
SET2 LO,HI,YA,YB Set channels LO-thru-HI of B2 to YA-thru-YB
ADD1 ICN,YV Add YV to channel ICN of B1
ADD2 ICN,YV Add YV to channel ICN of B2
ADD1 LO,HI,YV Add YV to channels LO-thru-HI of B1
ADD2 LO,HI,YV Add YV to channels LO-thru-HI of B2
ADD1 LO,HI,YA,YB Add YA-thru-YB to channels LO-thru-HI of B1
ADD2 LO,HI,YA,YB Add YA-thru-YB to channels LO-thru-HI of B2
(i.e. a strait line)

U300.280 Commands Related to Printer Plots

SKRZ Set to skip repeated-zeros for printer plots
PLRZ Set to plot repeated-zeros for printer plots

PLG ID,LO,HI,NCYC - Input to B1 & LOG plot
PLN ID,LO,HI,NCFS - Input to B1 & LIN plot
PLG IDA,IDB,LO,HI,NCYC - Input to B1 & LOG plot (for ID=IDA,IDB)
PLN IDA,IDB,LO,HI,NCFS - Input to B1 & LIN plot (for ID=IDA,IBD)

PLG1 LO,HI,NCYC Log Printer-plot of Buffer-1
PLG2 LO,HI,NCYC Log Printer-plot of Buffer-2
PLN1 LO,HI,NCFS Linear Printer-plot of Buffer-1
PLN2 LO,HI,NCYC Linear Printer-plot of Buffer-2

(NCFS = # of counts full-scale for LIN plots)
(NCYC = # of cycles for LOG plots)

U300.290 Discussion of Gain-Shifts and Compressions

Gain shifts are specified by five parameters - XI1, XI2, XF1, XF2 and NCF. XI1 and XI2 represent two locations (in channel-# units) in the initial 1-D histogram and XF1 and XF2 represent corresponding locations in the final histogram (i.e. after the transformation). That is:

XF=A+B*XI
where,
B=(XF2-XF1)/(XI2-XI1)
and
A=XF1-B*XI1

NCF gives the number of channels in the histogram after the transformation. If NCF=0, the final #-of-channels is determined by the initial #-of-channels NCI and the transformation specified. If NCF=-1 the final #-of-channels is set equal to NCI. Counts are redistributed into the final set of channels (bins) by assuming a uniform distribution of counts in the initial bins. Data shifted below channel-#-0 and above channel-#-NCF-1 are lost and gone forever.

Gain-shifts are always "in place"
CRUN IVAL (i.e. standard crunch) does it at "input time"
Data is kept internally as floating - is converted to fixed on output
All output from DAMM is 32 bits/channel

U300.300 Manipulation of HIS-files (Copy, Add, Gain-Shift, etc)

DAMM can copy, add (or subtract) and gain-shift 1-D or 2-D histograms from an input HIS-file to an output HIS-file.

(1)....All operations are from an INPUT-file and INPUT-ID (IDI) to an OUTPUT-file and OUTPUT-ID (IDO).

(2)....For HCOP and HADD operations, the output histogram must have the same dimensions and ranges as the input histogram.

(3)....For SHIF (gain-shift) and SHAD (gain-shift & add) operations, the dimensions of the output histogram need not match the input.

(4)....The number of bits-per-channel (16 or 32) need not be the same for input and output.

(5)....Gain-shifts are accomplished by converting the data to floating point, rebinning (with count fractionation) and finally converting back to integer.

(6)....Fractional copies and adds are also done in floating point.

(7)....Final conversion from floating point to integer involves the addition of a random number whose range is 0.0 to 1.0. This procedure results in slight differences in the total number of counts for the input and output histograms.

Use CHIL to create output DRR-file and allocate HIS-file as usual.

HOU DISK:FIL.HIS - Opens HIS-file for output

HCOP IDI,IDO <,F> - Copies F*IDI (input) to IDO (output)
(If F is not entered, F=1)

HADD IDI,IDO <,FI><,FO> - Adds FI*IDI to FO*IDO
(If FI is not entered, FI=1)
(If FO is not entered, FO=1)
(If FO is entered, FI must be entered)

HDIV IDI,IDO <,FI> - Divides FI*IDI by IDO & saves in IDO

SHIF IDI,IDO <,FI> - Gain-shifts IDI & stores in IDO
SHAD IDI,IDO <,FI><,FO> - Gain-shifts IDI & adds to IDO

GSX XI1,XI2 XF1,XF2 - Defines X-gain-shift (described below)
GSY YI1,YI2 YF1,YF2 - Defines Y-gain-shift (described below)
GSXOF - Turns off X-gain-shift
GSYOF - Turns off Y-gain-shift
HSET IDO,IV - Sets IDO on output-file to IV
HZOT IDO - Sets IDO on output-file to 0
HSTA - Shows files open & gain-shift data

U300.300 Manipulation of HIS-files (continued)

X- and Y-Gain-shifts

X-gain-shifts are specified by the parameters - XI1, XI2, XF1 & XF2.
Y-gain-shifts are specified by the parameters - YI1, YI2, YF1 & YF2.

For an X-gain-shift, XI1 and XI2 represent two locations (in channel # units) in the initial spectrum and XF1 and XF2 represent corresponding locations in the final spectrum (i.e. after transformation). that is:

XF=A+B*XI
where: B=(XF2-XF1)/(XI2-XI1)
and A=XF1-B*XI1

The "final" # of channels is determined by the "initial" # of channels and the transformation specified. Counts are redistributed into the final set of channels (bins) by assuming a uniform distribution of counts in the initial bins. Data shifted out of the range of the final histogram are lost and gone forever!

The rules and procedures are identical for Y-gain-shifts.

COMMENTS

(1)....If gain-shift specifications are not given (or turned off), bin-widths will be the same for output and input.

(2)....Any data which does not fall within the ranges of the output histogram will be lost (without comment).

(3)....Data will be properly positioned in the output histogram even if the ranges of the input and output are different. That is, data will appear in that region of the output histogram which overlaps the gain-shifted input histogram.

U300.350 Bananas - Definition, Rules, Construction & Display

Free-form-gates (or Banana-gates - Bananas for short) are 2-D regions of arbitrary shape which are specified by a list of X,Y-points (channel-# coordinates). Each Banana on a given BAN-file is stored and recalled by means of an identification number (ID #). Attempts to store two Bananas with the same ID will be rejected. The rules for Bananas are listed below:

(1) Banana coordinates nust be given in CLOCKWISE order.
(2) The Banana is formed by connecting X,Y-points with strait lines.
(3) The last point is connected to the first by the program.
(4) No line segment of the Banana should intersect another.
(5) A maximum of 63 points may be specified for any one Banana.
(6) A maximim of 880 Bananas may be stored on a given BAN-file.

Bananas may be displayed in two different forms (OPEN and CLOSED).

A CLOSED Banana is one which has just been recalled from or stored on a BAN-file (i.e. there is an exact image on disk). There may be up to 20 CLOSED Bananas displayed at once. You can do the following things with a CLOSED Banana:

GET - recall from disk (prompted for ID) by typing G
OPEN - for modification (change to OPEN) by typing O
FORGET - delete from display by typing F
KILL - delete from display and BAN-file by typing K
TOTALIZE - counts contained within Banana by typing T
PROJECT - (X & Y) and save on DAMQ8Q.SPK by typing P

An OPEN Banana is one which is open for creation or modification. If the Banana is being newly created there will be no corresponding image or partial image on a BAN-file. Only one such Banana can exist at a given time. You can do the following things with a OPEN Banana:

ADD - X,Y-point at cursor position by typing A

INSERT - X,Y-point at cursor position by typing I
MOVE - nearest X,Y-point to cursor position by typing M
SAVE - on BAN-file (prompted for ID) by typing S
REPLACE - on BAN-file (with old ID) by typing R
ZERO - all X,Y-points by typing Z
TOTALIZE - counts contained within Banana by typing T
PROJECT - (X & Y) and save on DAMQ8Q.SPK by typing P

All Banana references are made in cursor mode. ADD, INSERT, MOVE, SAVE, REPLACE and ZERO refer only to the OPEN Banana. Other references (except for GET) are made by moving the cursor such that it is closer to some point on the Banana of interest than it is to any point on any other Banana.

ALL BANANAS MUST BE CONSTRUCTED IN CLOCKWISE ORDER

U300.360 Bananas - Projections

Projections via the PJ-command

Data which fall within and on the boundries of a Banana are projected onto the X-axis of a coordinate system which is rotated through an angle DEGR with respect to the systen in which data channel-# (0,0) falls at the origin and the first and second indices of the histogram array define the X- and Y-axis, respectively. Channel-# XP in the projected histogram is calculated from channel-# X,Y in the 2-D histogram by an expression of the following form:

XP=A+COS(DEGR)*X+SIN(DEGR)*Y

where,

A=0.0 For DEGR = 0 - 90

A=-COS(DEGR)*XMAX For DEGR = 90 - 180

A=-COS(DEGR)*XMAX-SIN(DEGR)*YMAX For DEGR = 180 - 270

A=-SIN(DEGR)*YMAX For DEGR = 270 - 360

XMAX and YMAX are the "dimensions" of the 2-D histogram. The effect of this transformation is to make all channel numbers in the projected histogram positive.

NOTE: The "length" of the projected histogram may be as large as

SQRT(XMAX**2+YMAX**2).

Projections via the P-command

Each time DAMM is executed it will create a new version of the file DAMQ8Q.SPK for the storage of projections. The file is only created if projrctions are actually made.

When you project a Banana, both X- and Y-projections are stored on DAMQ8Q.SPK under the ID-numbers displayed. These 1-D histograms may be displayed (or otherwise used) in the normal manner for a SPK-file. Use the P-qualifier to display spectra from DAMQ8Q.SPK without explicitally opening it. For example, to display ID numbers 1,3,5 from DAMQ8Q.SPK, type:

D P 1,3,5

U300.400 FITTING - Introduction & General Features

You specify how fitting is to be carried out by supplying a number of Fit Specification Data Sets which may be given in any order. Many of these have default values (see SEC# U300.450). After the fitting process is specified, one or more Fit Requests are entered. Subsequently, some or all of the Fit Specifications may be changed and more Fit Requests entered etc. etc.

GENERAL FEATURES

(1)....Fit specifications may be entered interactively or read from a file or a combination of the two methods may be used.

(2)....Peak and background intensities are determined in a weighted linear least-squares fit while peak positions, widths, and asymmetry parameters are determined by a non-linear least-squares search (either Gradient search or Gauss method - See SEC#s U300.480 and U300.490).

(3)....Peak positions may be typed in or selected interactively or found automatically.

(4)....Spectra are fitted one section at a time and can be no more than 512 channels in length.

(5)....In the gradient search mode (FIT command), each section may contain a total of 16 peaks and background terms. That is, the number of linear coefficients to be determined in the linear least squares fit (# of peaks plus # of background terms) may not exceed 16. In

the Gauss mode, only 5 peaks are allowed and asymmetry is not supported.

(6)....Initial values of peak positions, widths and asymmetry parameters must be specified by the user. Different values of width and asymmetry may be assigned to each peak or all peaks may be assigned the same values.

(7)....The FWHM for peaks in a section may vary independently, conditionally, or be held fixed. All peaks in a section may be forced to have the same width or fixed relative widths.

(8)....Peak positions may be adjusted or held fixed.

(9)....Peaks may be gaussian or asymmetric (see SEC# U300.470 & U300.530

(10)...The background may be specified (by up to 50 X,Y-points) or be determined in the fit. If determined in the fit, the background takes the form, Y = A + B*X + C*X*X + D*X*X*X + ... with the number of terms in the power series specified by the user.

(11)...The output includes the Fit Specification Data, peak positions, widths, energies, areas and uncertainties (in percent) as well as a printer plot of the fit on a 0.5 to 5 cycle plot.

U300.410 Commands for 1-key (cursor mode)

One-Key cursor commands (valid following a DS or DSX command)

Type: C - To enter cursor-mode

P/UP Add/Delete peak to Library (pos specified by cursor)

M/M Move nearest displayed peak to cusor pos (FW, ASYM unchanged)

X/UX Fix/Free peak position (for displayed peak nearest to cursor)

W/UW Fix/Free peak width (for displayed peak nearest to cursor)

L/UL Fix/Free Lo-Side ASYM (for displayed peak nearest to cursor)

H/UH Fix/Free Hi-Side ASYM (for displayed peak nearest to cursor)

O/UO Turn peak ON/OFF (for displayed peak nearest to cursor)

B/UB Add/Delete background point at cursor position

<-- Set Expand Region Lo-Limit

--> Set Expand Region Hi-Limit

[Set Fit Region Lo-Limit

] Set Fit Region Hi-Limit

/ or ? Display chan#, cursor Y-value, chan contents

S Disp sum, cent & fwhm of Fit-Reg - DAT([),DAT(]) defines BGD

A Disp sum, cent & fwhm of Fit-Reg - CUR([),CUR(]) defines BGD

Q Return from cursor control routine

E Expand display (region defined by <-- -->)

See SEC# U300.110 or Type: h mous for use of Mouse Buttons.

U300.420 Setup Commands

Commands for entry of peak, background & skip-regions ----------

PZOT - Zero the Peak Library
PK X,W,ASLO,ASHI - List of complete peak specifications

BZOT - Delete Fixed Background array
BACK X1,Y1 X2,Y2 .. - X,Y-points for fixed background

SKIP - Without List turns SKIP OFF
SKIP I1,I2 J1,J2 .. - Up to 4 regions to omit from Fit

Commands for defining FWHM, ASYM, WLIM, ALIM, NBC, WOOD, ECAL ----------

FW FWA,FWB,FWC - Coefficients for standard width function
WLIM FWLO,FWHI - Variation limit factors for peak widths

ASYM ASLO,ASHI - Standard Lo-Side and Hi-Side asymmetries
ALIM FALO,FAHI - Variation limit factors for peak asymmetries

NBC NBC - Number of power series terms in variable BGD
WOOD ON/OFF - Turn Woods-Saxon BGD term ON/OFF (default OFF)

- ON creates an additional background component
- with a Woods-Saxon "jog" under each peak which
- is porportional to the peak intensity.

ECAL ECO,ECA,ECB - Coefficients for standard energy calibration

Commands for control of non-linear parameter variation ----------

DPX XSTEP,DXMAX - Initial step size and limit for peak pos
DEL DEL,DELFAC,NDEL - Initial step size, step size multiplier and
- number of DEL-values to use

VB - Use Variable Background (the default)
FB - Use Fixed Background if available

VX KVAR - Kind of variation for peak positions
VW KVAR - Kind of variation for peak widths
VALO KVAR - Kind of variation for Lo-Side asymmetries
VAHI KVAR - Kind of variation for Hi-Side asymmetries
KVAR = UIND - says vary Unconditionally, Independently
= CIND - says vary Conditionally, Independently
= ULOC - says vary Unconditionally, Locked
= CLOC - says vary Conditionally, Locked
= FIX - says keep Fixed - this the default assignmemt

Conditional says hold Fixed if peak so specified.
Unconditional says vary regardless of peak specifications.
Independent says given parameter-types are varied independently.
Locked says given parameter-types (width for example) are varied together
(multiplied by the same factor) in the non-linear search.

U300.430 Display of Data, Fits and Printer-plots

Commands for general display control ----------

FIG N - Select screen configuration number-N
WIN N - Select window-N for subsequent displays
LIN/LOG - Set graphic display to LIN (default) or LOG
DMM YMIN,YMAX - Set display-range (YMIN & YMAX)
DL ILO,IHI - Set display-range (channel# limits)
DS ID - Display spectrum# ID (range defined by DL)
DS ID,ILO,IHI - Display spectrum# ID (DL values replaced)
(MAX value of IHI-ILO = 4095)
DSX ID - Display Data defined by Expand Region
C - Enters 1-key cursor-mode

Commands related to display of FITS ----------

MON/MOF - Peak Markers ON/OFF for DF (default = ON)
DFI - Set to display (DF) DATA,FIT,BGD (default)
DPK - Set to display (DF) DATA,FIT,PEAKS,BGD
DPPB - Set to display (DF) DATA,FIT,(PEAKS+BGD),BGD
DF - Display Fit (channel-limits given by Fit-range)
DF ILO,IHI - Display Fit (channel-limits given by ILO,IHI)
DC NPK - Display Calculated peak # NPK+ RESIDUAL

PRP XLO,XHI - Display peaks from Library in range XLO thru XHI
PRP - Display all peaks from Library
PRB - Display all fixed Bgd-points
FSTAT - Display current fit-parameters
DR - List results of last Fit on VDT (terminal)

Commands related to printer-plots of results ----------

KPPL NONE - Says do no printer plots
KPPL FITS - Says plot FITS only (the default)
KPPL ALL - Says plot FITS, COMPONENTS and RESIDUALS

PR - Print and Plot results of last Fit on printer

Commands related to peak-finding ----------

FIND BIAS,FWHM - Enables peak-finding (see SEC# U300.155)
FIND - Enables peak-finding with (BIAS=5, FWHM=5)
NOFI - Disables peak-finding

If FIND is enabled (see SEC# U300.155 for general details), DAMM will do a peak find within the display region each time a DS (or DSX) command is given. An attempt will then be made to add the newly found peaks to the internal peak library. If a newly found peak is closer than 0.5*FWHM channels to an existing library peak, it will not be added. Finally, all library peaks will be marked on the display in the usual manner. No distinction is made between "found peaks" and "manually entered peaks".

U300.440 FIT Execution Commands

Commands for FIT execution ----------

FIT ID,ILO,IHI - Fit Request - (non-linear gradient search)
GFIT ID,ILO,IHI - Fit request - (gaussian method)
RFIT ID,ILO,IHI - Resume FIT/GFIT start with Parms from last Fit
LFIT ID,ILO,IHI - Linear Fit - no non-linear search

- (Fit-range specified by ILO,IHI)
FIT ID X - Fit Range specified by cursors (Fit Region)
GFIT ID X - Fit Range specified by cursors (Fit Region)
RFIT ID X - Fit Range specified by cursors (Fit Region)
LFIT ID X - Fit Range specified by cursors (Fit Region)

Ctrl/C - Terminates Fit-in-prograss (VAX)

U300.450 FIT Parameters - Saving, Setting, Default

Commands which save FIT parameters in memory library ----------

SAV I,J - Save all Parms from peaks I thru J of last Fit in PK-LIB
SAX I,J - Save X-Parms for peaks I thru J of last Fit
SAW I,J - Save W-Parms for peaks I thru J of last Fit
SAL I,J - Save ASL-Parms for peaks I thru J of last Fit
SAH I,J - Save ASH-Parms for peaks I thru J of last Fit
(If I,J ommitted, indicated Parms from ALL peaks are saved)

Commands which set FIT parameters ----------

SET- X1,X2 - Set STD WIDTH and ASYM for peaks in range X1-X2
values (defined by FWA, FWB, FWC, ASLO, ASHI)

SETW X1,X2 - Set WIDTH for peaks in range X1-X2 to STD value
SETW X1,X2,WA,WB,WC - Set WIDTH for peaks in range X1-X2 to value
defined by WA,WB,WC (FWA,FWB,FWC unchanged)

SETL X1,X2 - Set ASLO for peaks in range X1-X2 to STD value
SETL X1,X2,ASLOT - Set ASLO=ASLOT for peaks in range X1-X2

SETH X1,X2 - Set ASHI for peaks in range X1-X2 to STD value
SETH X1,X2,ASHIT - Set ASHI=ASHIT for peaks in range X1-X2

(If X1,X2,.. omitted, indicated Parms for ALL peaks are set)

List of default FIT parameters ----------

DEL = 0.05 FWLO = 0.5 VX = CIND NBC = 2
DELFAC=0.25 FWHI = 2.0 VW = CLOC WOOD = OFF
NDEL = 1 FALO = 0.5 VALO = FIX KPPL = FITS
XSTEP = 0.5 FAHI = 2.0 VAHI = FIX
DXMAX = 5.0 ASLO = 0.0
ASHI = 0.0

U300.460 FIT Specification Details

PK Data Set - Complete Peak Specifications

The PK Data Set accomodates a full specification of the characteristics of each individual peak. Up to 100 peaks may be included in the list. Each peak is specified by the following parameters.

X.....Gives the initial peak position in channels.

W.....Specifies the initial peak FWHM in channels. If not entered, FWHM is set to the value specified by FWA, FWB & FWC.

ASLO...Specifies the Lo-Side asymmetry parameter.

ASHI...Specifies the Hi-Side asymmetry parameter.

Other Specifications

ECO,ECA,ECB...Defines the spectrum energy calibration (not required for fitting) through the relation;

E = ECO + ECA*(CHAN #) + ECB*(CHAN #)**2

FWA,FWB,FWC...Defines the peak WIDTH as a function of channel number through the relation;

FWHM(CHANNELS)=FWA+FWB*SQRT(CHAN #)+FWC*(CHAN #)

ASLO,ASHI...Are the initial values of the Lo-Side and Hi-Side asymmetry parameters. If this specification is used, the initial values will be the same for all peaks.

FWLO...Is the minimum fraction of the initially specified value by which any peak width may be reduced.

FWHI...Is the maximum fraction of the initially specified value by which any peak width may be increased.

FALO...Is the minimum fraction of the initially specified value by which any peak asymmetry parameter may be reduced.

FAHI...Is the maximum fraction of the initially specified value by which any peak asymmetry parameter may be increased.

DEL....Specifies the fraction by which the peak width and the peak asymmetry parameters are to be changed in each step of the non-linear search. For example,

(NEW WIDTH) = (OLD WIDTH)*(1.0+-DEL)

DELFAC-Is a factor by which the current value of DEL is multiplied in order to obtain a new (smaller) value. Typically one starts with a fairly large value of DEL (say 0.02 to 0.05) and subsequently makes one or more reductions in order to achieve a greater convergence speed.

NDEL...Is the number of DEL-values to be used

XSTEP..Is the maximum amount (in channels) that a peak may be moved in any one step in the non-linear search for the best fit. XSTEP is reduced at the same time and by the same factor (DELFAC) that DEL is reduced.

DXMAX..Is the maximum number of channels (either + or -) that any peak is allowed to be moved from its original position.

SKIP...Defines up to four regions within the Fit Range which are to be ignored in doing the fit.

KPPL=..NONE says do no printer plots.

KPPL=..FITS says plot the FIT (experimental and calculated spectrum on the same graph).

KPPL=..ALL says plot the FIT (as in KPPL...=FITS) and in addition, plot each component (calculated peak) together with the corresponding "residual component". What do you mean by residual component, you ask. When plotting the Ith peak we calculate the Ith residual component by subtracting any background (specified or calculated) as well as all calculated peaks other than the Ith from the experimental spectrum.

NBC....Denotes the number of background components to be included in the power series. NBC=2 Says use the form Y=A+B*X and NBC=4 says use Y=A + B*X + C*X*X + D*X*X*X. The number of peaks in a section plus NBC must not exceed 16.

WOOD...ON/OFF says turn Woods-Saxon background term ON/OFF. The default is OFF. If WOOD is ON, an additional background component is included which has a Woods-Saxon type "jog" under each peak which is porportional to the peak intensity. The jog form is given by:

Y = 1.0/(1.0+EXP(ARG)) ;where

ARG = 4.714*(X0-X)/FWHM ;and

X0=peak-position, X=channel-of-interest, and FWHM=peak-FWHM.

The use of such a background form could be helpful in the analysis of weak peaks which are located on the low-energy side of strong peaks. You will have to be the judge.

U300.470 Discussion of Peak Shapes

The most general peak shape allowed is given by

YL=EXP(-(X-XO)**2/(A**2*(1+ASLO*(X-XO)/A)
YH=EXP(-(X-XO)**2/(A**2*(1+ASHI*(X-XO)/A)

Where A is the gaussian Half-Width at 1/e max and YL and YH describe the curve on the Lo- and Hi-Sides, respectively. If all asymmetry parameters are held to zero, the shape is gaussian. The ASLO/ASHI parameters broaden the Lo/Hi sides of the peak and result in an expodential fall-off (like EXP(-(X-X0)/(A*ASLO)) for example) As you move far away from the peak maximum (i.e. channel XO). To get some idea of what size asymmetry parameters to use see Fig 1.

U300.480 Gradient-search Method (FIT request)

Each time the program encounters a Fit Request, it searches the complete Library and includes in the Fit all peaks which are ON and whose positions lie within the Range of Fit (i.e. between ILO and IHI).

GENERAL PROCEDURE FOR THE NON-LINEAR SEARCH

(1)....The initial values of all parameters which are to vary in a non-linear way are set to the initial values specified by the user.

(2)....Each individual parameter is changed (both increased and decreased) by an amount determined by DEL or XSTEP in order to establish a "direction" (increase or decrease) for each parameter.

(3)....All parameters are changed in the direction determined in step (2) and in step sizes determined by DEL and XSTEP until the Quality of Fit is no longer improved.

(4)....Steps (2) and (3) are repeated until no improvment in the Fit can be made

(5)....DEL and XSTEP are multiplied by DELFAC and steps (2) and (3) are repeated until no improvment in the Fit can be made.

(6)....Step (5) is repeated (NDEL-1) times.

U300.490 Gaussian Method (GFIT request)

The GFIT (Gauss-method) fit request initiates an alternate non-linear procedure. Commands are:

GFIT ID,ILO,IHI
or
GFIT ID X

This command initiates a nonlinear least-squares search by Gauss' method as modified by Marquardt. (See, for example, P.R. Bevington's book, "Data Reduction and Error Analysis for the Physical Sciences", p. 235 ff. The routines used in GFIT are not Bevington's, but are those of M.J. Saltmarsh from the SEL 840-A program PKFT.)

The search continues until chi-squared per degree of freedom (QFN) has changed by less than 0.0001 or until 25 iterations have occurred. The search may be resumed by the RFIT command. The iteration number and QFN appear on the right side of the screen.

The printer output from GFIT includes absolute error estimates for the peak positions, widths, and areas which are derived from the correlation matrix of the fit. The percentage error in the area is printed in the column labeled PCE.

The DAMM commands VW ULOC or VW CLOC (the default option) ae interpreted to mean that the widths of the peaks are to be the same and vary together. Widths may be varied independently by VW UIND or VW CIND. Individual widths or positions may be frozen or released by the standard cursor commands of DAMM. If one or more widths are to be kept fixed while others are varied, the command VW CIND or VW CLOC should be given; if UIND or ULOC is given, the instruction to fix is ignored.

If GFIT is chosen, the program attempts to estimate the width of the tallest peak for its initial guess of width. If unsuccessful it reverts to the standard DAMM procedure of using whatever was stored from the last previous FW, SETW commands (or the default option, which is FW = 5).

At present The GFIT request is limited to fitting a sum of up to 5 Gaussian peaks with a linear background. The parameters ASLO and ASHI for asymmetric peaks are ignored.

U300.500 Estimated Uncertainties in Peak Areas

Estimated uncertainties should always be viewed with considerable skepticism, especially when non-linerar as well as linear fitting

processes are involved, as it is here. The uncertainties in the peak areas estimated by both the FIT and GFIT procedures are rather "standard" and involve CHISQ (of the overall fit) as well the diagonal element of the inverse matrix corresponding to the peak intensity in question. This inverse matrix is found in the standard linear least-squares fitting process. See a book like "Bevington" or Cziffra et. al. UCRL-8523, 1958 for a real discussion of this subject. I have used:

D(J) = SQRT(AINV(J,J)*QFN) where;

D(J) = the estimated uncertainty in the J-th fit parameter B(J)
(there is a B(J) for each peak-area(J))

QFN = CHISQ/(#data-points - #adjustable-parameters)

AINV(J,J) = the J-th diagonal element of the inverse matrix found in the
linear least-squares fitting process.

PCE(J) = 100*D(J)/B(J) = percent uncertainty in J-th peak-area.

U300.510 Reading Fit Results from damm.log

The table of fit-results recorded on damm.log, as a result of a PR command, includes flags of the form LAB$ to facillitate the location and decoding of relevant data by other programs. Formats associated with the different line-labels are listed below.

Label	Format
TIT$	(1X,15X,2I6,6X,20A4)
DEL$	(1X,2F10.0,I10,6F10.0)
SKP$	(1X,8I10)
CAL$	(1X,8F10.0,I10)
VAR$	(1X,6(6X,A4),I10)
FIT$	(1X,5F10.0,3F8.0,4F7.0,I5,I7)
GFI$	(1X,5F10.0,3F8.0,4F7.0,I5,I7)
BGD$	(1X,2F10.2)
QFN	(1X,6X,F10.0,11X,F10.0)

Alternatively, one may make use of the routines contained in the internally documented demonstration program samred. The source of this program is in /usr/users/milner/Ddamm/samred.f.

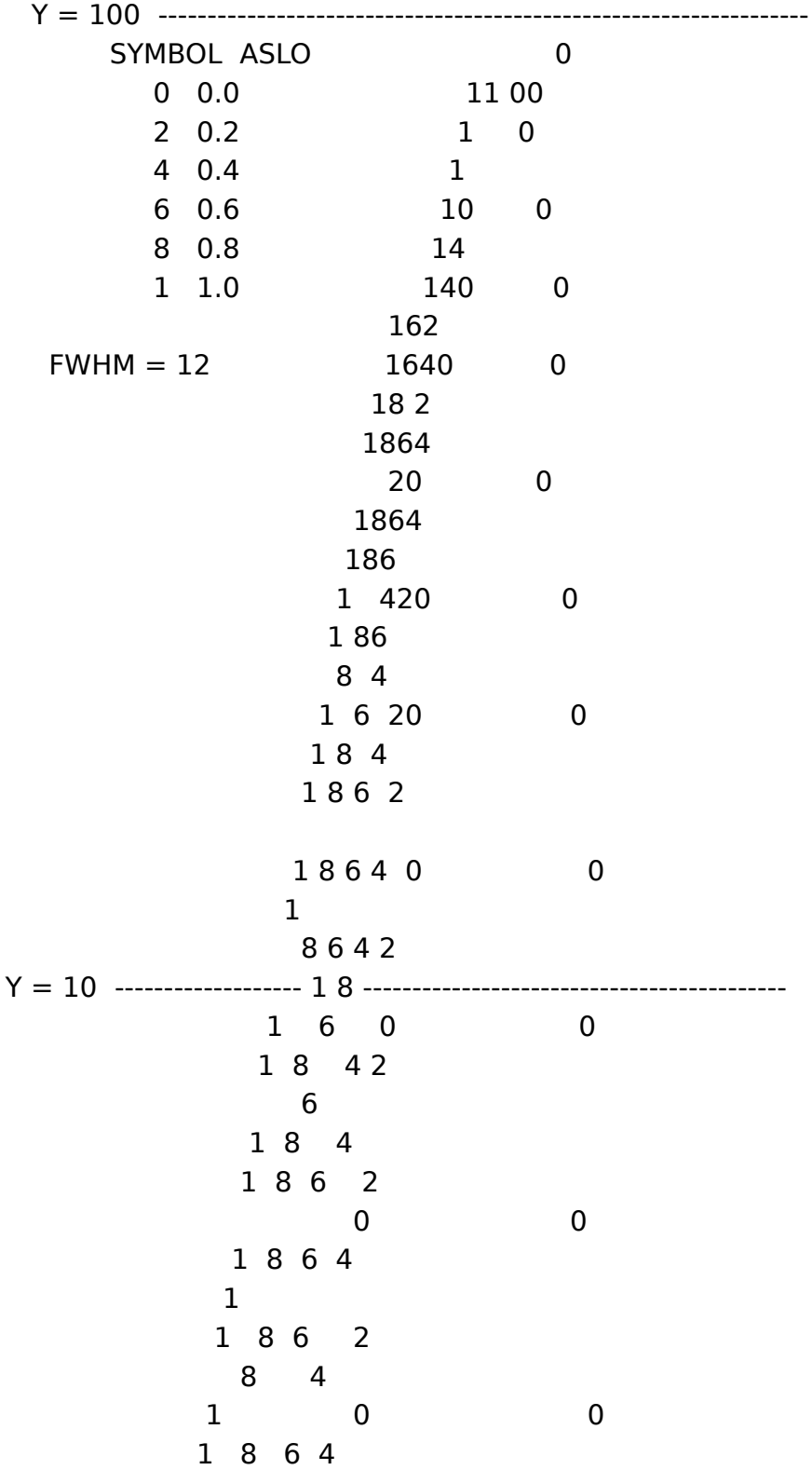
U300.520 Common Problems

(1)....If you define the standard FWHM (via command: FW FWA,FWB,FWC) or

standard asymmetry parameters (via command: ASYM ASLO,ASHI), this does not re-define such parameters for previously defined peaks. You must use SETW to do this (see SEC# U300.450).

U300.530 Peak Shape vs Asymmetry (Log plot)

YL=EXP(-(X-XO)**2/(A**2*(1+ASLO*(XO-X)/A)
YH=EXP(-(X-XO)**2/(A**2))



```
                                2
                  1    8    6
                 1            4
                1     8    6
                    8           2   0                           0
               1         6    4
              1     8
                       6
             1     8         4   2
Y = 1      ------------------------------------------------------------------------
```

Figure 1

U300.550 Screen Configurations

The sizes and locations of display windows are controlled by one or more of the commands - FIG, FIGF or FIGI.

FIG ID - Sets the number and size of display windows to that specified by configuration number ID. A given configuration ID number may specify up to 20 display windows. The default configuration library contains ID-numbers 1 thru 16. Type: H FIG for a display indicating the default sceen configurations or just Type: FIG 1, FIG 2 ... FIG 16 and see what you get.

FIGF FILENAME - Requests that a new configuration library be read from a file named FILENAME. The file U1:[MILNER.XWIN]XFIG1.DAT which contains the default configuration is listed on the next next two pages. The **** in col-1 of the table denote comment lines and are ignored in processing.

```
****DEFAULT SCREEN-CONFIGURATION TABLE
****-------------------------------------------------
****FIG-ID   X0(PIX)   Y0(PIX)    W(PIX)    H(PIX)
       1       30       84       540       480
****
       2        0       39       480       480
       2      495       39       480       480
****
       3       30      369       540       300
       3       30       39       540       300
       3      585       39       390       390
****
       4        0      369       480       300
       4        0       39       480       300
       4      495      369       480       300
       4      495       39       480       300
****
       5       30      489       480       195
       5       30      264       480       195
       5       30       39       480       195
       5      525       39       435       435
****
       6        0      489       480       195
       6        0      264       480       195
       6        0       39       480       195
       6      495      489       480       195
       6      495      264       480       195
```

```
      6      495       39      480      195
****
      7       30      534      480      135
      7       30      369      480      135
      7       30      204      480      135
      7       30       39      480      135
      7      525       39      435      435


****DEFAULT SCREEN-CONFIGURATION TABLE (continued)
****FIG-ID   X0(PIX)   Y0(PIX)    W(PIX)    H(PIX)
      8        0      534      480      135
      8        0      369      480      135
      8        0      204      480      135
      8        0       39      480      135
      8      495      534      480      135
      8      495      369      480      135
      8      495      204      480      135
      8      495       39      480      135
****
      9       30      579      480      105
      9       30      444      480      105
      9       30      309      480      105
      9       30      174      480      105
      9       30       39      480      105
      9      525       39      435      435
****
     10        0      579      480      105
     10        0      444      480      105
     10        0      309      480      105
     10        0      174      480      105
     10        0       39      480      105
     10      495      579      480      105
     10      495      444      480      105
     10      495      309      480      105
     10      495      174      480      105
     10      495       39      480      105
****
     11        0       39      960      525
****
     12        0      354      960      270
     12        0       54      960      270
****
     13        0      489      960      195
     13        0      264      960      195
     13        0       39      960      195
```

```
****
     14       0     549     960     135
     14       0     384     960     135
     14       0     219     960     135
     14       0      54     960     135
****
     15     240      39     750     750
****
     16       0      39     645     645
```

FIG-ID = Configuration ID number to be associated with window.
X0(PIX) = X-coordinate of upper left corner of window in pixels.
Y0(PIX) = Y-coordinate of upper left corner of window in pixels.
W(PIX) = Width of window in pixels.
H(PIX) = Height of window in pixels.
FIGI Restores the configuration library to the default state.

U300.560 Graphic Screen Color Mapping

The colors (or grey scale) is controlled by means of a color map which has 40 entries that specify (red, green, blue) intensities in the range 0-65535. The default table contained in /usr/hhirf/cmap.dat or in U1:[MILNER.XWIN]CMAP.DAT is listed below:

```
   RED  GREEN   BLUE   ;ENTRY# - NORMAL USE ------------------------------

     0       0       0  ;01 - BLACK
 65535  65535  65535  ;02 - 1-D DISPLAY - COL(1) - FIT DATA
 65535       0       0  ;03 - 1-D DISPLAY - COL(2)
     0  65535       0  ;04 - 1-D DISPLAY - COL(3) - FIT CALC
     0       0  65535  ;05 - 1-D DISPLAY - COL(4)
 65535  65535       0  ;06 - 1-D DISPLAY - COL(5) - FIT BACK
 65535       0  65535  ;07 - 1-D DISPLAY - COL(6)
     0  65535  65535  ;08 - 1-D DISPLAY - COL(7)
 65535  65535       0  ;09 - 1-D DISPLAY - COL(8)
 65535  65535       0  ;10 - NOT USED FOR NOW
     0       0  32767  ;11 - 2-D COLOR DISPLAY
     0       0  65535  ;12 - 2-D COLOR DISPLAY
 32767       0  32767  ;13 - 2-D COLOR DISPLAY
 65535       0  65535  ;14 - 2-D COLOR DISPLAY
 32767       0       0  ;15 - 2-D COLOR DISPLAY
 65535       0       0  ;16 - 2-D COLOR DISPLAY
 32767  32767       0  ;17 - 2-D COLOR DISPLAY
 65535  65535       0  ;18 - 2-D COLOR DISPLAY
 32767  32767  32757  ;19 - 2-D COLOR DISPLAY
 65535  65535  65535  ;20 - 2-D COLOR DISPLAY
```

```
 15000  15000  15000  ;21 - 2-D GREY-SCALE DISPLAY
 20600  20600  20600  ;22 - 2-D GREY-SCALE DISPLAY
 26200  26200  26200  ;23 - 2-D GREY-SCALE DISPLAY
 31800  31800  31800  ;24 - 2-D GREY-SCALE DISPLAY
 37400  37400  37400  ;25 - 2-D GREY-SCALE DISPLAY
 43000  43000  43000  ;26 - 2-D GREY-SCALE DISPLAY
 48600  48600  48600  ;27 - 2-D GREY-SCALE DISPLAY
 54200  54200  54200  ;28 - 2-D GREY-SCALE DISPLAY
 59800  59800  59800  ;29 - 2-D GREY-SCALE DISPLAY
 65535  65535  65535  ;30 - 2-D GREY-SCALE DISPLAY
 32767  32767  32767  ;31 - NOT USED FOR NOW
 32767  65535  65535  ;32 - NOT USED FOR NOW
 65535      0      0  ;33 - NOT USED FOR NOW
 65535  65535  65535  ;34 - GCOR(1)
 65535      0      0  ;35 - GCOR(2), SAM PK MARK,  FIT VAR MARK
     0  65535      0  ;36 - GCOR(3), 1-D PK LAB, 2-D BAN & EX-MARK
     0      0  65535  ;37 - GCOR(4), 1-D REG MARK, FIT VAR MARK
 65535  65535      0  ;38 - GCOR(5), CURSOR
 65535      0  65535  ;39 - GCOR(6)
     0  65535  65535  ;40 - GCOR(7)
```

Different color mapping is accomplished by the CMAP command as shown below:

```
CMAP FILENAME  ;Processes a file FILENAME of the structure shown above
               ;and maps as  specified  therein.  The new mapping only
               ;takes place subsequent to the next FIG command.
```

U300.570 Graphic Screen Black & White Mapping

When using black & white monitors, the table /usr/hhirf/bmap.dat, (or U1:[MILNER.XWIN]BMAP.DAT) listed below, may be more useful. If you are using the REVV mode, then table /usr/hhirf/bmapr.dat or U1:[MILNER.XWIN]BMAPR.DAT, not listed here, should be used as a template. You will probably need to make other adjustments in order to achieve semi-satisfactory results. Note: that table entries are labeled with their uses.

```
   RED  GREEN   BLUE   ;ENTRY# - NORMAL USE ------------------------------

     0      0      0  ;01 - BLACK
 65535  65535  65535  ;02 - 1-D DISPLAY - COL(1) - FIT DATA
 65535  65535  65535  ;03 - 1-D DISPLAY - COL(2)
 65535  65535  65535  ;04 - 1-D DISPLAY - COL(3) - FIT CALC
 65535  65535  65535  ;05 - 1-D DISPLAY - COL(4)
 65535  65535  65535  ;06 - 1-D DISPLAY - COL(5) - FIT BACK
 65535  65535  65535  ;07 - 1-D DISPLAY - COL(6)
 65535  65535  65535  ;08 - 1-D DISPLAY - COL(7)
 65535  65535  65535  ;09 - 1-D DISPLAY - COL(8)
```

```
65535  65535  65535  ;10 - NOT USED FOR NOW
    0      0  32767  ;11 - 2-D COLOR DISPLAY
    0      0  65535  ;12 - 2-D COLOR DISPLAY
32767      0  32767  ;13 - 2-D COLOR DISPLAY
65535      0  65535  ;14 - 2-D COLOR DISPLAY
32767      0      0  ;15 - 2-D COLOR DISPLAY
65535      0      0  ;16 - 2-D COLOR DISPLAY
32767  32767      0  ;17 - 2-D COLOR DISPLAY
65535  65535      0  ;18 - 2-D COLOR DISPLAY
32767  32767  32757  ;19 - 2-D COLOR DISPLAY
65535  65535  65535  ;20 - 2-D COLOR DISPLAY
15000  15000  15000  ;21 - 2-D GREY-SCALE DISPLAY
20600  20600  20600  ;22 - 2-D GREY-SCALE DISPLAY
26200  26200  26200  ;23 - 2-D GREY-SCALE DISPLAY
31800  31800  31800  ;24 - 2-D GREY-SCALE DISPLAY
37400  37400  37400  ;25 - 2-D GREY-SCALE DISPLAY
43000  43000  43000  ;26 - 2-D GREY-SCALE DISPLAY
48600  48600  48600  ;27 - 2-D GREY-SCALE DISPLAY
54200  54200  54200  ;28 - 2-D GREY-SCALE DISPLAY
59800  59800  59800  ;29 - 2-D GREY-SCALE DISPLAY
65535  65535  65535  ;30 - 2-D GREY-SCALE DISPLAY
32767  32767  32767  ;31 - NOT USED FOR NOW
32767  65535  65535  ;32 - NOT USED FOR NOW
65535      0      0  ;33 - NOT USED FOR NOW
65535  65535  65535  ;34 - GCOR(1)
45000  45000  45000  ;35 - GCOR(2), FIT PK MARK,  FIT VAR MARK
65535  65535  65535  ;36 - GCOR(3), 1-D PK LAB, 2-D BAN & EX-MARK
65535  65535  65535  ;37 - GCOR(4), 1-D REG MARK, FIT VAR MARK
65535  65535  65535  ;38 - GCOR(5), CURSOR
65535  65535  65535  ;39 - GCOR(6)
65535  65535  65535  ;40 - GCOR(7)
```

U300.600 Implementation

The XPAK tape that you receive will contain all of the files that I have on my Vaxstation-3100, whether you want them or not - it is just a lot easier for me to do that. Since I am running 5.3 on the Vaxstation, the EXEs may not work and you may have to re-link. To do this, type the following:

```
@DAMMLNK     ;To link DAMM
```

Final Note

Of course, you will probably need to change DISK and [DIRECTORY] names in files like LOGIN.COM, DAMM.COM, DAMMLNK.COM, etc. The same sort of things

you did for VAXPAK implementation.

Good Luck,
W. T. Milner

2D GATE WITH CURSOR - RVR

In order to set gates with cursor, you should first open the 2D matrix for INput (HIS or CMAT structures are accepted). This file should contain the 2D matrix as ID #1. Then open an SPK file for gates OUtput. This file can also contain a total projection of the matrix (constructed with the usual GY MILDO command) or another SPK file with the total projection can be opened e.g. with the QF command. Use the FIG command to set a WIN-configuration with at least 2 graphical windows (e.g. FIG 12 or FIG 14) and display the total projection or any previous gate on the fist window. Then enter in cursor mode.

New commands for gates with the cursor:

Z - (Zero) - Clear Buffer B2 and the **R** marks.

R - (Region) - Marks the peak limits. Op to 10 regions can be selected (the order is arbitrary, but the two channels of each region should be entered sequentially). All regions will be added to construct the gate. The pair (or single, if only one of a pair was entered) of coordinates of a region can be deleted with key **D** (the one that is closest to the cursor). **U R** removes all region marks.

B - (Background) - Marks up to 10 regions for Compton subtraction. The order is arbitrary, but members of a pair should be entered in sequence. **D** clears the pair of **B** marks closest to the cursor. **U B** removes all background marks.

J - (Janela/Window) - The gate is constructed automatically as a sequence of the following MILDO commands:

Z1
Z2
GY 1 R_L R_H - Where R_L , R_H are the two values of the peak region.
A12 1. - The Buffer 2 is accumulated with the actual peak gate.
.... - Repeat for all peak - regions
GY 1 B_1 B_2 - Where B_1 B_2 are the limits of the lower value of the BKG region (If any).
A12 F - Where F = -SUM (all R - channs.)/SUM(all B channs.)
.... - Repeat for all B regions.

After the **J** - command is given and these MILDO command are executed the result (Buffer 2) is displayed in the window above (IDW consecutive) the one where the graphic cursor is positioned at the moment the **J** command is entered. You can perform any valid cursor 1-KEY command in the window where the gate is plotted, including

setting new gates. The only limitation refers to previous (older) displayed gates: If you e.g. expand or perform any command that implies re-display, the result will refer to the last gate, since that is what is in Buffer 2.

T - (Totalize) - Similar to **J**, but Z1 and Z2 are **NOT** performed before constructing the gate. In this way, the present gate is accumulated to the previous one.

O - (Output) - After the command you are prompted to enter the new ID number to store the Buffer 2 on the output file. The channel limits are logged on the screen and DAMM.LOG after a successful command **O**.

X - Change the axis where gate is placed. Initially the gate is set in axis 2. Every time **X** is entered, gating axis is changed (1,2).

F - (Full) - This is not really a command related with gating, but a new command of the original cursor (D1) routine. After an expansion, the F - Key will return the display limits to the original limits of the D - command (entered with the DL - command).

DON'T forget to **U R** when setting new gates !!!