Use of COBOL II Macros for Database Access

Jerrel Baxter, CDP
School of Optometry

INTRODUCTION

IMAGE/3000 provides a powerful tool for development of data storage and retrieval applications. Its use of subroutine CALL's for operation invocation allows fairly simple inclusion into all programming languages for the Hewlett Packard 3000. However, this CALL level interface also places total responsibility for error checking and reporting on the programmer which can require rather voluminous coding. This paper describes one method for dealing with this problem for COBOL II programmers, use of COBOL II macros. The methods used are also applicable to use of VPLUS/3000 for terminal screen handling and DSG/3000 for graphics generation.

STANDARDS

In order to create an orderly process for generating code for database references the following standards were adopted.

1. A COPYLIB module defining a parameter set for use in all programs with database references was created. See Appendix A for a listing of the defined module.

2. Naming conventions for creating reference variables for database names, dataset names, search item names, search item values, and dataset record layouts were defined and made mandatory for all programs.

3. COBOL II macros were defined for making all database calls. Use of the COBOL II macros guarantees adherence to standard error checking and makes the purpose of many database calls simpler to recognize. The use of the macros also makes it easy to incorporate enhancements to the error notification function in the future by replacing the current error handling code within the macro definitions.

As an example the following shows DATA DIVISION and PROCEDURE DIVISION code segments for a simple database referencing program:

```
:FILE COPYLIB=COPYLIB.COBOL
:FILE COPYDEF=COPYDEF.COBOL
 .
 .
 .
$CONTROL USLINIT,NOMIXED,NOLIST
$INCLUDE COPYDEF.COBOL
 .
 .
 .
DATA DIVISION.
WORKING-STORAGE SECITON.
COPY DBPARAMS.
01  DB-CLIENT                  PIC X(10) VALUE "  CLIENT  ".
```

```
       01  DS-M-IDENTIFICATION      PIC X(16) VALUE "M-IDENTIFICATION".
       01  DS-D-PROJECTS            PIC X(16) VALUE "D-PROJECTS      ".
       01  DS-A-SHORTNAMES          PIC X(16) VALUE "A-SHORTNAMES    ".
```

(Continued on next page.)

```
       01  DI-ACCOUNT               PIC X(16) VALUE "ACCOUNT         ".
       01  DV-ACCOUNT               PIC 9(4) DISPLAY.
       01  DI-SHORTNAME             PIC X(16) VALUE "SHORTNAME       ".
       01  DV-SHORTNAME             PIC X(6).
       01  DR-M-IDENTIFICATION.
       COPY MIDENTIF.
       01  DR-D-PROJECTS.
       COPY DPROJECT.
       01  DR-A-SHORTNAMES.
       COPY ASHORTNA.
        .
        .
        .
       PROCEDURE DIVISION.
       MAIN-PROCESSING SECTION.
       INITIALIZATION.
           %DBOPEN(DB-CLIENT#,1#)
       CONTINUE-PROCESSING.
        .
        .
        .
      * Note use of V version of COBOL II macros to indicate VPLUS/3000
      * is active (error routine closes VPLUS form and terminal files
      * prior to aborting).
           %VDBFIND(DB-CLIENT#,DS-D-PROJECTS#,DI-SHORTNAME#
                   ,DV-SHORTNAME#)
           IF DB-MISSING
             MOVE "Client not found." TO V-MESSAGE
             GO TO CONTINUE-PROCESSING.
           %VDBGET(DB-CLIENT#,DS-D-PROJECTS#,5#,DB-ALL-LIST#
                   ,DR-D-PROJECTS#,DB-DUMMY#)
           IF DB-EOC
             MOVE "Client not found." TO V-MESSAGE
             GO TO CONTINUE-PROCESSING.
        .
        .
        .
      * End of program.
```

DEFINITIONS

The standard macro definitions and an example of their calling sequence are listed below:

```
      $DEFINE %DBOPEN=
      * %DBOPEN(DB-BASENAME#,1#)
      * Opens the specified database in the specified mode.  On open
      * failure an error message is output, the data base is closed
    * (per IMAGE manual recommendations), and the program is aborted.
       CALL "DBOPEN" USING !1
          DB-PASSWORD
          DB-MODE-!2
          DB-STATUS
       IF NOT DB-OKAY
        CALL "DBEXPLAIN" USING DB-STATUS
        CALL "DBCLOSE" USING !1
          DB-DUMMY
```

(Continued on next page.)

```
          DB-MODE-1
```

```
        DB-STATUS
      STOP RUN.#

$DEFINE %DBCLOSE=
* %DBCLOSE(DB-BASENAME#)
* Closes the specified data base. On close failure an error
* message is output and the program is aborted.
    CALL "DBCLOSE" USING !1
       DB-DUMMY
       DB-MODE-1
       DB-STATUS
    IF NOT DB-OKAY
      CALL "DBEXPLAIN" USING DB-STATUS
      STOP RUN.#

$DEFINE %DBREWIND=
* %DBREWIND(DB-BASENAME#,DS-M-SETNAME#)
* Rewinds the specified dataset within the specified database.
* On rewind failure an error message is output and the program
* is aborted.
    CALL "DBCLOSE" USING !1
       !2
       DB-MODE-3
       DB-STATUS
    IF NOT DB-OKAY
      CALL "DBEXPLAIN" USING DB-STATUS
      STOP RUN.#

$DEFINE %DBBEGIN=
* %DBBEGIN(DB-BASENAME#)
* Outputs a text message to the IMAGE logging file and marks the
* beginning of a logical transaction. On failure an error message
* is output and the program is aborted.
    CALL "DBBEGIN" USING !1
       DB-TEXT
       DB-MODE-1
       DB-STATUS
       DB-TEXT-LENGTH
    IF NOT (DB-OKAY OR DB-LOGGING-DISABLED)
      CALL "DBEXPLAIN" USING DB-STATUS
      STOP RUN.#

$DEFINE %DBEND=
* %DBEND(DB-BASENAME#)
* Outputs a text message to IMAGE logging file and marks the end of
* a logical transaction. On failure an error message is output and
* the program is aborted.
    CALL "DBEND" USING !1
       DB-TEXT
       DB-MODE-1
       DB-STATUS
       DB-TEXT-LENGTH
    IF NOT (DB-OKAY OR DB-LOGGING-DISABLED)
      CALL "DBEXPLAIN" USING DB-STATUS
      STOP RUN.#
$DEFINE %DBDELETE=
* %DBDELETE(DB-BASENAME#,DS-M-SETNAME#)
* Deletes the currently referenced data record in the specified
* dataset for the specified database. On failure an error
* message is output and the program is aborted. A missing record
* or chain head is not considered to be a failure.
    CALL "DBDELETE" USING !1
       !2
       DB-MODE-1
       DB-STATUS
    IF NOT (DB-OKAY OR DB-MISSING OR DB-CHAIN-HEAD)
      CALL "DBEXPLAIN" USING DB-STATUS
      STOP RUN.#
```

```
$DEFINE %DBFIND=
* %DBFIND(DB-BASENAME#,DS-D-SETNAME#,DI-KEYNAME#,DV-KEY-NAME#)
* Finds the search path for the specified key value for the
* specified key item in the specified dataset and database. On
* failure an error message is output and the program is aborted.
* A missing key value is not considered to be a failure.
    CALL "DBFIND" USING !1
       !2
       DB-MODE-1
       DB-STATUS
       !3
       !4
    IF NOT (DB-OKAY OR DB-MISSING)
     CALL "DBEXPLAIN" USING DB-STATUS
     STOP RUN.#

$DEFINE %DBGET=
* %DBGET(DB-BASENAME#,DS-M-SETNAME#,7#,DB-ALL-LIST#
*      ,DR-M-SETNAME#,DV-KEYNAME#)
* Gets the data record corresponding to the parameter values. On
* failure an error message is output and the program is aborted.
* Beginning of file, end of file, beginning of chain, end of chain,
* and no entry are not considered failures.
    CALL "DBGET" USING !1
       !2
       DB-MODE-!3
       DB-STATUS
       !4
       !5
       !6
    IF NOT DB-GET-OKAY
     CALL "DBEXPLAIN" USING DB-STATUS
     STOP RUN.#

$DEFINE %DBLOCKBASE=
* %DBLOCKBASE(DB-BASENAME#)
* Locks the specified database. On failure guarantees release of
* all locks, outputs an error message and aborts the program.
    CALL "DBLOCK" USING !1
       DB-DUMMY
       DB-MODE-1
       DB-STATUS
    IF DB-CONDITION > 0
     CALL "DBEXPLAIN" USING DB-STATUS
```

(Continued on next page.)

```
     CALL "DBUNLOCK" USING !1
        DB-DUMMY
        DB-MODE-1
        DB-STATUS
     STOP RUN
    ELSE
     IF NOT DB-OKAY
      CALL "DBEXPLAIN" USING DB-STATUS
      STOP RUN.#

$DEFINE %DBLOCKSET=
* %DBLOCKSET(DB-BASENAME#,DS-M-SETNAME#)
* Locks the specified dataset for the specified database. On
* failure guarantees release of all locks, outputs an error message
* and aborts the program.
    CALL "DBLOCK" USING !1
       !2
       DB-MODE-3
       DB-STATUS
    IF DB-CONDITION > 0
     CALL "DBEXPLAIN" USING DB-STATUS
```

```
      CALL "DBUNLOCK" USING !1
         DB-DUMMY
         DB-MODE-1
         DB-STATUS
       STOP RUN
     ELSE
      IF NOT DB-OKAY
       CALL "DBEXPLAIN" USING DB-STATUS
       STOP RUN.#

$DEFINE %DBLOCKENTRY=
* %DBLOCKENTRY(DB-BASENAME#,DE-LOCKSPECIFICATION#)
* Locks according to the lock specifications in the specified
* table. On failure guarantees release of all locks, outputs an
* error message and aborts the program.
    CALL "DBLOCK" USING !1
       !2
       DB-MODE-5
       DB-STATUS
    IF DB-CONDITION > 0
     CALL "DBEXPLAIN" USING DB-STATUS
     CALL "DBUNLOCK" USING !1
        DB-DUMMY
        DB-MODE-1
        DB-STATUS
     STOP RUN
    ELSE
     IF NOT DB-OKAY
      CALL "DBEXPLAIN" USING DB-STATUS
      STOP RUN.#

$DEFINE %DBPUTALL=
* %DBPUTALL(DB-BASENAME#,DS-M-SETNAME#,DR-M-SETNAME#)
* Puts the data record to the specified dataset and database using
* @; for the record list. On failure generates an error message,
* unlocks the database (assumes normal use of mode 1 for DBOPEN)
* and aborts the program.

(Continued on next page.)
    CALL "DBPUT" USING !1
       !2
       DB-MODE-1
       DB-STATUS
       DB-ALL-LIST
       !3
    IF NOT DB-OKAY
     CALL "DBEXPLAIN" USING DB-STATUS
     CALL "DBUNLOCK" USING !1
        DB-DUMMY
        DB-MODE-1
        DB-STATUS
     STOP RUN.#

$DEFINE %DBPUTSAME=
* %DBPUTSAME(DB-BASENAME#,DS-M-SETNAME#,DR-M-SETNAME#)
* Puts the data record to the specified dataset and database using
* *; for the record list. On failure generates an error message,
* unlocks the database (assumes normal use of mode 1 for the
* DBOPEN), and aborts the program.
    CALL "DBPUT" USING !1
       !2
       DB-MODE-1
       DB-STATUS
       DB-SAME-LIST
       !3
    IF NOT DB-OKAY
     CALL "DBEXPLAIN" USING DB-STATUS
```

```
      CALL "DBUNLOCK" USING !1
         DB-DUMMY
         DB-MODE-1
         DB-STATUS
      STOP RUN.#

$DEFINE %DBPUT=
* %DBPUT(DB-BASENAME#,DS-M-SETNAME#,DA-M-SETNAME#
*       ,DL-ITEMLIST#)
* Puts the specified record to the specified dataset and database
* using an abbreviated item list and record area. On failure
* outputs an error message, unlocks the database (assumes normal
* use of mode 1 for DBOPEN) and aborts the program.
    CALL "DBPUT" USING !1
       !2
       DB-MODE-1
       DB-STATUS
       !4
       !3
    IF NOT DB-OKAY
      CALL "DBEXPLAIN" USING DB-STATUS
      CALL "DBUNLOCK" USING !1
         DB-DUMMY
         DB-MODE-1
         DB-STATUS
      STOP RUN.#
$DEFINE %DBUNLOCK=
* %DBUNLOCK(DB-BASENAME#)
* Unlocks all held locks for the specified database. On failure
* outputs an error message and aborts the program.
    CALL "DBUNLOCK" USING !1
       DB-DUMMY
       DB-MODE-1
       DB-STATUS
    IF NOT DB-OKAY
      CALL "DBEXPLAIN" USING DB-STATUS
      STOP RUN.#

$DEFINE %DBUPDATEALL=
* %DBUPDATEALL(DB-BASENAME#,DS-M-SETNAME#,DR-M-SETNAME#)
* Updates the data record to the specified dataset and database
* using @; for the record list. On failure generates an error
* message, unlocks the database (assumes normal use of mode 1 for
* DBOPEN) and aborts the program.
    CALL "DBUPDATE" USING !1
       !2
       DB-MODE-1
       DB-STATUS
       DB-ALL-LIST
       !3
    IF NOT DB-OKAY
      CALL "DBEXPLAIN" USING DB-STATUS
      CALL "DBUNLOCK" USING !1
         DB-DUMMY
         DB-MODE-1
         DB-STATUS
      STOP RUN.#

$DEFINE %DBUPDATESAME=
* %DBUPDATESAME(DB-BASENAME#,DS-M-SETNAME#,DR-M-SETNAME#)
* Updates the data record to the specified dataset and database
* using *; for the record list. On failure generates an error
* message, unlocks the database (assumes normal use of mode 1 for
* the DBOPEN) and aborts the program.
    CALL "DBUPDATE" USING !1
       !2
       DB-MODE-1
```

```
      DB-STATUS
      DB-SAME-LIST
      !3
    IF NOT DB-OKAY
     CALL "DBEXPLAIN" USING DB-STATUS
     CALL "DBUNLOCK" USING !1
        DB-DUMMY
        DB-MODE-1
        DB-STATUS
     STOP RUN.#
$DEFINE %DBUPDATE=
* %DBUPDATE(DB-BASENAME#,DS-M-SETNAME#,DA-M-SETNAME#
*      ,DL-ITEMLIST#)
* Updates the specified record to the specified dataset and
* database using an abbreviated item list and record area. On
* failure outputs an error message, unlocks the database (assumes
* normal use of mode 1 for DBOPEN) and aborts the program.
    CALL "DBUPDATE" USING !1
      !2
      DB-MODE-1
      DB-STATUS
      !4
      !3
    IF NOT DB-OKAY
     CALL "DBEXPLAIN" USING DB-STATUS
     CALL "DBUNLOCK" USING !1
        DB-DUMMY
        DB-MODE-1
        DB-STATUS
     STOP RUN.#
```

The V versions at the UAB School of Optometry are created by prefixing a V before the macro name and placing the following code prior to the DBEXPLAIN call in all macro definitions:

```
MOVE 9 TO V-CONTROL
CALL "VIEWIO" USING V-COM-AREA, @DATA-BUFFER
```

The VIEWIO procedure is a general interface to VPLUS which considerably simplifies normal access to VPLUS. Please see the screen usage standards for more information.

Please see Appendix A for a description of the naming conventions and required COPYLIB module for macro definitions' references.

DISCUSSION

Use of COBOL II macros has resulted in a consistent use of error detection and recovery while requiring less input by programmers. It also enables replacement of error handling code with more comprehensive reporting by modification of the macro definitions and recompilation with no examination required of individual source programs. See Appendix B for an example describing enhancement of the previously defined error reporting. Other advantages of the use of macros include reduction in source code disc space requirements, reduction in segment traps caused by the prior method of using PERFORMed routines with set-up MOVEs for similar access without a significant code space increase, simplification of calling sequences for IMAGE references with related time reduction for coding efforts, and simplification of debugging efforts as code is in-line which reduces the normal requirement of searching throughout an often large program source listing for various PERFORMed routines. REFERENCES

Thanks are due to the originators of COBOL II for including the macro processor capability (see Appendix A of Hewlett-Packard's COBOL II reference manual for a description of the macro processor, part number 32233-90001 and product number 32233A) and to Gerald Weinberg, author of High Level COBOL Programming, for his influence in making use of macro and preprocessors for solving programming problems. Also thanks to David Greer for his article IMAGE/COBOL: Practical Guidelines and Peter Somers for his article

Using COBOL, VIEW and IMAGE: A Practical Structured Interface for the Programmer in the 1982 San Antonio HP 3000 IUG Proceedings. Last but not least thanks to Tony Abruzzio of Union Camp Corporation for his contribution to the Montreal Contributed Library tape, COB-CAN, a COBOL callable cancel routine which provided most of the ideas utilized within Appendix B.

APPENDIX A

Following are standard declarations used in all programs which access a database. They are used to guarantee consistent data names and availability of all necessary fields.

```
01  DB-PARAMETERS.
    05  DB-ALL-LIST            PIC X(2) VALUE "@ ".
    05  DB-SAME-LIST           PIC X(2) VALUE "* ".
    05  DB-NULL-LIST           PIC X(2) VALUE "  ".
    05  DB-DUMMY               PIC S9(4) COMP VALUE ZERO.
    05  DB-PASSWORD            PIC X(8) VALUE "APASS;  ".
    05  DB-MODE-1              PIC S9(4) COMP VALUE 1.
    05  DB-MODE-2              PIC S9(4) COMP VALUE 2.
    05  DB-MODE-3              PIC S9(4) COMP VALUE 3.
    05  DB-MODE-4              PIC S9(4) COMP VALUE 4.
    05  DB-MODE-5              PIC S9(4) COMP VALUE 5.
    05  DB-MODE-6              PIC S9(4) COMP VALUE 6.
    05  DB-MODE-7              PIC S9(4) COMP VALUE 7.
    05  DB-MODE-8              PIC S9(4) COMP VALUE 8.
    05  DB-MODE-OTHER          PIC S9(4) COMP VALUE ZERO.
    05  DB-TEXT-LENGTH         PIC S9(4) COMP VALUE 72.
    05  DB-TEXT                PIC X(72).
    05  DB-STATUS.
        10  DB-CONDITION       PIC S9(4) COMP.
            88  DB-OKAY        VALUE 0.
            88  DB-LOGGING-DISABLED
                               VALUE 71.
            88  DB-BOF         VALUE 10.
            88  DB-EOF         VALUE 11.
            88  DB-BOC         VALUE 14.
            88  DB-EOC         VALUE 15.
            88  DB-MISSING     VALUE 17.
            88  DB-CHAIN-HEAD
                               VALUE 44.
            88  DB-GET-OKAY    VALUE 0, 10, 11, 14, 15, 17.
        10  DB-STAT2           PIC S9(4) COMP.
        10  DB-STAT3-4         PIC S9(9) COMP.
        10  DB-CHAIN-LENGTH    PIC S9(9) COMP.
            88  DB-EMPTY-CHAIN
                               VALUE ZERO.
        10  DB-STAT7-8         PIC S9(9) COMP.
        10  DB-STAT9-10        PIC S9(9) COMP.
```

In addition to the above defined standard declarations all programs define the following data values for database access on an as needed basis:

(Continued on next page.)

```
01  DB-BASENAME                PIC X(10) VALUE "  BASENAME".
01  DS-D-BASEDETAIL            PIC X(16) VALUE "D-BASEDETAIL    ".
01  DS-M-BASEMASTER            PIC X(16) VALUE "M-BASEMASTER    ".
01  DS-A-BASEAUTOMATIC         PIC X(16) VALUE "A-BASEAUTOMATIC ".
01  DR-D-BASEDETAIL.
COPY BASEDETA.
    (Data declarations for @; reference to dataset, normally
```

```
    retrieved from a COBOL COPYLIB file named COPYLIB.COBOL.)
01  DR-M-BASEMASTER.
COPY BASEMAST.
01  DA-D-BASEDETAIL.
COPY BASEDE1A.
    (Data declarations for an item list reference to dataset,
    normally retrieved from a COBOL COPYLIB file named
    COPYLIB.COBOL.)
01  DL-D-BASEDETAIL.
COPY BASEDE1L.
    (Item list declarations for an item list reference to
    dataset, normally retrieved from a COBOL COPYLIB file named
    COPYLIB.COBOL.)
01  DE-LOCKSPECIFICATION.
COPY LOCKSP1.
    (Lock specifications for locking on an item value, etc.,
    normally retrieved from a COBOL COPYLIB file named
    COPYLIB.COBOL.)
01  DI-KEYVALUE              PIC X(16) VALUE "KEYVALUE        ".
    (This is a key name for DBFIND or DBGET access.)
01  DV-KEYVALUE              PIC 9(6) DISPLAY.
    (This is the search value for use with searches on data
    item referenced by DI-KEYVALUE.)
```

APPENDIX B

Additional macros have been defined to enhance the error reporting of the initial version of the IMAGE database access macros. They add the capability of logging all detected errors to the system console as well as to the user's terminal. This has proven especially useful for debugging of problems for remote users. The database parameters COPYLIB was modified to include the following at the end of its definition:

```
05  ERROR-REPORT-A.
    10  ER-DATE               PIC X(8) VALUE SPACES.
    10  FILLER                PIC X VALUE SPACE.
    10  ER-TIME               PIC X(6) VALUE SPACES.
    10  FILLER                PIC X VALUE SPACE.
    10  ER-DEVICE             PIC ZZZ9 VALUE 0.
    10  FILLER                PIC X VALUE SPACE.
    10  ER-USER               PIC X(8) VALUE SPACES.
    10  FILLER                PIC X VALUE ".".
    10  ER-ACCOUNT            PIC X(8) VALUE SPACES.
    10  FILLER                PIC X VALUE SPACE.
    10  ER-STATUS             PIC ZZZ9- VALUE 0.
    10  FILLER                PIC X VALUE SPACE.
    10  ER-SUBSYSTEM          PIC X(8) VALUE "IMAGE   ".
05  ERROR-REPORT-B.
    10  ER-PROGRAM            PIC X(35) VALUE SPACES.
    10  FILLER                PIC XXX VALUE " @ ".
    10  ER-PARAGRAPH          PIC X(30) VALUE SPACES.
```

The following macro is invoked at the initiation of each program to define the logon device, user name, and program:

```
$DEFINE %DBINIT=
* %DBINIT(LISTAPPT.PROG.SCHOOL#)
* Fills in the current date, time of day, logon device, user
* name, and program name for use in generation of error messages.
    CALL INTRINSIC "WHO" USING //,
        //,
        //,
        ER-USER,
        //,
        ER-ACCOUNT,
        //,
        DB-CONDITION
    MOVE DB-CONDITION TO ER-DEVICE
```

```
MOVE CURRENT-DATE TO ER-DATE
MOVE TIME-OF-DAY TO ER-TIME
MOVE "!!" TO ER-PROGRAM.#
```

Each of the macro definitions for database access is modified to place the following code prior to each call to DBEXPLAIN:

```
MOVE DB-CONDITION TO ER-STATUS
DISPLAY ERROR-REPORT-A UPON CONSOLE
DISPLAY ERROR-REPORT-B UPON CONSOLE
```

Optionally the programmer may include code at the beginning of each paragraph which will place the name of the paragraph in the variable ER-PARAGRAPH. The recommended method for this is to use the following macro definition:

```
$DEFINE %AT=
* %AT(CONTINUE-PROCESSING#)
* Places the current paragraph name into a variable for reporting
* of error messages and easing the debugging process.
    MOVE "!!" TO ER-PARAGRAPH#
```

Use of this macro as standard practice also provides an excellent method of providing program trace and profiling capabilities by replacement of the macro definition. A simple example to show the execution path of a program would use a macro definition as follows:

```
$DEFINE %AT=
* %AT(CONTINUE-PROCESSING#)
* Places the current paragraph name into a variable for reporting
* of error messages and easing the debugging process and also
* lists the current active paragraph.  (Note:  full value of this
* would require resetting of the paragraph name following return
* from each PERFORMed paragraph.)
    MOVE "!!" TO ER-PARAGRAPH
    DISPLAY ER-PARAGRAPH UPON CONSOLE#
```

When it is desirable to reduce the code requirements a null macro which generates no code could replace the normal definition. Any number of functions become practical when this scenario becomes accepted practice.

Jerrel Baxter graduated cum laude from Birmingham-Southern College, a United Methodist Church supported liberal arts college located in Birmingham, Alabama, in 1975 with a Bachelor of Science degree in Mathematics. Following graduation he was employed by BSC as a computer programmer and instructor in computer science.

In 1978 he became manager of data processing for Argo and Company, a General Electric industrial electrical parts distributor located in Birmingham, Alabama, and Southern Carbon Brush Company, a manufacturer of carbon brushes for electric motors, generators, etc.

Following stabilization of the operation at Argo and Company, he left in 1981 to become manager of data processing for American Intermedical Resources, providers of respiratory therapy department operations for hospitals throughout the United States, computerized electrocardiogram analysis, computerized pulmonary function and arterial blood gas analysis, and equipment rental for home-based health care.

In December of 1982, he became lead systems analyst for the School of Optometry of the University of Alabama in Birmingham, a new HP 3000 Series 44 installation. Currently he is heavily involved in computerization of the clinic operations of the school which see approximately 30,000 patients each year for vision tests by optometry students and necessary spectacle and/or contact lens fitting.

He has provided consulting and programming services to the North Alabama Conference of the United Methodist Church and to Innovest Systems, Inc. (provides investment advisory services to institutional investors and recently to brokers of E. F. Hutton).

As an adjunct instructor at Birmingham-Southern College he has taught courses in COBOL programming, FORTRAN programming, BASIC programming, APL programming, and Data Structures.
