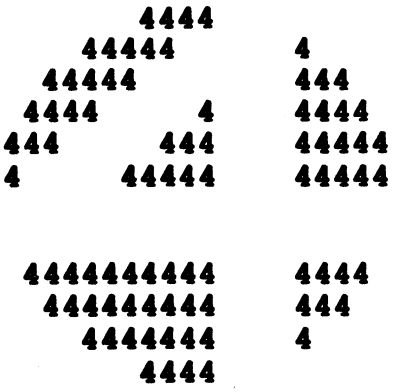
IRIS
CUSTOMER
SUPPORT
BULLETIN

September 1, 1983

POINT 4 Data Corporation



TECHNICAL
MEMORANDUM

TO: IRIS Subscription Service Customers

FROM: IRIS Customer Support

DATE: September 1, 1983

SUBJ: CUSTOMER SUPPORT BULLETIN

I. IRIS SYSTEM UPDATES

A. CUSTOMER SUPPORT NEWS

1. IRIS R8.2 Enhancements

2. R8.2 Enhancements to IRIS Business BASIC

B. SOFTWARE CHANGE ORDERS - R7.5

1. CONFIG75-00
   Problem: Identify SCOs released in the Customer Support Bulletin of September 1, 1983.

2. LCMAC75-03 (MARK 5 & 8 ONLY)
   Problem: Trap 5 may occur if LCMACTIVATE finds an error and aborts.

C. SOFTWARE CHANGE ORDERS - R8.2

1. CONFIG82-00
   Problem: Identify SCOs released in the Customer Support Bulletin of September 1, 1983.

2. BASIC82-03
   Problem: When listing a BASIC program, an extra space is sometimes inserted before numeric literals.

3. C71082-02 (MARK 5 & 8 ONLY)
   Problem: LOTUS 700 LARK driver entry points are to 710 driver not 700 driver.

4. CALLTBL82-02
   Problem: CALLTBL does not support subroutine #79 (i.e., $BAKUP).

5. DSP82-01
   Problem: Checksum capability, i.e., X<adr1>,<adr2>, gives trap 34.

6. DSUB182-05
   Problem: The "read RDA from extender block" function of READMAINTENANCE does not work and may cause a trap 17.

7. DSUB182-06
   Problem: NRCD is not updated correctly when doing a random write to text file resulting in premature record-not-written errors.

8. DSUB382-01
   Problem: Tape subsystem causes system to crash when inputting an extended file.

9. LIBR82-02
   Problem: Trap 34 occurs when doing a sorted LIBR of a large logical unit with an INDEX that is nearly full.

10. REXM382-07 (MARK 3 ONLY)
    Problem: ESCAPE does not work properly.

11. REXM582-06 (MARK 5 & 8 ONLY)
    Problem: ESCAPE does not work properly.

12. RUN82-05
    Problem: SPC (10) does not return the correct line number for errors 99 (ESCAPE) and 199 (CTRL-C).

13. S710CMD82-01 (MARK 5 & 8 ONLY)
    Problem: Infinite loop on LOTUS 700 & 710 Systems.

14. TERMSYS82-01
    Problem: When mnemonic not found in output translation table, character is sent out instead of a null.

D. SOFTWARE CHANGE ORDERS - R8.2 BASIC PROGRAMS

1. Patch to GUIDE.LPT (MARK 5 & 8 ONLY)

2. Patch to GUIDE.LPT (MARK 3 ONLY)

3. Changes to SETUP Programs and SU.ENTRIES File

4. Patch to SU1

5. Patch to SU112A

6. Patch to SU34

7. Patching BASIC Programs

II. DOCUMENTATION UPDATES

A. IRIS R7.5 DOCUMENTATION

1. R7.5 Release Notes Update Package
2. R7.5 Peripherals Handbook (MARK 5/8) Update Package
3. LCM Software Manual (R7.5) Update Package

B. IRIS R8 DOCUMENTATION

1. $FOREIGN Tech Memo Update Package
2. IRIS R8 Peripherals Handbook Update Package

III. APPLICATIONS UPDATES

A. APPLICATIONS SUPPORT BULLETIN

B. TYPIST INSTALLATION MANUAL UPDATE PACKAGE

POINT 4 DATA CORPORATION

```
   4444  4
  4444     444
 444    4  4444
 4   444  4444
```

CUSTOMER SUPPORT NEWS

```
 4444444  4444
  444444  444
    4444  4
```

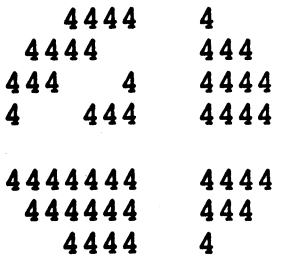
September 1, 1983

Tech memos describing the enhancements from IRIS R8.1 to R8.2 and the improvements to Business BASIC follow this month's News. One feature of R8.2 is that the Data File Table is located in upper memory. This modification allows more ports to be connected to the system than previously possible.

The LOTUS 710/LARK driver had a problem which sometimes resulted in an infinite loop in the event of an ECC error. This problem is corrected in R8.2.

Please note that DISCUTILITY's SAVE/RESTORE to diskette is not intended for use with CMD or LARK. Since those discs can be backed up to removable packs, we feel that a disc image copy on diskette is of little value.

POINT 4 DATA CORPORATION



M E M O

TO: All IRIS Users

FROM: IRIS Software Development

DATE: August 8, 1983

SUBJ: IRIS R8.2 Enhancements

IRIS R8.2 is the latest revision of the IRIS Operating System. It contains a number of new features which add to the versatility of IRIS and give it even better overall system performance.

New features under IRIS R8.2 include:

- A new and improved scheduler
- $CALLTBL - A new system driver aLLowing greater flexibility for implementing subroutines callable from BASIC
- Separate files for IRIS system and OEM-developed discsubs
- BASIC enhancements
- SETUP - A new interactive system configurator
- BAKUP - A new on-line, disc-to-disc copy utility

This Tech Memo gives a brief description of each of these new features.

I. THE SYSTEM SCHEDULER

A new scheduler has been written for IRIS R8.2 which effectively improves overall system performance.

The new scheduler ($SYS.SCHED) is implemented as a driver for convenient field upgrades. $SYS.SCHED should always be enabled before a full IPL; otherwise the system will halt. On a POINT 4 MARK 5/8 such a halt occurs at location 1442 (octal) and on a MARK 3, at location 1343 (octal).

Section 5.10 of the IRIS Installation and Configuration Manual, "Timesharing Parameters", has been rewritten to reflect the new priority system used by the IRIS R8.2 scheduler. Section 5.10.3 gives recommendations for tuning the scheduler to improve system performance based on individual system requirements.

II. $CALLTBL

$CALLTBL is a new system driver which has been added to the IRIS Operating System. Prior to IRIS R8.2, subroutine numbers used in the IRIS Business BASIC CALL statements were translated to actual DISCSUB numbers via a CALL Table residing in the RUN processor.. This restricted the size of the table and, as a result, the range of subroutine and DISCSUB numbers.

Under IRIS R8.2, the structure of the CALL Table has been changed and it has been moved to the new system driver. The restructuring of the table and its removal from the RUN processor has lifted the range restrictions and added a call-by-name feature.

The $CALLTBL system driver has the following overall structure:

HEADER - Starts at address 32200 and contains a pointer to the ATRIB table (see below)

TRANSLATION ROUTINES - Used by RUN to translate subroutine names and/or numbers to DISCSUB numbers

TRANSLATION TABLE

TRANSLATION TABLE EXPANSION AREA

ATRIB TABLE - Contains a pointer to the translation table and marks the end of the memory-resident section of the driver

The new CALL Table format consists of a list of 5-word entries. One 5-word entry is used for each DISCSUB. Each entry is terminated by 177777 (octal). Each entry has the following format:

Word	Description
0	Subroutine name (first three characters)
1	Second word of name (next three characters)
2	Third word of name (last three characters)
3	Subroutine number
4	DISCSUB number

A subroutine name may consist of 0 - 9 characters in radix 50 notation (i.e., a character set of at most 40 characters). In the radix 50 character set, the decimal values are as follows:

Decimal Value	Characters
1-10	0-9
11-36	A-Z

III. SEPARATE DISCSUBS FILES

A discsubs file (DISCSUBS.USER) is now available for listing OEM-supplied discsubs. POINT 4-supplied discsubs continue to be listed in the DISCSUBS file using numbers 1-177. By making a separate file available exclusively for the use of OEMs, previous restrictions on user-supplied discsubs have been removed.

The DISCSUBS.USER file must be a contiguous file and the rules for creating IRIS system discsubs apply to discsubs created by a user.

The discsub numbers in the DISCSUBS.USER file are totally independent of the system discsub numbers allocated by POINT 4. However, any user discsub must have the U-bit (U=2000) set in the discsub's ID number at the beginning of each discsub.

NOTE

> The information in Section 5.3 of the IRIS Installation and Configuration Manual covering the setting of the U-bit is misleading. It implies changing the file's header. No such change is required.

The procedure for entering user-supplied discsubs into the CALL Table is supplied in Section 5.3.1 of the IRIS Installation and Configuration Manual (Rev 07).

IV. BASIC ENHANCEMENTS

A file access and maintenance methodology has been incorporated into the IRIS R8.2 version of Business BASIC. This methodology allows convenient access to any system or data file on disc.

The procedure is called OPEN FILE MAINTENANCE. With the OPEN FILE MAINTENANCE capability, CALL 95 is no longer needed. The procedure is documented in the IRIS BASIC Enhancements Tech Memo.

V. SETUP

SETUP is an interactive utility program that is used to configure the following system and driver tables:

- System Information Tables in the CONFIG file
- Port Definition Table (PDT)
- Memory-Resident DISCSUBS Table
- Disc Driver Table

The SETUP utility uses a control file, which is set up by the user, and two parameter files provided by POINT 4. The control file is used to enter parameters for the desired configuration. It is essentially a scratch pad or a copy of the final configuration parameters. The system files are not configured until the update function of the SETUP program is invoked.

The parameter files provided by POINT 4 are called SU.ENTRIES and SU.DSUBS. SU.ENTRIES contains parameter information based on the disc spec parameters published in the IRIS R8 Peripherals Handbook and arranged by entry number. SU.DSUBS contains two listings. The first is a list of IRIS discsubs by name and discsub number. The list also indicates whether a discsub is in the preset memory-resident DISCSUB table. The second list is arranged by discsub type and discsub number (i.e., discsub numbers are listed under system, polyfile, tape, etc.).

SETUP uses the control file parameters and the two other parameter files to display default values for various parameters and to calculate others.

SETUP provides an easy, user-friendly method for configuring portions of the system without the necessity of using DSP or calculating the values for certain parameters such as the PHYU.

Procedures for using SETUP are fully documented in Section 6 of the IRIS Installation and Configuration Manual (Rev 07).

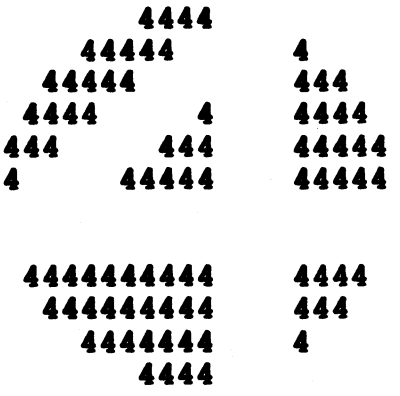
VI. BAKUP

BAKUP is a new on-line, disc-to-disc copy program intended for use as a backup utility. It must be run from a BASIC program which may be customized according to the drives used at any particular installation. Parameters are set in the BASIC program and the utility may then be run with minimal operator intervention.

BAKUP performs previously specified disc-to-disc copy operations without shutting down the system and, therefore, avoids time-consuming IPLs.

The procedure for initiating the copy process from the POINT 4-supplied BASIC program (BAKUP) is fully documented in the IRIS Operations Manual (Rev 06).

POINT 4 Data Corporation



TECHNICAL MEMORANDUM

TO: All BASIC Programmers

FROM: Systems Software Development

DATE: Revised August 29, 1983

SUBJ: R8.2 ENHANCEMENTS TO IRIS BUSINESS BASIC

This technical memorandum describes the enhancements available in the R8.2 release of Enhanced Business BASIC. This memo assumes the user is already familiar with existing BASIC.

CONTENTS

The following standard writing conventions are used in this technical memorandum:

variable	Underlined lowercase items represent a variable such as a filename, mode, value and so on.
<X>	Terminal keys are enclosed in angle brackets and are shown in upper case.
{optional}	Items enclosed in braces are optional.

I. PLACING MULTIPLE BASIC STATEMENTS ON A SINGLE LINE

More than one BASIC statement may now be placed on a single line, with each of the statements on the line separated by a backslash. The total length of any line remains limited to the size of the BASIC I/O and edit buffers.

The only exception to placing multiple statements on a single line is the DATA statement, which may not be included with any other statements on a single line. This exception and other restrictions are discussed in Section 1.5.

The end of a statement is defined by a backslash or by a <RETURN> at the end of a line.

This enhancement is fully compatible with the previous BASIC syntax. Any effect of multiple-statement lines on a given statement is described in that statement's subsection. Existing programs require no modification (unless they rely upon the generation of a syntax error).

This enhancement is compatible with the previous BASIC object files, with the exception that when an old BASIC object file is activated, it is converted to the new format and the GOSUB and FOR-NEXT stacks are cleared. This affects only those rare programs which have been saved with non-empty stacks and which are re-started using the command

line# RUN

This will probably remain unnoticed.

WARNING

Files saved under enhanced BASIC will not run under previous BASIC.

1.1 IF STATEMENTS ON MULTIPLE-STATEMENT LINES

The IF statement on a multiple-statement line controls the execution of all statements following it. If the condition stipulated in the IF statement is satisfied, all applicable following statements on that line are executed. If the condition is not satisfied, the remaining statements on that line are not executed and control passes to the next line.

For example, consider the following program lines

```
10 IF A=1 PRINT "A=1" \ GOTO 30
20 PRINT "A is not equal to 1"
30 END
```

If the value of A is equal to one, "A=1" is printed and execution continues at line 30. If the value of A is not equal to one, the two statements following the IF condition are not executed, and control passes to line 20.

1.2 GOSUB AND RETURN STATEMENTS ON MULTIPLE-STATEMENT LINES

The GOSUB and RETURN statements may be placed on multiple-statement lines. GOSUB functions as before, but RETURN now returns to the next statement, not line, following its associated GOSUB. RETURN+ now returns from the subroutine and skips the designated number of statements, not lines, relative to its associated GOSUB.

For example, consider the following program lines

```
100 INPUT "CONTINUE?  (TYPE YES OR NO)" R$ \ PRINT
110 GOSUB 1000 \ GOTO 500 \ GOTO 600 !CHECK IF YES OR NO
120 PRINT "PLEASE TYPE 'YES' OR 'NO'" \ GOTO 100

500 REM Perform 'YES' alternative
510 REM
520 REM

600 REM Perform 'NO' alternative
610 REM
620 REM

1000 REM Routine to check for 'YES' and 'NO'
1010 IF R$="YES" RETURN !return to next statement if YES
1020 IF R$="NO" RETURN 1 !return and skip next statement if NO
1030 RETURN 2 !return and skip 2 statements if neither
```

This program uses a subroutine to check whether the input string R$ is equal to "YES", "NO" or neither. If R$ is equal to "YES", the subroutine returns to the next statement following the GOSUB, which is "GOTO 500".

If R$ is equal to "NO", the subroutine returns to the program, skips one statement, and continues execution with the following statement, which is "GOTO 600".

If R$ is not equal to "YES" or "NO", the subroutine returns to the program, skips two statements, and continues execution with the following statement, which is "PRINT "PLEASE TYPE 'YES' OR 'NO'".

In existing BASIC, statements and lines are equivalent because there is only one statement per line. Because of this, existing BASIC programs will not be affected by this distinction.

1.3 FOR-NEXT STATEMENTS ON MULTIPLE-STATEMENT LINES

FOR-NEXT loops are now statement-oriented, rather than line-oriented. This allows FOR-NEXT loops to be placed within a single line or across lines.

For example, consider the following program lines

```
10 IF F=9 FOR I=1 TO 10 \ LET A[I]=0 \ NEXT I
20 LET B=1 \ FOR I=3 TO 8 \ LET A[I]=B
30 LET C[I]=B \ NEXT I \ PRINT "DONE"
```

If the variable F is equal to nine, the first FOR-NEXT loop sets elements one through ten of array A to zero. The variable B is set to one before the second FOR-NEXT loop is entered.

The second FOR-NEXT loop sets elements three through eight of arrays A and C to the value of B. When the second FOR-NEXT loop is completed, the PRINT statement is executed.

1.4 IF ERR STATEMENTS ON MULTIPLE-STATEMENT LINES

Every IF ERR statement must be the last statement on the line on which it occurs. If the IF ERR statement has no error handling statements following it, error control is reset. Any code following IF ERR must consist of a single statement which will be executed when an error occurs.

For example, consider the following program lines. This program's intent is that lines 100 through 500 not be interrupted by the entry of <ESCAPE> or <CTRL-C>.

```
100 IF ERR 0 GOSUB 1000
110 REM
120 REM Perform file I/O which must not be
130 REM interrupted by Escape or Control-C

500 IF ERR 0   !reset error control

1000 IF SPC(8)=99 RETURN -1  !ignore escape, continue pgm
1010 PRINT "ERROR", SPC(8), "ON LINE", SPC(10)
1020 STOP
```

To prevent user interrupts, the program uses the IF ERR statement at line 100 to invoke a GOSUB when an error occurs. The called subroutine checks the error code to determine if either <ESCAPE> or <CTRL-C> was entered. If so, "RETURN -1" executes the statement which was interrupted before it was completed.

Previously, "RETURN -1" would have re-executed the line on which the error occurred, but now "RETURN -1" re-executes the statement in which the error occurred.

If an error other than <ESCAPE> or <CTRL-C> occurred, an error message is printed and the program stops.

This enhancement has no effect on existing programs.

1.5 EXCEPTION AND RESTRICTIONS

The only statement which may not be included in a multiple-statement line is the DATA statement.

Functions are restricted to a single statement. Any statements following a DEF FN statement are not included as part of the function.

Every IF ERR statement must be the last statement on the line on which the IF ERR statement occurs.

Comments may be placed only at the end of a line. Backslash characters within a REM statement or following an exclamation point are treated as part of the comment.

II. CALLING SUBROUTINES BY NAME

Subroutines may now be CALLed by a number or by a mnemonic, instead of by a number only. The mnemonics which may be used are listed below. Programs may be input using subroutine numbers or mnemonics. This enhancement increases readability and makes subroutine identification easier to remember.

A subroutine which has no mnemonic, such as a user-defined discsub, is still referenced by number. If an illegal subroutine mnemonic is input, Error 87 is generated.

The following columns list the subroutine number, new subroutine mnemonic, and a brief description of each subroutine's function. $STRING and $LOGIC have been added to enhanced BASIC. $STRING is described in Section IV and $LOGIC in Section V.

Number	Mnemonic	Function
CALL 79	$BAKUP	BAKUP Assembly language support
CALL 82	$STRING	Perform string functions
CALL 88	$LOGIC	Use logical operators
CALL 96	$FINDF	Find a file
CALL 97	$RDFHD	Read file header
CALL 98	$TRXCO	Transmit system command
CALL 99	$TIME	Convert date and time

The following program lines show the use of subroutine mnemonics:

```
10 REM Print date and time
20 DIM T$(40) \ T$=""
30 CALL $TIME, T$   !get date and time
40 PRINT T$
```

In this program, the CALL statement uses the subroutine mnemonic "TIME" instead of the number "99".

This enhancement is compatible with existing BASIC programs.

III. INPUT LENGTH LIMIT

The length of user input may now be limited by including a LEN x clause in an input statement, where x is the number of characters which may be input. Both string and variable input may be limited.

The LEN clause affects only the next input of the INPUT statement in which the clause occurs. The LEN clause may follow or precede the INPUT prompt and cursor positioning specifications.

For example, consider the following program lines

```
10 INPUT @0,10;LEN 1; "Yes (Y) or No (N)?" A$
20 IF A$="Y" GOTO 100
30 IF A$="N" GOTO 200
40 PRINT "PLEASE ANSWER YES OR NO" \ GOTO 10
```

This example accepts only the letters "Y" or "N" as input. The user may type "YES" or "NO", but the system acts as though return were pressed as soon as the specified input length was reached. The excess letters ("ES" or "O") are queued up as type-ahead.

Setting the input length to zero (LEN 0) has the same effect as not including a LEN clause. The system ignores any input length which is set greater than the I/O Buffer size.

This enhancement has no effect on existing BASIC programs.

IV. $STRING SUBROUTINE

BASIC has been enhanced to include a new subroutine which provides the following string functions: convert string to upper case, convert string to lower case, convert one or two characters to a related numeric value, convert an 8-bit or 16-bit value to a character or characters, and read the I/O Buffer. This subroutine may be called by its discsub number, 82, or by its mnemonic, $STRING. These functions are discussed in the following subsections.

4.1 CONVERT STRING TO UPPER OR LOWER CASE

The syntax for upper and lower case conversion is:

CALL $STRING,mode,string variable

where

mode - variable equal to the number of the appropriate case conversion, where

1 - specifies conversion to upper case
2 - specifies conversion to lower case

This subroutine converts alphabetical letters only to the appropriate case. Any characters already in the case being converted to or any characters other than letters of the alphabet remain unchanged.

4.2 CONVERT CHARACTERS TO A NUMERIC VALUE

The syntax for character conversion is:

CALL $STRING,mode,string var,value

where

mode - variable equal to the number of the appropriate character conversion, where

- 3 - converts a single character to an ASCII value
- 6 - converts two characters to a 16-bit number

string var - for mode=3, converts the first character of this string to its ASCII value

for mode=6, converts the first two characters of this string to a value determined by the following equation

(A * 256) + B = value

where

- A - ASCII value of the first character
- B - ASCII value of the second character

value - calculated result

value - the ASCII or calculated value is returned here

If the mode is set to 3, the subroutine returns an ASCII value in the range zero to 255 for the first character of the string, depending on the character and parity setting.

If the mode is set to 6, the subroutine returns a 16-bit value in the range zero to 65535 which is calculated using the equation above and depends on the characters and parity setting.

Error 38 is generated if the parameters do not meet the specifications described above. Error 15 is generated when the mode is set to 6 and the numeric variable is not of sufficient dimension to hold the returned value.

4.3 CONVERT A NUMERIC VALUE TO CHARACTERS

The syntax for numeric value conversion is:

```
CALL $STRING,mode,value,string var
```

where

mode - variable equal to the number of the appropriate numeric conversion, where

- 4 - converts an ASCII value to a single character
- 7 - converts a given value to two characters

value - converts the value entered here to the appropriate character or pair of characters

string var - the character or pair of characters are returned here

Only values in the range zero to 65535 may be converted to characters using this subroutine.

The subroutine checks the type and magnitude of the parameter "value". For mode 4, if the value is greater than 255, the value Modulus 256 is used in the conversion.

For mode 4, the generated character overlays the first character of the string, if any, and the second character is overlayed with a null. For mode 7, the pair of generated characters overlays the first two characters of the string.

Error 38 is generated if the parameters do not meet the specifications described above.

4.4 READ THE COMMAND LINE

This function reads the Command Line (the contents of the I/O Buffer) from within a BASIC program. This function allows programs to operate more like processors or system commands. The syntax for reading the Command Line is:

```
CALL $STRING,mode,string var
```

where

mode - variable set equal to 5 for reading the Command Line

string var - the string is returned here

The returned string consists of the contents of the I/O Buffer up to the first <RETURN> found in the buffer. Because of this, the CALL must precede any input or output (READ, WRITE, INPUT, or PRINT) within the program; otherwise, the contents of the I/O Buffer are lost.

If the Command Line is empty, the string variable remains unchanged.

When CHAINing to a program which uses CALL $STRING, the I/O Buffer should be cleared immediately before the CHAIN statement. For example, consider the programs ALPHA and BETA:

```
100 REM This is the program ALPHA
110 SIGNAL 3,0
120 PRINT "   ";
130 SIGNAL 3,0
140 CHAIN "BETA"

500 REM This is the program BETA
510 DIM A$ [50]
520 A=5
530 CALL $STRING, A, A$
```

Lines 110, 120 and 130 in ALPHA are required to ensure that the I/O Buffer contains exactly three spaces and a terminator. This prevents BETA from accessing the last output as type-ahead input.

The use of $STRING in mode 5 allows the user to enter

```
#DISPLAY REPORT
```

to use the program "DISPLAY" on a file named "REPORT".

V. $LOGIC SUBROUTINE

BASIC has been enhanced to include a new subroutine which provides the logical operators AND, OR, XOR and NOT. This subroutine may be called by its discsub number, 88, or by its mnemonic, $LOGIC. This subroutine is called using the following format:

```
CALL $LOGIC,operator,p1,p2,result
```

where

operator - variable set to the number of the appropriate logical operator, where
- 1 - specifies AND
- 2 - specifies OR
- 3 - specifies XOR
- 4 - specifies NOT

p1 - first operand

p2 - second operand (for NOT, p2 is used as a dummy)

result - a variable to receive the returned result

Although NOT requires only one operand, the second operand must be specified and is used as a dummy to satisfy syntax requirements.

$LOGIC operates on either numeric integers in the range 0 to 65535 or on character strings. The type of operand specified in the parameter list determines whether numeric values or character strings are being used. All parameters must be of the same type (either string or numeric); otherwise, an error is generated.

For character strings, the operation is performed on a byte-by-byte basis until reaching the dimensioned length of the shortest string. Numeric values are converted to unsigned 16-bit integers before the logical operation is performed.

This subroutine does not recognize null as a string terminator. Because of this, the subroutine may be used to copy strings which include nulls (by ANDing the string with itself), to fill a string with nulls (by XORing it with itself), and so on.

The following program lines demonstrate the use of $LOGIC

```
10 A=1  !used as a parameter to specify the logical operator
20 P1=3 \ P2=22  !the two operands
30 CALL $LOGIC,A,P1,P2,R  !set R to the logical AND of P1 and P2
40 PRINT R
```

In the CALL statement, the value of A is set to one to specify the logical operator AND, the values of P1 and P2 are the operands, and the result is returned as the value of R.

The following lines demonstrate the AND operation.

```
      Base 2   Base 10

       00011 = 3

   AND 10110 = 22
       -----------

       00010 = 2
```

The program prints "2" as the result of the operation.

Error 38 is generated if the parameters do not meet the specifications described above. Error 15 is generated if the variable result is not of sufficient dimension to hold the returned result.

VI. ERROR HANDLING

Error handling enhancements to BASIC involve error trapping during program initialization, different error codes when <ESC> and <CTRL-C> are pressed, generation of an error when a record is locked and enabling the new error-handling mode. These enhancements are discussed in the following subsections.

6.1 FIRST TRAPPABLE ERROR

When chaining from one BASIC program to another BASIC program, existing BASIC allows the new program to be interrupted by <ESC> or <CTRL-C> before it can establish error control through the IF ERR statement. This can occur after initiation of the RUN processor but before execution of the first statement. This problem is most common when chaining is used frequently.

This problem may be avoided in enhanced BASIC. Pressing <ESC> or <CTRL-C> during the initialization period now sets a flag which is detected as soon as the first executable statement has been completed. By including IF ERR as the first executable statement of each program, the errors will be trapped. IF ERR must be the first executable statement because RUN does not detect <ESC> or <CTRL-C> until after the first executable statement has been executed. Executable statements include all statements except REM and DATA.

6.2 RECORD LOCKING

This enhancement provides a method of generating an error when a program is paused because a record or device is locked longer than the specified period of time. This enhancement may be used in READ, WRITE and PRINT statements using the following syntax:

READ #channel,{record,{item,{delay}}};data

WRITE #channel,{record,{item,{delay}}};data

PRINT #channel,{record,{item,{delay}}};expressions/strings

Each parameter is described below.

channel - channel number expression

record - record number expression

item - item number expression

delay - delay limit in tenths of a second

data - appropriate data (optional for WRITE #channel;;)

expressions/strings - mathematical expressions or literal strings

If the program is paused due to a locked record longer than the time specified in "delay", error 123 is generated. The delay is specified in tenths of a second.

Setting the delay to -1 (default) allows an unlimited delay period.

Setting the delay to 0 indicates no delay and no I/O retries. Note that the delay limit specifies the maximum amount of time to be spent retrying input or output. For example, a delay limit of 600 (60 seconds) allows 20 retries, given a three-second delay between retries.

Some programs must continue the input or output which caused a record-locked error. The method of recovery depends on the type of file or device involved, the statement used and whether access was random or sequential. For formatted or contiguous files, the aborted I/O may simply be re-executed. For text files, the statement should be re-executed only if random access was used. If the error occurred during sequential access to a text file, the operation might have been only partially completed and the file position might have changed. A record-locked error on any data file may always be followed by random access to the same record or to a different record.

The delay clause may also be used to return an error message when a printer is locked longer than the specified period of time. The length of time which should be specified varies according to the printer's buffer size and printing speed, and the system's baud rate. Using a delay of two minutes or more usually permits recovery from a normal buffer-full condition but still detects a true failure such as the printer being out of ribbon or paper. A program could recover by requesting that the condition be corrected, waiting for the correction to be made and then reprinting appropriate pages. A retry should not be attempted because the record lock may have occurred when the statement was only partially executed.

The use of a delay limit on devices other than printers is possible but not recommended. The method of recovery depends on the device being used.

6.3 ERROR 99 AND 199 GENERATION THROUGH IF ERR 1

In existing BASIC, Error 99 is generated when error trapping is enabled and either <ESC> or <CTRL-C> is pressed. Enhanced BASIC is able to generate different error numbers for each of these commands. If the new error mode and error trapping are enabled, pressing <CTRL-C> generates Error 199 and pressing <ESC> generates Error 99.

To enable generation of Error 199 when <CTRL-C> is pressed, a statement of the following form must be executed

IF ERR 1 {statement}

The IF ERR 1 statement is functionally identical to the existing IF ERR 0 statement, except that the "1" enables generation of Error 199. This may be disabled using an IF ERR 0 statement.

For example, the following statement passes control, including new errors, to a routine at line 9000:

```
IF ERR 1 GOTO 9000
```

VII. NEW SPECIAL FUNCTIONS

Two new special functions have been added to enhanced BASIC. The first provides the current input buffer length and the second returns the system base year.

7.1 PROVIDE THE INPUT BUFFER LENGTH

Existing BASIC does not provide a way to determine the input buffer size. The new special function SPC(17) returns the length of the input buffer which was allowed in the last input statement. If the buffer has been shortened by the new INPUT LEN statement (described in Section III), the shortened length is returned.

This function is particularly useful when cursor tracking is in effect and extremely long inputs may be terminated without the operator's knowledge. It may also be used to dimension a string to the maximum length allowed.

7.2 RETURN THE SYSTEM BASE YEAR

Existing BASIC provides no standard function to return the system base year. The base year has been changed for IRIS R8 and is also changed whenever the 16-bit hour count may overflow (approximately every eight years). Application programs frequently use the base year to calculate dates through use of the SPC(2) function. The new special function SPC(18) returns the system base year. Under IRIS R8, the SPC(18) function currently returns the value 1980.

For example, consider the following program lines:

```
10 LET X=SPC(2)                    !get current date in hours
20 LET X=INT (X/(12*31*24))        !convert hours to years
30 LET Y=SPC(18)+X                 !calculate current year
```

The first line returns the current date in hours and assigns the value to the variable X. The second line converts the hours to years and assigns the value to X. The final line calculates the current year by adding the base year to X.

POINT 4 recommends <u>not</u> storing data values in the SPC(2) form, because the system base year changes every seven years. Most databases store the date in a D2 value, which can hold six digits. These six digits should be used in the form

```
YYMMDD
```

(where YY is the year, MM is the month and DD is the day) in order to provide a value which is directly readable and will sort.

VIII. CALCULATOR MODE

In calculator mode, existing BASIC outputs a Carriage Return/Line Feed after executing any statement except PRINT. BASIC now outputs a Carriage Return/Line Feed after executing every statement including PRINT when in calculator mode. Thus, a PRINT statement now has the same effect in calculator mode as in program execution.

A semicolon may be appended to any PRINT statement to disable the output of Carriage Return/Line Feed.

IX. CORRECTIONS IN ENHANCED BASIC

The corrections described in the following paragraphs are included in enhanced BASIC. Corrections which resulted from enhancements are not discussed.

- The function table and FOR-NEXT stack are now properly adjusted when BASIC program lines are inserted, deleted, or replaced. Because of this, user-defined functions and FOR-NEXT loops now execute correctly after a program has been interrupted, modified and continued using the command

  line# RUN

- User-defined functions will now work in calculator mode.

- The DUMP command no longer produces extra useless text when a selective dump, such as

  line# DUMP filename line#

  is used.

- The RENUMBER command now sets any undefined line number to zero when renumbering a program.

- The usage of the character "q" in a literal string in an IF statement will no longer result in runtime errors.

- In IF statements, string expressions (rather than single strings) may be used on both the left and right sides of the relational operator; for example

  ```
  IF A$,B$=C$,D$ PRINT "EQUAL"
  ```

X. OPEN FILE MAINTENANCE

Certain system functions are restricted from most programs because general access to those functions would endanger the integrity and security of the system. For example, the logical unit index may not be opened for READ and WRITE by all users. If it were, it might be damaged, causing many files to be inaccessible; a user might also take advantage of it to find other users' files.

Open file maintenance provides a way for the system manager to grant access by specific programs to otherwise restricted files. Access is granted by setting the appropriate designated-operator-oriented mode (DOOM) bits. The process of setting a program's DOOM bits is called "GUARDing"[1] a program. Processors, discsubs and programs written in assembly code are allowed to access files regardless of file type or protection. By setting the appropriate DOOM bits of a program, the system manager may grant this ability to individual BASIC programs. Only the manager may grant this access, and it is allowed only for specific programs, so system security is maintained.

In general, IRIS protects files in two ways: by file type and by file protection. When a user attempts to open a file from a BASIC program, IRIS checks the following:

- the file type
- the protection of the file against the privilege level of the account which owns the file
- the privilege level of the account attempting to open the file
- the DOOM bits of the program attempting to open the file

If the file type is legitimate and the protection and privilege levels are in accord, the user is granted access to the file in question. If they are not, access may be granted depending on how the DOOM bits are set. The DOOM bit settings provide two modes of open file maintenance, as described in the following section.

[1]A program named GUARD, which manipulates the DOOM bits, will be supplied shortly. Prior to GUARD's release, the user must use DSP. Refer to the DEFS listing in Appendix B of the IRIS Installation and Configuration Manual for a complete definition of the DOOM bits.

10.1 OPEN FILE MAINTENANCE MODES

There are two open file maintenance modes. The capability of each mode is described below:

Mode	Capability
1	Allows the program to open a file regardless of the file's type (but not regardless of protection).
2	Allows the program to open a file regardless of the file's type and regardless of protection.

10.2 SECURITY OF GUARDED PROGRAMS

If a program has been granted open file capability, the program is execute-only to all users except the system manager. If any user other than the manager attempts to bring the program into his or her active file, BASIC immediately clears the program from the user's partition (as though "NEW" had been entered) and unsets the DOOM bits, thereby taking away the program's open file capability. This prevents users from overwriting programs which have been granted open file capability. The program still resides on disc, so it is safe.

To provide additional security, a special DOOM bit may be set to make a program execute-only to everyone, including the manager. The system checks this bit when the manager attempts to bring any program into his or her active file. This bit should be set to protect any program used by the system manager to set the DOOM bits of other programs. Any program used to set the DOOM bits of other programs should require entry of a password in order to prevent any user who happens to be on the manager account from setting the DOOM bits of programs without the manager's knowledge or consent.

10.3 BASIC PROGRAM INTERFACE

This section describes the BASIC statements used to take advantage of each maintenance mode.

10.3.1 OPENING A FILE

To open a file to which access has been granted via maintenance mode one or two, the OPEN statement is used with the following syntax:

```
OPEN #channel=maintenance mode,filename
```

For example

```
70 OPEN #1=2, "INDEX"
```

opens the INDEX file on channel 1 in mode 2.

10.3.2 READING A FILE

Reading a file which has been opened in maintenance mode one or two requires the use of a READ statement with a special syntax. The syntax varies slightly depending on whether the program will read a word in a data block or header, or access the RDA list.

10.3.2.1 Reading a Word in a Data Block or Header

The syntax of the READ statement for reading a word in a data block or header is:

READ #channel,block no,(format*256+displacement);variable list

where

channel - the number of a channel with a file OPENed in maintenance mode one or two

block number - the relative block number of the file; 0 is the first block, 1 the second, and so on. Set the block number equal to 65533 to reference the file's header. Setting it equal to -1 or -2 obeys the usual use of those values in a READ statement.

format - indicates the data type at the requested disc file location:

0 = unsigned BCD integer
1 = signed BCD integer
2 = 2-word BCD number
3 = 3-word BCD number
4 = 4-word BCD number
5 = reserved
6 = reserved
8 = indeterminate; pass as is
9 = IRIS string
10 = binary number

displacement - the word displacement within the block for the data; the displacement is advanced by the quantity transferred up to word 255 (decimal).

variable list - variables to which the data is transferred

An error occurs in the following two cases: if the format equals 1, 2, 3, 4, 5, 6 or 10 and the variable list is not numeric; and if the format equals 9 and the variable list does not include strings. READ terminates on a zero byte if the format equals 9.

10.3.2.2 Accessing the RDA List

The syntax of the READ statement for accessing the RDA list from extended headers is:

READ #channel,65532,data block number;variable list

where

channel - the number of a channel with a file OPENed in maintenance mode one or two

data block number - a value from 0 to 65535

variable list - receives the RDA of the requested data block

This access is invalid if the file is not extended. A "record not written" error occurs if there is no extender associated with the data block.

10.3.3 WRITING TO A FILE

Writing to a file which has been opened in maintenance mode one or two requires the use of a WRITE statement with the following special syntax:

WRITE #channel,block no,(format*256+displacement);variable list

where

channel - the number of a channel with a file OPENed in maintenance mode one or two

block number - the relative block number of the file; 0 is the first block, 1 the second, and so on. Set the block number equal to 65533 to reference the file's header. Setting it equal to -1 or -2 obeys the usual use of those values in a READ statement. When writing a word to memory, the block number may not be set to 65532.

format - indicates the data type at the requested disc file location:

```
 0 = unsigned BCD integer
 1 = signed BCD integer
 2 = 2-word BCD number
 3 = 3-word BCD number
 4 = 4-word BCD number
 5 = reserved
 6 = reserved
 8 = indeterminate; pass as is
 9 = IRIS string
10 = binary number
```

displacement - the word displacement within the block for the data; the displacement is advanced by the quantity transferred up to word 255 (decimal). When writing a word to memory, the displacement may not be set greater than 126 if the block number equals 65533, because that would be the RDA list.

variable list - variables to which the data is transferred

XI. SUMMARY OF BASIC ENHANCEMENTS

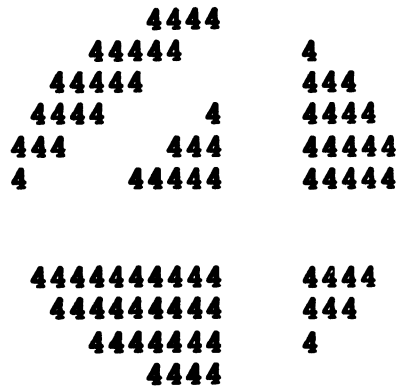
The following table summarizes the enhancements to Business BASIC.

error trapping	entry of <ESC> and <CTRL-C> when chaining between BASIC programs is detected if IF ERR is the program's first executable statement
Errors 99 and 199	when an IF ERR 1 statement has been executed, entry of <CTRL-C> generates Error 199 and <ESC> generates Error 99
limit input length	may be accomplished by including a LEN x clause
$LOGIC subroutine	provides the logical operators AND, OR, XOR and NOT
multiple statements	may now be placed on a single line (except DATA); affects IF, GOSUB, RETURN, FOR-NEXT, and IF ERR statements and calculator mode
open file maintenance	allows the manager to grant special file maintenance capabilities to certain programs
record locked	delay clause may be included in READ, WRITE and PRINT statements to generate Error 123 when a record is locked longer than the time specified
special functions	SPC(17) provides the Input Buffer length; SPC(18) returns the system base year
$STRING subroutine	converts: • string to upper case • string to lower case • character(s) to number(s) • number(s) to character(s) and reads the I/O Buffer
subroutine mnemonics	79 - $BAKUP --BAKUP Assembly language support 82 - $STRING--perform string functions* 88 - $LOGIC--use logical operators* 96 - $FINDF--find a file 97 - $RDFHD--read file header 98 - $TRXCO--transmit system command 99 - $TIME--convert date and time *new subroutines

R7.5 SCOs

September 1, 1983

POINT 4 Data Corporation



T E C H N I C A L
M E M O R A N D U M

TO: IRIS System Managers

FROM: IRIS Customer Support

DATE: Revised September 1, 1983

SUBJ: INSTALLING SCOs

This Customer Support Bulletin contains a number of Software Change Orders (SCOs). It is imperative that every SCO which pertains to the version of IRIS (R7.5 or R8.2) on your system be installed unless the SCO is marked as optional, or special instructions specify a configuration that is different from your system. Future SCOs will assume that all previous SCOs have been installed.

The procedure for installing Software Change Orders is as follows:

1. Back up your system and data.

2. Ensure that you are the only one on the system.

3. Enter DSP and do not exit until all the updates have been entered.

   a. Compare the contents of each location with the old contents listed on the patch.

NOTE

If the old contents do not agree, do not apply the patch! Call IRIS Customer Support.

   b. If old contents agree, enter the patch.

   c. Exit DSP.

4. SHUTDOWN and do an IPL to affect the changes in your system.

Point 4 Data Corporation
830753

+--- P4DC production manager only ---+

| Status _____ |
| Production System |
| Update by_____ Date__/__/__ |
| |
| SCO # _____ Date__/__/__ |
| Master File |
| Update by_____ Date__/__/__ |
| Master File |
| Name _____ |

+-----------------------------------+

Patch #: CONFIG75-00

Product: CONFIG

Detail: CONFIG

Asm Date: 11-6-81 Release #: 7.5

Update Date: 01-Sep-83 By: RDC

Review Date: dd-Mon-yy By: _____

Problem: TO IDENTIFY SCOs RELEASED IN THE CUSTOMER SUPPORT BULLETIN OF SEPTEMBER 1, 1983.

Special Instructions: If the old contents do NOT agree DO NOT enter ANY patches in this bulletin. CALL POINT 4 CUSTOMER SUPPORT.

Location (Octal)	New Contents (Octal a/o Symbolic)	Comments (Describe Solution)	Old Contents
600	77740	;	73450

Point 4 Data Corporation

+--- P4DC production manager only ---+
|
| Status _____
| Production System
| Update by_____ Date__/__/__
|
| SCO # _____ Date__/__/__
| Master File
| Update by_____ Date__/__/__
| Master File
| Name _____
+-----------------------------------+

Patch #: LCMAC75-03

Product: LCMACT

Detail: LCMACTIVATE

Asm Date: 1030 Release #: 7.5

Update Date: 25-Aug-82 By: RDC1

Review Date: dd-Mon-yy By: _____

Problem: Sometimes a TRAP 5s will occur if LCMACTIVATE found an error and aborted.

Special Instructions: DO NOT ENTER PATCH IF OLD CONTENTS DO NOT AGREE.

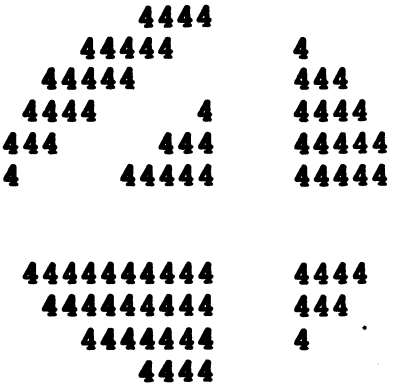
FOR LCMACTIVATE THAT IS NON-DYNAMIC !

Location (Octal)	New Contents (Octal a/o Symbolic)	Comments (Describe Solution)	Old Contents
012153	100000	;RESET SFLU	140000

R8.2 SCOs

September 1, 1983

POINT 4 Data Corporation



T E C H N I C A L
M E M O R A N D U M

TO: IRIS System Managers

FROM: IRIS Customer Support

DATE: Revised September 1, 1983

SUBJ: INSTALLING SCOs

This Customer Support Bulletin contains a number of Software Change Orders (SCOs). It is imperative that every SCO which pertains to the version of IRIS (R7.5 or R8.2) on your system be installed unless the SCO is marked as optional, or special instructions specify a configuration that is different from your system. Future SCOs will assume that all previous SCOs have been installed.

The procedure for installing Software Change Orders is as follows:

1. Back up your system and data.

2. Ensure that you are the only one on the system.

3. Enter DSP and do not exit until all the updates have been entered.

   a. Compare the contents of each location with the old contents listed on the patch.

NOTE

> If the old contents do not agree, do not apply the patch! Call IRIS Customer Support.

   b. If old contents agree, enter the patch.

   c. Exit DSP.

4. SHUTDOWN and do an IPL to affect the changes in your system.

Point 4 Data Corporation
830823

+--- P4DC production manager only ---+

| Status _____ |
| Production System |
| Update by_____ Date__/__/__ |
| SCO # _____ Date__/__/__ |
| Master File |
| Update by_____ Date__/__/__ |
| Master File |
| Name _____ |

+-----------------------------------+

Patch #: CONFIG82-00

Product: CONFIG

Detail: CONFIG

Asm Date: 3162 Release #: 8.2

Update Date: 01-Sep-83 By: RDC

Review Date: dd-Mon-yy By: _____

Problem: TO IDENTIFY SCOs RELEASED IN THE CUSTOMER SUPPORT BULLETIN OF SEPTEMBER 1, 1983.

Special Instructions: If the old contents do NOT agree DO NOT enter ANY patches in this bulletin. CALL POINT 4 CUSTOMER SUPPORT.

Location (Octal)	New Contents (Octal a/o Symbolic)	Comments (Describe Solution)	Old Contents
000600	72100	;	72020

Point 4 Data Corporation
830823

+--- P4DC production manager only ---+

| Status _____ |
| Production System |
| Update by_____ Date__/__/__ |
| |
| SCO # _____ Date__/__/__ |
| Master File |
| Update by_____ Date__/__/__ |
| Master File |
| Name _____ |

+-----------------------------------+

Patch #: BASIC82-03

Product: BASIC

Detail: BASIC

Asm Date: 3158 Release #: R8.2

Update Date: 15-Aug-83 By: TWME

Review Date: dd-Mon-yy By: _____

Problem: When listing a BASIC program, an extra space is sometimes inserted before numeric literals.

Special Instructions:

Location (Octal)	New Contents (Octal a/o Symbolic)	Comments (Describe Solution)	Old Contents
040604	LDA 0,30	;Load output function code	020027

Point 4 Data Corporation
830823

+--- P4DC production manager only ---+

| Status _____ |
| Production System |
| Update by_____ Date__/__/__ |
| SCO # _____ Date__/__/__ |
| Master File |
| Update by_____ Date__/__/__ |
| Master File |
| Name _____ |

+-----------------------------------+

Patch #: C71082-02

Product: C710

Detail: CONFIG

Asm Date: 3162 Release #: 8.2

Update Date: 05-Aug-83 By: RDC1

Review Date: dd-Mon-yy By: _____

Problem: LOTUS 700 LARK driver entry points are to 710 driver not 700 driver.

Special Instructions:

Location (Octal)	New Contents (Octal a/o Symbolic)	Comments (Describe Solution)	Old Contents
014130	45216	;lufix address	062356
014131	37404	;bzud address	062004

Point 4 Data Corporation
830823

+--- P4DC production manager only ---+

| Status _____ |
| Production System |
| Update by_____ Date__/__/__ |
| |
| SCO # _____ Date__/__/__ |
| Master File |
| Update by_____ Date__/__/__ |
| Master File |
| Name _____ |

+-----------------------------------+

Patch #: CALLTBL82-02

Product: CALLTBL

Detail: $CALLTBL

Asm Date: 3159 Release #: R8.2

Update Date: 20-Jul-83 By: RMSA

Review Date: dd-Mon-yy By: _____

Problem: CALLTBL does not support subroutine #79 i.e. $BAKUP

Special Instructions: This patch completely replaces patch # CALLTBL82-01.

Only the contents of address 32423 have been changed.

Location (Octal)	New Contents (Octal a/o Symbolic)	Comments (Describe Solution)	Old Contents
032422	46315	;BAK radix 50	046315
032423	142720	;UP radix 50	027270
032424	0	;	000000
032425	117	;79 (decimal)	000117
032426	173	;BAKUP discsubs #	000173
032427	177777	;terminator	177777

Patch Received Made 11/15/83

Point 4 Data Corporation
830823

+--- P4DC production manager only ---+

| Status _____ |
| Production System |
| Update by_____ Date__/__/__ |
| SCO # _____ Date__/__/__ |
| Master File |
| Update by_____ Date__/__/__ |
| Master File |
| Name _____ |

+-----------------------------------+

Patch #: DSP82-01

Product: DSP

Detail: DSP

Asm Date: 3162 Release #: R8.2

Update Date: 13-Aug-83 By: RMSA

Review Date: dd-Mon-yy By: _____

Problem: Checksum capability i.e. X<adr1>,<adr2> gives trap #34

Special Instructions:

Location (Octal)	New Contents (Octal a/o Symbolic)	Comments (Describe Solution)	Old Contents
035206	LDA 1,56,3	;base register left off	024056
;;			
035210	LDA 1,55,3	;base register left off	024055
;;			
035215	LDA 0,53,3	;base register left off	020053
035216	LDA 1,54,3	;base register left off	024054

Point 4 Data Corporation
830823

+--- P4DC production manager only ---+

| Status _____ |
| Production System |
| Update by_____ Date__/__/__ |
| |
| SCO # _____ Date__/__/__ |
| Master File |
| Update by_____ Date__/__/__ |
| Master File |
| Name _____ |

+----------

Patch #: DSUB182-05

Product: DSUB1

Detail: DISCSUBS

Asm Date: 3162 Release #: R8.2

Update Date: 10-Aug-83 By: TWMA

Review Date: dd-Mon-yy By: _____

Problem: The "read rda from extender block" function of READMAINTENANCE does not work and may cause a trap 17.

Special Instructions:

Location (Octal)	New Contents (Octal a/o Symbolic)	Comments (Describe Solution)	Old Contents
014702	6110	;DATAPUMP	111000
014703	0	;GETBLOCK	006110
014704	100000	;@0	000000
014705	LDA 0,14601	;Load hdr bufr address	100000
014706	STA 2,14601	;Save extdr bufr address	020673
014707	MOV 0,2	;Move hdr addr for UNLATCH	050672

Point 4 Data Corporation
830823

+--- P4DC production manager only ---+

| Status _____ |
| Production System |
| Update by_____ Date__/__/__ |
| SCO # _____ Date__/__/__ |
| Master File |
| Update by_____ Date__/__/__ |
| Master File |
| Name _____ |

+-----------------------------------+

Patch #: DSUB182-06

Product: DSUB1

Detail: DISCSUBS

Asm Date: 3162 Release #: R8.2

Update Date: 13-Aug-83 By: RMSD

Review Date: dd-Mon-yy By: _____

Problem: NRCD is not updated correctly when doing random write to text file resulting in premature record not written errors

Special Instructions:

Location (Octal)	New Contents (Octal a/o Symbolic)	Comments (Describe Solution)	Old Contents
012041	JMP 12055	;NRCD will be handled by	021010
012042	77177	;GETRW there is no need to	034043
012043	77177	;arbitrarily set it to	163401
012044	77177	;NBLK-1	000030
012045	77177	;clear unused area to halts	034777
012046	77177	;	116404
012047	77177	;	000406
012050	77177	;	021011
012051	77177	;	162000
012052	77177	;	041016
012053	77177	;	020511
012054	77177	;	006143

Point 4 Data Corporation
830823

+--- P4DC production manager only ---+
| Status _____ |
| Production System |
| Update by_____ Date__/__/__ |
| SCO # _____ Date__/__/__ |
| Master File |
| Update by_____ Date__/__/__ |
| Master File |
| Name _____ |
+-----------------------------------+

Patch #: DSUB382-01

Product: DSUB3

Detail: DISCSUBS

Asm Date: 3162 Release #: R8.2

Update Date: 11-Aug-83 By: RMSA

Review Date: dd-Mon-yy By: _____

Problem: Tape subsystem causes system to crash when inputting an extended file

Special Instructions:

Location (Octal)	New Contents (Octal a/o Symbolic)	Comments (Describe Solution)	Old Contents
034336	SLS 2,0	;to be in HXA is before end	142032
;;			
034226	STA 2,34173	;replace patched code	77377
034227	JMP 34162	;clr HXA since wrong extr	77377
;;			
034113	JSR 34226	;create next extender	050460

Point 4 Data Corporation
830823

+--- P4DC production manager only ---+

| Status _____ |
| Production System |
| Update by_____ Date__/__/__ |
| SCO # _____ Date__/__/__ |
| Master File |
| Update by_____ Date__/__/__ |
| Master File |
| Name _____ |

+-----------------------------------+

Patch #: LIBR82-02

Product: LIBR

Detail: LIBR

Asm Date: 3120 Release #: R8.2

Update Date: 10-Aug-83 By: RDC1

Review Date: dd-Mon-yy By: _____

Problem: Trap 34 when doing a sorted LIBR of large logical unit with an INDEX that is nearly full.

Special Instructions:

Location (Octal)	New Contents (Octal a/o Symbolic)	Comments (Describe Solution)	Old Contents
000351	LDA 0,74	;ETSF	77377
000352	SKZ 0,0	;END OF TIME SLICE ?	77377
000353	6117	; YES, BUMPUSER	77377
000354	2401	; NO, CONTINUE	77377
000355	33343	;	77377
;;			
000206	351	;	033343

Point 4 Data Corporation
830823

```
+--- P4DC production manager only ---+
|                                     |
| Status _________                    |
| Production System                   |
| Update by__________ Date__/__/__    |
|                                     |
| SCO # _____________ Date__/__/__    |
| Master File                         |
| Update by__________ Date__/__/__    |
| Master File                         |
| Name _______________________________|
+-------------------------------------+
```

Patch #: REXM382-07

Product: REXM3

Detail: REX

Asm Date: 3167 Release #: R8.2

Update Date: 09-Aug-83 By: RMSC

Review Date: dd-Mon-yy By: _____

Problem: ESCAPE DOESN'T WORK PROPERLY

Special Instructions: This patch totally replaces REXM382-02

<Companion patch is REXM582-06>

Location (Octal)	New Contents (Octal a/o Symbolic)	Comments (Describe Solution)	Old Contents
000153	10723	;	77377
;;			
000706	10747	;	011334
;;			
001065	STA 0,73	;	042517
;;			
001077	LDA 0,73	;save escape status	175113
001100	STA 0,@1204	;in PCES	001400
001101	SSN 3,3	;replace displaced code	020073
001102	JMP 0,3	;	042502
001103	2401	;branch to patch	177240
001104	10743	;	003400
;;			
005700	6153	;	006102

Point 4 Data Corporation
830823

Patch #: REXM382-07

Product: REXM3 Asm Date: 3167 Release #: R8.2

Location (Octal)	New Contents (Octal a/o Symbolic)	Comments (Describe Solution)	Old Contents
010723	LDA 2,@.+2	;	77377
010724	2102	;	77377
010725	5345	;	77377
;;			
010743	SUB 0,0	;clear escape flag since	N/A
010744	STA 0,73	;we're making proper entry	N/A
010745	ADDOR 3,3	;replace patched code	N/A
010746	3400	;	N/A

Point 4 Data Corporation
830823

+--- P4DC production manager only ---+

| Status _____ |
| Production System |
| Update by_____ Date__/__/__ |
| SCO # _____ Date__/__/__ |
| Master File |
| Update by_____ Date__/__/__ |
| Master File |
| Name _____ |

+-----------------------------------+

Patch #: REXM582-06

Product: REXM5

Detail: REX

Asm Date: 3162 Release #: R8.2

Update Date: 09-Aug-83 By: RMSC

Review Date: dd-Mon-yy By: _____

Problem: ESCAPE DOESN'T WORK PROPERLY

Special Instructions: This patch totally replaces REXM582-02

<Companion patch is REXM382-07>

Location (Octal)	New Contents (Octal a/o Symbolic)	Comments (Describe Solution)	Old Contents
000153	11314	;	77377
;;			
000706	11340	;	011334
;;			
001163	STA 0,73	;	042517
;;			
001175	LDA 0,73	;save escape status	175113
001176	STA 0,@1302	;in PCES	001400
001177	SSN 3,3	;replace displaced code	020073
001200	JMP 0,3	;	042502
001201	2401	;branch to patch	177240
001202	11334	;	003400
;;			
006054	6153	;	006102
;;			

Point 4 Data Corporation
830823

Patch #: REXM582-06

Product: REXM5 Asm Date: 3162 Release #: R8.2

Location (Octal)	New Contents (Octal a/o Symbolic)	Comments (Describe Solution)	Old Contents
011314	LDA 2,@.+2	;	77377
011315	2102	;	77377
011316	5443	;	77377
;;			
011334	SUB 0,0	;clear escape flag since	N/A
011335	STA 0,73	;we're making proper entry	N/A
011336	ADDOR 3,3	;replace patched code	N/A
011337	3400	;	N/A

Point 4 Data Corporation
830823

+--- P4DC production manager only ---+
| |
| Status _____ |
| Production System |
| Update by_____ Date__/__/__ |
| |
| SCO # _____ Date__/__/__ |
| Master File |
| Update by_____ Date__/__/__ |
| Master File |
| Name _____ |
+-----------------------------------+

Patch #: RUN82-05

Product: RUN

Detail: RUN

Asm Date: 3158 Release #: R8.2

Update Date: 15-Aug-83 By: TWMF

Review Date: dd-Mon-yy By: _____

Problem: SPC(10) does not return the correct line number for errors 99 (Escape and 199 (Control-c)

Special Instructions:

Location (Octal)	New Contents (Octal a/o Symbolic)	Comments (Describe Solution)	Old Contents
036777	JMP @.+1	;Jump to patch	041051
037000	40436	; (patch address)	002375
;;;			
040436	STA 0,51,2	;Set error code	077377
040437	LDA 3,2,2	;Load PLC. of error line	077377
040440	ADD 2,3	;Calc address of stmt desc	077377
040441	LDA 0,0,3	;Load line number	077377
040442	STA 0,53,2	;Set error line number	077377
040443	JMP @375	;Execute error handler code	077377

Point 4 Data Corporation

```
+--- P4DC production manager only ---+
|                                     |
| Status _________                    |
| Production System                   |
| Update by__________ Date__/__/__    |
|                                     |
| SCO #  __________   Date__/__/__    |
| Master File                         |
| Update by__________ Date__/__/__    |
| Master File                         |
| Name   ____________________________ |
+-------------------------------------+
```

Patch #: S710CMD82-01

Product: S710CMD

Detail: REX

Asm Date: 3162 Release #: R8.2

Update Date: 18-Jul-83 By: DF1

Review Date: dd-Mon-yy By: _____

Problem: INFINITE LOOP ON LOTUS 700 & 710 SYSTEMS.

Special Instructions: For LOTUS 700 & 710 8.2 systems only. Peripherals handbook entry numbers: 36, 37, 38, 39, 41, 46, 54, 55 & 56.

Location (Octal)	New Contents (Octal a/o Symbolic)	Comments (Describe Solution)	Old Contents
027345	DOA 0,27	;	004661
;;			
027447	JMP 27342	;TO BS	027342
027450	JMP 27344	;TO JRET	027344

Point 4 Data Corporation
830823

+--- P4DC production manager only ---+

| Status _____ |
| Production System |
| Update by_____ Date__/__/__ |
| SCO # _____ Date__/__/__ |
| Master File |
| Update by_____ Date__/__/__ |
| Master File |
| Name _____ |

+-----------------------------------+

Patch #: TERMSYS82-01

Product: TERMSYS

Detail: $TERMS

Asm Date: 3120 Release #: R8.2

Update Date: 13-Aug-83 By: RMSA

Review Date: dd-Mon-yy By: _____

Problem: When mnemonic not found in output translation table, char. is sent ou as is. Should send out a null.

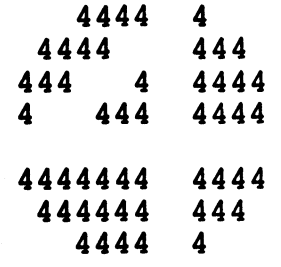
Special Instructions:

Location (Octal)	New Contents (Octal a/o Symbolic)	Comments (Describe Solution)	Old Contents
032636	JMP 32707	;	000452

R8.2 BASIC PROGRAM SCOs

September 1, 1983

POINT 4 Data Corporation



TO: All IRIS 8.2 Users

FROM: Systems Software Department

DATE: SEPTEMBER 1, 1983

SUBJ: Patch to GUIDE.LPT for MARK 5 & MARK 8

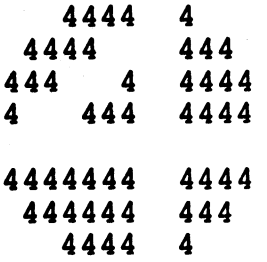
This patch is for the BASIC program 'GUIDE.LPT'. It should only be applied if your current operating system is IRIS 8.2 on a MARK 5 or MARK 8 CPU, and the current check code for GUIDE.LPT matches the OLD CHECK CODE given in the patch below.

DO NOT apply this patch if your software does not meet the above specifications.

You should patch the BASIC program 'GUIDE.LPT' using the procedure for patching BASIC programs described in this bulletin.

```
790 REM PROGRAM NAME: GUIDE.LPT  (FOR IRIS 8.2 MARK 5 ONLY)
790 REM
790 REM DATE: 8-16-83
790 REM
790 REM OLD CHECK CODE: 12E4A
790 REM NEW CHECK CODE: 377B
790 REM
790 REM REMARK: INCORRECT SDATE (LPTM-LPTP) & ABSOLUTE ADDRESSES
790 REM         FOR LPTP
790 REM
790 IF P=1 LET A=32561
1170 IF P=1 LET X=32565
1350 IF P=1 LET A9=32317
2040 IF P=0 PRINT #1;"\215\  32206:25  71360" !  ABS ADDR: ID TAGS
2050 IF P=1 PRINT #1;"\215\  32206:17  71360"
```

POINT 4 Data Corporation



TO: All IRIS 8.2 Users with MARK 3

FROM: Systems Software Department

DATE: SEPTEMBER 1, 1983

SUBJ: Patch to GUIDE.LPT for MARK 3

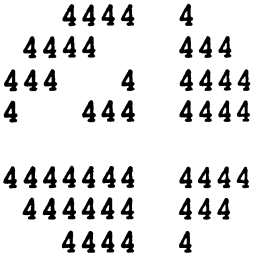
This patch is for the BASIC program 'GUIDE.LPT'. It should only be applied if your operating system is IRIS 8.2 on a MARK 3 CPU, and the check code for GUIDE.LPT matches the OLD CHECK CODE given in the patch below.

DO NOT apply this patch if your software does not meet these specifications.

You should patch the BASIC program 'GUIDE.LPT' using the procedure for patching BASIC programs described in this bulletin.

```
220 REM PROGRAM NAME: GUIDE.LPT
220 REM
220 REM IRIS 8.2  MARK 3 ONLY
220 REM DATE: 8-16-83
220 REM
220 REM OLD CHECK CODE: 7454
220 REM NEW CHECK CODE: A7EC
220 REM
220 REM REMARK: INCORRECT SDATE
220 REM
220 LET C[27] = 750 !  ASM DATE WORD FOR MARK 3
```

POINT 4 Data Corporation



TO: All IRIS 8.2 Users

FROM: Systems Software Department

DATE: SEPTEMBER 1, 1983

SUBJ: Changes to 'SETUP' Programs & SU.ENTRIES File

Included in this bulletin are patches to three SETUP modules. These patches are for SU1, SU112A and SU34.

In addition to these patches, the 'SU.ENTRIES' file must be updated. Revision 07 of the IRIS R8 PERIPHERALS HANDBOOK should be used to update the file. If Revision 07 is not available, updated information is listed on the following page of this bulletin (R8 DISC SPECIFICATION).

PROCEDURE:

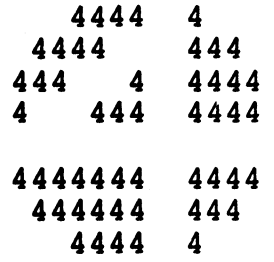
1. Obtain a listing of SU.ENTRIES file. This is done by selecting option (4) Disc Drive Entries File Maintenance from the Main Menu of SETUP, followed by selection of option (2) List the Drives Entry File.

2. Select option (1) Drives Entry File Maintenance, and make all necessary changes.

ENTRY NO.	DEVICE CODE	NO. CYLS IN LU/0	MAX. CYLS OTHER LUS	NPTC	DFLG	NTRS	PHYU CODE
1	33	200	626	2	100500	214	1
2	40	200	630	2	100500	214	1
5	30	200	626	2	104500	214	1
6	36	34	1123	5	40500	1213	0
7	36	34	626	5	40500	1213	0
8	36	24	631	5	40500	1220	0
9	36	10	234	23	40500	4613	0
10	36	5	153	23	40500	4620	0
11	40	156	626	2	100500	216	1
12	36	22	615	5	40500	1713	0
13	36	22	615	5	40500	1713	0
14	36	140	1450	1	40500	313	2
22	33	115	115	1	121000	110	0
23	40	200	626	2	100500	214	1
24	40	115	115	1	121000	110	0
26	33	100	626	2	100500	414	1
29	60	140	1450	1	500	220	2
30	60	24	630	5	40500	1220	0
32	33	313	313	2	105000	214	0
36	27	140	1462	1	500	220	2
37	27	10	252	14	40500	3020	0
38	27	24	626	5	40500	1220	0
39	27	6	153	23	40500	4620	0
40	60	6	153	23	40500	4620	0
41	27	14	377	10	40500	2020	0
43	27	32	1042	5	40500	1214	0
44	36	10	234	23	40500	4613	0
45	27	60	311	2	40500	420	1
46	27	12	314	12	40500	2420	0
47	73	200	626	2	100500	214	1
48	73	115	115	1	121000	110	0
49	40	313	313	2	105000	214	0
51	73	313	313	2	100500	214	0
54	27	16	444	7	40500	1620	0
55	27	34	775	5	40500	1213	0
56	27	60	311	2	40500	420	1
301	52	140	1462	1	500	220	2
302	52	60	311	2	500	420	1
303	52	34	775	5	40500	1213	0
304	52	57	1123	3	40500	613	0
305	52	24	626	5	40500	1220	0
306	52	6	153	23	40500	4620	0
307	52	16	444	7	40500	1620	0
308	52	14	377	10	40500	2020	0
309	52	12	314	12	40500	2420	0
310	52	10	252	14	40500	3020	0
360	52	115	115	2	20500	217	0

*** END OF DATA ***

POINT 4 Data Corporation



TO: IRIS 8.2 Users

FROM: Systems Software Department

DATE: SEPTEMBER 1, 1983

SUBJ: Patch to SU1

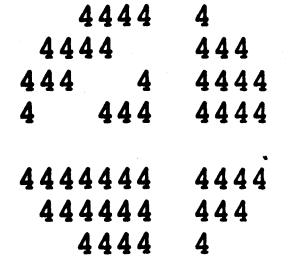
This patch is for the BASIC program 'SU1'. It should only be applied if your operating system is IRIS 8.2 and the check code for SU1 matches the OLD CHECK CODE given in the patch below.

DO NOT apply this patch if your software does not meet these specifications.

This patch should be applied using the procedure for patching BASIC programs described in this bulletin.

```
970 REM PROGRAM NAME: SU1
970 REM
970 REM DATE: 8-16-83
970 REM IRIS 8.2 ONLY
970 REM
970 REM OLD CHECK CODE: 8501
970 REM NEW CHECK CODE: C069
970 REM
970 LET A8$="MARK2T"
1960 PRINT @ 55,4;" 1 = MARK2T"@ 55,6;" 2 = MARK3";
```

POINT 4 Data Corporation



TO: All IRIS 8.2 Users

FROM: Systems Software Department

DATE: SEPTEMBER 1, 1983

SUBJ: Patch to SU112A

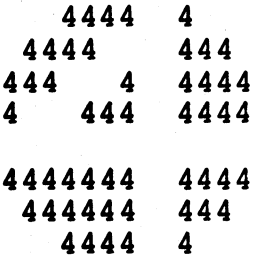
This patch is for BASIC program 'SU112A'. It should only be applied if your operating system is IRIS 8.2 and the check code for SU112A matches the OLD CHECK CODE given in the patch below.

DO NOT apply this patch if your software does not meet these specifications.

This patch should be applied using the procedure for patching BASIC programs described in this bulletin.

```
1980 REM PROGRAM NAME: SU112A
1980 REM
1980 REM DATE: 8-15-83
1980 REM IRIS 8.2 ONLY
1980 REM
1980 REM OLD CHECK CODE: 11879
1980 REM NEW CHECK CODE: 11492
1980 REM
1980 LET C8=2
2242 IF A8$[1,5]="MARK3" LET D8=140235
2262 IF A8$[1,5]="MARK3" LET A3$=140235
5090 PRINT @ 58,2;" 0 - NO TRANSLATION"@ 58,3;"15 - ADDS";
5095 PRINT @ 58,4;" 1 - ADM1A/31"@ 58,5;" 3 - ADM3A";
5100 PRINT @ 58,6;"10 - BEEHIVE100"@ 58,7;"11 - DG6052/6053";
5105 PRINT @ 58,8;"17 - DIALOGUE80"@ 58,9;" 6 - ELITE1520A";
5110 PRINT @ 58,10;" 7 - ELITE1521A"@ 58,11;" 5 - GE TERMINET";
5115 PRINT @ 58,12;"12 - HAZELTINE1500"@ 58,13;" 9 -
HAZELTINE2000";
5120 PRINT @ 58,14;"13 - MT ACT-V"@ 58,15;" 1 - SOROC IQ120";
5125 PRINT @ 58,16;"14 - TV912/920"@ 58,17;" 4 - TV950";
5130 PRINT @ 58,18;" 8 - VT100";
```

POINT 4 Data Corporation



TO: All IRIS 8.2 Users

FROM: Systems Software Department

DATE: SEPTEMBER 1, 1983

SUBJ: Patch to SU34

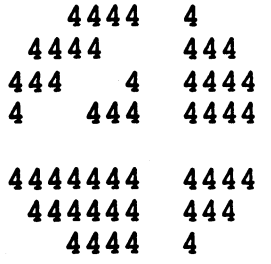
This patch is for the BASIC program 'SU34'. It should only be applied if your operating system is IRIS 8.2 and the check code for SU34 matches the OLD CHECK CODE given in the patch below.

DO NOT apply this patch if your software does not meet these specifications.

This patch should be applied using the procedure for patching BASIC programs described in this bulletin.

```
952 REM PROGRAM NAME: SU34
952 REM
952 REM DATE: 8-15-83
952 REM IRIS 8.2 ONLY
952 REM
952 REM OLD CHECK CODE: 6313
952 REM NEW CHECK CODE: D7BE
952 REM
952 LET A=6144+(B4[1]-1)*2
954 LET B=INT (A/256)
956 LET D=A-B*256
960 READ #2,B,C4+D;C5,C6
1602 IF B4[B6]=2 DEF FNP(X)=64+256*B2[B6]+8*B2[B6]+32768*B7[B6]
1642 IF B4[B6]=14 DEF FNP(X)=64*B2[B6]+B7[B6]+B5[B6]*32768
1720 IF B4[B6]=36 DEF FNP(X)=24576+B7[B6]*4+B2[B6]+B5[B6]*32768
1730 IF B4[B6]=37 DEF FNP(X)=B2[B6]+8240
1732 IF B4[B6]=38 DEF FNP(X)=B2[B6]+8212
1734 IF B4[B6]=39 DEF FNP(X)=B2[B6]+8268
1736 IF B4[B6]=40 DEF FNP(X)=B2[B6]
1740 IF B4[B6]=41 DEF FNP(X)=B2[B6]+8224
1780 IF B4[B6]=46 DEF FNP(X)=B2[B6]+8232
1810 REM ENTRY 50, 52, 53 NOT IN USE
1830 IF B4[B6]=54 DEF FNP(X)=B2[B6]+8220
1832 IF B4[B6]=55 DEF FNP(X)=B2[B6]+5652
1834 IF B4[B6]=56 DEF FNP(X)=8320+B2[B6]+B7[B6]*32768
```

POINT 4 Data Corporation



TO: All IRIS Users

FROM: Systems Software Department

DATE: SEPTEMBER 1, 1983

SUBJ: Patching BASIC Programs

Making patches to a protected BASIC program is a one-way street. That is, if an error is made during the patch process, such as entering a patch with the wrong line number, you can destroy program code that cannot be restored. In an effort to help you avoid possible errors, we propose the following procedure.

STEP 1. Use the COPY command to make a copy of the BASIC program module to be patched. Copy the program to another or the same Logical Unit, but give it a new name.

EXAMPLE: #COPY LU/PROGRAM.BU=LU/PROGRAM

NEVER patch this backup program !!

STEP 2. Create an IRIS text file consisting of all the patch lines of code for the module. Note---REM lines are optional, but may be included for documentation purposes. Be sure to enter each patch line into the text file in the same order shown on the patch listing. If lines are NOT entered in the same order shown, the new check code may not match.

STEP 3. Bring the program module into BASIC, load the patch text file, and SAVE the program under the original name.

```
EXAMPLE: #BASIC LU/PROGRAM
          LOAD LU/Patchfile
          EXIT
         #SAVE LU/PROGRAM!
```

IRIS will respond with: SAVED !! CHECK CODE = NNNN

NOTE: The "EXIT" command is used in IRIS R8 only. In IRIS 7.5, use a "CONTROL C" to return to SCOPE.

Keep the copy of the unpatched program available. Also, keep the patch text file on your system and use it to enter any subsequent patches to the same program.

If the above procedure is followed, your BASIC programs should be patched without error and the CHECK CODES will agree.

For the Applications Packages (STYLUS, TYPIST and FORCE) the patcher utility programs supplied with the Application Package should be used to apply any patches.