ICL

International Computers Limited

User notice

PUBLICATION (NOTICE NO.) 31/1/73

4322 PLAN REFERENCE MANUAL (1)

File one copy of this notice with each of the publications indicated.

This notice corrects some errors in Appendix 4 of the new edition, and reproduces some of the information in late User Notices of the previous edition.

CORRECTIONS TO APPENDIX 4

Page 27 (a) Mode 7 should read:

Move read/write head to given bucket number.

(b) The second note 2 should be renumbered 3, and subsequent notes should be renumbered 4, 5 and 6.

Page 29 (a) Note 3 should begin:

Additive mode #4000 may be ...

(b) Note 10 should begin:

For close file mode #1000 only ...

Page 33 (a) Word 1, Bit 0 = 0, the third sentence should read:

Modes #1100, #1400, #1500:
Serial number of storage unit

(b) Note 2 should begin:

Additive mode #6000 may be ...

[redacted]

Issued by Technical Publications Service,

Page 37 (a) Note 4 should read:

For close modes #1000, #1400 only Words 0 to 3 are required in the control area.

(b) Note 6 should begin:

For modes #100, #200 on successful...

XPLX/29 AND XPLZ/28A

There is a restriction on the above compilers such that the first segment of a program presented to the compiler must be in source format.

DISC COSY PROGRAMS - #XPMY, #XPMZ AND #XPMX

The following rules should be observed if the user wishes to specify a change of file generation number and a check on the cartridge serial number for the output and/or workfiles used by these programs:

for example, OUTE (FILENAME(FGN1 = FGN2),CSN)

Output file

Two parameters should be used:

for example, OUTE (FILENAME(FGN1),CSN)

where FGN1 = the file generation number currently existing on the file.

followed by

RENE (FILENAME(FGN2))

where FGN2 = the new file generation number to be written to the output file.

Work file (does not apply to #XPMX)

When using this facility for the OUTW parameter, care should be taken since if more than one file of the same specified filename and file generation number should be on-line, the wrong file may be opened and the file generation number updated before the CSN is checked.

PLAN 3 AND 4 DISC COMPILERS #XPLF/28, #XPLM/28, #XPLT/7D, #XPLX/29 AND #XPLZ/28A

The following rules should be observed if the user wishes to specify a change of file generation number and a check on the cartridge serial number for the output and/or work files used by these compilers. An example of an incorrect format would be:

OUT(*filename*(*fgn*1=*fgn2*),*csn*)

Output file

Two parameters of the following form should be used:

OUTE(*filename*(*fgn*1),*csn*)

where *fgn*1 = the file generation number currently existing on the file.

followed by

REN (*filename*(*fgn*2))

where *fgn*2 = the new file generation number to be written to the output file.

Work file (applies to #XPLM/28 and #XPLX/29 only)

When using this facility for the WORK parameter, care should be taken, since, if more than one file of the same specified filename and fgn should be on-line, the wrong file may be opened and the fgn updated before the csn is checked.

AMENDMENTS TO CHAPTERS 10 AND 11

Chapter 10, page 27

The paragraph preceding the heading *6 Amendment type* should read:

> The #COPY parameter can be used in a dummy amendment, that is one from which the amendment type parameter is omitted. If a dump of the whole program is required the operand field of the #IDENTITY parameter should be punched with the subfile name only. If a dump of only one segment is required, the operand of the #IDENTITY parameter should be punched with the subfile name, a solidus and the segment name.

Chapter 10 page 40

Add to the list under the heading *Operating instructions for #XPMJ*:

7 It is not possible to make a copy of a subfile to a slow peripheral and to delete that subfile in the same run of #XPMJ. If this is required, two #XPMJ runs must be made, the first to copy the subfile and the second to delete it.

Chapter 11, page 33, line 14 should read:

(d) the work file (amend-in-situ and copy-and-amend modes) or the output file (copy-and-amend mode only).

Chapter 11, page 34. The following should be added to item 4 of the list:

When the OUTW parameter is omitted, a scratch file is opened on the same cartridge or unit as the file used for output (copy-and-amend mode) or for input/output (amend-in-situ mode).

PLAN INSTRUCTION MPR

1 The last line of the first paragraph under the heading 'Execution' on page 147 of Chapter 4 should read:

"The result is thus a rounded single-length product in X. The sign bit of X* is always made zero."

2 Note 2 on page 147 of Chapter 4 should read:

"The contents of X and of N(M) may be regarded as integers, fractions, or mixed numbers. However, this instruction should not be used if the contents of both X and N(M) are integers as the product will normally only occur in X*, which is destroyed by the rounding process. When fractions or mixed numbers are involved, the number of bits after the binary point in the double-length product is equal to the sum of the numbers of bits after the binary points in the factors."

THE DIRECTIVE #MACRO

The second paragraph under the heading *Format* on page 21 of Chapter 6 should read:

"The name of the macro-instruction may consist of up to five alphanumeric characters, of which the first must be alphabetic. This name must be unique in the program, except as explained below. It cannot be the same as the name of any PLAN pseudo-operation (see Chapter 5) and it can only be the same as the name of a standard PLAN function if the number of accumulators is greater than the number in the standard function.

The name can be identical to that of a previous user-defined macro provided that the number of accumulators is different."

The following note should be added at the end of page 23 of Chapter 6:

10 A #MACRO directive may, but need not necessarily, be used in a steering segment.

ICL

International
Computers
Limited

User
notice

PUBLICATION (NOTICE NO.)

14/2/73

4322 PLAN REFERENCE (2)

File one copy of this notice with each of the publications indicated.

PLAN compilers

All the PLAN compilers will flag unlabelled data symbols as 'N' if these have been previously defined by #DEFINE or #SET.

This situation can be avoided by including a label, for example

```
#DEFINE             SYMBOL = 100
#LOWER
PRESET              1
                    / BUFFER
                    2
PRESET3             SYMBOL
```

Issued by Technical Publications Service, 30 - 31 Friar Street, Reading, Berkshire. RG1 1JP

ICL

International Computers Limited

User notice

PUBLICATION (NOTICE NO.) 6/6/73

4322 PLAN REFERENCE MANUAL (3)

File one copy of this notice with each of the publications indicated.

Chapter 10, Page 27

The following paragraph should be inserted before the heading 6 Amendment type:

It is not possible to make a copy of a subfile to a slow peripheral and to delete that subfile in the same run of #XPMJ. If this is required, two #XPMJ runs must be made, the first to copy the subfile and the second to delete it.

Chapter 11, Page 19

The paragraph under the heading OUTW(#XPMZ only) should read:

This parameter is optional. It defines the disc file to be opened as the work file. In a #XPMX update run, or in a #XPMZ update run where this parameter is omitted, a scratch file will be opened on the same cartridge or unit as the file used for output (copy-and-amend mode) or input/output (amend-in-situ mode).

Chapter 11, Page 34

The message for Exception Condition 4 has been printed incorrectly; it should include three space characters before the word WORK, for example,

```
O#XPM-; HALTED:- ∇∇∇WORK FILE NOT FOUND
```

Paragraph 4(c) should read:

(c) a scratch file cannot be opened on the cartridge or unit containing the file used for output (copy-and-amend mode) or for input/output (amend-in-situ mode).

Chapter 11, Page 32

The error message PREVIOUS RUN ABANDONED and the paragraph that explains it should be deleted from the list of Line Printer Error Messages. The following should be inserted as Exception Condition 13:

13. O#XPM-; HALTED :- PREVIOUS RUN ABANDONED

Acopy-and-amend or amend-in-situ run has been attempted but the input file has been corrupted by abandoning a previous amend-in-situ run. The corrupted file should be recreated from the previous dump and brought up-to-date either by using tag dumps or by carrying out any outstanding amendments.

ICL

International
Computers
Limited

User
notice

PUBLICATION (NOTICE NO.)

12/9/73

4322 PLAN REFERENCE MANUAL (4)

File one copy of this notice with each of the publications indicated.

ASKUNIT Overlay Interrogation Routine

The specification of ASKUNIT previously found in User Notice 42 to the first edition of the *Plan Reference Manual* (TP 4004) has now been transferred to the *Compiling Systems Manual* (TP 4241).

Issued by Software Distribution Department,

ICL

International Computers Limited

User notice

PUBLICATION (NOTICE NO.) 27/2/74

4322 PLAN REFERENCE (5)

File one copy of this notice with each of the publications indicated.

PLAN 3 COMPILERS - OPERATING INSTRUCTIONS

On page 42 of Chapter 7, lines 12 and 13 (operating instruction 9) should read:

9 Except where a successor program has been specified by compiler parameters, either delete #XPL?, or repeat steps 1, 3 and 6 to compile further programs. For #XPLX, #XPLZ, #XPLM and #XPLF step 2 must also be repeated.

ICL

International
Computers
Limited

User
notice

PUBLICATION (NOTICE NO.) 1/5/74

4322 PLAN REFERENCE (6)

File one copy of this notice with each of the publications indicated.

SOFTWARE NOTICE CORRECTIONS

#XPMY/4D

#XPMY/4D will be available on Release MLT 28. This will correct the error described in Software Notices 1900 PLAN 65/110.

#XPMZ

#XPMZ/4F will be available on Release MLT 28. This version will correct the errors described in Software Notices 1900 PLAN 65/111, 85/142.

ssued by Software Distribution Department,

ICL

International Computers Limited

User notice

PUBLICATION (NOTICE NO.)

18/9/74

4322

PLAN REFERENCE (7)

File one copy of this notice with each of the publications indicated.

AMENDMENTS TO MANUAL

Chapter 4 Page 149

Line 6 of Execution should read.

"word addresses and the contents of B2 to B8 are cleared; when in extended data mode, the least significant 22"

Chapter 6 Page 5

The heading for this page should be '#'.

Chapter 8 Page 27 final line

This line should be omitted.

Chapter 8 Page 28 line 23

Alter to read as follows:

"if compilation has been abandoned or if an error was found in the AMPL amendments for this program".

Chapter 8 Page 29 line 17

Alter to read as follows:

"has to incorrect format. The rest of the amendments for this segment will be ignored, and at the end of compilation the message 'SERV OK' or 'SERV ER' will be displayed".

Chapter 8 Page 29 line 19

Alter to read as follows:-

"continues. The rest of the amendments for this segment will be ignored, and at the end of compilation the message 'SERV OK' or 'SERV ER' will be displayed".

Appendix 7 Page 71 Section 'Output'

The second sentence should read "A binary program on disc can be obtained by a separate run using #XPCK or by parameter calling #XPCK at the end of the run."

Appendix 7 Page 91 Section 'Output'

The second sentence should read "A binary program can be obtained by a separate run using #XPCK."

Appendix 7 Page 91 Section "Switch Settings"

In addition to the two switches add

"11 if it is required to suppress the command issuing facilities."

Appendix 7 Page 91 Section "Input Parameters"

In addition to the parameters listed add

"ABN, ABNE, ABNF"

Chapter 6 Page 2

The following section should replace the first paragraph under the title Steering Segements.

STEERING SEGMENTS

The descriptions of some PLAN directive statements specify that the directive concerned must be in the steering segment. A steering segment is required in the following circumstances:

1 In any Plan 3/4 program which is overlaid and uses one of the standard overlay packages

2 In any Plan 4 program in order to specify the address and branch mode in which the program is to operate

A non-overlay Plan 3 program which is being compiled by a Plan 4 compiler does not require a steering segment. A steering segment, where one is required, must be the first segment of a program, and may not be presented as a semi-compiled segment. It may only contain permitted PLAN directives i.e. directives which in their individual descriptions are stated as being appropriate to a steering segment. A summary of which directives may be used follows:

Major Directives	Must be present in a steering segment	If present must be in a steering segment	May not be present in a steering segment	Optionally present in a steering segment	Not applicable	Minor Directives	Must be present in a steering segment	If present must be in a steering segment	May not be present in a steering segment	Optionally present in a steering segment	Not applicable
CMODE				√		CUE			√		
ELASTIC				√		ENTRY			√		
END	√					MONITOR			√		
FINISH					√	COMPLETE			√3		
LOWER				√1		#comment				√	
MACRO				√		DEFINE			√		
OMIT				√		HMODE		√			
OVERLAY		√				LIBRARY				√	
PERIPHERAL				√		ORDER		√			
PERMANENT				√		OUST					√
PLOWER			√			PAGE				√	
PMODE		√				SET			√		
PUPPER			√			SWITCH				√	
PROGRAM	√2					ERRORSEG				√	
STOP					√						
UPPER				√1							

Notes:

1 May only be used if the block defined is marked as OVERCOMMON

2 May only be used to introduce the segment

3 Must not be used if standard run-time overlay package is required

If illegal directives are included in the Steering Segment they will not be flagged as errors and the action of the compiler is unpredictable. In particular, steering segments should not produce any object code.

ICL

International Computers Limited

User notice

PUBLICATION (NOTICE NO.)

13/11/74

4322 PLAN REFERENCE (8)

File one copy of this notice with each of the publications indicated.

#XPLF/30B, #XPLZ/30B, #XPLT/8B
PLAN 3 AND 4 DIRECT ACCESS COMPILERS

New versions of these PLAN 3 and 4 direct access compilers are shortly to be issued. Besides correcting a number of errors these versions also provide the interface to enable users to take advantage of both the Mark III 1900 Direct Access Housekeeping Package, and the Dump and Restart Facilities Mark IV. These changes are now described in more detail.

Errors corrected

The following PLAN Software Notice items have been corrected in these new versions.

S/N	*Item*
38	66
42	73
42	74
44	79
58	99
59	102
79	132
79	133
80	135
81	138
82	139
86	145
86	146
91	151
92	152
100	161
100	163

Interface with Mark III 1900 Direct Access Housekeeping package

These versions will provide all the facilities necessary for the user to take advantage of the Mark III 1900 Direct Access Housekeeping package. (For a full description of this package please refer to the 1900 Series manual *Direct Access* (Edition 2, TP 4385).

#CMODE DIRECTIVE

1 A new device code 'MDA' has been defined as an operand to the #CMODE directive. This device code should be used in place of 'DA' for all unit numbers which are to use the DAHK Improved Performance package

For example

```
#CMODE      MDA (1, 3-5)
```

2 Unit numbers in the range 0-63 may now be specified in the list for device code 'MDA' only

SDDEF MACRO FOR AN MDA UNIT

1 A new parameter 'UWA' is defined for the SDDEF macro. If required this is included as the last in the parameter sequence:

```
SDDEF  X  N,L,FDA,LABEL,MODE,PROC,O,L1,LABEL1,UWA
```

The 'UWA' parameter can take the value 'U' or 'V' or be omitted (in which case the preceding comma should also be omitted). The parameter determines how a Unit Work Area is to be allocated for the MDA unit concerned (see *Unit work Areas for MDA units* below)

(a) If it is set to 'U' the allocation of the Unit Work Area is left to the user by means of a subsequent SDUWA macro (see *SDUWA : Allocate/deallocate Unit Work Area : for use with multithreading*, below)

(b) If it is set to 'V' the compiler allocates the Unit Work Area as a 100-word area in Upper Variable. Such an area will be used exclusively for the unit concerned and cannot be shared among units

(c) If it is omitted, the compiler will cause the unit concerned to use the Lower Common Variable area HMTHREAD as the Unit Work Area. The 100-word area HMTHREAD will be shared by all MDA units not having a 'UWA' parameter in their associated SDDEF macro.

If a 'UWA' parameter is given for a non-MDA unit, the compilers flag error 'O'

2 The meaning of the 'L1' parameter has changed. The compiler uses it together with the 'LABEL1' parameter to determine what overflow routines are required. The following parameter combinations are now permitted:

(a)	L1 omitted LABEL1 omitted	No overflow routines required
(b)	L1=1 LABEL1 omitted	Housekeeping 1st level only
(c)	L1 = *bucket* size of the file (in words) or set negative LABEL1 omitted	Housekeeping 1st and 2nd level

(d)	L1=1 or omitted
	LABEL1 = address of users 1st level routine
		User 1st level only

(e)	L1 = *bucket* size of the file (in words) or set negative LABEL1 = address of users 1st level routine	User 1st level and Housekeeping 2nd level

In cases (c) and (e) above, if L1 is positive, the compiler allocates as a contingency bucket buffer the Common Upper Variable area HDMLV. This buffer will be shared by all MDA units requiring it, and its length will be equal to the maximum value specified for an 'L1' parameter plus ten words. If 'L1' is expressed as a negative number, Housekeeping routines dealing with second level overflow are incorporated, but the compiler does not allocate the contingency buffer

(3) If an SDDEF macro has 'D' in the accumulator field (see *Replaced Unit Numbers for MDA Units* below) it will be flagged 'X' and the compilation of the macro is abandoned

(4) If any parameter is omitted from an SDDEF statement, its following comma must still be present unless all subsequent parameters are also omitted

SDIND MACRO FOR AN MDA UNIT

A third parameter 'A3' is defined for the SDIND macro. If required this is included as the last in the parameter sequence:

```
          SDIND  X  A1,A2,A3
```

The new parameter specifies the start address of an area in store to hold the Third Level Index. If it is omitted the preceding comma should also be omitted. A3 must not be modified. A3 may be present even if A1 and/or A2 are omitted, but in this case the two commas should still be present

NEW MACROS FOR MDA UNITS

The specifications of four new macros which generate calls to new Housekeeping routines are given below:

1 SDBFF: ALLOCATE/DEALLOCATE/WRITE AWAY BUFFERS: FOR USE WITH MULTITHREADING

Format: SDBFF X N(M)

where X — is the Unit Number (0 to 63)

N(M) — is the address of a word in either Lower or Upper data store containing the address of a 3 word area which holds pointers to the first home buffer, second home buffer and overflow buffer respectively. A zero in any word of the area means there is no corresponding buffer. N(M) may be an accumulator

SDBFF has three main uses:

(a) To allocate buffers to unit X. On entry to the macro, bit 3 of word 31 of the File Definition Area should be set and the three word area addressed by location N(M) should contain the addresses of the buffers. The macro allocates the buffers and initialises them for use by unit X. Bit 3 of word 31 of the FDA is cleared

(b) To deallocate buffers from unit X. On entry to the macro, bit 3 of word 31 of the FDA should be set and the three word area addressed by location N(M) should contain zeros. The macro writes away to the appropriate file any buffers allocated to unit X which need to be written, and then deallocates the buffers (if any) from Unit X. Bit 3 of word 31 of the FDA is cleared

(c) To reinitialise for use by unit X buffers which are allocated to that unit but which currently contain data from another unit. On entry to the macro bit 3 of word 31 of the FDA should be clear. The macro writes away to the appropriate file any buffers allocated to unit X which need to be written and then reinitialises the buffers for use by unit X. Housekeeping does not refer to location N(M). Bit 3 of word 31 of the FDA is left clear.

Note: The three word area must not be included in words 0 - 15 of the user program

2 SDCON: RESTART SUSPENDED THREAD OR DESTROY THREAD: FOR USE WITH MULTITHREADING

Format: SDCON X S

where X is the Unit number (0 to 63)

S is an optional field. If it is present the routine destroys the thread to which the unit is attached; if it is absent, the thread is restarted. In both cases, the unit number must be associated with a suspended thread.

If the S parameter is absent and work on the thread is restarted, control returns to the user program, either at the last instruction of the calling sequence for the original macro or to the exception routine. If the macro is used to destroy a thread which was not suspended with exception code #04 or #06, control returns to the user program at the last instruction of the SDCON calling sequence.

3 SDNXB: INITIATE TRANSFER INTO BUFFER

Format: SDNXB X

where X is the Unit number (0 to 63)

SDNXB initiates transfer of a bucket (specified in word 25 of the FDA) into a home buffer and returns to the calling routine which can then do other processing while the transfer is in progress. Word 25 of the FDA is not altered.

Note: If the unit has two home buffers, the transfer is initiated to read into the one which is not current. Thus SDNXB can provide double buffering when processing is non-automatic. This macro is intended for use with single thread units.

4 SDUWA: ALLOCATE/DEALLOCATE UNIT WORK AREA : FOR USE WITH MULTITHREADING

Format: SDUWA X N(M)

where	X	is the Unit number (0 to 63)
	N(M)	is the address of a word in either Lower or Upper data store which contains the address of a UWA or zero. N(M) may be an accumulator

The macro allocates and/or deallocates a UWA depending on the contents of the word addressed by N(M). If this word contains an address, this is taken as the address of a UWA which is to be allocated to the unit, any current UWA for this unit having been deallocated first. If the word contains zero the current UWA is deallocated

Note: This macro is intended for use when multithreading units are sharing a UWA. Each unit is opened with an SDDEF which has a 'U' parameter, and a UWA is allocated to each via a series of SDUWAs.

Whenever a UWA is allocated or deallocated it must not be attached to a thread (i.e. the UWA must not be in use).

REPLACED UNIT NUMBERS FOR MDA UNITS

All SD macro statements except those for SDDEF can specify a replaced unit number. The method of specification for all macros is as follows:

If the statement when the unit number is not replaced is

```
SD ---    X    parameter list
```

the statement when it is replaced is

```
SD ---    D    UNIT, parameter list
```

UNIT is a one word location, anywhere in Lower data store or accumulators X3 to X7, which will contain the unit number at run time. UNIT must not be modified

RANGE OF UNIT NUMBERS FOR MDA UNITS

Housekeeping allows the use of unit numbers in the range 0 to 63, but the user program may only use those unit numbers that are supported by the Executive or operating system being used.

UNIT WORK AREAS FOR MDA UNITS

Each unit which is being used in multithreading mode must have exclusive use of a UWA (allocated by the compiler if the 'UWA' parameter of the unit's SDDEF macro is set to V, or allocated by the user by means of the SDUWA macro) when attached to a thread. All other units normally use the common UWA, HMTHREAD (allocated by the compiler if the 'UWA' parameter of the unit's SDDEF macro is omitted).

MODE OF OPERATION

There is only one set of the new Housekeeping routines, which is called by both PLAN 3 and 4 Compilers. These routines are mode compatible and have 'HDM' as the first three letters of their names.

1 COMMON AREAS

The compilers, when generating calls to the new Housekeeping routines, may cause a number of common areas to be allocated. They are as follows:

HMTHREAD (Lower Common Variable)
Length = 100 words
Use: Common Unit Work Area.

HDMLV (Upper Common Variable)
Length = Maximum value of L1 parameter + 10 words
Use: Contingency bucket buffer for use by Second Level Overflow routine.

HMLVSIZE (Lower Common Preset)
Length = 1 word
Use: Contains length of HDMLV

HMMAXUNO (Lower Common Preset)
Length = 1 word
Use: Contains maximum MDA unit number used, plus one.

HMUWAPTR (Lower Common Variable)
Length = (Contents of HMMAXUNO) words
Use: Table of pointers for Housekeeping

HMFDAPTR (Lower Common Variable)
Length =(Contents of HMMAXUNO) words
Use: Table of pointers for Housekeeping

2 MACROS APPLICABLE TO MDA UNITS

All macros which are applicable to DA units are also applicable to MDA units.

Dump and Restart

USE OF MDA DEVICE CODE

Any unit number specified under a #CMODE directive for the device code 'MDA' can be given as a parameter to the SDUMP macro. The dump will be sent to the relevant DA device.

INTERFACE WITH DUMP AND RESTART FACILITIES MARK IV

These versions enable the user to take advantage of the Dump and Restart Facilities Mark IV. (For a full description of the package please refer to the *Direct Access Manual* (Edition 2 TP 4385). The compilers now accept a new parameter to the SDUMP macro, a description of which follows:

HDUMPFILE

A new parameter, HDUMPFILE, is defined for the SDUMP macro. If required, this is included as the last in the parameter sequence

SDUMP DPT, x_1, x_2, x_3 ....,HDUMPFILE

The parameter is required if any Direct Access overlay file is to be dumped. Otherwise, it and its preceding comma should be omitted. Inclusion of the parameter causes a call to the routine HDUMPFILE to be generated when the macro is compiled. The routine will be incorporated into the program at consolidation.

Compiler sizes

The minimum core requirement of these new versions is now as follows:

#XPLF	13504	words
#XPLZ	13632	words
#XPLT	18688	words

#XPLZ/30B: POSTMORTEM ROUTINE

This version of #XPLZ uses a different postmortem routine from previous versions. Its operation is as follows:

The postmortem procedure, should the compiler go ILLEGAL or be suspected of looping, can be initiated manually by inputting the console message

```
GO #XPLZ 29
```

The postmortem procedure, whether initiated manually by the operator or automatically by the compiler, first attempts to open a scratch file of 120 one block buckets as ED1. If the file is successfully opened, the postmortem procedure continues and finally halts with the message

```
0 #XPLZ; HALTED:-END OF PM
```

If the compiler fails to open a scratch file, it outputs the message

```
0 #XPLZ; HALTED:-SF
```

In this event, re-entering the compiler will only cause the message to be repeated, so an Executive core print of the compiler's area of store should be taken instead.

If it ever becomes necessary to inhibit the compiler's automatic postmortem, this may be done by setting switch 14 on before commencing the run. Should the circumstances then arise which, were switch 14 not on, would initiate the automatic postmortem, the compiler will halt with the message

```
0 #XPLZ; HALTED:-PM
```

In this event, the postmortem can be initiated manually by typing

```
GO #XPLZ
```

or else an Executive core print of the compiler's area of store should be taken. All postmortem prints or Executive core prints should be forwarded, together with the console log and printouts of the disc files used, through the normal channels for software errors.

AMENDMENTS TO MANUAL

With the issue of the versions of the compilers described above, Appendix 7 of the *Plan Reference Manual* should be amended as follows:

Specification of #XPLF page 71 line 5	Core requirement is now 13504 words
Specification of #XPLT page 91 line 5	Core requirement is now 18688 words

Specification of #XPLZ	page 101 line 5	Ccre requirement is now 13632 words
	page 102 line 6	Insert the section below:

ED1	scratch file (optional)	The file is assigned when a postmortem is initiated either automatically by the compiler or manually by the operator.

ICL

International Computers Limited

User notice

PUBLICATION (NOTICE NO.)

25/6/75

4322

PLAN REFERENCE (9)

File one copy of this notice with each of the publications indicated

PLAN4T/3 - GEORGE 3 SYSTEM MACRO

A new version of the GEORGE 3 macro PLAN4T will shortly be available. The new macro, the specification of which remains unchanged, enables the Fullist option to be used when running under GEORGE 3 Mark 8.30. Previous versions of the macro will fail in this respect if used with this mark of GEORGE 3. PLAN4T/3 may also be safely used with other recent versions of GEORGE 3. The new version of the macro also corrects the errors reported in Software Notice 1900/PLAN/103 item 180.

#XPLM/30C, #XPLX/30C - PLAN 3 DIRECT ACCESS COMPILERS

New versions of these PLAN 3 direct access compilers are shortly to be issued. Besides correcting a number of errors these versions also provide the interface to enable users to take advantage of both the Mark III 1900 Direct Access Housekeeping Package, and the Dump and Restart Facilities Mark IV. These changes are now described in more detail:

Errors Corrected

The following PLAN Software Notice items have been corrected in these new versions.

S/N	*Item*
38	66
42	73

Interface with Mark III 1900 Direct Access Housekeeping Package and Dump and Restart Facilities Mark IV

These versions will provide all the facilities necessary for the user to take advantage of the above software packages. The new facilities provided by the compilers are identical to those described for #XPLF/30B, #XPLZ/30B and #XPLT/8B on pages 2 to 8 of User Notice 8 to the PLAN Reference Manual.

Compiler Sizes

The minimum core requirement of these new versions is now:

#XPLM	5888 words
#XPLX	5888 words

#XPLG, #XPLH, #XPLV, #XPLW, #XPLN - PLAN 3 AND 4 MAGNETIC TAPE COMPILERS

It should be noted that the above compilers have not been changed to support the new facilities provided by either the Mark III 1900 Direct Access Housekeeping Package or the Dump and Restart Facilities Mark IV. These compilers cannot be used to generate calls to the former package, and although the new Dump/Restart routines can be incorporated, it is not possible to use the HDUMPFILE facility or specify MDA as a device type in a #CMODE directive.

AMENDMENTS TO MANUAL

Chapter 8, page 28

The following section should be inserted before step 5 of the operating instructions for #XPLT:

If, in a paged environment, the purity of a segment is not defined under a #PROGRAM directive, the segment by default will be made impure. If it is required to change this default to pure, input:

```
     ON #XPLT 15
```

If the steering information specifies LIST or SHORTLIST, and it is required to throw to a new page when channel 8 of the control loop is detected, input:

```
     ON #XPLT 18
```

If the steering information specifies LIST or SHORTLIST, and it is required to use a 96 print position printer, input:

```
     ON #XPLT 21
```

Appendix 7, page 91

The following should be added to the section headed "Switch settings":

15   If, in a paged environment, a segment is to be made pure by default.

ICL

International Computers Limited

User notice

PUBLICATION (NOTICE NO.)

4322 PLAN REFERENCE (10) 7/1/76

File one copy of this notice with each of the publications indicated

#XPMR/8C, #XPMY/4E, #XPMZ/4G
COSY EDITOR PROGRAMS

New versions of these Cosy editors are shortly to be issued. These correct errors reported in 1900 PLAN Software Notices as given below.

#XPMR/8C

This version will correct the errors described in the software Notice 1900 Plan 103/173.

Please note that the XPMR console message

HALTED:- INCORRECT FILE PARAMETERS. FIX & RESTART.

on page 38 Chapter 10 has been replaced by the console message

HALTED: PE

#XPMY/4E, #XPMZ/4G

These versions will correct the errors described in Software Notices 1900 Plan 101/165 and 101/166.

Please note that, with #XPMY/4E and #XPMZ/4G, under certain circumstances records containing sequencing errors in a subfile specified in a #MFILE or a #DFILE parameter will be listed to the line printer as if these lines were part of the parameter input.

The conditions under which this will occur are as follows:

#XPMY	When a #VARIABLE parameter and #MFILE parameter are present
#XPMZ	When an amendments type parameter specifies 'V' and the amendment includes a #MFILE or #DFILE parameter.

Issued by Software Distribution Department,30-31 Friar Street,Reading,Berkshire RG1 1JP

ICL

International Computers Limited

User notice

PUBLICATION (NOTICE NO.)

28/4/76

4322 PLAN REFERENCE (11)

File one copy of this notice with each of the publications indicated

SUBPROGRAMMING

Subprogramming is the term given to the facility whereby parts of a program, called *members*, can time-share with each other. In many ways subprogramming is similar to multiprogramming, except that whereas programs occupy discrete areas of store protected by hardware, subprogramming allows order numbers to be given to members of the program which share the program's store but follow their own sequence of instructions. The subprogramming facility is available with all Executives that have multiprogramming or dualprogramming facilities.

The purpose of subprogramming

The purpose of subprogramming, in general terms, is to enable programs to run more efficiently. One use is allowing calculation to be time-shared with input/output in a fairly straightforward commercial type of program where the degree of time-sharing provided by the use of double-buffering techniques and by the basic autonomy of peripheral transfers is not adequate. In such cases the use of subprogramming allows input/output routines of a more complete nature to be written. This particular use is usually relevant only in fairly modest operating environments, as the problem of obtaining the best performance from such a program is usually overcome by the off-lining of peripherals by operating systems such as GEORGE.

An extension of the example is the use of subprogramming in connection with real time equipment where it is essential to answer a request for acceptance of incoming data promptly to avoid possible loss of data. Subprogramming provides the means by which one part of the program can get incoming data safely into the processor whilst another part of the program is processing previously received data. Without the subprogramming facility, the processing routine would have to look at the real time devices very frequently and, even if it did so, the access time to service a request would be much greater than with subprogramming.

Members of a program

A program may consist of three or four members. Each member has its own priority. During loading or dumping, member 0 acts as a master member. At all other times any member may issue control instructions to suspend or activate itself or any other member.

Two forms of suspension may occur. A member may suspend itself by means of the appropriate subprogram control instruction or it may be suspended for a reason that is not relevant to subprogram control, for example while awaiting the terminator of a peripheral transfer. The former case is described as de-activation to avoid confusion with the latter case which is the generally accepted meaning of the term suspension.

Each member of a program has certain information that is permanently associated with it and that is stored each time the member is suspended. This information includes the contents of the floating-point accumulator and its overflow indicator (FOUR), the normal overflow indicator (V), the carry indicator (C), the address of the next instruction to be obeyed, the object program modes that are under the control of a program, that is the addressing mode, branch mode and zero suppression mode, and the accumulators. This information is stored in the first 16 words of the member's area.

These first 16 words contain for each member the same information normally held for a program in the first sixteen words of store. The first 16 words of each member's area are distinct and reserved for use by that member. These words are stored consecutively for each member from location 32 onwards of the program area. Thus:

Words 32 to 47	Words 0 to 15 of member 0
Words 48 to 63	Words 0 to 15 of member 1
Words 64 to 79	Words 0 to 15 of member 2
Words 80 to 95	Words 0 to 15 of member 3

Note: The GO AT directive renders to contents of words 0 to 15 indeterminate for all members. The rest of the program area is available to all members, so that each member's area consists of its own first 16 words plus the rest of the program area minus the other members first 16 words and any reserved areas.

MEMORY INDICATORS

Each member has associated with it two memory indicators: M and P. These indicators are used to remember attempts to activate a member while it is already active. The indicators each consist of a single bit that is set if an attempted activation is to be remembered; subsequent attempts to activate the member before the memory indicator has been cleared will be forgotten.

M is the indicator set by a 163, *N(M)*=0(AUTO) instruction issued in respect of a member that is active. It must be remembered that a member may be active but suspended. The M indicator remains set until cleared by a 164 (SUSAR or SUSIN) instruction.

P is the indicator set when an event occurs, while the member is active, on a direct response device that the member controls. An event is said to occur on a flag-setting direct response device if the reply word for the device changes from *transfer in progress* to *transfer completed* or if a condition requiring object program action occurs. An event is said to occur on a suspension device operating in direct response mode if the former condition above occurs or if a device that was disengaged becomes engaged. A member is said to control a device if it is the member from which the most recent non-discrete instruction was accepted for that device. A non-discrete instruction is one that cannot be assumed to have been completed at the time that execution of the following instruction begins. If there was no such instruction, the controlling member is the one that established the devices flag area or caused the device to be switched to direct response mode.

The P indicator remains set until cleared by a 164, *X*= 2 or 3 (SUSIN) instruction.

Priorities

The priority assigned to each member is that supplied in the request block; the number of each member in no way affects the member's priority. It should be noted that the priorities of members of a program can be changed by means external to the program without the program being aware of the change; accordingly, a program must not be logically dependent upon the priorities of its members.

Address and branch modes

The initial mode setting of member 0 is determined by the supplementary request block. Members other than member 0 obtain their initial mode setting from the mode of the member that first activates them. When a multi-member program is dumped, the mode setting of member 0 only is recorded. On subsequent reloading, all members will again take their initial setting from the member that activates them initially.

Loading

When a program is first loaded member 0 will be active and suspended awaiting operator action. All other members will be inactive awaiting activation by a 163, *N(M)*≠0 (AUTO) instruction.

Dumping

Orders 154 (CONT) and 155 (SUSDP) are illegal if issued by any member other than member 0. All members of a program are suspended while either of these two instructions is being carried out so that words 0 to 15 of all members are in their respective storage areas and thus will be dumped correctly for subsequent reload.

In the case of an operator initiated dump, the operator must ensure that all members are suspended before the dump is initiated. The members may be suspended by the use of a SUspend directive in respect of each member.

Reference to common storage areas

As stated in *Members of a program*, page 1, words 0 to 15 of each member are protected from corruption; the rest of the program area is common to all members and it is therefore possible for one member to corrupt another. This means that where an area of store is to be used by more than one member the program must include appropriate lock-out routines. In simple cases lock-out can be performed by using 163 and 164 instructions; in more complex cases it has to be performed by means of indicators that record the state of the common areas. In the latter cases no assumption can be made as to the relative priorities of members, since these are not under members' control. It is always possible that another member's instructions may be obeyed between the obeying of any pair of successive instructions in a member's routine, so that the programmer must ensure that data being processed by one member is always protected from interference by other subprograms.

A consequence of this last point is that the alteration of an indicator that is also altered by another member, and whose altered value is in some way dependent upon its original value, must be carried out by a single instruction that alters the quantity directly in its commonly accessed location. If more than one instruction is used to alter an indicator, it is impossible for two members to be altering the same indicator at the same time, with indeterminate results. The 162 (SUSMA) instruction is provided to help overcome this problem, since SUSMA causes Executive to be entered, so that no other member can interrupt.

Two further important consequences are as follows:

1 If, under certain conditions, an indicator is set by a member and is nowhere reset by that member, then, if another member determines that the indicator is set, it can assume that the setting conditions existed in the first member. However, if it detects that the indicator is not set, it cannot assume that the setting conditions did not exist

2 Only when a routine in a program is pure may it be obeyed by more than one member of the program; this applies particularly to subroutines. In this context, a routine is deemed to be pure if the only words of the program area that it attempts to change lie in the range 0 to 15 inclusive or are reserved for that member and are assessed by use of a modifier. The area used for dumping words 0 to 15 of the member's area must not be used by a pure routine

Subprogramming control instruction

The instructions provided for the control of communication between members of a program are the 162 (SUSMA), 163 (AUTO) and 164 (SUSAR or SUSIN) instructions.

THE 162 (SUSMA) INSTRUCTIONS

The action of the 162 (SUSMA) instruction depends on the contents of word *N(M)*+1, as follows:

1 If the contents of word *N(M)*+1 are non-zero, the program continues at the instruction in the word following that which contains the 162 instruction

2 If the contents of word *N(M)*+1 are zero, they are made non-zero and the contents of *X* are copied into word *N(M)*. The program then continues at the instruction contained in the second word after that which contains the 162 instruction

Note: *N(M)* must not be in a reserved area of store.

THE 163 (AUTO) INSTRUCTION

The 163 (AUTO) instruction takes two forms:

1 *N(M)* is non-zero; this form is provided for the initial activation of a member after the program is loaded

2 *N(M)* is zero; this form is provided for subsequent activation of a member

Initial activation

X contains the number of the member to be activated, the first instruction to be obeyed being that in word *N(M)*. Activation will cause member *X* to assume the same address and branch modes applicable to the member that issues the 163 instruction at the time that the instruction is issued. The state of the zero suppression mode in member *X* will be indeterminate.

Restrictions applicable to this form of the instruction are:

1 *N(M)* must not be in a reserved area of store

2 The instruction may be obeyed only when member *X* is inactive in state SL, that is in the state assumed by members other than member 0 as a result of initial loading or the GO AT directive

3 Member *X* can never be member 0 or the member that issues the 163 instruction

Subsequent re-activations

X contains the number of the member to be re-activated and word *N(M)* is always zero. If member *X* is currently inactive due to a 164 instruction, it is re-activated at the instruction following that 164, with the state of address, branch and zero suppression modes the same as when the 164 instruction was issued. If member *X* is currently active, the memory indicator M of member *X* will be set and will remain so until member *X* issues a 164 instruction which will then clear the M indicator but not de-activate member *X*.

Restrictions applicable to this form of the instruction are:

1 Word *N(M)* must be zero

2 The instruction must not refer to a member that is inactive in state SL

THE 164 (SUSAR OR SUSIN) INSTRUCTION

This instruction provides the means by which a member can de-activate itself until a specified type of event occurs provided that no such event has occurred since the previous equivalent instruction was issued by the member. It will be noted that this instruction has variants dependant upon the value of *X* (*N(M)* is always zero), and that successive variants include all preceding variants. The definition of all variants is such that spurious re-activation, that is re-activation of a member that should not be re-activated may occur. All programs must be coded to allow for spurious re-activation.

The 164, *X*=1 (SUSAR) variant

Unless the M indicator of the member issuing this instruction is set, the member is de-activated until either a 163 instruction referring to this member is issued. If the M indicator is set; it is cleared and the member proceeds at the next instruction.

The 164, *X*=2 (SUSIN) variant

Unless the M or P indicators of the member issuing this instruction are set, the member is de-activated until either a 163 instruction referring to this member or an event on a direct response device controlled by this member occurs. If either or both of the M and P indicators are set, they are then cleared and the member proceeds at the next instruction.

States of members

When a program is loaded, member 0 is deemed to be active, although it will probably be suspended awaiting some operator message, for example, GO. Member 0 may then activate some other member and de-activate itself. It is important to realise that a member may be active, that is have current use of the central processor *as far as that program is concerned*, and yet be suspended awaiting operator action or an event such as the terminator of a peripheral transfer.

A program may be in any one of a number of inactive states. These inactive states are considered below.

STATE TRANSITION TABLES

Tables 1 and 2 summarize the effect of various events under all possible valid conditions. The explanatory notes that follow should be read in conjunction with the diagrams.

1 The word *invalid* indicates that a restriction has been violated

2 At any time a member is in one of the following states:

NS Active but may be suspended

SL In an inactive state because it has not had an initial activation since the program was loaded or since a GO AT directive

SM Inactive because it has issued a 164, *X*=1 instruction

SMP Inactive because it has issued a 164, *X*=2 instruction

SMPE Inactive because it has issued a 164, *X*=3 instruction

Instruction or event	*Applying to member*	*Member being in state*				
		SL	*NS*	*SM*	*SMP*	*SMPE*
163, *N(M)*≠0	Y	Y becomes active	Invalid	Invalid	Invalid	Invalid
163, *N(M)*=0	Y	Invalid	M indicator of Y set	Y becomes active	Y becomes active	Y becomes active
Direct re response peripheral event	Y or all members	Invalid	P indicator of Y set	P indicator of Y set	Y becomes active	Y becomes active

Table 1 The effect of the 163 instruction and direct response peripheral events in member's state

Memory indicators of Y that are set	*Effect of member Y issuing a 164 instruction*		
	164, X=1	*164, X=2*	*164, X=3*
None	Y assumes state SM	Y assumes state SMP	Y assumes state SMPE
P only	Y assumes state SM	P cleared, Y remains in state NS	P cleared, Y remains in state NS
M only	M cleared, Y remains in state NS	M cleared, Y remains in state NS	M cleared, Y remains in state NS
M and P only	M cleared, Y remains in state NS	M and P cleared, Y remains in state NS	M and P cleared, Y remains in in state NS

Table 2 The effect of the 164 instruction on memory indicators

ICL

International
Computers
Limited

User
notice

PUBLICATION (NOTICE NO.)

15/7/76

4322

PLAN REFERENCE (12)

File one copy of this notice with each of the publications indicated

#XPMZ/4H

COSY editor program

A new version of this program is shortly to be issued.

Errors corrected

The errors described in the following 1900 Series/PLAN Software Notices are corrected in this version.

112 item 2, 113 item 2

500

ICL International Computers Limited

User notice

PUBLICATION (NOTICE NO.) 5/1/77

4322 PLAN REFERENCE (13)

File one copy of this notice with each of the publications indicated

PLAN 3 Compilers #XPLZ and #XPLF

The following amended versions of the above compilers will shortly be issued:

#XPLZ/30D
#XPLF/30C

Errors corrected

These compilers correct the errors published in the following 1900 Series/ PLAN Software Notices.

103 item 168
103 item 169
106 items 1 and 2
120 item 2

Issued by UK Software & Literature Distribution Centre, 60 Portman Road, Reading, Berkshire RG3 1NR