The Bell System Technical Journal
Vol. 53, No. 1, January 1974
Printed in U.S.A.

Automatic Intercept System:

Development Support Systems

By I. D. BUCK, G. D. CRUDUP, C. W. KEMP, and G. C. VOGEL

(Manuscript received September 29, 1971)

A series of hardware-software packages has been developed to aid the Automatic Intercept System (AIS) development. These packages involve a PDP-9 support computer connected to the AIS control unit through specially designed hardware. Described in this paper are a real-time debugging system, a series of three support computer programs which aid in the administration of AIS program development, and two utility programs used in connection with the AIS file subsystem.*

I. INTRODUCTION

Development of a real-time system such as the Automatic Intercept System[1] requires many support programs. The assembler, loader, and simulator used by AIS were developed by the No. 2 ESS Switching System development.[2] In addition, a real-time debugging system called "ORACLE" and a series of utility and program administration programs have been developed and are the subject of this paper.

1.1 Real-time debugging

A debugging aid for real-time program development should provide all the features of nonreal-time debugging aids but in addition be noninterfering with the system operation. The objectives of giving all the features of normal debugging aids and being noninterfering are difficult to meet since most debugging routines suspend program execution to obtain data concerning the state of the program.

* PDP is a trademark of Digital Equipment Corp., Maynard, Mass.

ORACLE, the real-time debugging system, uses a combination of software, special interface hardware, and a PDP-9 computer to provide a noninterfering debugging package. ORACLE uses the PDP-9 [3] support computer to monitor, collect data, and print the results of AIS operations. Basically, ORACLE provides three things:

(*i*) Records of the "from" and "to" addresses of branch (transfer) instructions executed by the AIS control unit.
(*ii*) Dumps of the contents of AIS temporary memory (call store).
(*iii*) Means by which the printing of the above data can be activated.

1.2 Program administration

Program administration is the process of controlling, documenting, and verifying the correctness of a total system program. In the AIS development, programmers produce separate programs which are assembled on an IBM 360. The results of many such assemblies are linked together by a loader program to form an image of the AIS program store on magnetic tape. This image must be transferred onto "magnet cards" which form the AIS memory.[4] AIS uses the PMT (permanent magnet twistor) memory. In this memory, information is stored in small bar magnets attached to an aluminum card. It is the use of this type of memory in AIS, and the lack of high-speed input such as magnetic tape or high-speed paper tape, that has prompted the development of the three program administration programs (MAGNUS, PSUTY, and OVRWRT).

MAGNUS reads AIS program store data from tape and controls an MCW (memory card writer)[5] which records the data on PMT memory cards. PSUTY (*P*rogram *S*tore *UT*ilit*Y*) is used to compare magnetic tape images of the AIS program store with the actual contents of the store. It can also generate tape images from the actual program store contents. OVRWRT (*OVeRWRiTe*) is an assembler which produces program patches to the AIS program. It allows the AIS programmers to rapidly produce changes for fixing program bugs. OVRWRT translates assembly language statements into machine language, overlays the assembled code on top of what already is in the AIS program store, and produces new PMT cards via the MCW. Each of these programs will be described in detail in later sections.

1.3 Disc backup and system parameter initialization

Two programs, DISKUS and PDATA, have been developed to back up the AIS file data base and to change the parameter data stored on that file.[6,7]

1.4 AIS—PDP-9 hardware interfaces

Figure 1 shows the three hardware interface circuits used by the PDP-9 development support programs. The real-time debugging interface provides the means by which AIS program execution is monitored. It provides a path for passing the contents of AIS call store into the PDP-9's memory. The asynchronous communication register, connected between the PDP-9 I/O bus and the AIS scan answer bus, provides a path from the PDP-9 to the AIS. Over this path both commands and data are passed to an AIS program called CPTSTA. The memory card writer interface connects the PDP-9 to the memory card writer and is used by MAGNUS and OVRWRT.

II. REAL-TIME DEBUGGING

Debugging of a system such as AIS is more difficult than that involved with nonreal-time systems for several reasons:

(*i*) Real-time operation.
(*ii*) Random input traffic.
(*iii*) Added reliability requirements that make system malfunctions an added input rather than a reason for stopping the system.
(*iv*) Multiprogramming and multiprocessing.
(*v*) Large, complex programs.

These reasons indicate a need for a debugging aid which monitors the program execution in the AIS without any disturbance of the AIS. ORACLE is such a debugging aid. Within the support processor's memory a complete image of the AIS call store and a trace of transfers that occur in AIS are recorded. The transfer trace contains up to 128 "from-to" address pairs. Match circuits monitor the program address, call store address, and call store information to control the collecting and printing of call store images and transfer trace data.

2.1 Hardware description

The real-time debugging system uses a wired logic interface controlled by support computer programs to make logical decisions

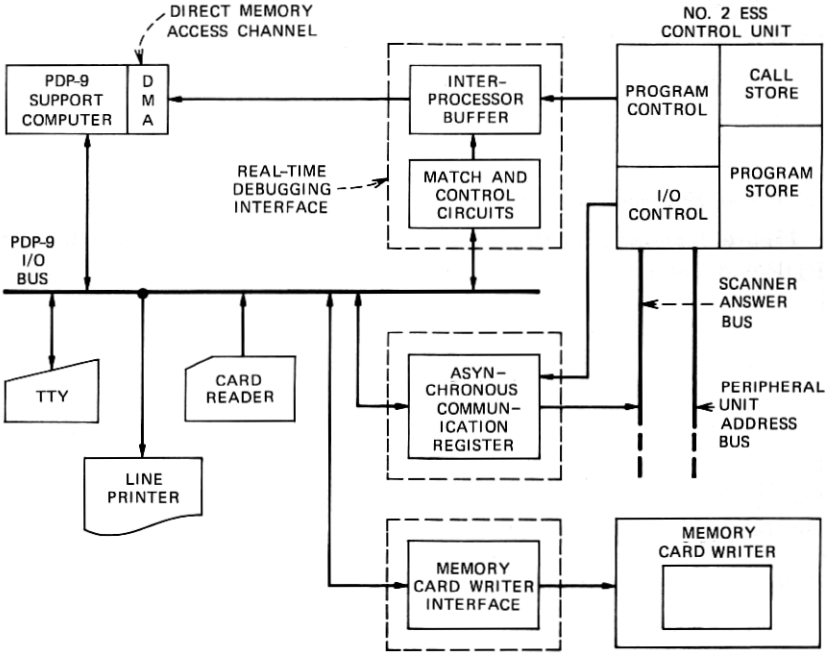

Fig. 1—Block diagram of the AIS development support system.

based upon data inputs from AIS and to control the flow of data from the AIS to PDP-9 memory. Results of the wired logic decisions initiate program action to tabulate and print the desired output data from the PDP-9's core memory.

A block diagram of the real-time debugging hardware is shown on Fig. 2. Three sets of data leads are utilized from the AIS processor. They are the PA (program store address), CSA (call store address), and CSI (call store input) data. The PA leads address the PMT memory and are used for records of transfer instructions executed by the AIS control unit. The CSA and CSI leads provide an image of the AIS call store in PDP-9 core for call store dumps. Eight matcher circuits (M0–M7) and three registers, CR (control register), FR (function register), and SR (status register), form the logic which controls data flow from the AIS to the support computer through a DMA (direct memory access) channel.

2.1.1 Matcher setup

A user inputs requests to the utility system via the support processor TTY or card reader. ORACLE sets up these requests via IOT

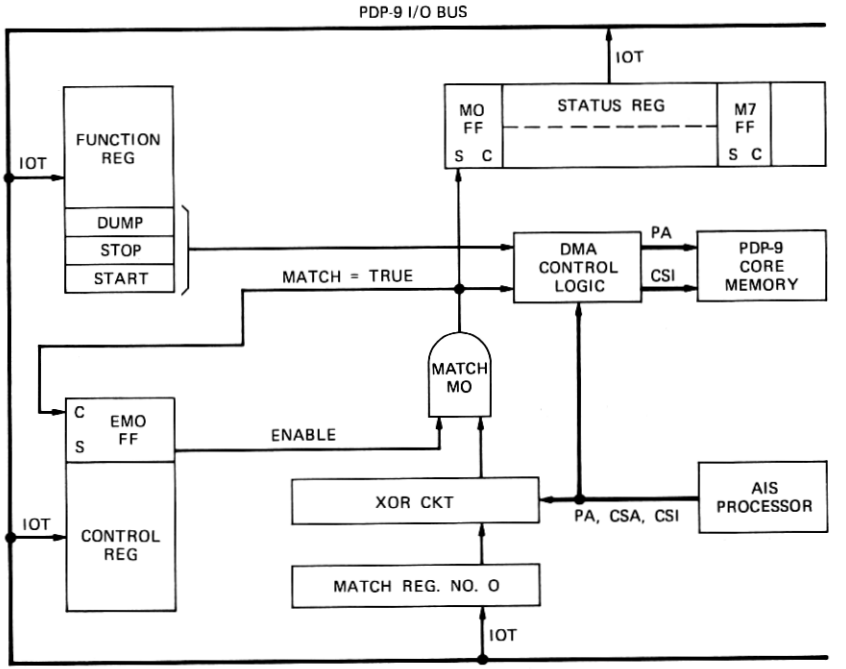

Fig. 2—Real-time debugging system: hardware interface between AIS and the PDP-9.

(input/output transfer) instructions to the match register and control registers in the utility hardware. When a match register is enabled, its contents are continuously matched against the PA leads by an exclusive "OR" circuit. If a match occurs, a bit corresponding to the matcher number is set in the SR, the match circuit is disabled by clearing its associated enable bit in the CR, and action is taken by the DMA control logic to start or stop the data flow to the support computer's memory.

2.1.2 Registers

The CR contains two bits per match circuit to enable each matcher circuit either conditionally or unconditionally. When a match circuit fires, its enable bit is cleared to prevent any further action by that matcher. The FR contains three bits per match circuit. These three bits determine what action will be taken by the DMA control logic, on the data flow, when a matcher fires. The "start" bit starts the flow of transfer addresses to be stored in the PDP-9. The "stop" bit stops the storing of transfer addresses. "Dump" stops the data

flow on the CSI leads which provides a call store image. The SR contains one bit for each matcher. These bits are set when the corresponding matcher "fires." Matcher firing also causes a program interrupt in the support computer to signal ORACLE that some action should be taken. The status register may be read by the program via an IOT instruction.

2.1.3 Direct memory access control

The DMA control logic derives all the control signals for strobing matchers and controls the storing of transfer trace and call store data in memory. Decision logic is located here to start and stop data flow dependent upon a matcher firing and the contents of the FR bits for that matcher.

2.1.4 Special match functions

Matchers 0–6 compare the data loaded in their match register against the PA leads from the AIS control unit. Match circuit 0 (M0) is unique in that it may be set up by program to conditionally enable one or more of the remaining match circuits. Until M0 fires, the conditionally enabled matchers are disabled. Matcher 7 compares its data against the CSA leads. This matcher provides all the functions of matchers 0–6, and in addition performs a bit matching function. Matcher 7 is divided into three major hardware sections:

(*i*) CSA match register,
(*ii*) CSI match register, and
(*iii*) CSI mask register.

Two modes of operation may be used with this configuration. If the CSA match register is loaded with an address and the mask register is zero, then the matcher acts as a CSA matcher. If the CSA and CSI are loaded with information and the mask register is set to all ones, the match looks for an exact match on the CSA and all the bits of the CSI. A zero mask bit is a "don't care." With this circuitry, the reading or writing of specified data at a given call store address can initiate utility functions.

2.2 Control program

2.2.1 General organization

Figure 3 is a functional diagram of the real-time debugging software. Primary control is from the PDP-9 console TTY with data coming from the card reader. An input deck, shown in the upper

left of Fig. 3, contains a description of what data are to be collected and when they are to be printed out. The input program has responsibility for checking the syntax of the UCL (utility control language) statements on each input card and translating them into a sequence of numerical codes in the table LCTAB. At the end of each card the input program passes control to the matcher setup program which checks for consistency between all cards processed so far, builds an image of the hardware registers, and stores data concerning each matcher in the MCB (matcher control block) table. An end of file in the input deck causes the matcher setup program to load the hardware registers and turn the hardware on with a final "GO" command. The "firing" of one or more of the matchers causes a PDP-9 program interrupt, passing control to the utility function execution program which collects hardware status information from the interface circuits and causes data to be printed out based on the number of the matcher that fired and the contents of the corresponding MCB. The overall operation is explained by the following example.

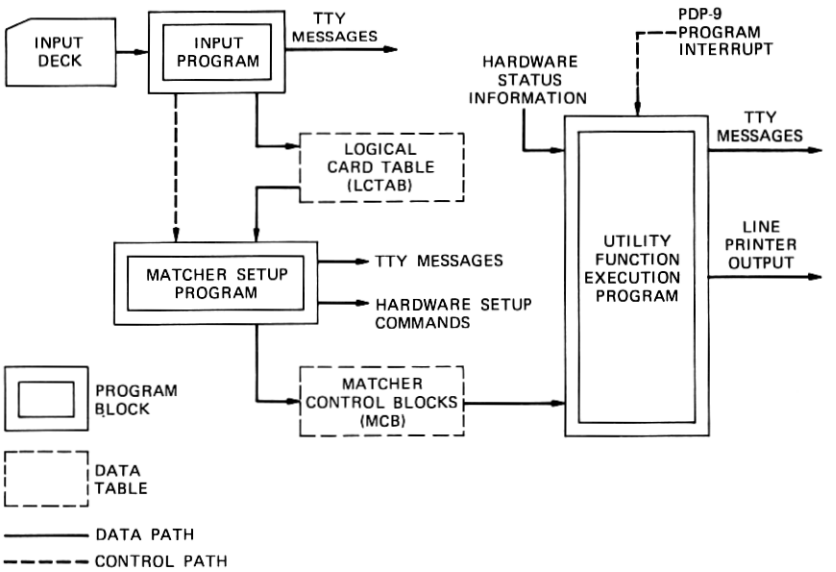

Fig. 3—Real-time debugging system: functional software diagram.

2.2.2 Example

Consider the input UCL statement

```
//START MATCH M0, PA=30000, STARTT, ENAB(M1, M7)
//SYMB. A MATCH M1,PA=17,DUMP (250, 500)
//SYMB. B MATCH M7,BIT(360, 40),ENDTT
/*
```

The first statement says "set matcher M0 to 30000 and, when a match occurs, start the transfer trace hardware and enable matcher M1 and M7." Because of the ENAB(M1, M7) function on M0, matchers M1 and M7 will not be "turned on" until after matcher M0 has "fired." Statement number 2 specifies that matcher M1 be set up for a program address match at 17 and when the match occurs, a dump of the AIS call store image between 250 and 500 will be printed. Matcher 7 specifies a match when 40 is written into or read from location 360. When M7 "fires" the transfer trace data are printed due to the ENDTT (*END* *T*ransfer *T*race) function. The /* terminates the matcher setup phase of the debugging system.

Figure 4 shows the output that results from the sample input statements. Lines 1–4 are the input statements. Line 5 indicates that the matcher with the symbol START has fired and the SEQ# = 01 says that it was the first matcher to fire. At line 6 we see that the matcher labeled SYMB. B has fired and that a transfer trace will follow. The PA field indicates the program address which caused the bit match to occur and the sequence number indicates that this was the second matcher to fire. Lines 7–24 are the transfer trace data. Line 25 states that the matcher labeled SYMB. A has fired and that a call store dump will follow. Line 37 shows that groups of all zero words are compressed into a single line.

III. PROGRAM ADMINISTRATION AND FILE UTILITY SYSTEM

The program administration and file utility system comprises software and hardware to provide such functions as changing AIS file parameters, comparing AIS PMT store contents with store images on magnetic tape, and the actual writing of PMT cards from magnetic tape via the memory card writer.

3.1 Hardware description

The program administration hardware is shown on Fig. 5. Two registers are necessary for the functions described. The MCW output register passes information from the PDP-9 to the MCW upon request to magnetize AIS PMT cards. The ACR (asynchronous com-

```
LINE 1.... //START MATCH M0,PA=30000,STARTT,ENAB(M1,M7)
           //SYMB.A MATCH M1,PA=17,DUMP(250,500)
           //SYMB.B MATCH M7,BIT(360,40),ENDTT
LINE 4.... /*

LINE 5....    START              SEQ#= 01

LINE 6....    SYMB.B TRA TRACE  PA= 030240 SEQ#= 02
                        FROM ADDRESS
                           TO ADDRESS
LINE 7.... 030001 033764
           033766 033770
           033772 030002
           030005 030027
           030031 030033
           030035 030101
           030101 033653
           033666 030102
           030103 030117
           030121 030160
           030160 030161
           030172 030202
           030214 032556
           032566 032561
           032565 032576
           032577 030215
           030215 030217
LINE 24... 030240 032421

LINE 25...    SYMB.A CS DUMP    SEQ#= 03
                        CALL STORE ADDRESS X
                                  DATA FOR CALL STORE
                                  ADDRESS X THROUGH X + 7
LINE 26... 000250 045636 000400 000000 000000 000000 000000 000000 000000 000000
           000260 045636 000400 000000 010710 000000 000000 000000 000000 000000
           000270 136573 000000 001200 010710 000323 005102 000000 005317
           000300 077731 700130 001200 011262 000323 005116 000000 000717
           000310 045636 000400 001200 000000 000323 005116 000002 005317
           000320 045636 000000 000002 010710 070005 001414 070005 001622
           000330 045636 000400 001200 010515 000323 005102 000000 005317
           000340 136773 000000 001200 010370 000100 000400 000002 005317
           000350 064363 000100 014675 010370 000001 000400 000000 000007
           000360 100430 000030 014536 010373 000000 007500 000221 000520
           000370 000000 000000 005345 004070 000020 000400 000000 000460
LINE 37...    000400  -  000467  =  0
           000470 000000 000000 000000 000000 000000 000000 000000 177777
LINE 39...    000500  -  000507  =  0
```

Fig. 4—Sample output of the real-time debugging system.

munication register) provides a data link from the support computer to the AIS processor.

3.1.1 Asynchronous communication register

The ACR is connected to the AIS scan answer bus and is loaded from the PDP-9 I/O bus by an IOT instruction. A program (CPTSTA) running in the AIS processor scans the ACR for both instructions and data. If control bits ENV and ASW both equal 1, then the 16 data bits are passed over the scan answer bus to the control unit. CPTSTA issues a data acknowledge to clear the ENV and ASW

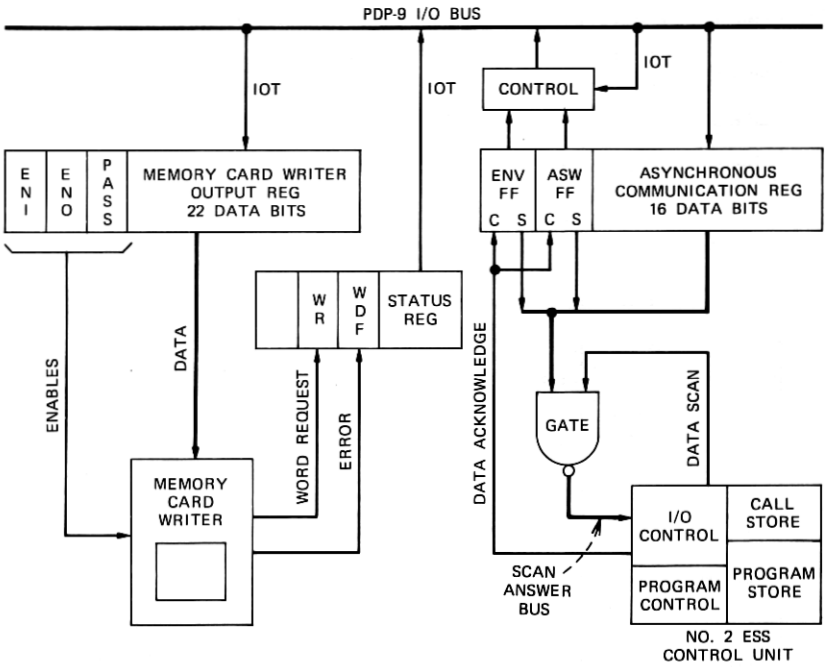

Fig. 5—Asynchronous communication register and memory card writer interfaces.

flip-flops. This indicates to the PDP-9 program that the transfer is complete and new data may be sent.

3.1.2 Memory card writer

The MCW is an electromechanical device capable of magnetizing PMT cards for AIS memories. One PMT card consists of 64 rows of small bar magnet with 44 magnets per row. A write head is driven over the PMT card, writing one row at a time. An internal 44-bit register stores the information for each row. As the write head approaches each row of bar magnets, a word request is sent to the SR which in turn creates an interrupt to the control program (MAGNUS). MAGNUS responds with a 44-bit data message via the MCW output register. The process continues until all 64 rows are written on the PMT card.

3.1.3 MCW output register

Upon receiving a word request from the SR, the MCW output register is loaded with its data and enables. The 44-bit data register

in the MCW is loaded by two paralled 22-bit transfers from the MCW output register. Two enables, EN0 and EN1, are used to steer each 22-bit word transfer into the proper position in the MCW input register.

If either a mechanical or electrical malfunction occurs during the writing of a PMT card, a WDF (word delivery failure) is sent to the SR. A program interrupt signals the support computer program that an error has occurred.

3.2 Program store utility program

PSUTY (*P*rogram *S*tore *UT*ilit*Y*) program uses the AIS—PDP-9 interface to match AIS program store contents with program store images on magnetic tape and to create program store images on tape. PSUTY determines which function to perform from the user's TTY messages. On match functions between tape and the AIS program store, mismatches are listed on the line printer. The tape creation and tape copy functions indicate their completion via a TTY message.

PSUTY communicates with the AIS via the ACR and the DMA channel. The AIS program CPTSTA contains the AIS software necessary to interface with PSUTY. Requests for the contents of a program store card are sent to the AIS via the ACR. CPTSTA reads the contents of the specified plane into the AIS call store. As these data are written in call store, they are also written into the PDP-9's memory via the DMA channel. When this operation is complete, CPTSTA clears the ENV and ASW bits in the ACR which tells PSUTY to begin processing the requested program store data. This process is repeated until all AIS program store data have been processed.

3.3 PMT card magnetization

MAGNUS magnetizes AIS PMT cards from program store images on magnetic tape. MAGNUS is a conversational mode program in which the user can request specific store modules or single planes for magnetization. When the specified data have been found on the magnetic tape input file, the program arranges the data for output and types a message on the TTY. The user may then magnetize the PMT card(s). Error routines are included in MAGNUS which allow automatic restart in the case of a word delivery failure.

MAGNUS communicates with the MCW via the memory card writer interface. MAGNUS reads enough data from magnetic tape

to magnetize one PMT card. When the MCW is ready to start magnetizing the card it sends a WR (word request) signal through the interface circuit to the status register. MAGNUS answers the WR by sending 44 bits of data through the MCW output register. The MCW uses the 44 data bits to magnetize one row of magnets on the PMT card. A series of 64 such signals and answers occurs to magnetize one plane.

3.4 Laboratory change assembler

The laboratory change assembler, OVRWRT, is used to assemble and insert program changes into the AIS program store. OVRWRT (Fig. 6) is a conventional two-pass assembler with a magnetizing routine added. Eventually laboratory program changes must be incorporated into the program source files using the EDITOR program and reassembled by the SWAP assembler.[2] Therefore, OVRWRT has been designed to accept both assembly language statements which are compatible with SWAP and control statements required by the EDITOR program.

3.4.1 Special pseudo-operations

Since a program change in one part of a program may use information such as symbol definitions derived from other parts of the program, it is necessary to provide special pseudo-operations for supplying this information. Special pseudo-operations begin with a percent sign (%). The most basic of the special pseudo-operations is %SET. It is generally used to define symbols for use in a program

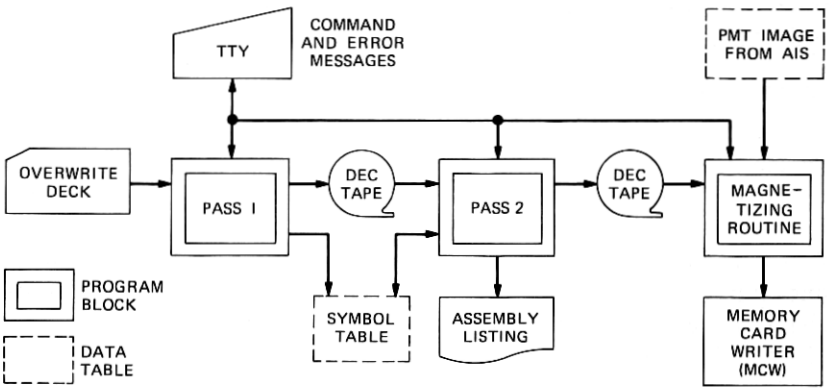

Fig. 6—Laboratory change assembler: functional software diagram.

patch which otherwise would be undefined. For example, the following statement would assign the value 12345 to the symbol TAG.

```
TAG         %SET        12345
```

Symbols may be redefined by %SET. The corresponding regular pseudo-operations (no percent sign) may also be used. However, special pseudo-operations are ignored by the EDITOR program while the regular ones are not. This scheme automatically eliminates inserting unnecessary statements in the user's source program.

3.4.2 System macros for program patching

Program changes requiring the addition or removal of instructions rather than substitution are generally made using system macros. The use of system macros facilitates the subsequent job of editing the program source files. Three system macros, PATCH, TAKEOUT, and WIPEOUT, are used for program patching. The PATCH macro is used for program changes requiring additional instructions to be added in the middle of the program. This is accomplished by overwriting a full word in the program with a transfer (branch instruction) to an unused area in program store. The overwritten instruction as well as the instructions to be added may then be written in the unused area followed by a transfer back to the main body of the program. The following lines show an example of the patch macro:

```
NAME        PCHORG      5000
NAME        PATCH 1
KH          ZGL
            GRXLR
            HLR
            GGR
NAME        PCHEND
```

Expansion of this sample can be found in the assembly listing of Fig. 7. TAKEOUT and WIPEOUT are degenerate forms of the PATCH macro and are used to remove blocks of instruction. TAKEOUT overlays the unwanted instructions with NOP (no operation) instructions while WIPEOUT inserts code which will cause an error condition if an attempt is made to execute it.

3.4.3 The assembly listing

Figure 7 shows an assembly listing produced by OVRWRT. Since input statements are free-field format, the fields are aligned in the

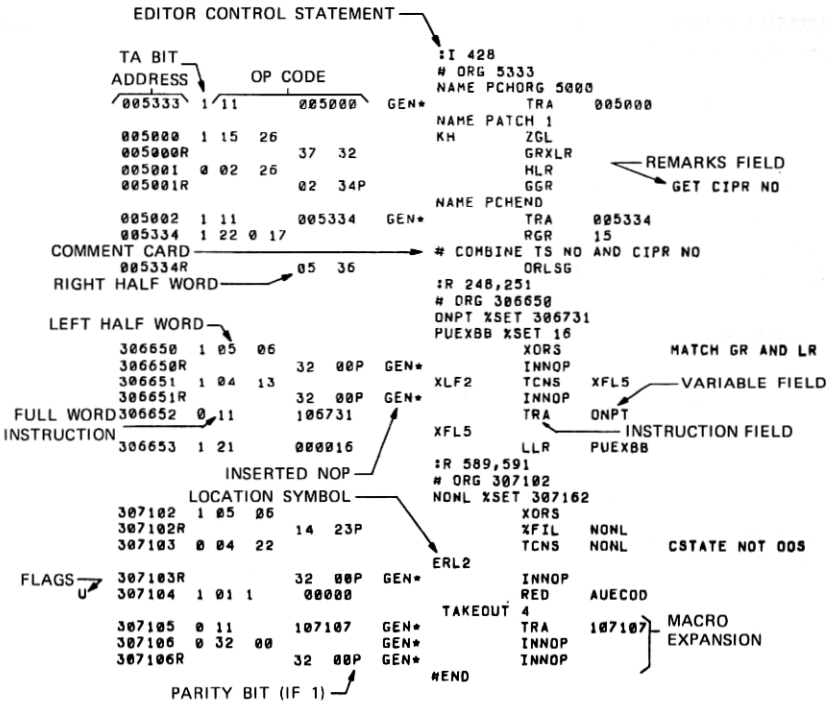

Fig. 7—Example program listing produced by OVRWRT.

listing for greater readability. Assembly errors are indicated by flagging the appropriate statement. Machine language code is presented in one of several field octal formats depending on the instruction classification. Instructions generated by OVRWRT are labeled as such and are listed with both the machine language and assembly language representations. Comments are printed as they appear on the input statements. When a program change has been inserted, the assembly listing is used for program documentation.

3.5 Disc utility programs

The AIS disc utility programs (DISKUS and PDATA) use the AIS—PDP-9 interface circuits to record, restore, and match the contents of the AIS disc file to disc file images on magnetic tapes.[6] These two programs are development tools. DISKUS is used to create backup copies of the AIS intercept number data base. PDATA was developed to record and change the AIS parameter data which

are recorded on the AIS disc. In the testing of AIS programs it is necessary to change the configuration of the AIS laboratory model and to change the parameter data accordingly.

3.5.1 Reading and writing AIS discs

Communication between the PDP-9 and the AIS disc proceeds via the DMA channel and the ACR. To initiate a transfer of a block of AIS disc data to the support computer, the ACR is loaded with a function code. When the flag bits ENV and ASW are zeroed, the ACR is loaded with the desired disc address. The AIS program (CPTSTA) locates the disc data and transfers them through the DMA channel into the PDP-9's memory. Communication from the PDP-9 to the AIS disc is established by loading the ACR with another function code. When the code is accepted, the ACR is loaded with the disc address, and then, sequentially, with the 80 data words to be written in the specified disc block.

3.5.2 Disc backup and loading

DISKUS provides AIS disc file backup and restoration of the file data. A complete disc image is stored on a set of 16 low-density magnetic tapes. The user has a choice of three operations: copying the disc contents onto the tapes, restoring the disc from the tapes, or matching the contents of the disc against the tapes. Each of the 16 tapes is assigned a specific portion of the disc. The user has the option of using any one or all of the tapes for each of the three operations.

3.5.3 System parameter initialization

System parameter initialization on the AIS disc is accomplished by PDATA. A maximum of 16 different sets of AIS system parameters may be recorded on one DECtape. PDATA users have the option of: copying current system parameters on tape, changing AIS configuration via a previously recorded set of system parameters, or matching current system parameters to a previously recorded set. Each set of system parameters recorded is accompanied by a description record and a unique name. System parameters occupy a fixed location on the AIS disc.

IV. SUMMARY

Each of the systems described has had a profound impact on the AIS development project. The real-time debugging system and the

laboratory change assembler were at first met with skepticism as to the need for such systems, or with at least a wait-and-see attitude. After the systems were introduced, the AIS programmers became heavily dependent on ORACLE and OVRWRT. If the support computer is down because of a hardware failure, the programmers would rather wait for it to be fixed than return to the debugging and program patching methods previously used.

PSUTY and MAGNUS have greatly helped the program administration of AIS. PSUTY reduced the time required to compare AIS program load images from the IBM 360 to the actual program in the AIS machine from several hours to a matter of minutes. The authors strongly feel that every large real-time system should have a series of utility systems such as the ones described and that no proposal for a real-time system development is complete without an outline of what development support systems will be needed.

REFERENCES

1. C. J. Byrne, W. A. Winckelmann, and R. M. Wolfe, "Automatic Intercept System: Organization and Objectives," B.S.T.J., this issue, pp. 1–18.
2. M. E. Barton, N. M. Haller, and G. W. Ricker, "[No. 2 ESS] Service Programs," B.S.T.J., *48*, No. 8 (October 1969), pp. 2865–2896.
3. *PDP-9 User Handbook*, Maynard, Mass., Digital Equipment Corp.
4. T. E. Browne, T. M. Quinn, W. N. Toy, and J. E. Yates, "[No. 2 ESS] Control Unit System," B.S.T.J., *48*, No. 8 (October 1969), pp. 2619–2668.
5. C. F. Ault, L. E. Gallaher, T. S. Greenwood, and D. C. Koehler, "No. 1 ESS Program Store," B.S.T.J., *43*, No. 5 (September 1964), pp. 2097–2146.
6. J. W. Hopkins, P. D. Hunter, R. E. Machol, J. J. DiSalvo, and R. J. Piereth, "Automatic Intercept System: File Subsystem," B.S.T.J., this issue, pp. 107–132.
7. H. Cohen, D. E. Confalone, B. D. Wagner, and W. W. Wood, "Automatic Intercept System: Operational Programs," B.S.T.J., this issue, pp. 19–69.