WINDRUSH
Micro Systems Ltd.

Worstead Laboratories (Reg Office)
North Walsham, Norfolk NR28 9SA
Tel (0692) 405189
Telex: 97360 SHARET G
MAPCON approved consultants

1 JUNE 1984

Dear Customer,

Enclosed you will find a set of bug patches for your copy of PL/9 version 3.XX.

We are about to release version 4.XX of PL/9. Details of the enhancements are enclosed.

You may purchase the upgrade for $25.00 (if outside of the U.K.) or 18.00 pounds (plus VAT) if you are within the U.K. Further details can be found on the enclosed order form.

If you wish to purchase the upgrade follow this procedure:

1. Locate the original disk we supplied with your copy of PL/9. Make a copy of it for your archives.

2. Complete the enclosed order form. This is very important as it will be the document U.K. customs will look for if they open the parcel for inspection. The order form substantiates the fact that the software is of U.K. origin so that it will clear customs without duties or delay.

3. If you are outside of the U.K. complete the 'GREEN CUSTOMS DECLARATION', which is enclosed, as follows:

   "COMPUTER SOFTWARE OF U.K. ORIGIN BEING RETURNED FOR UPGRADE"

   Declare the value of the goods as: $198.00

4. Insure the parcel for $198.00 (the cost of replacement).

5. Post the parcel with your remittance and the enclosed order form.

NOTES

A. YOU MUST RETURN THE ORIGINAL DISK SUPPLIED WITH YOUR COPY OF PL/9. WE WILL NOT ACCEPT ANY ORDERS FOR UPGRADES UNLESS THEY ARE ACCOMPANIED BY THE ORIGINAL DISK.

B. If you fail to fix the customs declaration to the parcel OR complete it incorrectly OR fail to complete and enclose the order form supplied the parcel may be siezed by U.K. customs. It will then take about two to three weeks to clear it.

C. If you fail to insure the parcel and we don't receive it you are out of luck! You will have to purchase the latest version at the full price. WE WILL NOT BE HELD RESPONSIBLE FOR THE PARCEL UNTIL IT IS IN OUR POSSESSION!

D. The parcel MUST be accompanied with payment for the upgrade. We will NOT supply upgrades against purchase orders on an open account basis.

CHANGES TO PL/9 FOR VERSION 4.XX

1. The STACK keyword

The STACK keyword has been enhanced, while maintaining compatibility with the old version. STACK can be used either outside or inside a procedure. In the former case, its purpose is to initialise the stack pointer at the start of a program. The operand may be Immediate (as before), Program Counter Relative (also as before) and now Extended, as follows:

Immediate:

```
STACK = $BFFF;
STACK = VALUE; (following CONSTANT VALUE = <NUMBER>)

These generate    LDS #$XXXX
```

Program Counter Relative:

```
STACK = *;   generating    LEAS *,PCR
```

Extended:

```
STACK = MEMEND; (following AT $CC2B: INTEGER MEMEND;)

generating   LDS $XXXX
```

Inside a procedure, the STACK keyword may be used on either the left or right of an expression. Its effect is to access or assign the 6809 CPU's Stack Pointer register, as in the following:

```
STACK = <Expression>;          giving   <get expression in ACCD>
                                        TFR D,S

<Variable> = STACK;            giving   TFR S,D
                                        STD <Variable>
```

Please note that the stack is not assigned or read directly; the former statement causes <Expression> to be evaluated in the D accumulator, regardless of complexity, and copied into the Stack Pointer, while the second copies the Stack Pointer into the D accumulator before using it in an expression. The effect is that in each case the accumulator is modified. This is important in the sort of programs that need to load or save the stack pointer!

2. Code improvements

The code generated by PL/9 has been improved for INTEGER vectors, such as the previously poor

```
VECTOR(INDEX) = VECTOR(INDEX) + ......
```

Any program that uses INTEGER vectors much will see a worthwhile decrease in size, with some improvement in speed (depending if the code is in a critical area).

3. Compiler options

The compile options T, P and L can now be used in combination, e.g. printout with output to screen. The listing output is now slightly different, in that lines now end with carriage return and line feed instead of just the former. This is to help some print spoolers do their job properly.

There is now an 'R' option which forces the compiler to use the MC6809 interrupt vectors at $FFF2 - $FFFF in lieu of those defined by SETPL9. This enables you to use SETPL9 to configure the copy of PL/9 for your system interrupt RAM vectors. Any compilation without the 'R' option (A:0, etc) will produce overlays for the RAM vectors. If you include the 'R' option (A:0,R) the MC6809 hardware vectors will be substituted. This was previously only possible if you created two versions of the compiler using SETPL9.

The result of these changes, together with fixes for all the bugs reported for the earlier version, was to take the size of the compiler to over 16k bytes. This limit is self-imposed in order to keep as much space as possible for source and object. The solution was to remove the error messages from the body of the compiler and put them into their own file. PL9.ERR has the same structure as ERRORS.SYS and is substituted for that file by the compiler when required, a facility thoughtfully included in FLEX by TSC. The effect is to slow down error reporting by an amount depending upon the efficiency of your disc drivers. If you find this a problem you'd better not make so many errors!

This version is also accompanied by a new manual which is essentially a reorganised version of the one supplied with version 3.XX. The main changes are that the manual is now supplied in two sections, a REFERENCE manal and a USERS GUIDE. It was felt that the existing manual was simply too bulky for most peoples tastes.

Along with the reorganisation we have taken the oportunity to correct several typographical errors. We have also expanded the sections covering:

a. POINTERS.
b. ASMPROCS.
c. ROM based programs.

As a result the manual now pushes 400 pages and hence the need to split it in two.

Please allow 6 weeks for delivery. Thats 3 weeks to us and 3 weeks back!

PL/9 VERSION 2.XX ... 3.XX UPGRADE ENHANCEMENTS

15 October 1983

Dear Registered PL/9 user,

Your copy of PL/9 may be upgraded to the current release. The main enhancements in the product are outlined below. Some of the enhancements will require slight modifications to the source files written for the older versions of the compiler. The main area where there are syntax differences that require modification are in the use of unsigned BYTES and INTEGERS.

VERSION 3.XX ENHANCEMENT SUMMARY

A. The manual has been revised to 300+ pages. A complete re-organization of the manual has taken place and a massive 'USERS GUIDE' with plenty of examples is the focus of the new document. The manual has been expanded to provide a full tutorial on programming in PL/9 and now includes a full description of all of the library routines provided.

B. Source files may now be written in upper or lower case. The compiler does not make any distinction between the case of letters however. Thus 'TEST', 'test' and 'TeSt' are all the same as far as the compiler is concerned.

C. You now have direct access to the 'X' register in the same way as you previously had access to 'A', 'B' and 'D'. This is provided primarily to improve communication with assembly language routines.

D. Several new forms of the 'Pointer' structure are provided:

`.POINTER = .BUFFER(18);`	Compute the address of the 18th element of BUFFER and place it in the memory location reserved for pointer.
`POINTER = BUFFER(18);`	Take the 18th element of BUFFER and place it in the memory location pointed to by POINTER.
`.POINTER = .POINTER + 1;`	Take the value contained in the memory location reserved for pointer and increment it by one.
`.POINTER = .POINTER(1);`	Step POINTER to the next element of the data it is pointing to. In the case of BYTE size data this is the same as the previous example. If the data is INTEGER then it is the same as '.POINTER = .POINTER + 2'. If the data is REAL it is the same as '.POINTER = .POINTER + 4'.
`POINTER(5) = BUFFER(3);`	Take the element of data at BUFFER(3) and copy it to POINTER(5).

E. SWI, SWI2, SWI3, and RESET procedure names (with the associated function) are now provided.

F. The previous restrictions involving the use of 'BREAK' have been removed.

G. The speeds of the REAL and INTEGER mathematics routines have been substantially improved.

H. The handling of unsigned BYTE and INTEGER variables has beem improved with the use of HEX notation in evaluation or assignment being taken as a desire to have the operation performed as unsigned. (no minor trick in a compiler designed to work with signed numbers!)

I. The tracer now resides in the FLEX transient command area thus enabling many extra features to be added to it.

J. The compiler may now be called directly from the FLEX command line to compile a program. A typical call looks like this:

```
+++PL9,1.SOURCE.PL9,0=1.OBJECT.BIN,P,C
```

Compile the program called SOURCE.PL9 to a disk object file called OBJECT.BIN make a printed listing of the compiled source with object code.

K. A:N option produces a symbol table only. Symbol table now includes GLOBALS and their offset from the 'Y' register to assist assembly language programmers wishing to access the GLOBAL variables on the stack.

UPGRADE PROCEDURE

If you wish to take advantage of our upgrade service follow the procedure outlined below:

1. Locate the ORIGINAL disk we supplied with the product. We will only accept the original disk (with our label on it) for upgrade. This is an unbreakable company policy and has been adopted to prevent 'kloning' of the original package without introducing massive overheads into our internal procedures. The only way we can offer this service at the price is to rigidly enforce this basic rule.

2. Make a working master of the original disk and put it in a safe place. This is just in case you crash your work disk while we have the original or in case you want to continue using the old version for a while.

3. For customers outside of the U.K. the upgrade fee is $25.00. For customers within the U.K. the upgrade fee is 18.98 pounds (15.00 + 1.50 PP + 2.48 VAT). Payment by bankers draft or international money order is preferred. Personal cheques are acceptable but they can take up to four weeks to clear. VISA/ACCESS/MASTER/EURO credit cards are all acceptable.

ALL UPGRADES MUST BE PRE-PAID. NO OPEN ACCOUNT FACILITIES ARE AVAILABLE.

3. Forward the ORIGINAL disk to us via REGISTERED post insuring it for the full purchase price of the product. We will not accept any responsibility for disks that fail to reach us. Mark the customs declaration: GOODS OF U.K. ORIGIN BEING RETURNED FOR REPAIR.

4. Unless otherwise specified the upgrade disk will be returned in the same format as the disk returned to us. If you wish to swap it for another disk size or track density PLEASE STATE YOUR REQUIREMENTS as we are not clairvoyant!

5. We will forward you the upgrade package via REGISTERED AIR MAIL within one working week of clearing your payment.

PL/9 COMPILER, VERSION 3.01 AND 3.02 BUG FIXES as at 15 OCTOBER 1983

The following patches cure all known problems with the above versions of the compiler and upgrade it to the current version; 3.03.

If you are aware of any other problems we would be most obliged if you would complete the enclosed 'BUG REPORT' and return it to us. We will endeavor to fix any reported bug that we can verify within three weeks.

To implement the patches follow this procedure:

1. Use the FLEX 'GET' command to load the PL/9 object code into memory:

   +++GET,0.PL9.CMD <RETURN>

2. Use the FLEX 'RENAME' command to change the name of the old PL9.CMD as follows:

   +++RENAME,0.PL9.CMD,0.PL9-301.CMD <RETURN>

3. Use your system monitor memory examine and change command to implement the patches by altering the contents of the various memory locations as indicated.

4. Once all of the patches have been made (make sure you get it right!) save the binary out to disk using the FLEX 'SAVE' command as follows:

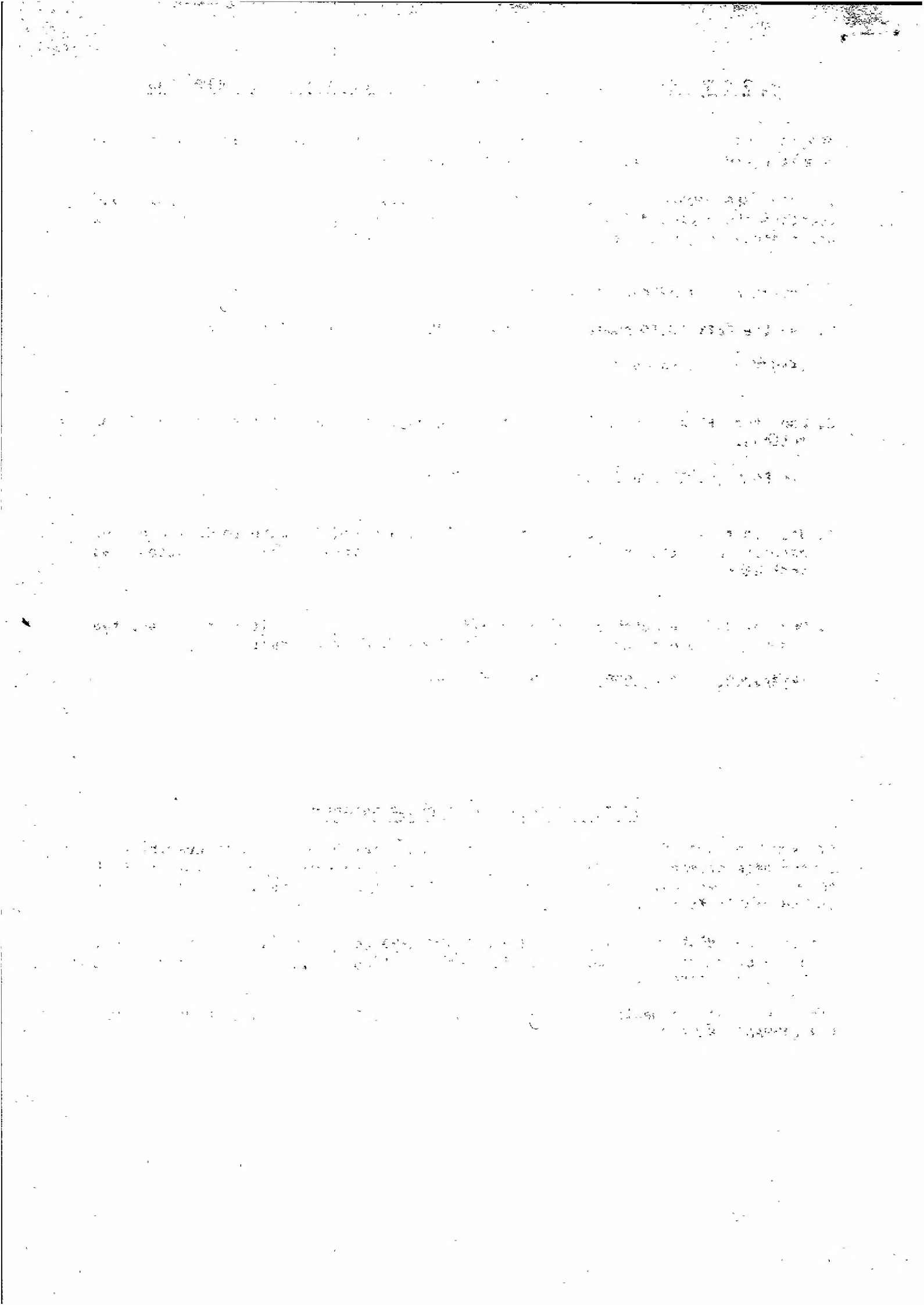
   +++SAVE,0.PL9.CMD,0000,3FFC,0000 <RETURN>

NOTE TO NON-ASSEMBLY LANGUAGE PROGRAMMERS

If you do not feel competent to perform these patches (not everybody is comfortable at machine code level) you are welcome to return your original disk to us and we will perform them for you free of charge. We will turn the disk around within five working days.

The original disk must be returned via REGISTERED post and insured for the full purchase price of the product. We will not accept any responsibility for disks that fail to reach us.

The green customs declaration should state: GOODS OF U.K. ORIGIN BEING RETURNED FOR WARRANTY REPAIR.



SOFTWARE BUG REPORT

PRODUCT _____ REVISION _____ SERIAL NUMBER _____

NAME OF DEALER _____

DATE _____

NAME _____

ADDRESS _____

_____

_____

_____

DESCRIBE YOUR SYSTEM HARDWARE AND ANY MODIFICATIONS YOU MAY HAVE MADE.

_____

_____

_____

_____

_____

_____

_____

Describe, in detail, the problem you are having. Include examples which we can run that clearly and REPEATEDLY demonstates the problem. The simpler the example the quicker we will be able to locate the cause of it.

Attach extra sheets or send us a SS/SD disk with upper case file names.

_____

_____

_____

_____

_____

_____

_____

_____

SEND THIS FORM AND ANY ASSOCIATED DOCUMENTATION TO:

Windrush Micro Systems, Worstead Labs., N. Walsham, Norfolk, NR28 9SA, England

TROUBLE WITH THE 'INCHNE' VECTOR?

We have encountered a lot of problems with our use of this FLEX vector located at $D3E5. It is supposed to point to the 'INPUT CHARACTER, NEVER ECHO ROUTINE', which, like 'INCH' is supposed to strip parity.

One of the enclosed patches (Patch 8) does not need to be performed if your version of FLEX has the INCHNE vector at $D3E5 implemented correctly. Most of the problems with our software involve SWTP versions of FLEX. Either this vector has not been implemented at all or, if it is implemented, it does not strip the parity bit out of the incoming code.

Either of these situations will result in PL/9, SETPL9, MACE, XMACE, SETMACE, and SETXMACE either locking up or crashing the system.

The root cause of the problem appears to be a revision in the FLEX I/O routine vector table that was made around 1980 by TSC. Earlier versions of FLEX started the vector table at $D3E7 and did not have an INCHNE vector whilst the current versions start the vector table at $D3E5 and have the INCHNE vector at this location. In most of the SWTP versions we have encountered problems with $D3E5 points to the MC6850 ACIA location.

The best solution to the problem is to upgrade your version of FLEX to the latest version which has the INCHNE vector at $D3E5 installed correctly. Alternatively you can build an INCHNE routine into PL/9 as described in Patch 8.

Since SETPL9 also uses the INCHNE routine (it is part of the 'IOSUBS.LIB' file) it must be recompiled once you have patched PL/9 itself. If you call PL/9 from the FLEX command line you will be able to recompile the SETPL9 program AFTER you have modified the 'GETCHAR_NOECHO' routine in the IOSUBS library as required by your system configuration. The FLEX call looks like this:

```
+++PL9,0.SETPL9.PL9,0=0.SETPL9.CMD <RETURN>
```

If your system uses 'SBUG-E', 'GMX-BUG', 'CHIEFBUG' or 'GT-BUG' or any other system monitor that has the following vector table located at $F800 the modifications required will be quite simple.

```
$F800   RESET     COLD START
$F802   CNTRL     WARM START
$F804   INCHNE    INPUT CHARACTER, NEVER ECHO
$F806   INCHAE    INPUT CHARACTER, ALWAYS ECHO
$F808   INCHECK   TEST FOR INCOMING CHARACTER (<> 0 IF CHAR IS WAITING)
$F80A   OUTCH     OUTPUT A CHARACTER
$F80C   PDATA     OUTPUT STRING POINTED TO BY 'X' ($04 TERMINATES)
$F80E   CRLF      OUTPUT CR-LF STRING
$F810   PSTRNG    OUTPUT CR-LF FOLLOWED BY STRING POINTED TO BY 'X' ($04 TERM)
$F812   LRA       LOAD REAL ADDRESS (USED MAINLY BY DMA DISK CONTROLLERS)

$F814   JMP CONTROL  (not present in some monitors)
```

In the above circumstances the 'GETCHAR_NOECHO' routine in IOSUBS can be modified as follows:

```
PROCEDURE GETCHAR_NOECHO;
   GEN $AD,$9F,$F8,$04;        /*  JSR [$F804]                  */
   GEN $84,$7F;                /*  ANDA #$7F   (STRIPS PARITY)  */
ENDPROC ACCA;
```

If you are using another system monitor you can construct an INCHNE routine as follows:

```
PROCEDURE GETCHAR_NOECHO;
   REPEAT;
      CALL $CD4E;                /*  FLEX 'STAT'                          */
   UNTIL CCR AND $04 = 0;        /*  THERE IS A KEY WAITING!              */

   GEN $B6,$E0,$05;              /*  LDA $E005  (READ ACIA DATA PORT)     */
   GEN $84,$7F;                  /*  ANDA #$7F   (STRIP PARITY)           */
ENDPROC ACCA;
```

In the above example we are using the FLEX 'STAT' routine to test the status of the system console keyboard. Once we have determined that a key has been pressed we then simply read the ACIA data port, strip the parity bit, and return the code in the 'A' accumulator.

We could just as easily read the data from a PIA data port or whatever I/O device you have in the system.

Patch 1

This patch fixes the problem of the wrong code being generated for pointers with constant indices.

```
25D1  12
```

Patch 2

This patch fixes the problem of the compiler crashing on an assignment split between two lines.

```
1E43  16
1E44  21
1E45  6E
1E46  5F
1E47  4F
1E48  DD
1E49  6D
1E4A  35
1E4B  90

399F  16
39A0  E4
39A1  A4
```

Patch 3

This patch fixes the problem of symbols that contain numbers being confused with symbols containing letters in the same position.

```
0CD6  16
0CD7  32
0CD8  F2
```

Patch 4

This patch fixes (yet another) problem with multiple ORIGINs, this time where accessing DATA previously located at a higher address generates the wrong code.

```
279E  12
279F  17
27A0  18
27A1  37
27A2  26
27A3  11
```

Patch 5

This patch fixes the problem of RETURN and ENDPROC not forcing INTEGER when it is specified.

```
1AAC  34
1AAD  04
1AAE  16
1AAF  25
1AB0  3A
```

Patch 6

This patch fixes the problem of ENDPROC not always cleaning up the stack properly if the last constant encountered was hex. (How about that for cause-and-effect?)

```
1B0C  17
1B0D  24
1B0E  E9
```

Patch 7

This patch fixes the problem in some versions of FLEX that the interrupt vectors IRQ and SWI3 get altered by FLEX while it is loading the tracer.

```
353B  17
353C  09
353D  13

3E51  EE
3E52  9F
3E53  00
3E54  07
3E55  10
3E56  AE
3E57  9F
3E58  00
3E59  0F
3E5A  34
3E5B  60
3E5C  BD
3E5D  CD
3E5E  4B
3E5F  35
3E60  60
3E61  EF
3E62  9F
3E63  00
3E64  07
3E65  10
3E66  AF
3E67  9F
3E68  00
3E69  0F
3E6A  39
```

Patch 8

This patch does not need to be performed in the vast majority of systems

This patch is only required in systems that use a version of FLEX that does not have the INCHNE vector at $D3E5 installed or in versions that have the vector installed but the routine does not strip the parity bit out of the incoming code.

Most of the systems we have had problems with in this area are early SWTP 6809 systems. Since these systems all have 'SBUG-E' this patch is organized around this system monitor. The basic principle of the patch is to 're-aim' four calls in PL/9 that use the FLEX 'INCHNE' vector at $D3E5 to a vector at $3E70. The vector at $3E70 (which is installed by this patch) points to a routine at $3E72:

```
02CD   3E    (CURRENTLY D3)
02CE   70    (CURRENTLY E5)

3A29   3E    (CURRENTLY D3)
3A2A   70    (CURRENTLY E5)

3A32   3E    (CURRENTLY D3)
3A33   70    (CURRENTLY E5)

3F57   3E    (CURRENTLY D3)
3F58   70    (CURRENTLY E5)

3E70   3E    :
3E71   72    :  (NEW VECTOR IN LIEU OF $D3E5)
```

Next we instal the INCHNE routine itself. This routine can take many forms but since the majority of problems appear to be in SWTP systems we are simply going to use the INCHNE vector in 'SBUG-E' itself:

```
3E72   AD    :
3E73   9F    :  ( JSR [$F804] )
3E74   F8    :
3E75   04    :

3E76   84    :
3E77   7F    :  ( ANDA #$7F ... STRIP PARITY )

3E78   39    :  ( RTS         )
```

If your system monitor does not have an INCHNE routine that you can use you can construct one in the 14 bytes available between $3E72 and $3E7F as indicated in the following example:

```
3E72   BD    :
3E73   CD    :  (JSR $CD4E ... 'STAT' IN FLEX)
3E74   4E    :

3E75   27    :
3E76   FB    :  (BEQ $3E72 ... LOOP IF NO KEY PENDING)

3E77   B6    :  (LDA DATAPORT  ... IN THIS CASE AN MC6850 ACIA AT $E004      )
3E78   E0    :  (Note:  This could have just as easily been a PIA at $E040 by)
3E79   05    :  (       using: B6, E0, 40.                                   )

3E7A   84    :
3E7B   7F    :  ( ANDA #$7F ... STRIP PARITY )

3E7C   39    :  ( RTS         )
```

Patch 9

This is the extra code needed for patches one (1) through seven (7). It is located at the end of the code for PL/9.

```
3FB4   8C                +------------> 3FD9   10
3FB5   41                |              3FDA   83
3FB6   B3                |              3FDB   FF
3FB7   10                |              3FDC   83
3FB8   27                |              3FDD   2D
3FB9   DE                |              3FDE   09
3FBA   E7                |              3FDF   10
3FBB   A6                |              3FE0   83
3FBC   82                |              3FE1   00
3FBD   81                |              3FE2   82
3FBE   20                |              3FE3   2E
3FBF   27                |              3FE4   03
3FC0   03                |              3FE5   1A
3FC1   34                |              3FE6   04
3FC1   02                |              3FE7   39
3FC3   5C                |              3FE8   1C
3FC4   9C                |              3FE9   FB
3FC5   6D                |              3FEA   39
3FC6   22                |              3FEB   17
3FC7   EC                |              3FEC   E1
3FC8   16                |              3FED   5C
3FC9   DE                |              3FEE   6D
3FCA   85                |              3FEF   E0
3FCB   17                |              3FF0   27
3FCC   CE                |              3FF1   03
3FCD   54                |              3FF2   17
3FCE   10                |              3FF3   DD
3FCF   26                |              3FF4   14
3FD0   CD                |              3FF5   16
3FD1   1C                |              3FF6   DA
3FD2   8B                |              3FF7   CE
3FD3   20                |              3FF8   0F
3FD4   A1                |              3FF9   22
3FD5   3F                |              3FFA   16
3FD6   16                |              3FFB   CE
3FD7   CD                |              3FFC   67
3FD8   01 >--------------+
```

Patch 10

These final patches change the version number to 3.03 and modify the the END of PL/9 pointer (used by 'SETPL9') to point to the new end of the PL/9 code defined by these patches.

```
005A   33

006E   3F
006F   FC
```

PL/9 COMPILER, VERSION 3.03 BUG FIXES as at 15 NOVEMBER 1983

The following patches cure all known problems with the above versions of the compiler and upgrade it to the current version; 3.04.

If you have versions 3.01 or 3.02 you must incorporate all of the modifications outlined on the previous 'bug-patches' letter before incorporating these patches.

If you are aware of any other problems we would be most obliged if you would complete the enclosed 'BUG REPORT' and return it to us. We will endeavor to fix any reported bug that we can verify within three weeks.

To implement the patches follow this procedure:

1. Use the FLEX 'GET' command to load the PL/9 object code into memory:

   +++GET,0.PL9.CMD <RETURN>

2. Use the FLEX 'RENAME' command to change the name of the old PL9.CMD as follows:

   +++RENAME,0.PL9.CMD,0.PL9-303.CMD <RETURN>

3. Use your system monitor memory examine and change command to implement the patches by altering the contents of the various memory locations as indicated.

4. Once all of the patches have been made (make sure you get it right!) save the binary out to disk using the FLEX 'SAVE' command as follows:

   +++SAVE,0.PL9.CMD,0000,3FFC,0000 <RETURN>

NOTE TO NON-ASSEMBLY LANGUAGE PROGRAMMERS

If you do not feel competent to perform these patches (not everybody is comfortable at machine code level) you are welcome to return your original disk to us and we will perform them for you free of charge. We will turn the disk around within five working days.

The original disk must be returned via REGISTERED post and insured for the full purchase price of the product. We will not accept any responsibility for disks that fail to reach us.

The green customs declaration should state: GOODS OF U.K. ORIGIN BEING RETURNED FOR WARRANTY REPAIR.

SOFTWARE BUG REPORT

PRODUCT _____ REVISION _____ SERIAL NUMBER _____

NAME OF DEALER _____

DATE _____

NAME _____

ADDRESS _____

_____

_____

_____

DESCRIBE YOUR SYSTEM HARDWARE AND ANY MODIFICATIONS YOU MAY HAVE MADE.

_____

_____

_____

_____

_____

_____

_____

Describe, in detail, the problem you are having. Include examples which we can run that clearly and REPEATEDLY demonstates the problem. The simpler the example the quicker we will be able to locate the cause of it.

Attach extra sheets or send us a SS/SD disk with upper case file names.

_____

_____

_____

_____

_____

_____

_____

_____

SEND THIS FORM AND ANY ASSOCIATED DOCUMENTATION TO:

Windrush Micro Systems, Worstead Labs., N. Walsham, Norfolk, NR28 9SA, England

TROUBLE WITH THE 'INCHNE' VECTOR?

We have encountered a lot of problems with our use of this FLEX vector located at $D3E5. It is supposed to point to the 'INPUT CHARACTER, NEVER ECHO ROUTINE', which, like 'INCH' is supposed to strip parity.

One of the enclosed patches (Patch 1) does not need to be performed if your version of FLEX has the INCHNE vector at $D3E5 implemented correctly. Most of the problems with our software involve SWTP versions of FLEX. Either this vector has not been implemented at all or, if it is implemented, it does not strip the parity bit out of the incoming code.

Either of these situations will result in PL/9, SETPL9, MACE, XMACE, SETMACE, and SETXMACE either locking up or crashing the system.

The root cause of the problem appears to be a revision in the FLEX I/O routine vector table that was made around 1980 by TSC. Earlier versions of FLEX started the vector table at $D3E7 and did not have an INCHNE vector whilst the current versions start the vector table at $D3E5 and have the INCHNE vector at this location. In most of the SWTP versions we have encountered problems with $D3E5 points to the MC6850 ACIA location.

The best solution to the problem is to upgrade your version of FLEX to the latest version which has the INCHNE vector at $D3E5 installed correctly. Alternatively you can build an INCHNE routine into PL/9 as described in Patch 8.

Since SETPL9 also uses the INCHNE routine (it is part of the 'IOSUBS.LIB' file) it must be recompiled once you have patched PL/9 itself. If you call PL/9 from the FLEX command line you will be able to recompile the SETPL9 program AFTER you have modified the 'GETCHAR_NOECHO' routine in the IOSUBS library as required by your system configuration. The FLEX call looks like this:

```
+++PL9,0.SETPL9.PL9,0=0.SETPL9.CMD <RETURN>
```

If your system uses 'SBUG-E', 'GMX-BUG', 'CHIEFBUG' or 'GT-BUG' or any other system monitor that has the following vector table located at $F800 the modifications required will be quite simple.

```
$F800   RESET     COLD START
$F802   CNTRL     WARM START
$F804   INCHNE    INPUT CHARACTER, NEVER ECHO
$F806   INCHAE    INPUT CHARACTER, ALWAYS ECHO
$F808   INCHECK   TEST FOR INCOMING CHARACTER (<> 0 IF CHAR IS WAITING)
$F80A   OUTCH     OUTPUT A CHARACTER
$F80C   PDATA     OUTPUT STRING POINTED TO BY 'X' ($04 TERMINATES)
$F80E   CRLF      OUTPUT CR-LF STRING
$F810   PSTRNG    OUTPUT CR-LF FOLLOWED BY STRING POINTED TO BY 'X' ($04 TERM)
$F812   LRA       LOAD REAL ADDRESS (USED MAINLY BY DMA DISK CONTROLLERS)

$F814   JMP CONTROL  (not present in some monitors)
```

In the above circumstances the 'GETCHAR_NOECHO' routine in IOSUBS can be modified as follows:

```
PROCEDURE GETCHAR_NOECHO;
   GEN $AD,$9F,$F8,$04;        /*  JSR [$F804]                  */
   GEN $84,$7F;                /*  ANDA #$7F   (STRIPS PARITY)  */
ENDPROC ACCA;
```

If you are using another system monitor you can construct an INCHNE routine as follows:

```
PROCEDURE GETCHAR_NOECHO;
   REPEAT;
      CALL $CD4E;              /*  FLEX 'STAT'                          */
   UNTIL CCR AND $04 = 0;      /*  THERE IS A KEY WAITING!              */

   GEN $B6,$E0,$05;            /*  LDA $E005  (READ ACIA DATA PORT)     */
   GEN $84,$7F;                /*  ANDA #$7F   (STRIP PARITY)           */
ENDPROC ACCA;
```

In the above example we are using the FLEX 'STAT' routine to test the status of the system console keyboard. Once we have determined that a key has been pressed we then simply read the ACIA data port, strip the parity bit, and return the code in the 'A' accumulator.

We could just as easily read the data from a PIA data port or whatever I/O device you have in the system.

Patch 1

This patch does not need to be performed in the vast majority of systems

This patch is only required in systems that use a version of FLEX that does not have the INCHNE vector at $D3E5 installed or in versions that have the vector installed but the routine does not strip the parity bit out of the incoming code.

Most of the systems we have had problems with in this area are early SWTP 6809 systems. Since these systems all have 'SBUG-E' this patch is organized around this system monitor. The basic principle of the patch is to 're-aim' four calls in PL/9 that use the FLEX 'INCHNE' vector at $D3E5 to a vector at $3E70. The vector at $3E70 (which is installed by this patch) points to a routine at $3E72:

```
02CD   3E    (CURRENTLY D3)
02CE   70    (CURRENTLY E5)

3A29   3E    (CURRENTLY D3)
3A2A   70    (CURRENTLY E5)

3A32   3E    (CURRENTLY D3)
3A33   70    (CURRENTLY E5)

3F57   3E    (CURRENTLY D3)
3F58   70    (CURRENTLY E5)

3E70   3E    :
3E71   72    :  (NEW VECTOR IN LIEU OF $D3E5)
```

Next we instal the INCHNE routine itself. This routine can take many forms but since the majority of problems appear to be in SWTP systems we are simply going to use the INCHNE vector in 'SBUG-E' itself:

```
3E72   AD    :
3E73   9F    :  ( JSR [$F804] )
3E74   F8    :
3E75   04    :

3E76   84    :
3E77   7F    :  ( ANDA #$7F ... STRIP PARITY )

3E78   39    :  ( RTS         )
```

If your system monitor does not have an INCHNE routine that you can use you can construct one in the 14 bytes available between $3E72 and $3E7F as indicated in the following example:

```
3E72   BD    :
3E73   CD    :  (JSR $CD4E ... 'STAT' IN FLEX)
3E74   4E    :

3E75   27    :
3E76   FB    :  (BEQ $3E72 ... LOOP IF NO KEY PENDING)

3E77   B6    :  (LDA DATAPORT  ... IN THIS CASE AN MC6850 ACIA AT $E004      )
3E78   E0    :  (Note:  This could have just as easily been a PIA at $E040 by)
3E79   05    :  (       using: B6, E0, 40.                                   )

3E7A   84    :
3E7B   7F    :  ( ANDA #$7F ... STRIP PARITY )

3E7C   39    :  ( RTS         )
```

Patch 2

This patch fixes the problem of the compiler crashing when it encounters an error in a call to a procedure using a subscripted pointer.

1657 16
1658 00
1659 5B

Patch 3

This patch fixes the problem of F!/string or C!/string either not working or crashing the compiler.

0409 CC
040A FF
040B FF
040C DD
040D 0C
040E 30
040F 01
0410 39

Patch 4

This final patch changes the version number to 3.04.

005A 34

PL/9 COMPILER, VERSION 3.04 BUG FIXES as at 1 DECEMBER 1983

The following patches cure all known problems with the above versions of the compiler and upgrade it to the current version; 3.05.

If you have versions 3.01, 3.02 or 3.03 you must incorporate all of the modifications outlined on the previous 'bug-patches' letters before incorporating these patches.

If you are aware of any other problems we would be most obliged if you would complete the enclosed 'BUG REPORT' and return it to us. We will endeavor to fix any reported bug that we can verify within three weeks.

To implement the patches follow this procedure:

1. Use the FLEX 'GET' command to load the PL/9 object code into memory:

   +++GET,0.PL9.CMD <RETURN>

2. Use the FLEX 'RENAME' command to change the name of the old PL9.CMD as follows:

   +++RENAME,0.PL9.CMD,0.PL9-304.CMD <RETURN>

3. Use your system monitor memory examine and change command to implement the patches by altering the contents of the various memory locations as indicated.

4. Once all of the patches have been made (make sure you get it right!) save the binary out to disk using the FLEX 'SAVE' command as follows:

   +++SAVE,0.PL9.CMD,0000,3FFC,0000 <RETURN>

NOTE TO NON-ASSEMBLY LANGUAGE PROGRAMMERS

If you do not feel competent to perform these patches (not everybody is comfortable at machine code level) you are welcome to return your original disk to us and we will perform them for you free of charge. We will turn the disk around within five working days.

The original disk must be returned via REGISTERED post and insured for the full purchase price of the product. We will not accept any responsibility for disks that fail to reach us.

The green customs declaration should state: GOODS OF U.K. ORIGIN BEING RETURNED FOR WARRANTY REPAIR.

TROUBLE WITH THE 'INCHNE' VECTOR?

We have encountered a lot of problems with our use of this FLEX vector located at $D3E5. It is supposed to point to the 'INPUT CHARACTER, NEVER ECHO ROUTINE', which, like 'INCH' is supposed to strip parity.

One of the enclosed patches (Patch 1) does not need to be performed if your version of FLEX has the INCHNE vector at $D3E5 implemented correctly. Most of the problems with our software involve SWTP versions of FLEX. Either this vector has not been implemented at all or, if it is implemented, it does not strip the parity bit out of the incoming code.

Either of these situations will result in PL/9, SETPL9, MACE, XMACE, SETMACE, and SETXMACE either locking up or crashing the system.

The root cause of the problem appears to be a revision in the FLEX I/O routine vector table that was made around 1980 by TSC. Earlier versions of FLEX started the vector table at $D3E7 and did not have an INCHNE vector whilst the current versions start the vector table at $D3E5 and have the INCHNE vector at this location. In most of the SWTP versions we have encountered problems with $D3E5 points to the MC6850 ACIA location.

The best solution to the problem is to upgrade your version of FLEX to the latest version which has the INCHNE vector at $D3E5 installed correctly. Alternatively you can build an INCHNE routine into PL/9 as described in Patch 8.

Since SETPL9 also uses the INCHNE routine (it is part of the 'IOSUBS.LIB' file) it must be recompiled once you have patched PL/9 itself. If you call PL/9 from the FLEX command line you will be able to recompile the SETPL9 program AFTER you have modified the 'GETCHAR_NOECHO' routine in the IOSUBS library as required by your system configuration. The FLEX call looks like this:

```
+++PL9,0.SETPL9.PL9,0=0.SETPL9.CMD <RETURN>
```

If your system uses 'SBUG-E', 'GMX-BUG', 'CHIEFBUG' or 'GT-BUG' or any other system monitor that has the following vector table located at $F800 the modifications required will be quite simple.

```
$F800   RESET     COLD START
$F802   CNTRL     WARM START
$F804   INCHNE    INPUT CHARACTER, NEVER ECHO
$F806   INCHAE    INPUT CHARACTER, ALWAYS ECHO
$F808   INCHECK   TEST FOR INCOMING CHARACTER (<> 0 IF CHAR IS WAITING)
$F80A   OUTCH     OUTPUT A CHARACTER
$F80C   PDATA     OUTPUT STRING POINTED TO BY 'X' ($04 TERMINATES)
$F80E   CRLF      OUTPUT CR-LF STRING
$F810   PSTRNG    OUTPUT CR-LF FOLLOWED BY STRING POINTED TO BY 'X' ($04 TERM)
$F812   LRA       LOAD REAL ADDRESS (USED MAINLY BY DMA DISK CONTROLLERS)

$F814   JMP CONTROL  (not present in some monitors)
```

In the above circumstances the 'GETCHAR_NOECHO' routine in IOSUBS can be modified as follows:

```
PROCEDURE GETCHAR_NOECHO;
   GEN $AD,$9F,$F8,$04;        /*  JSR [$F804]                  */
   GEN $84,$7F;                /*  ANDA #$7F   (STRIPS PARITY)  */
ENDPROC ACCA;
```

If you are using another system monitor you can construct an INCHNE routine as follows:

```
PROCEDURE GETCHAR_NOECHO;
   REPEAT;
      CALL $CD4E;              /*  FLEX 'STAT'                        */
   UNTIL CCR AND $04 = 0;      /*  THERE IS A KEY WAITING!            */

   GEN $B6,$E0,$05;            /*  LDA $E005  (READ ACIA DATA PORT)   */
   GEN $84,$7F;                /*  ANDA #$7F   (STRIP PARITY)         */
ENDPROC ACCA;
```

In the above example we are using the FLEX 'STAT' routine to test the status of the system console keyboard. Once we have determined that a key has been pressed we then simply read the ACIA data port, strip the parity bit, and return the code in the 'A' accumulator.

We could just as easily read the data from a PIA data port or whatever I/O device you have in the system.

Patch 1

This patch does not need to be performed in the vast majority of systems

This patch is only required in systems that use a version of FLEX that does not have the INCHNE vector at $D3E5 installed or in versions that have the vector installed but the routine does not strip the parity bit out of the incoming code.

Most of the systems we have had problems with in this area are early SWTP 6809 systems. Since these systems all have 'SBUG-E' this patch is organized around this system monitor. The basic principle of the patch is to 're-aim' four calls in PL/9 that use the FLEX 'INCHNE' vector at $D3E5 to a vector at $3E70. The vector at $3E70 (which is installed by this patch) points to a routine at $3E72:

```
02CD   3E    (CURRENTLY D3)
02CE   70    (CURRENTLY E5)

3A29   3E    (CURRENTLY D3)
3A2A   70    (CURRENTLY E5)

3A32   3E    (CURRENTLY D3)
3A33   70    (CURRENTLY E5)

3F57   3E    (CURRENTLY D3)
3F58   70    (CURRENTLY E5)

3E70   3E    :
3E71   72    :  (NEW VECTOR IN LIEU OF $D3E5)
```

Next we instal the INCHNE routine itself. This routine can take many forms but since the majority of problems appear to be in SWTP systems we are simply going to use the INCHNE vector in 'SBUG-E' itself:

```
3E72   AD    :
3E73   9F    :  ( JSR [$F804] )
3E74   F8    :
3E75   04    :

3E76   84    :
3E77   7F    :  ( ANDA #$7F ... STRIP PARITY )

3E78   39    :  ( RTS         )
```

If your system monitor does not have an INCHNE routine that you can use you can construct one in the 14 bytes available between $3E72 and $3E7F as indicated in the following example:

```
3E72   BD    :
3E73   CD    :  (JSR $CD4E ... 'STAT' IN FLEX)
3E74   4E    :

3E75   27    :
3E76   FB    :  (BEQ $3E72 ... LOOP IF NO KEY PENDING)

3E77   B6    :  (LDA DATAPORT  ... IN THIS CASE AN MC6850 ACIA AT $E004      )
3E78   E0    :  (Note:  This could have just as easily been a PIA at $E040 by)
3E79   05    :  (       using: B6, E0, 40.                                   )

3E7A   84    :
3E7B   7F    :  ( ANDA #$7F ... STRIP PARITY )

3E7C   39    :  ( RTS         )
```

Patch 2

This patch fixes the problem with ENDPROC END not adjusting the stack properly if local variables have been used in the procedure.

```
1A74   17
1A75   23
1A76   F4
1A77   27
1A78   08
1A79   8D
1A7A   74

3E6B   DC
3E6C   12
3E6D   D3
3E6E   10
3E6F   39
```

Patch 3

This final patch changes the version number to 3.04.

```
005A   35
```

PL/9 COMPILER, VERSION 3.05 BUG FIXES as at 1 JANUARY 1983

The following patches cure all known problems with the above versions of the compiler and upgrade it to the current version; 3.06.

If you have versions 3.01, 3.02, 3.03 or 3.04 you must incorporate all of the modifications outlined on the previous 'bug-patches' letters (which will bring your copy of PL/9 up to version 3.05) BEFORE incorporating these patches.

If you are aware of any other problems we would be most obliged if you would complete the enclosed 'BUG REPORT' and return it to us. We will endeavor to fix any reported bug that we can verify within three weeks.

To implement the patches follow this procedure:

1. Use the FLEX 'GET' command to load the PL/9 object code into memory:

   +++GET,0.PL9.CMD <RETURN>

2. Use the FLEX 'RENAME' command to change the name of the old PL9.CMD as follows:

   +++RENAME,0.PL9.CMD,0.PL9-305.CMD <RETURN>

3. Use your system monitor memory examine and change command to implement the patches by altering the contents of the various memory locations as indicated.

4. Once all of the patches have been made (make sure you get it right!) save the binary out to disk using the FLEX 'SAVE' command as follows:

   +++SAVE,0.PL9.CMD,0000,3FFC,0000 <RETURN>

NOTE TO NON-ASSEMBLY LANGUAGE PROGRAMMERS

If you do not feel competent to perform these patches (not everybody is comfortable at machine code level) you are welcome to return your original disk to us and we will perform them for you free of charge. We will turn the disk around within five working days.

The original disk must be returned via REGISTERED post and insured for the full purchase price of the product. We will not accept any responsibility for disks that fail to reach us.

CUSTOMERS OUTSIDE OF THE U.K. PLEASE NOTE

You must affix a customs declaration to the parcel/envelope containing the disk. The description of contents should state:

GOODS OF U.K. ORIGIN BEING RETURNED FOR WARRANTY REPAIR.

If you fail to do this we have to process additional paperwork to clear the disk through customs and as a duty exempt import. This process usually takes two to three weeks!

SOFTWARE BUG REPORT

PRODUCT _____ REVISION _____ SERIAL NUMBER _____

NAME OF DEALER _____

DATE _____

NAME _____

ADDRESS _____

_____

_____

_____

DESCRIBE YOUR SYSTEM HARDWARE AND ANY MODIFICATIONS YOU MAY HAVE MADE.

_____

_____

_____

_____

_____

_____

_____

Describe, in detail, the problem you are having. Include examples which we can run that clearly and REPEATEDLY demonstates the problem. The simpler the example the quicker we will be able to locate the cause of it.

Attach extra sheets or send us a SS/SD disk with upper case file names.

_____

_____

_____

_____

_____

_____

_____

_____

SEND THIS FORM AND ANY ASSOCIATED DOCUMENTATION TO:

Windrush Micro Systems, Worstead Labs., N. Walsham, Norfolk, NR28 9SA, England

TROUBLE WITH THE 'INCHNE' VECTOR?

We have encountered a lot of problems with our use of this FLEX vector located at $D3E5. It is supposed to point to the 'INPUT CHARACTER, NEVER ECHO ROUTINE', which, like 'INCH' is supposed to strip parity.

One of the enclosed patches (Patch 1) does not need to be performed if your version of FLEX has the INCHNE vector at $D3E5 implemented correctly. Most of the problems with our software involve SWTP versions of FLEX. Either this vector has not been implemented at all or, if it is implemented, it does not strip the parity bit out of the incoming code.

Either of these situations will result in PL/9, SETPL9, MACE, XMACE, SETMACE, and SETXMACE either locking up or crashing the system.

The root cause of the problem appears to be a revision in the FLEX I/O routine vector table that was made around 1980 by TSC. Earlier versions of FLEX started the vector table at $D3E7 and did not have an INCHNE vector whilst the current versions start the vector table at $D3E5 and have the INCHNE vector at this location. In most of the SWTP versions we have encountered problems with $D3E5 points to the MC6850 ACIA location.

The best solution to the problem is to upgrade your version of FLEX to the latest version which has the INCHNE vector at $D3E5 installed correctly. Alternatively you can build an INCHNE routine into PL/9 as described in Patch 8.

Since SETPL9 also uses the INCHNE routine (it is part of the 'IOSUBS.LIB' file) it must be recompiled once you have patched PL/9 itself. If you call PL/9 from the FLEX command line you will be able to recompile the SETPL9 program AFTER you have modified the 'GETCHAR_NOECHO' routine in the IOSUBS library as required by your system configuration. The FLEX call looks like this:

```
+++PL9,0.SETPL9.PL9,0=0.SETPL9.CMD <RETURN>
```

If your system uses 'SBUG-E', 'GMX-BUG', 'CHIEFBUG' or 'GT-BUG' or any other system monitor that has the following vector table located at $F800 the modifications required will be quite simple.

```
$F800   RESET     COLD START
$F802   CNTRL     WARM START
$F804   INCHNE    INPUT CHARACTER, NEVER ECHO
$F806   INCHAE    INPUT CHARACTER, ALWAYS ECHO
$F808   INCHECK   TEST FOR INCOMING CHARACTER (<> 0 IF CHAR IS WAITING)
$F80A   OUTCH     OUTPUT A CHARACTER
$F80C   PDATA     OUTPUT STRING POINTED TO BY 'X' ($04 TERMINATES)
$F80E   CRLF      OUTPUT CR-LF STRING
$F810   PSTRNG    OUTPUT CR-LF FOLLOWED BY STRING POINTED TO BY 'X' ($04 TERM)
$F812   LRA       LOAD REAL ADDRESS (USED MAINLY BY DMA DISK CONTROLLERS)

$F814   JMP CONTROL  (not present in some monitors)
```

In the above circumstances the 'GETCHAR_NOECHO' routine in IOSUBS can be modified as follows:

```
PROCEDURE GETCHAR_NOECHO;
   GEN $AD,$9F,$F8,$04;        /*  JSR [$F804]                  */
   GEN $84,$7F;                /*  ANDA #$7F   (STRIPS PARITY)  */
ENDPROC ACCA;
```

If you are using another system monitor you can construct an INCHNE routine as follows:

```
PROCEDURE GETCHAR_NOECHO;
   REPEAT;
      CALL $CD4E;              /*  FLEX 'STAT'                          */
   UNTIL CCR AND $04 = 0;      /*  THERE IS A KEY WAITING!              */

   GEN $B6,$E0,$05;            /*  LDA $E005  (READ ACIA DATA PORT)     */
   GEN $84,$7F;                /*  ANDA #$7F   (STRIP PARITY)           */
ENDPROC ACCA;
```

In the above example we are using the FLEX 'STAT' routine to test the status of the system console keyboard. Once we have determined that a key has been pressed we then simply read the ACIA data port, strip the parity bit, and return the code in the 'A' accumulator.

We could just as easily read the data from a PIA data port or whatever I/O device you have in the system.

Patch 1

This patch does not need to be performed in the vast majority of systems

This patch is only required in systems that use a version of FLEX that does not have the INCHNE vector at $D3E5 installed or in versions that have the vector installed but the routine does not strip the parity bit out of the incoming code.

Most of the systems we have had problems with in this area are early SWTP 6809 systems. Since these systems all have 'SBUG-E' this patch is organized around this system monitor. The basic principle of the patch is to 're-aim' four calls in PL/9 that use the FLEX 'INCHNE' vector at $D3E5 to a vector at $3E70. The vector at $3E70 (which is installed by this patch) points to a routine at $3E72:

```
02CD   3E    (CURRENTLY D3)
02CE   70    (CURRENTLY E5)

3A29   3E    (CURRENTLY D3)
3A2A   70    (CURRENTLY E5)

3A32   3E    (CURRENTLY D3)
3A33   70    (CURRENTLY E5)

3F57   3E    (CURRENTLY D3)
3F58   70    (CURRENTLY E5)

3E70   3E    :
3E71   72    :  (NEW VECTOR IN LIEU OF $D3E5)
```

Next we instal the INCHNE routine itself. This routine can take many forms but since the majority of problems appear to be in SWTP systems we are simply going to use the INCHNE vector in 'SBUG-E' itself:

```
3E72   AD    :
3E73   9F    :  ( JSR [$F804] )
3E74   F8    :
3E75   04    :

3E76   84    :
3E77   7F    :  ( ANDA #$7F ... STRIP PARITY )

3E78   39    :  ( RTS         )
```

If your system monitor does not have an INCHNE routine that you can use you can construct one in the 14 bytes available between $3E72 and $3E7F as indicated in the following example:

```
3E72   BD    :
3E73   CD    :  (JSR $CD4E ... 'STAT' IN FLEX)
3E74   4E    :

3E75   27    :
3E76   FB    :  (BEQ $3E72 ... LOOP IF NO KEY PENDING)

3E77   B6    :  (LDA DATAPORT  ... IN THIS CASE AN MC6850 ACIA AT $E004      )
3E78   E0    :  (Note:  This could have just as easily been a PIA at $E040 by)
3E79   05    :  (       using: B6, E0, 40.                                   )

3E7A   84    :
3E7B   7F    :  ( ANDA #$7F ... STRIP PARITY )

3E7C   39    :  ( RTS         )
```

Patch 2

This patch fixes problems associated with comparisions of unsigned integers.

```
1FFD   16
1FFE   14
1FFF   D8

349E 59                  +-------------> 34D0 33
349F 4F                  |               34D1 20
34A0 55                  |               34D2 2A
34A1 27                  |               34D3 2A
34A2 52                  |               34D4 2A
34A3 45                  |               34D5 0D
34A4 20                  |               34D6 0A
34A5 4F                  |               34D7 04
34A6 4E                  |               34D8 0D
34A7 20                  |               34D9 1C
34A8 59                  |               34DA 27
34A9 4F                  |               34DB 09
34AA 55                  |               34DC 30
34AB 52                  |               34DD 8C
34AC 20                  |               34DE 0C
34AD 4F                  |               34DF 17
34AE 57                  |               34E0 EE
34AF 4E                  |               34E1 83
34B0 20                  |               34E2 16
34B1 4E                  |               34E3 EE
34B2 4F                  |               34E4 36
34B3 57                  |               34E5 17
34B4 21                  |               34E6 EE
34B5 0D                  |               34E7 3F
34B6 0A                  |               34E8 16
34B7 2A                  |               34E9 EB
34B8 2A                  |               34EA 15
34B9 2A                  |               34EB 02
34BA 20                  |               34EC 34
34BB 52                  |               34ED 06
34BC 45                  |               34EE 02
34BD 2D                  |               34EF EC
34BE 45                  |               34F0 62
34BF 4E                  |               34F1 02
34C0 54                  |               34F2 A3
34C1 45                  |               34F3 E4
34C2 52                  |               34F4 02
34C3 20                  |               34F5 32
34C4 50                  |               34F6 64
34C5 4C                  |               34F7 00
34C6 2F                  |
34C7 39                  |
34C8 20                  |               (most of this 'code' is used to shorten
34C9 41                  |                the banner printed by PL/9 when
34CA 54                  |                entering the system monitor to make
34CB 20                  |                room for the additional program code
34CC 24                  |                required for this bug-fix which starts
34CD 30                  |                at $34D8)
34CE 30                  |
34CF 30 -----------------+
```

Patch 3

This final patch changes the version number to 3.06.

```
005A   36
```

PL/9 COMPILER, VERSION 3.06 BUG FIXES as at 20 FEBRUARY 1983

The following patches cure all known problems with the above version of the compiler and upgrade it to the current version; 3.07.

If you have versions 3.01, 3.02, 3.03, 3.04 or 3.04 you must incorporate all of the modifications outlined on the previous 'bug-patches' letters (which will bring your copy of PL/9 up to version 3.06) BEFORE incorporating these patches.

If you are aware of any other problems we would be most obliged if you would complete the enclosed 'BUG REPORT' and return it to us. We will endeavor to fix any reported bug that we can verify within three weeks.

To implement the patches follow this procedure:

1. Use the FLEX 'GET' command to load the PL/9 object code into memory:

   +++GET,0.PL9.CMD <RETURN>

2. Use the FLEX 'RENAME' command to change the name of the old PL9.CMD as follows:

   +++RENAME,0.PL9.CMD,0.PL9-306.CMD <RETURN>

3. Use your system monitor memory examine and change command to implement the patches by altering the contents of the various memory locations as indicated.

4. Once all of the patches have been made (make sure you get it right!) save the binary out to disk using the FLEX 'SAVE' command as follows:

   +++SAVE,0.PL9.CMD,0000,3FFC,0000 <RETURN>

NOTE TO NON-ASSEMBLY LANGUAGE PROGRAMMERS

If you do not feel competent to perform these patches (not everybody is comfortable at machine code level) you are welcome to return your original disk to us and we will perform them for you free of charge. We will turn the disk around within five working days.

The original disk must be returned via REGISTERED post and insured for the full purchase price of the product. We will not accept any responsibility for disks that fail to reach us.

CUSTOMERS OUTSIDE OF THE U.K. PLEASE NOTE

You must affix a customs declaration to the parcel/envelope containing the disk. The description of contents should state:

GOODS OF U.K. ORIGIN BEING RETURNED FOR WARRANTY REPAIR.

If you fail to do this we have to process additional paperwork to clear the disk through customs and as a duty exempt import. This process usually takes two to three weeks!

Patch 1

This patch fixes a bug in the compiler routine which scans the source line for a number or a constant. The symtoms of this bug are generally as obscure as the above description.

```
0D76  26
0D77  1F

0D92  32
0D93  61
0D94  10
0D95  9E
0D96  6B
0D97  16
0D98  27
0D99  5E

34F8  A6
34F9  A4
34FA  81
34FB  27
34FC  10
34FD  26
34FE  D8
34FF  9F
3500  31
3501  21
3502  16
3503  D8
3504  95
```

Patch 2

This patch prevents PL/9 from appending a control-Z to the output file when (S)aving or (W)riting.

```
35A3  20
35A4  31
```

Patch 3

This final patch changes the version number to 3.07.

```
005A  37
```

PL/9 COMPILER, VERSION 3.07 BUG FIXES as at 15 APRIL 1984

The following patches cure all known problems with the above version of the compiler and upgrade it to the current version; 3.08.

If you have versions 3.01, 3.02, 3.03, 3.04, 3.05 or 3.06 you must incorporate all of the modifications outlined on the previous 'bug-patches' letters (which will bring your copy of PL/9 up to version 3.07) BEFORE incorporating these patches.

If you are aware of any other problems we would be most obliged if you would complete the enclosed 'BUG REPORT' and return it to us. We will endeavor to fix any reported bug that we can verify within three weeks.

To implement the patches follow this procedure:

1. Use the FLEX 'GET' command to load the PL/9 object code into memory:

   +++GET,0.PL9.CMD <RETURN>

2. Use the FLEX 'RENAME' command to change the name of the old PL9.CMD as follows:

   +++RENAME,0.PL9.CMD,0.PL9-307.CMD <RETURN>

3. Use your system monitor memory examine and change command to implement the patches by altering the contents of the various memory locations as indicated.

4. Once all of the patches have been made (make sure you get it right!) save the binary out to disk using the FLEX 'SAVE' command as follows:

   +++SAVE,0.PL9.CMD,0000,3FFC,0000 <RETURN>

NOTE TO NON-ASSEMBLY LANGUAGE PROGRAMMERS

If you do not feel competent to perform these patches (not everybody is comfortable at machine code level) you are welcome to return your original disk to us and we will perform them for you free of charge. We will turn the disk around within five working days.

The original disk must be returned via REGISTERED post and insured for the full purchase price of the product. We will not accept any responsibility for disks that fail to reach us.

CUSTOMERS OUTSIDE OF THE U.K. PLEASE NOTE

You must affix a customs declaration to the parcel/envelope containing the disk. The description of contents should state:

GOODS OF U.K. ORIGIN BEING RETURNED FOR WARRANTY REPAIR

If you fail to do this we have to process additional paperwork to clear the disk through customs and as a duty exempt import. This process usually takes two to three weeks!

IF YOU DEVELOP YOUR PROGRAMS IN A SEPERATE EDITOR

If you prepare text for PL/9 in an editor other than the resident editor you must ensure that the line length never exceeds 127 characters. If you exceed this line length the compiler will crash as only 127 characters have been reserved on the stack for the line buffer during program compilation. The resident editor it will not allow you to enter a line longer than 127 characters and hence the reason why the compiler does not check line length.

Patch 1

This patch fixes a bug reported by Brutech of Holland. If a forward branch fixup (indicated on the A:C,T listing by a parenthesised entry) occurs at the point that a block of binary is about to be written to disc, the data in the binary file may be incorrect. The patch is fairly lengthy - our apologies!

```
3BB8   34            +--------->  3BE2   5D
3BB9   36            |            3BE3   26
3BBA   DC            |            3BE4   06
3BBB   14            |            3BE5   DC
3BBC   93            |            3BE6   14
3BBD   B8            |            3BE7   D3
3BBE   25            |            3BE8   54
3BBF   19            |            3BE9   ED
3BC0   10            |            3BEA   1D
3BC1   83            |            3BEB   6C
3BC2   01            |            3BEC   1F
3BC3   FE            |            3BED   D6
3BC4   24            |            3BEE   74
3BC5   0E            |            3BEF   C1
3BC6   8E            |            3BF0   01
3BC7   43            |            3BF1   26
3BC8   F0            |            3BF2   16
3BC9   30            |            3BF3   E6
3BCA   8B            |            3BF4   1F
3BCB   A6            |            3BF5   27
3BCC   E4            |            3BF6   12
3BCD   A7            |            3BF7   31
3BCE   84            |            3BF8   1D
3BCF   8E            |            3BF9   86
3BD0   45            |            3BFA   02
3BD1   F1            |            3BFB   17
3BD2   20            |            3BFC   00
3BD3   1F            |            3BFD   95
3BD4   17            |            3BFE   CB
3BD5   00            |            3BFF   03
3BD6   CB            |            3C00   17
3BD7   20            |            3C01   00
3BD8   E1            |            3C02   8E
3BD9   8E            |            3C03   5A
3BDA   45            |            3C04   26
3BDB   F1            |            3C05   FA
3BDC   E6            |            3C06   7F
3BDD   1F            |            3C07   45
3BDE   A6            |            3C08   F0
3BDF   E4            |            3C09   35
3BE0   A7            |            3C0A   B6
3BE1   85  ---->----+
```

Patch 2

This final patch changes the version number to 3.08.

```
005A   38
```

PL/9 COMPILER, VERSION 3.08 BUG FIXES as at 15 MAY 1984

The following patches cure all known problems with the above version of the compiler and upgrade it to the current version; 3.09.

If you have versions 3.01, 3.02, 3.03, 3.04, 3.05, 3.06 or 3.07 you must incorporate all of the modifications outlined on the previous 'bug-patches' letters (which will bring your copy of PL/9 up to version 3.08) BEFORE incorporating these patches.

If you are aware of any other problems we would be most obliged if you would complete the 'BUG REPORT' form supplied with your copy of PL/9 and return it to us. We will endeavor to fix any reported bug that we can verify within three weeks.

To implement the patches follow this procedure:

1. Use the FLEX 'GET' command to load the PL/9 object code into memory:

   +++GET,0.PL9.CMD <RETURN>

2. Use the FLEX 'RENAME' command to change the name of the old PL9.CMD as follows:

   +++RENAME,0.PL9.CMD,0.PL9-308.CMD <RETURN>

3. Use your system monitor memory examine and change command to implement the patches by altering the contents of the various memory locations as indicated.

4. Once all of the patches have been made (make sure you get it right!) save the binary out to disk using the FLEX 'SAVE' command as follows:

   +++SAVE,0.PL9.CMD,0000,3FFC,0000 <RETURN>

NOTE TO NON-ASSEMBLY LANGUAGE PROGRAMMERS

If you do not feel competent to perform these patches (not everybody is comfortable at machine code level) you are welcome to return your original disk to us and we will perform them for you free of charge. We will turn the disk around within five working days.

The original disk must be returned via REGISTERED post and insured for the full purchase price of the product. We will not accept any responsibility for disks that fail to reach us.

CUSTOMERS OUTSIDE OF THE U.K. PLEASE NOTE

You must affix a customs declaration to the parcel/envelope containing the disk. The description of contents should state:

GOODS OF U.K. ORIGIN BEING RETURNED FOR WARRANTY REPAIR

If you fail to do this we have to process additional paperwork to clear the disk through customs and as a duty exempt import. This process usually takes two to three weeks!

Patch 1

This patch changes the version number of PL/9 to 3.09. There are no actual bugs in the compiler itself. The bug lies in the TRACER but we keep track of revision levels in any part of the compiler in the compilers version number, hence the need for this patch. To implement this patch follow the instructions on the first sheet.

005A 39

Patch 2

NOTE: This patch applies to the file 'PL9_TD.CMD' NOT the file 'PL9.CMD'.

This patch fixes the problem of breakpoints not working when the R command is used.

C23E 0D
C23F BE
C240 27
C241 0B
C242 0D
C243 BC
C244 27
C245 42
C246 DC
C247 C2
C248 26
C249 03

C167 31

To implement the above patches follow this procedure:

1. Use the FLEX 'GET' command to load the PL/9 TRACER object code into memory:

   +++GET,0.PL9_TD.CMD <RETURN>

2. Use the FLEX 'RENAME' command to change the name of the old PL9_TD.CMD as follows:

   +++RENAME,0.PL9_TD.CMD,0.PL9_TD8.CMD <RETURN>

3. Use your system monitor memory examine and change command to implement the patches by altering the contents of the various memory locations as indicated.

4. Once all of the patches have been made (make sure you get it right!) save the binary out to disk using the FLEX 'SAVE.LOW' command as follows:

   +++SAVE.LOW,0.PL9_TD.CMD,C100,C556,CD03 <RETURN>

COPYRIGHT NOTICE

SERIAL NUMBER NOTICE

This product has been assigned a unique serial number at the time of manufacture. This serial number is encrypted into the body of the product. This product is therefore traceable to the original purchaser in the event of plagiarized copies being discovered.

This product is sold on the basis of being used on a SINGLE microcomputer system by a SINGLE user.

We shall consider it to be an attempt to criminally plagiarize us if duplicate copies of this manual or the accompanying disk are made available for use by other parties, or on other microcomputers. This consideration also applies to, but is not limited to, duplicate copies being produced for use within the original purchasers organisation, establishment, or home for anything other than archival purposes.

WARNING

We at Windrush Micro Systems Limited and the author consider the recognition we receive as a result of the sale of our programs and manuals to be of vital importance in remaining in business.

Unless written arrangements to the contrary have been made between authorized agents of Windrush Micro Systems Limited and the purchaser of this manual and the accompanying computer program we shall consider it to be an attempt to criminally plagiarize us if our company name, the program name, or the authors name is altered, changed or removed on or from any of the materials purchased from us regardless of the means by which accomplished. This consideration shall include, but not be limited to, the re-writing of this manual, or its reproduction for distribution under another company, program or trade name, or any like modification of the accompanying computer program.

WARRANTY NOTICE

Although every effort has been made to insure the accuracy of this material, it is sold AS IS and without warranty. No claim as to the suitability or workability of this material for any particular application or on any particular computer is made. This statement is in lieu of any other statement whether expressed or implied.

Windrush Micro Systems, Worstead Labs, N. Walsham, Norfolk, NR28 9SA, England

W A R N I N G

We at Windrush Micro Systems Limited are in the business of selling and supporting software NOT rebuilding disks crashed by carelessness.

The first thing you should do with the disk(s) we supply is to make a working copy of it on a system known to be performing properly. Once this has been done the original disks should be stored in a safe place.

You should NEVER use the original disk in a system that you suspect has hardware problems ... it can prove to be very expensive if you do!

If you accidentally wipe out the disk we supply we will REQUIRE that you return the disk(s) supplied with this product; disks other than those bearing our labels WILL NOT BE ACCEPTED. The disk(s) MUST be returned via REGISTERED mail with a cheque or money order for $25.00 (15.00 Pounds Sterling + VAT in U.K.). This is to cover our costs in rebuilding the disk and the return postage. This will prove to be a time consuming exercise at the best of times.

THERE ARE NO EXCEPTIONS TO THIS POLICY

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                     *
*   DON'T EVER USE THE DISK WE SUPPLY FOR ANYTHING OTHER THAN         *
*   MAKING A WORKING COPY OF ITSELF ON A KNOWN GOOD MACHINE!!         *
*                                                                     *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

A N O T H E R W A R N I N G

Whenever you wish to return a disk to us for a product enhancement upgrade you MUST return the original disk we supplied with this product via REGISTERED mail insured for the full value of the new product. If the disk fails to reach us you then can make a claim and recover the cost of the product and purchase a new one. If you send the disk to us by any other means and it fails to reach us you will have to purchase the product all over again. We cannot, and will not, be held responsible for losses outside of our control.

We will not accept any reason for your inability to return the original disk. Therefore take whatever steps you consider necessary to protect the disk from theft, flood, fire, your cat/dog/budgie, mother-in-law, etc. The only way we can keep the cost of our upgrade service down to $25.00 is to eliminate administrative overheads. We do this by ONLY upgrading original disks.

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                     *
*          THERE ARE NO EXCEPTIONS TO THIS POLICY EITHER             *
*                                                                     *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

If you purchase this product through a dealer and he has had to copy it from our original disk onto another disk for compatibility with your hardware ENSURE that you get our original disk from the dealer.

We apologize for the harshness of the wording in these warnings. From our past experience in these areas it seems that abruptness is the only way we can drive home the fact that we must follow rigid procedures that prevent unauthorized copies of disks from being produced. In this way we can provide better after-sales support and an upgrade service that is unique in the industry.