CONF-841007--48

CONF-841007--48

DE85 005356

CHIL - A COMPREHENSIVE HISTOGRAMMING LANGUAGE

W. T. Milner and J. A. Biggerstaff

Oak Ridge National Laboratory*

Oak Ridge, TN 37831

Abstract

A high level language, CHIL, has been developed for use in processing event-by-event experimental data at the Holifield Heavy Ion Research Facility (HHIRF) using PERKIN-ELMER 3230 computers. CHIL has been fully integrated into all software which supports on-line and off-line histogramming and off-line pre-processing. CHIL supports simple gates, free-form-gates (2-D regions of arbitrary shape), condition test and branch statements, bit-tests, loops, calls to up to three user supplied subroutines and histogram generating statements. Any combination of 1, 2, 3 or 4-D histograms (32 megachannels max) may be recorded at 16 or 32 bits/channel. User routines may intercept the data being processed and modify it as desired. The CPU-intensive part of the processing utilizes micro-coded routines which enhance performance by about a factor of two.

Introduction

All experimental data from nuclear and atomic physics experiments at HHIRF are acquired in 16-bit list-mode (i.e. event-by-event). Each event normally consists of several parameter identification words, one or more pattern words, about 10 to 100 data words. (experimental parameters from ADC's, TDC's etc.) and an end-of-event word. The HHIRF list-data structure is shown below (all numbers are in hexadecimal).

```
FFFF    ;End of preceding event
8000+N  ;Parameter ID word
P(N)    ;Parameter N
P(N+1)  ;Parameter N+1
.       ;etc
8000+M  ;Parameter ID word
P(M)    ;Parameter M
P(M+1)  ;parameter M+1
.       ;etc, etc
FFFF    ;End of event
```

Processing of such data usually involves tests on the pattern word bits, simple (1-D) gating, free-form (2-D) gating, tests for the existance of parameters and the generation of a number of one- and two-parameter histograms. In some cases, a pre-processing step is also carried out in which selected events (of a given type for example) are retrieved from an input data tape, possibly modified, and saved on an output tape for subsequent processing.

A general purpose high-level language, CHIL, has been devised and implemented on the HHIRF PERKIN-ELMER 3230 computer system which supports both pre-processing and histogram generation. CHIL allows the user to specify test conditions and histogramming specifications in a way which is much more simple and compact than an equivalent FORTRAN program. A special set of microcoded routines (which execute in Writable Control Store and define new machine instructions) are used for most of the basic CPU-intensive operations. This results in a program which is often more efficient than an equivalent assembly language program tailored to the specific task.

*Operated by Martin Marietta Energy Systems, Inc. under contract DE-AC05-840R21400 with the U.S. Department of Energy.

Although CHIL supports many of the basic data processing requirements, it does not attempt to cover everything that all users may wish to do. For this reason, a provision is made for calling user supplied subroutines from the CHIL program on an event-by-event basis. These routines receive the event in expanded form, modify the event data as desired and pass it back to the CHIL program for additional processing. This type of user customization has proved to be a very effective means of meeting special requirements while maintaining a high degree of generality, standardization and associated reliability.

Chil Syntax

A summary of the CHIL syntax is given in Table 1. This table does not include all possible variations of the language but is intended to illustrate the basics. Most of the statement types are described below in more detail.

$LPR - Parameter Lengths

The parameter length defines the maximum number of channels (always a power of 2) to be associated with a given parameter. Two syntax forms are supported: An implied loop and an explicit list. Lengths may be re-defined as many times as convenient but all lengths must be given at least once before another statement type is encountered.

$DIP & $ASS - Parameter Names

Names (up to 8 characters in length) may be associated with some or all parameter numbers and be used to reference said parameters in the CHIL program. Two steps are required: First, the name must appear in a $DIP statement (dimension parameter statement) which is much like the FORTRAN dimension statement. Next, values must be assigned via the $ASS statement which has both implied loop and explicit forms. For the implied loop case, JLO is the first value to assign and JNC is the value increment to be added for successive assignments.

$DIM & $DAT - Indexed Symbols

Indexed symbols are arrays of constants which are named and dimensioned via the $DIM statement and have values assigned by means of the $DAT statement or by an equate of the form: SYM(J) = EXPRESSION. Values associated with such symbols may be re-defined any number of times within the CHIL program but the value at a given point in the program is fixed at compile time. All symbols (indexed or simple) are variables at compile time but constants at run time.

$BAN & $BAF - Free-Form Gates

Free-Form-Gates must be created outside of the CHIL program (usually via the HHIRF display program RIP[1] and stored on a special file (BAN file). Incorporation of such 2-D gates into a CHIL program requires one or more $BAN statements which specifify the gate ID-numbers (IDA, IDB ...) to be retrieved from the file and the X-length (LENGTH) to be allocated for storage

```
SETUP STATEMENTS (Processed at compile time only) ..............................

$LSTL = NBYTES                      ;Tape record length (bytes)
$NPR  = NPAR                        ;Maximum number of parameters per event
$LPR  PA TO PB,STEP = LENGTH        ;Assign parameter lengths (max-value -1)
$LPR  P1,P2,P3,.... = LENGTH        ;Assign parameter lengths (max-value -1)
$DIP  NAMA(IV),NAMB(JV)...          ;Define & dimension parameter names
$ASS  NAMA(I1 TO I2,INC)=JLO,JNC    ;Assign values to parameter names
$ASS  NAMB(I1)          =JV         ;Assign value  to parameter name
$DIM  SYMA(IV),SYMB(JV)...          ;Define & dimension indexed symbols
$DAT  SYMA(I1 TO I2) = V1,V2....    ;Assign values to indexed symbols
$DAT  SYMB(I)        = VAL          ;Assign value  to indexed symbol
$BAN  (LENGTH) ID1,ID2,ID3 .....    ;List of free-form-gate ID's to get from
$BAF  FILENAME                      ;Free-form-gate file - FILENAME
$GLST LENGTH,IS LO,HI LO,HI ....    ;Gate-list specification
$MAPL LENGTH,IS LO,HI LO,HI ....    ;Gate-list specification (mapped)
$H16                                ;Specify 16-bits/channel
$H32                                ;Specify 32-bits/channel
$HID  NXID                          ;Specify next histogram-ID to use
$TIT  TITLE                         ;Specify next title (40 bytes max)

DO LABEL I=IA,IB,STEP               ;FORTRAN-like DO
CONTINUE                            ;FORTRAN-like CONTINUE
GO TO    LABEL                      ;FORTRAN-like GO TO
SYMBOL=EXPRESSION                   ;FORTRAN-like definition of simple  symbol
SYM(I)=EXPRESSION                   ;FORTRAN-like definition of indexed symbol

IF, BIT-TEST, CALL AND ASSIGN STATEMENTS (Executed at run time) ................

IFS(CONDITION) LABEL                ;If CONDITION    satisfied, GO TO LABEL
IFU(CONDITION) LABEL                ;IF CONDITION  unsatisfied, GO TO LABEL
IFX(PARAMETER) LABEL                ;IF PARAMETER       exists, GO TO LABEL
IFN(PARAMETER) LABEL                ;IF PARAMETER    non-exist, GO TO LABEL
IFC(COND-SET ) L1,L2,L3...          ;IF CONDITION-J  satisfied, GO TO LJ
IFP(PARAMETER) L1,L2,L3...          ;IF PARAMETER = J-1         GO TO LJ
                                    ;Otherwise, drop through
BTAB(P,MASK) LALL,LSOME,LNONE       ;BIT-TEST AND BRANCH (3-WAY)
                                    ;GO TO LALL, LSOME, LNONE if
                                    ;       ALL,  SOME,  NONE of bits
                                    ;Set in MASK match bits in parameter-P
CALL USERSUBX                       ;CALL USERSUBX  (X=1,2 or 3)
CALL REPACKX   PA,PB                ;Output parameters PA-PB to stream-X (X=1,2,3)
ASP(P,V)                            ;Assign to parameter-P the value V

HISTOGRAM GENERATING STATEMENTS (Executed at run time) .........................

 H(PI,PJ..) <L(LI,LJ..)> <R(LOI,HII LOJ,HIJ..)> <CONDITIONS>  or
OH(PI,PJ..) <L(LI,LJ..)> <R(LOI,HII LOJ,HIJ..)> <CONDITIONS>

Where:                              ;< > denotes an optional field
 H(PI,PJ..)                         ;Means histogram parameters PI,PJ..
OH(PI,PJ..)                         ;Means overlay histogram parameters PI,PJ..
                                    ;in space defined by last H-statement
L(LI,LJ,LK..)                       ;Defines basic histogram lengths (PWR of 2)
                                    ;Implies number of bits to shift raw data
                                    ;Length of PI,PJ,PK..= LI,LJ,LK..
R(LOI,HII LOJ,HIJ ..)               ;Sets histogram-range actually stored
                                    ;for PI to be LOI thru HII,
                                    ;for PJ to be LOJ thru HIJ, etc.

CONDITION SPECIFICATION SYNTAX .................................................

G(P LO,HI)                          ;Single gate (P = test parameter,
                                    ;LO,HI = test limits)
G(P LO,HI LO,HI..)                  ;Gate-list
GS(P,IS NA,NB)                      ;Gate-set (IS = set number and NA,NB gate
                                    ;numbers - from $GLST or $MAPL
B(PX,PY ID)                         ;Single 2-D gate (PX,PY are test parameters
                                    ;and ID is the ID-number from $BAN)
B(PX,PY IDA,IDB)                    ;2-D gate set
```

Table 1. Summary Of CHIL Syntax

of the gate tables in memory. The BAN-file to be processed may be specified in the CHIL program by the $BAF statement or may be deferred until run time.

SGLST & $MAPL - Gate Sets

Multiple gates which are to be imposed on a given parameter may be specified by the $GLST statement. Here LENGTH denotes the "parameter length" basis on which the gates are defined and ISN the set *number* by which the gate list is to be be referenced. Gates specified in a given $GLST statement may not overlap. If a large number (5 or more) gates are to be imposed on a given parameter, processing will proceed faster if the gates are "mapped"[2]. To do this one specifies the gates by means of the $MAPL statement. Otherwise, the rules are exactly the same as for the $GLST statement. As many continuation lines as required are allowed.

Expressions

The CHIL compiler supports simple expressions which are evaluated left to right. Let V represent a single value (number or previously defined symbol) and S an algebraic sum of V's. The left hand side (A) as well as V and terms in S may be indexed symbols. Parenthesis are not allowed except for indexed symbols and the (MOD) expression. Legal forms follow:

```
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 are allowed
                 ;(same argument definition as FORTRAN)
A=S+[S,S,..]     ;[ ] encloses a list of bit numbers
```

Gating conditions

Two types of gating conditions are supported: One parameter (1-D) gates of the lo-limit, hi-limit type and two parameter (2-D) gates which are defined by a clockwise set of X,Y coordinates in two parameter space. The latter type may also be referred to as a free form gate or by the slang term Banana gate due to fact that such regions defined on E vs E*DE plots often exhibit this shape. Gates may be specified singly or by multiple gate sets (.OR. lists). See Table 1 for a list of legal forms.

IFS & IFU - Single Condition Test And Branch

```
Legal forms are: IFS(G(P  LO,HI)) LABEL
                 IFU(G(P  LO,HI)) LABEL
                 IFS(B(PX,PY ID)) LABEL
                 IFU(B(PX,PY ID)) LABEL
```

These two IF statements may be applied to a single 1-D gate: G(P LO,HI) (where P denotes the test parameter and LO,HI the test limits) or to a single 2-D gate: B(PX,PY ID) (where PX and PY denote the test parameters and ID the ID number given in the $BAN specification). If the stated requirement is met, the program branches to LABEL, otherwise, it drops through to the next run time executable statement.

IFX & IFN - Test And Branch On Parameter Existance

These two IF statements test the specified parameter value for hexadecimal FFFF (non existant) and branch to the given label if the requirement is met. All legal forms are given in Table 1.

IFP - Multiple Branch On Parameter Value

IFP tests the specified parameter (P) and branches to L1 if P=0, L2 if P=1, etc. This test provides a means by which a test parameter created in a USERSUB may be used to direct the flow of the CHIL program.

IFC - Condition-Set Testing

```
Legal forms are: IFC(GS(P,IS   NA,NB)) L1,L2,L3,...
                 IFC(B(PX,PY IDA,IDB)) L1,L2,L3,...
```

The IFC test may be applied to a multiple 1-D or 2-D gate set as indicated. The first condition in the set to be satisfied determines the branch to be taken. If no condition is satisfied, the program drops through to the next run time executable statement.

H-Statements - Histogram Generation

The histogram statement (H-statement) specifies which parameters are to be histogrammed, the length of each histogram dimension, the range of interest and gating conditions. 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.

In general, the CONDITION field may actually include several single and/or multiple condition specifications (gate lists or sets). Each specification is considered to be an OR-list which contains one or more members. For the case where more than one OR-list is specified, these are "ANDED" together to produce multiple histograms. In other words, you will get a count "somewhere" if and only if some member of each OR-list is satisfied. If you have N OR-lists and NG(I) represents the number of gates in the Ith list, then the number of histograms (NH) actually implied is given by: NH = NG(1)*NG(2)*....NG(N-1)*NG(N).

OH-Statements - Histogram Overlays

The OH-statement has exactly the same meaning as the H-statement except that the histogram origin is not advanced. That is, histogramming takes place in the "space" defined by the most recent H-statement.

Loops

Loops are implemented so as to look like FORTRAN but are actually expanded by the compiler so that loop parameters are constants. In the following example we 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
```

CALL REPACK - Pre-Processing

Pre-processing (as interpreted here) involves the processing of an input data stream to produce one or more output data streams (usually from and to mag tape). The processing may include selection of certain events or parameters, the modification or creation of parameters or any combination thereof. A CHIL based preprocessing task involves the use of HHIRF program

LEMO (or a customized version thereof) to control the process combined with a CHIL program which directs the processing and selection. Modification or creation of parameters will normally require one or more user supplied subroutines (USERSUBS). The CHIL program may include any legal statement except for the H-statement. The statement CALL REPACK- initiates the "saving of an event" into the output data stream. In the following simple example, all parameters of the event are saved only if parameter 1 exists.

```
$LSTL = 8192        ;Tape record length (bytes)
$NPR 18             ;Number of parameters
$LPR 1 TO 18 = 2048 ;Length of parameters 1 - 18
     IFN(1)100      ;Test for existance of P-1
     CALL REPACK1 1,18 ;Save event in stream-1
 100 CONTINUE
```

CALL USERSUBX - User-Supplied Subroutines

One 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. The interception occurrs after the event is unpacked (i.e. all parameters are properly positioned in the event array and any parameters not present in the raw event are initialized to hexadecimal FFFF). Created or modified parameters may be, subsequently, tested and histogammed in the same way as any others. Stock processing tasks include dummy usersubs while customized tasks are produced by linking the user's special routines, which may include a command processor and other support routines, with the standard main package. A skeleton USERSUB which just sets P-50 = P-1 is shown below:

```
SUBROUTINE USERSUB1(IBUF)
INTEGER*2 IBUF(512)
IBUF(50)=IBUF(1)
RETURN
END
```

CHIL Compiler Output

The CHIL compiler processes the source program and generates an instruction list which contains opcodes, gate values and/or addresses, test parameter and histogram parameter addresses, histogram base offsets etc. and stores this "object code" in the Instruction Region of an array (CHIL array). Gate lists, maps and 2-D gate tables are stored in a Data Region of this array as indicated below.

Instruction Region - 64 KB Contains: opcodes, instruction list displacements, histogram base addresses, lengths & ranges pointers to histogram & test parameters pointers to gate lists, etc.
Data Region - 192 KB Contains: 1-D gate lists, 1-D mapped gates and 2-D gate tables

The CHIL array is written onto a disk file which the processing program, subsequently, reads and uses to direct the operation of the microcoded procedures which do most the actual processing (histogram generation or new data stream construction). The CHIL processor goes through the complete instruction list once per event - branching as dictated by the data.

Microcoded Procedures

The Instruction Region of the CHIL array is composed of combinations of the opcodes (and associated date elements) given in the following list (i.e. these are the procedures available to the compiler). In this list ACC denotes the one accumulator at our disposal.

```
Opcode/Data     ;Definitions And Comments .....

CODE(17)        ;STORE ACC IN PARAMETER .....
PDISP           ;Parameter displacement

CODE(18)        ;LOAD ACC WITH PARAMETER .....
PDISP           ;Parameter displacement

CODE(19)        ;SHIFT ACC BY P1 (SIGNED) .....
P1              ;Shift count

CODE(20)        ;ADD PARAMETER TO ACC .....
PDISP           ;Parameter displacement

CODE(21)        ;MULTIPLY ACC BY P1 .....
P1              ;Multiplier

CODE(22)        ;1-D GATE-LIST PROCESSING .....
PDISP           ;Test parameter displacement
CDISP(FAIL)     ;CHIL displacement for fail
CDISP(HIT)      ;CHIL displacement for hit
INCR            ;Value added to ACC for partial miss
-#GATES         ;Minus number of gates in list
GDISP           ;CHIL displacement of gate list (fullwd)

CODE(23)        ;2-D (FREE-FORM) GATE PROCESSING .....
PDISP(1)        ;Displacement of 1st test parameter
CDISP(FAIL)     ;CHIL displacement for fail
CDISP(HIT)      ;CHIL displacement for hit
INCR            ;Value added to ACC for partial miss
-#FFGS          ;Minus number of free-form gate vectors
SHIFT           ;Shift to apply to 1st test parameter
PDISP(2)        ;Displacement of 2nd test parameter
0               ;Pad for alignment
SIZE            ;Length of each 2-D gate vector
GDISP           ;Displacement of first 2-D gate vector

CODE(24/25)     ;FIRST/SUBSEQUENT CHAN# CALCULATION ....
CDISP(FAIL)     ;CHIL displacement for out-of-range data
SHIFT           ;Right shift to apply to parameter
PDISP           ;Histogram parameter displacement
MIN             ;Baseline to subtract after shift
MAX             ;# of channels after shift and subtract

CODE(26/27)     ;16/32 BIT/CHANNEL INCREMENTING .....
0               ;Pad
HBASE           ;Histogram base channel number

CODE(28)        ;LOAD ACC WITH P1 (IMMEDIATE) .....
P1              ;Value to load

CODE(29)        ;TABLE LOOKUP / MAPPED GATES .....
PDISP           ;Test parameter displacement
CDISP(FAIL)     ;CHIL displacement for fail
CDISP(HIT)      ;CHIL displacement fot hit
SHIFT           ;Shift to apply to test parameter
SIZE            ;Size of table
TDISP           ;Table displacement in CHIL
                ;ACC contains TABLE(parameter-value)

CODE(30)        ;SIMPLE IN-LINE GATE .....
PDISP           ;Parameter displacement
CDISP(FAIL)     ;CHIL displacement for FAIL
CDISP(HIT)      ;CHIL displacement for HIT
HI-LIMIT        ;Gate high-limit
LO-LIMIT        ;Gate low -limit
```

```
CODE(31)        ;GO TO (ADD P1 TO CHIL POINTER) .....
P1              ;CHIL displacement

CODE(32)        ;EXIT WRITABLE CONTROL STORE .....
P1              ;P1=0 says done
                ;P1=1,2,3 says CALL USERSUB1,2,3
                ;P1=4,5,6 says CALL  REPACK1,2,3
PMIN            ;Min parameter to output (REPACK only)
PMAX            ;Max parameter to output (REPACK only)

CODE(33)        ;BIT TEST AND THREE-WAY BRANCH .....
PDISP           ;Parameter displacement
CDISP(SOME)     ;CHIL displacement for some bits match
CDISP(NONE)     ;CHIL displacement for no   bits match
CDISP(ALL)      ;CHIL displacement for all  bits batch
MASK            ;Bit mask

CODE(34)        ;COMPUTED GO TO .....................
                ;(ACC must contain dispatch table index)
P1              ;Displacement for 1st branch
PN              ;Displacement for Nth branch, etc.
```

A very short and simple example of a histogram generating CHIL source program is shown below followed by the instruction-list generated by the compiler.

CHIL Source Program

```
$LSTL 8192              ;Tape record length (bytes)
$NPR  2                 ;Max # of parameters/event
$LPR (1 TO 2) = 8192    ;Max value of parameters
$DIP GE(2)              ;Define & dimension names
$ASS GE(1 TO 2) = 1,1   ;Assign values to names
     H(GE(1))           ;Histogram GE(1) (i.e. P-1)
     &L(4096)           ;Compress from 8192 to 4096
     &R(2000,4000)      ;Store channels 2000 to 4000
     &G(GE(2) 200,400)  ;Gate on GE(2) 200 to 400
```

Instruction List Generated By Compiler

```
Code   Type    - Definition And Comments
  28   OPCODE  - Load accumulator (immediate)
   0   P1      - Immediate value
  30   OPCODE  - Simple in-line gate
   2   PDISP   - Test parameter displacement
  32   CDISP   - (FAIL) (MISS minus last OPCODE loc)
  12   CDISP   - (HIT )
 400   1HI     - Single in-line gate HI-limit
 200   ILO     - Single in-line gate LO-limit
  24   OPCODE  - Prepare first histogram parameter
  20   CDISP   - (FAIL) (MISS minus last OPCODE loc)
   1   SHIFT   - Right shift (+) to apply to abscissa
   0   PDISP   - Histogram parameter displacement
2000   MIN     - Base subtracted from shifted parameter
2001   MAXNC   - Maximum number of channels to store
  26   OPCODE  - Increment 16-bit histogram
   0   PAD
   0   HISBASE- Histogram base address (full word)
  32   OPCODE  - Exit Writable Control Store
   0   END     - End of instruction-list
```

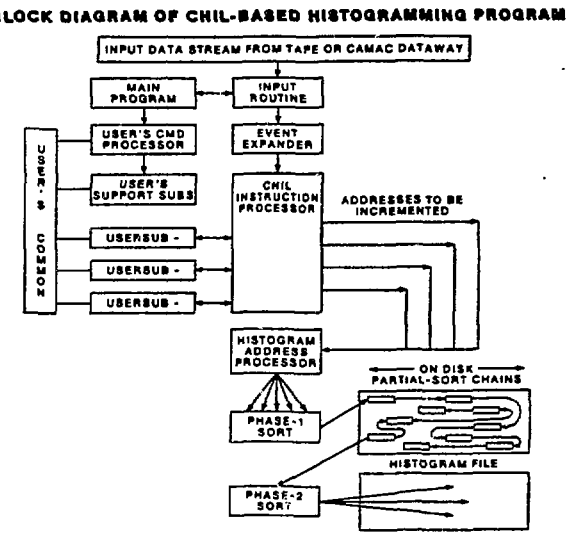

Figure 1. Block Diagram of CHIL Based Histogramming Program.

A block diagram of a customized CHIL based histogramming program is shown in Figure 1. Histogram addresses to be incremented are pushed into partial sort buffers and processed via two phase sorting2 to produce histograms on disk. On-line monitor tasks accumulate the first 512K channels in memory.

Conclusions

CHIL has been in general use at HHIRF for about six months and has provided major improvements in both performance and ease of use of our processing software. An important feature of CHIL is it's robustness. First, the microcode is hardened against unexpected (illegal) data. Secondly, the CHIL compiler was designed with thorough syntax and logic checking to insure that illegal instructions are never generated. User acceptance of the new system was so immediate and trouble-free that CHIL has completely superseded all our older histogramming software.

References

1. R. O. Sayer, RIP - A General Purpose Interactive Display At HHIRF, IEEE Transactions on Nuclear Science, Vol. NS-30, No 5, October 1983, pp. 3833-3837.

2. W. T. Milner, J. A. Biggerstaff, D. C. Hensley, and R. O. Sayer, Data-Acquisition and Analysis System For The Holifield Heavy Ion Research Facility, IEEE Transactions on Nuclear Science, Vol. NS-26, No 4, August, 1979, pp. 4399-4404.

DISCLAIMER