# Extension of Automatic Flow Charting Capabilities

R. J. Margolin and W. O. Paine
Quality Assurance DSN and Mechanical Hardware Section

*A new macro generation facility within the AUTOFLOW II flow charting system was used to process assembly language programs for the MODCOMP II computer. This article describes the nature of this new facility and how it was used, as well as describing other capabilities for automatic flow charting.*

In its search for tools to aid in auditing DSN software, the Software Quality Assurance Group has made use of graphic macro facilities within AUTOFLOW II[1] to enable automatic flow charting of assembly language source programs for the MODCOMP II[2] computer used in the Deep Space Network.

This facility in AUTOFLOW II is known as *macro definition*. It permits the representation of the individual instructions of a target computer (in this case, the MODCOMP II) in terms of flow chart symbols with accompanying text information. A group of macro definition statements is used to express an instruction and each time that instruction occurs:

(1) The type and number of flow chart symbols for that instruction are generated.

(2) The text associated with each symbol is generated, including literal expressions and indicated parts of the source instructions.

(3) In the case of decision, branch, and subroutine symbols, the location of the field which specifies the destination data is given.

(4) For decision symbols, the in-line and out-of-line path labels are supplied.

Over 500 lines of macro definition statements were used to form 236 macro definitions of assembler directives and instructions for the MODCOMP assembler with some of the macro assembler features added. Twenty-three of the macro definitions were for assembler directives. Twenty-seven of the remaining 213 macro definitions were for

[1] AUTOFLOW II is a program product of Applied Data Research, Inc., Princeton, New Jersey. There are many modules in AUTOFLOW II. One module, the chart/assembly module, accepts programs written in IBM 360/370 assembly language and automatically provides flow charts and analytical listings which make the structure of the program clearer. This system has been installed on the IBM 360/75 since early 1973.

[2] The MODCOMP II computer is a product of Modular Computer Systems, Fort Lauderdale, Florida.

separate definitions of those instructions (indicated in MODCOMP code by an asterisk as the last character of the operation) where the memory reference was indirect. In addition, a group of specific MODCOMP executive service macros was defined.

A simple and conventional approach to graphic macro definition would have been to select the appropriate boxes and merely repeat the source coding along with the comment. It was our desire to provide a type of documentation suitable even for one not reasonably familiar with MODCOMP coding. In an effort to do this, we have expressed the instructions in a combination of English, logical, and program language-related symbolic notation. This enables a new reader to understand the operations more easily without knowing the many details of addressing modes. A certain lack of flexibility in the macro definition feature may cause pairs of parentheses to appear with null contents. This serves to indicate non-use of indexing where potentially available.

Where an assembler itself offers macro capabilities, it is possible to provide parallel AUTOFLOW macro definitions, depending on the complexity of the assembler macros used. This was done by Quality Assurance in the case of the Communication Buffer and Quad Standard Interface Adaptor (SIA) Test Program, part of which is shown as a sample. Part of the macro definitions, source code, and chart output are illustrated in Figs. 1, 2, and 3. Normally, the parallel AUTOFLOW macro definitions would be prepared to the general standard used for the original assembler macros.

In addition to handling 360/370 assembly language programs, AUTOFLOW II also has additional capabilities. AUTOFLOW has a built-in facility known as chart code. It is unrelated to any programming language, but is a vehicle for indicating the structure of the program at a design level. AUTOFLOW can also handle assembly language programs for a number of other computers and a variety of languages, such as PL/1, COBOL, JCL. Some of these are options which may be delivered with the basic AUTOFLOW II, while others are preprocessors which dovetail with AUTOFLOW or produce output that may be input to AUTOFLOW.

The status of other AUTOFLOW activities is as follows:

(1) The current options in extensive use are FORTRAN and XDS Sigma Assembly Language.

(2) The installed preprocessors are SDS 920, UNIVAC 1108, and CDC 3100.

(3) Other preprocessors (which use the macro definition facility) are for the INTERDATA 4 and INTERDATA 70.

(4) Other uses have been examined and it is possible to automatically chart assembly code for the PDP-8, NOVA, MAC-16, and PDP-11.

## Bibliography


*AUTOFLOW II Assembly Series Reference Manual P402A*, Applied Data Research, Inc., Princeton, N.J., 1974.

*The New AUTOFLOW Sigma Series Reference Manual*, Applied Data Research, Inc., Princeton, N.J., 1972.

*AUTOFLOW Preprocessor System Maintenance and Technical Manual*, Applied Data Research, Inc., Princeton, N.J., 1969 (prepared for NASA-GSFC under Contract No. NAS5-10587).

*MODCOMP II Computer Reference Manual 210-102000*, Modular Computer Systems, Fort Lauderdale, Fla., 1973.

*MODCOMP Assembler/Macro Assembler Reference Manual 210-600101*, Modular Computer Systems, Fort Lauderdale, Fla., 1971.

```
06/23/75        INPUT LISTING                          AUTOFLOW CHART SET - MODCOMP  TEST PROGS (SCQ.HND.ERP.TIM)

 CARD NO        ****                                CONTENTS                                          ****

   222          * MDEF IF                                                                        *D IF    1
   223          * MDEF     INC                                                                    *D INC
   224          * MDEF INS                                                                        *D INS
   225          * MDEF INT.EN.D1/OP(ALL)/                                                         *D INT   1
   226          * MDEF     ISA.I                                                                  *D ISA   1
   227          * CON /'INPUT STATUS FROM DEVICE',Q(2),'TO REG.',Q(1),':',OP(ALL),C/              *DISA   2
   228          * MDEF     ISB.I                                                                  *D ISB   1
   229          * CON /'INPUT STATUS FROM DEVICE',Q(2),'TO REG.',Q(1),':',OP(ALL),C/              *DISB   2
   230          * MDEF     ISC.I                                                                  *D ISC   1
   231          * CON /'INPUT STATUS FROM DEVICE',Q(2),'TO REG.',Q(1),':',OP(ALL),C/              *DISC   2
   232          * MDEF     ISD.I                                                                  *D ISD   1
   233          * CON /'INPUT STATUS FROM DEVICE',Q(2),'TO REG.',Q(1),':',OP(ALL),C/              *DISD   2
   234          * MDEF     LAD.P                                                                  *D LAD   1
   235          * CON /'SHIFT LT ARITH DBL REG.',Q(1),Q(2),'BITS:',OP(ALL),C/                     *D LAD   2
   236          * MDEF     LAS.P                                                                  *D LAS   1
   237          * CON /'SHIFT  LT.AR.SNGL REG.',Q(1),Q(2),'BITS:',C/                              *D LAS   2
   238          * MDEF     LBR.P                                                                  *D LBR   1
   239          * CON /'LOAD BIT',Q(2),'INTO REG.',Q(1),':',OP(ALL),C/                            *D LBR   2
   240          * MDEF     LBRB.P                                                                 *D LBRB  1
   241          *          CON  /'LOAD BIT',Q(2),'INTO REG.',Q(1),':',C/                          *D LBRB  2
   242          * IBID B.D1/OP(1)/                                                                 *D LBRB  3
   243          * CON /OP(1)/                                                                     *D LBRB  4
   244          * MDEF LBX.P                                                                      *D LBX   1
   245          * CON /'LOAD BYTE INTO REG.',Q(1),'PER REG.',Q(2),':',OP(ALL),C/                  *D LBX   2
   246          * MDEF LDI.P                                                                      *D LDI   1
   247          * CON /'LOAD',OP(1),'INTO REG.',Q(1),':',C/                                       *D LDI   2
   248          * MDEF LDM.P                                                                      *D LDM   1
   249          * CON /'LOAD REG.',Q(1),'WITH (',OP(1),'+(REG.',Q(2),')):',C/                     *D LDM   2
   250          * MDEF LDM*.P                                                                     *D LDM*  1
   251          * CON /'LOAD REG.',Q(1),'WITH ((',OP(1),'+(REG.',Q(2),'))):',C/                   *D LDM*  2
   252          * MDEF LDS.P                                                                      *D LDS   1
   253          * CON /'LOAD REG.',Q(1),'WITH ((R1)+',Q(2),'):',OP(ALL),C/                        *D LDS   2
   254          * MDEF LDX.P                                                                      *D LDX   1
   255          * CON /'LOAD REG.',Q(1),'FROM ((REG.',Q(2),')):',OP(ALL),C/                       *D LDX   2
   256          * MDEF LFM.P                                                                      *D LFM   1
   257          * CON /'LOAD FILE TO REG',Q(1),'ETC FROM',OP(1),'+(',Q(2),'):',C/                 *D LFM   2
   258          * MDEF LFM*.P                                                                     *D LFM*  1
   259          * CON /'LOAD FILE TO REG',Q(1),'ETC FROM (',OP(1),'+(',Q(2),')):',C/              *DLFM*  2
   260          * MDEF LFS.P                                                                      *D LFS   1
   261          * CON /'LOAD FILE TO REG',Q(1),'ETC FROM((R1)+',Q(2),'):',OP(ALL),C/              *DLFS   2
   262          * MDEF LFX.P                                                                      *D LFX   1
   263          * CON /'LOAD FILE TO RG.',Q(1),'ETC FRM ((RG.',Q(2),')):',OP(ALL),C/              *DLFX   2
   264          * MDEF     LLD.P                                                                  *D LLD   1
   265          * CON /'SHIFT LEFT LOG DBL REG',Q(1),Q(2),'BITS:',OP(ALL),C/                      *D LLD   2
   266          * MDEF     LLS.P                                                                  *D LLS   1
   267          * CON /'SHIFT LEFT LOGIC REG',Q(1),Q(2),'BITS:',OP(ALL),C/                        *D LLS   2
   268          * MDEF     LRS.P                                                                  *D LRS   1
   269          * CON /'LEFT ROTATE REG.',Q(2),'INTO REG.',Q(1),':',OP(ALL),C/                    *D LRS   2
   270          * MDEF LST                                                                        *D LST
   271          * MDEF MAC                                                                        *D MAC
   272          * MDEF MBL.P                                                                      *D MBL   1
   273          * CON /'MOVE BYTE LEFT FROM REG.',Q(2),'TO REG.',Q(1),':',OP(ALL),C/              *DMBL   2
   274          * MDEF MBR.P                                                                      *D MBR   1
   275          * CON /'MOVE BYTE RIGT FROM REG.',Q(2),'TO REG.',Q(1),':',OP(ALL),C/              *DMBR   2
   276          * MDEF MLR.P                                                                      *D MLR   1
```

Fig. 1. Part of macro definition statements for MODCOMP instructions

```
                LDI,R2      #4020          PICK UP OCX INSTRUCTION          HND04110
                ORS,R2,0                   ADD GROUP AND UNIT NUMBERS       HND04120
                STM,R2      M3COND         STORE CCX INSTR IN CONDITIONING  HND04130
                STM,R2      M3WRIT         AND WRITE MODE INSTRUCTIONS      HND04140
                LDS,R2,5                   PICK UP IOS AND FUNCTION CODE    HND04150
                MLR,R2,R2                  ZERO I/O STATE                   HND04160
                ORI,R2      #0100          SET I/O STATE TO 1  WRITING MODEHND04170
                STS,R2,5                                                    HND04180
                BLM,R14     SETIME         SET TIMER                        HND04190
                LDI,R2      #4000          PICK UP CONDITIONING COMMAND     HND04200
        M3COND  OCB,R2,0                   OUTPUT CONDITIONING COMMAND      HND04210
                LDI,R2      #D000          PICK UP TI COMMAND               HND04220
        M3WRIT  OCB,R2,0                   OUTPUT TI COMMAND                HND04230
                BRU         DQINT          DEQUEUE NEXT INTERRUPT           HND04240
                EJT                                                         HND04250
        M3TERM  BLM,R14     CKSTAT         CHECK STATUS                     HND04260
                LDS,R2,5                   PICK UP I/O STATE                HND04270
                MBR,R2,R2                                                   HND04280
                CRMB,R2     ONE,M3CKSM,M3HLT  IF IOS = 1 GO CHECK SIMUL I/O    HND04290
                CRMB,R2     THREE,M3STRD,M3HLT IF IOS = 3 GO SET MEMORY READ   HND04300
                CRMB,R2     FOUR,M3BEGN,M3HLT IF IOS = 4 GO BEGIN AGAIN        HND04310
        M3HLT   HLT                        INVALID I/O STATE                HND04320
        M3CKSM  TBRB,R4,2   M3ST3          IF SIMUL I/O NOT SET GO READ MEMHND04330
                BRU         SIMUL          GO PROCESS SIMULTANEOUS I/O      HND04340
        M3ST3   LDI,R2      #FFF8          PICK UP LENGTH OF 8 WORDS        HND04350
                STM*,R2,R1  1              STORE IN TC LOCATION             HND04360
                LDI,R2      16             PICK UP LENGTH OF 16 BYTES       HND04370
                STS,R2,3                   STORE IN TABLE                   HND04380
                LDI,R2      M3BUF          PICK UP ADDR OF MEMORY BUFFER    HND04390
                STM*,R2,R1  2              STORE IN TA LOCATION             HND04400
                STS,R2,4                   STORE IN TABLE                   HND04410
                LDI,R2      #4020          PICK UP OCX INSTRUCTION          HND04420
                ORS,R2,0                   ADD GROUP AND UNIT NUMBERS       HND04430
                STM,R2      M3READ         STORE OCX IN READ INSTRUCTION    HND04440
                LDS,R2,5                   PICK UP IOS AND FUNCTION CODE    HND04450
                MLR,R2,R2                  ZERO I/O STATE                   HND04460
                ORI,R2      #0300          SET I/O STATE TO 3  READING MEM  HND04470
                STS,R2,5                                                    HND04480
                BLM,R14     SETIME         SET TIMER                        HND04490
                LDI,R2      #D800          PICK UP TI COMMAND               HND04500
        M3READ  OCB,R2,0                   OUTPUT TI COMMAND                HND04510
                BRU         DQINT          DEQUEUE NEXT INTERRUPT           HND04520
        M3STRD  LDS,R2,4                   SET INDICATOR FOR MEMORY PRINT   HND04530
                STS,R2,15                  STORE BUFFER ADDR IN TABLE       HND04540
                LDS,R2,5                   PICK UP IOS AND FUNCTION CODE    HND04550
                MLR,R2,R2                  ZERO I/O STATE                   HND04560
                ORI,R2      #0200          SET IOS TO 2 WAITING             HND04570
                STS,R2,5                                                    HND04580
                BLM,R14     DELAY          GO SET DELAY TIMER               HND04590
                BRU         DQINT          GO DEQUEUE NEXT INTERRUPT        HND04600
                EJT                                                         HND04610
        M3TIME  LDS,R2,5                   PICK UP IOS AND FUNCTION CODE    HND04620
                MBR,R2,R2                  ZERO FUNCTION CODE               HND04630
                CRMB,R2     FOUR,M3TIMO,M3CHK3 IF IOS = 4 GO ABORT AND START   HND04640
        M3CHK3  CRMB,R2     THREE,M3TIMO,M3CK2 IF IOS = 3 GO ABORT & START     HND04650
        M3CK2   CRMB,R2     TWO,M3BEGN,M3CK1   IF IOS = 2 GO START AGAIN       HND04660
        M3CK1   CRMB,R2     ONE,M3TIMO,M3HLTT  IF IOS = 1 GO ABORT & START     HND04670
```

Fig. 2. Part of sample source program

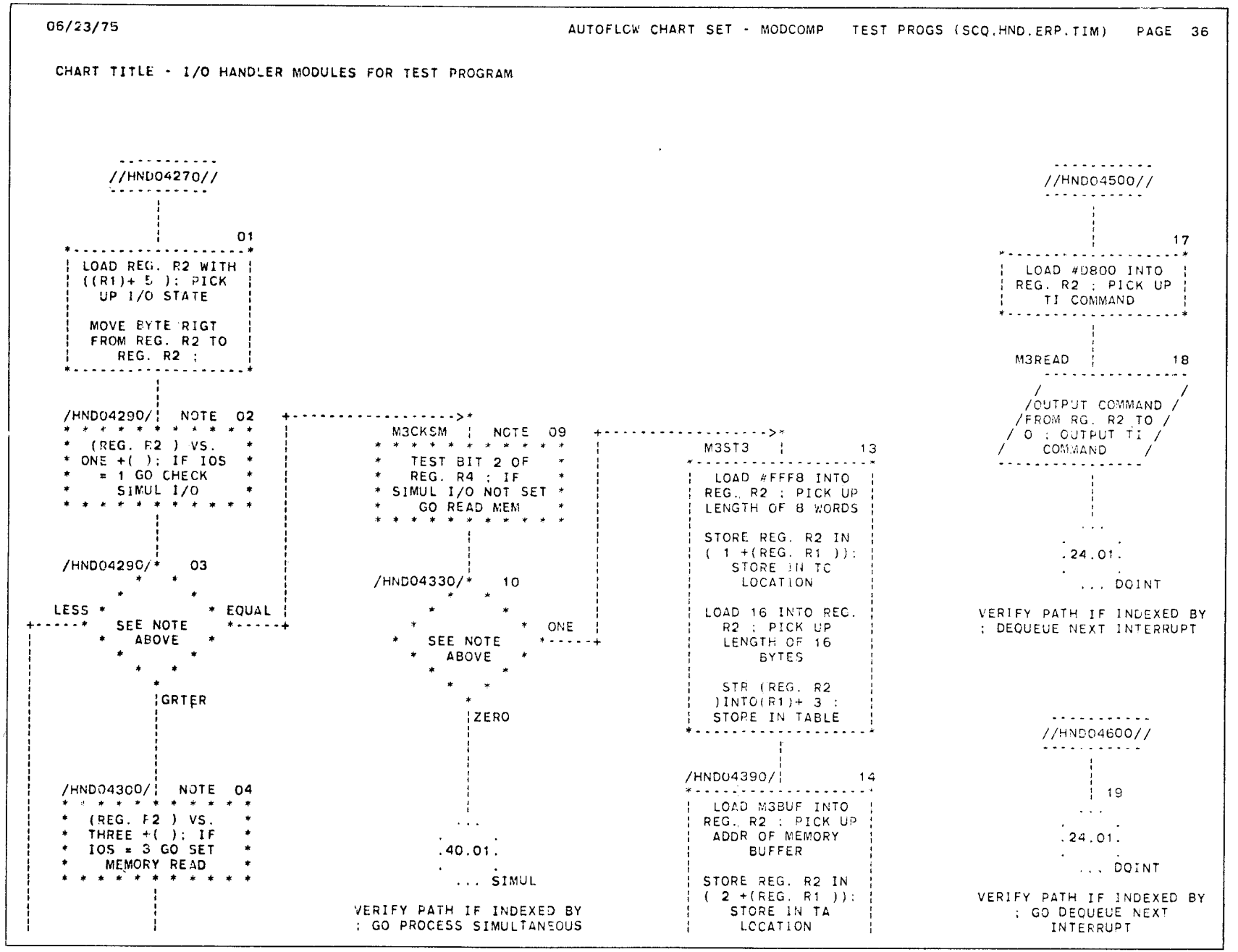


Fig. 3. Part of generated flow chart corresponding to sample in Fig. 2