INTERNATIONAL COMPUTERS AND TABULATORS LIMITED

ERRATA AND AMENDMENTS NO.2 TO THE ATLAS PROVISIONAL PROGRAMMING MANUAL (CS348)

The documents which have been published to correct and elucidate the Atlas Provisional Programming Manual are

Errata and Amendments No.1 (September 1963). Input and Output For Atlas Computers 1 and 2 (April 1964; replaces Chapter 8 of the manual; not however for use with Atlas 2) Preparing a Complete Program for Atlas 1 (August 1964; replaces Chapter 10 of the manual) Errata and Amendments No.2 (August 1964)

AUGUST 1964

I.C.T.LIMITED
68, NEWMAN STREET
LONDON, W.1.

I.C.T. LIMITED
TELEVISION HOUSE
PETER STREET
MANCHESTER 2

INTRODUCTION

The following is a list of notes to be inserted in the ABL Provisional Programming Manual. Some of the notes take the form SEE AMENDMENT so-and-so, and a numbered list of amendments follows the list of notes. The remaining notes are themselves short corrections or amendments. A typical reference to be written in the manual at the appropriate point might be SEE A3/2 (AMENDMENT 3 OF E & A 2).

CONTENTS

PAGE	SECTION	INSERT
6	after 2.5	AMENDMENT 1 of E & A No.2 contains a new section on the complete range of Atlas addresses.
14	4.3	If Bm is B0 in the 164 and 165 instructions, then bm&n gives n rather than 0.
17	4.6	About instructions 200, 201, 202, 203 : If Ba and Bm are the same B-line and the test succeeds, its final contents are n. If Bm is B127 (and Ba is not), these instructions give an unpredictable result.
18	4.7	About instructions 220, 221, 222, 223 : If Ba and Bm are the same B-line and the test succeeds, its final contents are n. If Bm is B127 (but Ba is not), these instructions give an unpredictable result.
20	4.10	See AMENDMENT 2 of E & A No.2 for a more complete version of this section.
20	4.11	See AMENDMENT 3 of E & A No.2 for a new section on the B-carry digit.
21	5.1	See AMENDMENT 4 of E & A No.2 for more on ABL format.
22	5.3	See AMENDMENT 5 of E & A No.2 for a more complete version of this section.
22	5.3.1	See AMENDMENT 6 of E & A No.2 on setting preset parameters by an interlude.
25	5.8	If * is not set at the head of an ABL program, * = 1: is assumed.
27	5.11	See AMENDMENT 7 for more information on ABL floating point numbers.
28	5.13	Any optional parameter settings to be made for a library routine should occur before the L directive that calls that routine.
29	5.13	B1 to B88 are cleared before a program is entered by E, ER or EX. B89, however, contains the current value of *. After an E directive B90 contains J3; after ER and EX B90 is clear.

PAGE	SECTION	NOTE
33	6.2.3	The inequality 11 lines down in this section should read sx < $\frac{1}{2}$.
43	7.4.6	1400 - the imaginary part of the complex logarithm will lie in the range $-\pi$ (not inclusive) to π (inclusive). 1410 - of the two possible values of the complex square root, the one computed here has a non-negative real part; the remaining ambiguity about the square roots of negative real numbers is removed by computing the one whose imaginary part is positive.
48 to 56		Chapter 8 has been replaced altogether by a separately produced document. (See the cover page of the present document). The following fact should be mentioned on page 24 of that document: When 5 hole tape is read in binary the order of the bits as they appear in the store is the reverse of the order shown in the Internal Character Code table (pages 100 to 103) in the programming manual. For example open brackets '(' is listed on page 100 as FS 001.01 and when stored in binary it appears as 000000 010100. The card showing the Atlas/ Orion punched tape code (CS 308B) gives the bits in the order given by the manual.
57	9.1	The range of tape numbers (3rd paragraph) should begin at zero rather than one.
59	9.3	Paragraph 5, line 3 should state that "the start instruction must be preceded by the word search instruction 1044".
60		See AMENDMENT 8 of E & A No.2 for more details on the 1032 and 1040 instructions.
61		For a correct definition of the Where am I? extracode (1024, not 1022 as listed) see page 67 of the manual.
61		See AMENDMENT 9 E & A No.2 for more details on the 1044 word search extrocode.
65 & 66		Replace everything from (and including) the final paragraph of section 9.5.4 (Safeguards) to (and including) the 1015 (Read title) extracode on page 66 with AMENDMENT 10 of E&A No.2.

PAGE	SECTION	NOTE
68 to 76		Chapter 10 has been replaced entirely by a separately produced document. (See the cover page of the present docuement.) The following additional rule for the preparation of document and tape titles should be inserted on page 3 of that document (Preparing A Complete Program - section 10.3.3): d) a title must not contain three successive asterisks (***). Under section 10.4.2 on the OUTPUT section of the job description it should be stated that a request will be made to the operator to mount special stationery for a given output stream if an asterisk is placed in front of the word LINE PRINTER. Thus if output stream 3 is to be printed on special stationery, the output section should contain *LINE PRINTER m BLOCKS.
78		When the rename extracode (1164) is used there must be at least one more block allowed for in the job description than is defined at that moment.
78		The list of definitions of P_1, P_2, etc. should have added to it: sector number k= bits 21 to 23 of ba.
79		See AMENDMENT 11 of E & A No.2 for two new extracodes and corrected versions of the drum extracodes 1175 and 1176.
85	11.4	See AMENDMENT 12 in E & A No.2 for information on ABL monitoring.
86		See AMENDMENT 13 in E & A No.2 for an up-to-date version of the Supervisor fault table.
87	11.4.2 (line 9)	The value of main control left in the B-line is the value when the fault was detected and may not be the address of the instruction which caused the fault.
87	11.4.2	The entry opposite n = 0 in the table explaining the 1114 (exit from trap) extracode should read 'compile the program again and enter it at the beginning, making the original data available to the re-compiled program.'

PAGE	SECTION	NOTE
87	11.4.7.	Atlas restart procedures are being reorganised. At present when a computer failure occurs all jobs not using magnetic tapes are recomputed and entered at the beginning. The restart extracode 1113 should not be used until further notice.
88	11.4.4.	The address left in B92 is the value of main control when the fault was detected, which may not be the address of the instruction which caused the fault.
89	11.4.5.	See AMENDMENT 14 in E & A No.2 for a corrected description of the 1117 'end program' extracode
89	11.4.6.	This example is incorrect.
92.		The 1115 extracode writes all the blocks of the program (not the input or output streams) to private tape in ascending order of black labels. Omit the words 'instead of onto systems tape' - no such dump is ever automatically made.
92		The 1116 extracode is not currently useful since programs are not automatically dumped onto systems tape when they are monitored.
93		See AMENDMENT 14 or E & A No.2 for a full description of the 1117 extracode.
95		See AMENDMENT 15 of E & A No.2 for additional documents concerning Atlas.
99		The V-store address J60091000 given for type 4 equipment refers to the Analex Line Printers (not the I.C.T. Hammer Printers).
100		& (ampersand) is 8,5 on cards.

Other characters available on cards but not listed in the manual are:

Colon	6;8
Open square brackets	-;7,8
Close square brackets	-,6,8
Underline	+;6;8
Vertical bar	+,7,8

E & A NUMBER 2

AUGUST 1964

AMENDMENTS

FOR

ATLAS PROVISIONAL

PROGRAMMING MANUAL

(CS348)

2.6 THE FULL RANGE OF ATLAS ADDRESSES

As explained in 2.4, address bits 1 to 11 represent the number of the 512 word main store block to which that address refers, so that an Atlas main store address refers to one of 2048 blocks. In octal (for the J notation see 4.3) these block addresses are J0000, J0001, J0002,...., J3777 and in their decimal representation (see 5.6) 0:, 1:, 2:,..., 2047:. The ABL compiler and the program it is compiling share the same range of block numbers, with ABL occupying blocks in the range J3 to J34. Consequently to avoid over-writing itself ABL will refuse to compile program into any of the 256 blocks 1536: to 1791: Once the program is compiled and ABL has withdrawn from the store these blocks become available again and can be used as working space by the program .

The remainder of the main store block numbers J34 to J4 are illegal. ABL will not store program in block J3 or above and the Supervisor will fault the program if the compiled program attempts to refer to block J34 or above.

In fixed store and private store addresses bit 0 is a 1. There are therefore another 2048 blocks that can be addressed and they have octal addresses J4000 to J7777. The first 16 of these (J4 to J4017) are the block numbers of the fixed store and may be referred to by the programmer if he wishes, although there is generally no reason why he should do so. The block numbers J4020 to J4777 are also quite legal. In effect these block numbers refer to 31 consecutively stored copies of the fixed store. For example either of the instructions

```
101   3   0   J4017777
101   3   0     -1J5
```

would place the same half word in B3, namely the left half of the last word of the fixed store. In other words addresses in the range J4 to J5 are in effect masked with J40177777 before execution (for a definition of masking, or what is the same thing collating, see 4.3)

The private store block addresses J5 through J7777 are completely forbidden to the ordinary programmer. These addresses ca n be referred to on extracode control. However it is impossible for the programmer to force an extracode to refer to the private store. He is prevented by faulting an extracode instruction in which the modified address is in the private store but the unmodified address is not. Thus for example a program containing the instruction

```
1730  0   0   J6    am' = sin (J6)
```

would be thrown off because the first instruction of the extracode routine is

```
324   0  119  0      am' = (b119)
```

(B119 will contain the address J6 upon entry to the extracode. See 7.2).

AMENDMENT 2

4.11 Restrictions on the Use of B-registers

Although B81 - B119 were included in section 4.1 as general purpose B-registers, they are of limited utility for the ordinary programmer, since they are each used by one or more of the system routines which may assume control during the running of the object program. Before using any of these B-registers, the B-test register, the substitution register, or the B-carry digit, the programmer must check to see that there is no danger of their contents being overwritten before he has finished with them.

The routines which these B-registers are as follows:-

B81-89	Library routines
B90	Return link from library routines
B91-97	Extracodes
B98-99	The logical Accumulator and some less common extracodes
B100-110	Supervisor
B111-118	Interrupt routines
B119	Extracode operand address
B121, 122	Extracodes, library routines
Bt, Bc	Extracodes

It should be noted that the library routines may use extracodes. This means that when library programs are in use, no B-line above B80 should be used (except for B90). Provided no reference is made to library routines, B81 - B90 may be freely used. Similarly B81 to B99, B121, B122, Bt and Bc are safe to use when neither extracodes nor library routines are in use. It is never safe for an ordinary program to use B100 - B118, since an interrupt can occur at any time and cause control to be transferred to the Supervisor.

AMENDMENT 3

4.11 The B-carry digit

When any one of the four addition codes

104	ba' = ba + s
114	s' = ba + s
124	ba' = ba + n
164	ba' = ba + (bm & n)

is used to add two 24-bit quantities, bit 23 of line 6 of the V-store is set to 1 if there is a 'carry' from the addition.

Thus for example the addition of any two 24-bit numbers whose leftmost bit is a 1 sets the 'B-carry digit' to one. If there is no 'carry', the B-carry digit is set to 0. When an ABL program is entered the B-carry digit is clear.

The singly - modified extracode 1223 loads Ba with n if the B-carry digit is set to 1 and does nothing if it is not set. (The extracode does not affect the state of the B-carry digit.) The following example uses 1223 to add b1 to b2 and then add the 'carry', if any, to the bottom of B3. Thus the contents of B1 and B2 are here regarded as 24-bit positive integers whose double length sum is placed in B3 and B2 with the most significant half in B3.

Example

```
124      2      1      0
1223     3      3      Y1
```

Similarly each of the ten instructions

100	ba' = s - ba	102	ba' = ba - s
110	s' = s - ba	112	s' = ba - s
120	ba' = n - ba	122	ba' = ba - n
150	bt' = s - ba	152	bt' = ba - s
170	bt' = n - ba	172	bt' = ba - n

set the B-carry digit to 1 when, regarded as 24-bit positive integers, a larger number is subtracted from a smaller. Otherwise these codes set B-carry to zero. For example, the B-carry digit is set to 1 by the instruction 172, 0, 0, 1.

Example

In the previous example b1 was added to b2 and the double length sum held in B3 and B2. The following two instructions would subtract b1 off again from the double length sum.

```
122      2      1      0
1223     3      3      -Y1
```

AMENDMENT 4

A complete line of ABL input is read, and an image of the print-out is formed, taking correct account of the characters SPACE, BACKSPACE, and TAB. TAB is interpreted assuming 9 fixed TAB positions, at 8, 16, 24, 32, and then every 16 up to 112, character positions from the left-hand margin; TAB always moves the current 'carriage position' along at least two character positions. A maximum of 128 character positions along the line is allowed for; any characters beyond position 127 are ignored. A backspace beyond the left-hand margin is ignored.

In interpreting a line, ERASE, or a composite character including ERASE, is everywhere ignored (except in circumstances where a direct copy of a string of characters is called for - see section 5.10.)

The character small l is an illegal character. Otherwise ABL treats upper and lower case letters as being identical. The letters capital O and capital I are treated as alternatives to zero and one.

AMENDMENT 5

5.3 PRESET PARAMETERS

Preset parameters are not associated w ith any particular routine and are meant for use by the program as a whole.

One hundred preset parameters P0 to P100 are allowed. P0 may also be written P. Certain preset parameters with numbers greater than 100 have special effects, described in amendment 12.

Preset parameters can only be set directly by an equation (P3 = 10) and cannot be set by labelling. Unlike routine parameters they cannot be referred to before they have been set (see Amend. 12 for faulting action). However, preset parameters can be reset to a new value later in the program (later in the sense of being compiled later). Thus, the same preset parameter can be used as often as is convenient with different values, e.g. see the use of P123 described in 5.10., (Taken from Page 4 of E&A No.1).

In connection with the optional setting of a preset parameter (see 5.5) one may direct the ABL compiler to regard a preset parameter as never having been set by writing Ua (unset Pa). Similarly Ua-b means unset Pa through Pb. The directives P=1, U1, P?=2, compiled in the order given, result in preset parameter P0 being set to the value 2.

The compiler always sets a preset parameter immediately upon finding it in the program. Thus, everything on the right side of the equation which sets a preset parameter must itself already have a value. This means that P = 61A/3 would be faulted if it occurred in the program before label C of routine 3 had been defined.

It takes somewhat less time to compile preset parameters than routine parameters.

AMENDMENT 6

5.3.1. PRESET PARAMETERS SET BY PROGRAM

Normally the values of all the parameters used in the program are determined during compilation of the program and before the program is entered. Indeed when compilation is terminated by an E directive the ABL compiler along with its lists of parameters is removed from the main store. However, the compiler and its lists are still in the store when the program is entered by an EX or an ER directive, although only the list of preset parameters is made available to the (partially) compiled program. By using the special preset parameters described in Amend 12 the programmer may then enter a part of his program and set a preset parameter, then re-enter the compiler to compile the remainder of his program.

In the following example P40 is set to the value of the integer which is the first item of input stream 7. Note that the program input stream (input 0 in this example, although in general the lowest numbered input) must be reselected before ABL is entered again. Otherwise ABL would treat input 7 as if it contained the program.

```
JOB
F5006/PROGRAM WITH INTERLUDE
INPUT
7 NUMBERS HEADED BY AN INTEGER
COMPILER ABL
L100  -  LIBRARY ROUTINES USED IN AN
      -  INTERLUDE MUST BE CALLED EXPLICITLY
1050  0    0   7          - CALL INPUT 7
1362  0    0   A2/L100    - b81' = INTEGER
113   81   0   40P121     - P40 = INTEGER
121   1    0   J2'        - b1'  = MASK
117   1    0   40P122     -'P40 IS SET'
1050  0    0   0          - RESELECT INPUT 0
121   127  0   P120       - RE-ENTER COMPILER
EXA
(PROGRAM USING P40)       - COMPILATION CONTINUES HERE
E(ENTRY POINT)
***Z
```

AMENDMENT 7

Some notes on ABL floating point numbers

If the program contains a floating point number that is to large to be stored in Atlas standardized form the program will be monitored during compilation and the fault indicated by the monitor printing EXPONENT OVERFLOW. If a floating point number is too small to be represented in standardized form ABL will store floating point zero in its place and continue compiling the program.

If the exponent of a floating point number is forced to a value that is too small to allow the number to be represented in standardized form, the program is monitored and A0 on fixing is printed to identify the fault. For example, +1:0 requests that 1 be stored in floating point form with exponent zero, which cannot be done -1:0 would be acceptable).

AMENDMENT 8

1032 Prepare to write forwards on tape Ba in variable length mode.

This instruction must be preceded by a 1044 word search instruction, even when the desired word is the first in the block. (A block search should not be used and may result in the record being written starting at the wrong word word on the tape).

A buffer will be set up by this extracode in the K+1 blocks P, P+1,...., P+K (These blocks must have been allowed for in the job description. If one of these blocks is already in use by the program, the information in it will be lost. Strings of information with markers at either end are transferred to these buffer blocks by the 1040 instruction and, when a buffer block is full, it is written to the tape Ba mentioned in this instruction. The buffer blocks are not protected from the main program but they should not be referred to directly under any circumstances.)

When the first record is written with the next 1040 instruction it will begin with a 7 marker. (If a different marker is required the 1042 'mark' instruction must be used prior to the 1040 instruction.)

Select tape Ba for succeeding variable-length operations. (This means that if no other tape is selected in the meantime, all succeeding transfer (1040), skip (1041) and

mark (1042) instructions apply to this tape.)

If there is already a 'write buffer' set up for tape Ba (by an earlier 1032 instruction) the information it contains will be written to the tape and the second 'write buffer' will be set up. The first record written via the 'new' buffer will then immediately follow the last record written from the 'old' buffer and will begin with the same marker as the one which ends the previous record.

It may be desirable to read variable length records from a tape for a while and then begin writing to the very next record on that tape. The 1032 instruction will switch modes in this way. The first record written will begin with the same marker at the one which ends the previous record.

Whenever variable length writing is terminated on a given tape each buffer block that contains information there by a 1040 instruction is written to tape. (A buffer block is not written to tape unless it contains such information.) This means that the first few words of the last buffer block written may contain the end of the final record (or even just the final marker) and the rest of the block contain perhaps the remains of some previously transferred records. This fact permits one to overwrite individual records without disturbing the records on either side. This is done by filling the buffer with the record that is to be overwritten along with the records (or part of the records) on either side of it. One aligns on the marker at the beginning of the record and starts reading forward (1030) with a buffer large enough to contain the whole record at one time. One then skips' (1041) the record in question, starts reading backward (1031, with the same buffer) and skips' back over the record to the beginning marker. One switches to write mode (1032, same buffer again) and writes the new string in its proper place. A 'stop' instruction (1043) will then finish the job by causing the buffer blocks, now containing the new string in place of the old, to be written back to the tape. The two uses of the skip instruction are necessary to make the marker at the beginning of the old string available for the construction of the ma rker at the beginning of the new string.

EXAMPLE

There is a 50 word record in section 10 of tape 33 which is preceded by a '7' marker in word 177. Replace that record with the 50 consecutive words beginning in the store at A3, leaving the '7' marker undisturbed.

```
3)  H10,177                                  Section 10, word 177
    1044        33        0        A3       search for marker
    1030        33        0        1        Fill the one block buffer
    1041        0         0        0        skip to end of next record
```

(program continues on next page)

1031	33	0	1:	Read backward
1041	0	0	0	skip back to head of record
1032	33	0	1:	switch to write mode
121	75	0	50	
1040	75	0	A3	replace record in buffer
1043	33	0	0	write block 1: to section 10

Although in the example just given the length of the string was known in advance and BO mentioned in both of the skip instructions, one could use one of the skip instructions to pick up the length of the string in a B-line.

If in the above example the 1032 write instruction is replaced by a 1042 mark instruction, the program will replace only the mark at the head of the string.

NOTE ON THE 1040 TRANSFER INSTRUCTION

It is not advisable to write with the octal fraction $b_K=0$, because when the resulting string, which therefore ends with a zero marker, is read back, the octal fraction bK will be zero regardless of whether reading ended at the zero marker or somewhere short of the marker within the string itself.

AMENDMENT 9

1044 Word Search

This instruction positions the tape before section A of tape Ba and defines the 'next word' W at which extracodes 1030 and 1032 begin reading and writing forward. A must be stored as a 21 bit integer in the left half of the full word whose address is S. W occupies the right half word and is also a 21 bit integer. If tape Ba is not defined the program is terminated by the Supervisor. The program is also terminated if A≤3 or if A≥4999. This instruction (rather than the block search 1001) must be used before starting to read (1030, 1031) or to write (1032) variable length records.

AMENDMENT 10

9.5.4 (last paragraph)

When a magnetic tape has useful information on it, a descriptive title of that information is stored in block 0 of the tape. This is in addition to the tape serial number, which is permanently associated with the tape and is unalterable by the user. This tape title can be up to 80 characters long, though the Supervisor prints out only the first 30 characters in operator messages. It is stored on tape in Atlas Internal Code with tab and multiple spaces (including tab) replaced by a single space; initial spaces, tabs, full stops, and commas are ignored. Redundant shifts are ignored throughout the title. See Chapter 10.

9.6 Organizational Instructions

A number of organizational instructions are provided to cope with special situations which will arise in some magnetic tape programs. Many of the operations which these instructions perform are usually required near the beginning or end of the program, and they are then best left to the job description or the 1117 (end program) instruction. One exception to this rule is the instruction 1017 (Free Tape), which should be used before the 1117 instruction if the information on any titled tape is not required again.

Programs frequently require to use magnetic tapes as working space, without wanting to keep them after the job has been run. Such magnetic tapes should be requested under a heading TAPE COMMON in the job description.

For a long job, it may be desired to restart the job after a machine failure severe enough for common tapes to be lost; in such a case tapes should be requested under TAPE NEW. In exceptional circumstances a title may be written to a common tape, which is then kept by the user after the job has ended; the operator will be notified, and the Atlas installation may take action to discourage a user from doing this at all often.

The title stored in section 0 of the tape is referred to by extracodes 1014 and 1015 (Write Title & Search for Block 1, and Read Title or Tape Number). These refer by its main store address to a copy of the tape title, stored as one record; the 6-bit characters are packed 8 to a word and the last character is binary zero.

9.6.1 Mount Instructions

If a program requires to use magnetic tape, its job description must indicate the number of magnetic-tape mechanisms required. Normally this is done by listing the magnetic tapes which are required to be mounted on these mechanisms initially: the Supervisor will then ensure that these tapes are mounted before the program is entered. The details of how the job description is prepared are given in Chapter 10.

If further magnetic tapes are required after a program has been entered, they should be listed in the job description and may then be called in by obeying "mount" instructions. However, the total number of mechanisms in use at any one time must not exceed the number reserved in the job description. A mount instruction should, if possible, be obeyed at least 2 minutes before the tape to which it refers is required; otherwise the tape may not be ready in time and the program will have to wait. Note that the program will be monitored if it calls for a new tape to be mounted at a time when none of its reserved mechanisms has been made free. If there is a spare tape mechanism, the tape may be mounted on it by the operator before the program calls for it.

The tape reference letter referred to in the mount extracodes is a letter associated with the tape in the job description (see chapter 10).

The mount instructions are as follows:-

1010 Mount

Allocate the number Ba to the tape whose tape reference letter is in the 6 least significant bits of n, in internal code. If this tape is not already available, instruct the operator to mount it on any available tape mechanism. If the tape is in the TAPE category, check its title; if in the TAPE NEW category, write into the section 0 the title given in the job description: if in the TAPE COMMON category, leave the tape untitled.

1011 Mount Free

Allocate the number Ba to a free tape. If no free tape is available, instruct the operator to mount one on any available tape mechanism.

This extracode should only be used in exceptional circumstances; normally all tapes required should be listed in the job description. The operator will be informed whenever this extracode is used, and an installation may take action if it is used overmuch.

1007 Mount Next Reel of File

Allocate the number n to the next reel of file Ba. If this tape is not mounted, instruct the operator to mount it on any available mechanism.

The following mount instructions refer to logical channel number (K= 0 to 3) of a given program. These logical channel numbers do not each refer to a fixed magnetic-tape channel: they are merely a device to enable the program to separate on to different channels the magnetic tapes that it requires to operate simultaneously. On an installation with only one tape mechanism on each of the eight tape channels, there is no advantage in specifying channel numbers. On larger installations, the tape mechanisms are grouped together on pairs of channels, and the logical channel numbers then refer to these pairs.

A program's referring to logical channel number K has the following effect. If no channel has been previously designated logical channel K of the program, the new tape is mounted, if possible, on a channel different to any which has been previously designated a logical channel of the program; that channel is then designated logical channel K of the given program. If a channel has been previously designated logical channel K of the program, then the new tape is mounted on the same channel, if possible. A channel may also be designated logical channel K of the program by the Job Description; see chapter 10.

1012 Mount on Channel K

Allocate the number Ba to the tape whose tape reference letter is in the 6 least significant bits of n, in internal code. If this tape is not already available, instruct the operator to mount it if possible, on channel K of this program, where K is the most significant octal digit of n. If the Tape is in the TAPE category, check its title; if in the TAPE NEW category, write into section 0 the title given in the job description; if in the TAPE COMMON category, leave the tape untitled.

1013 Mount Free on Channel K

Allocate the number Ba to a free tape. If no free tape is available, instruct the operator to mount one, if possible, on channel K of this program, where K is the most significant octal digit of n.

Note that this extracode, as 1011, should be used only in very exceptional circumstances; normally all tapes required should be listed in the job description. The operator will be informed whenever this extracode is used and an installation may take action if it is used overmuch.

9.6.2 Other organizational Extracodes

1014 Write Title & Search for Block 1

Write to section 0 of tape Ba the title stored in locations S onwards, overwriting any title that may be there. Inform the operator that this has been done.

1015 Read Title or Tape Number

If S is even (least significant bit is 0), then store in locations S onwards the title of tape Ba, i.e. that currently in section 0.

If S is odd (least significant bit is 1), then store in location S the tape number of tape Ba.

In 1014 and 1015 S is taken as a half word address. The tape serial number stored by 1015 will be in internal code, left justified, 8 characters in length.

AMENDMENT 11

Two new extracodes for use in keeping track of the blocks which have been defined in a program are:

1135 b91' = c and c' = n if block number ≥ ba newly defined
Henceforth, each time a block with a number ≥ ba is newly defined by a non-equivalence, store current control in B91 and jump to n. The block number in ba occupies bits 1 to 11 and the remaining bits of ba are ignored. The contents of Ba are undisturbed. The instruction causing the non-equivalence is not executed. n is singly modified.

1155 ba' = smallest block label ≥ n defined. Place in bits 1 to 11 of ba the smallest block number ≥ n which is defined for this program. The remaining bits of ba are left cleared. (Only bits 1 to 11 of n are used. n is singly modified). If all the program's blocks are < n, then bit 0 of ba' is made 1 and the remaining bits are cleared

Descriptions of the extracodes for doing multiple drum transfers

1175 Read K + 1 blocks from band d, starting at sector k.

k is the octal fraction of ba; K is the octal fraction of the singly modified address in the extracode; d must already be defined by extracode 1174. The K + 1 successive sectors k, k + 1,..., k + K are read to store blocks P, P + 1,..., P + K. If K is 6 (or 7), sectors k (or k and k + 1) will be read twice. Thus if K = 6, blocks P and P + 6 will both contain sector k. (If k is 6 or 7, it is taken as 0 or 1 respectively.) All blocks involved are locked down until the entire transfer is complete.

1176 Write K + 1 blocks to drum band d starting at sector k.

The definitions of K, k, d are as in 1174. This extracode writes store blocks P, P + 1,..., P + K to drum sectors k, k + 1,..., k + K. Sectors 6 and 7 are the same as sectors 0 and 1. If K exceeds 5 some of the earlier blocks are overwritten. Thus if K = 6, sector k will finally contain block P + K rather than block P.

AMENDMENT 12

ABL fault detection

Section 1. Locating and identifying faults, permissible density of errors, entering a faulty program, program monitored during compilation.

Section 2. ILLEGAL BLOCK and SACRED VIOLATION monitoring of compiled program.

Section 3. A list of faults detected by ABL

Section 4. Optional fault printing during compilation, total number of errors permitted, entry despite errors, relocating compiled program, changing program input streams, re-entry to compiler, preset parameters set at run time, character count for C directives.

SECTION 1 Errors detected during compiling will be located and identified by one line of fault printing, often followed by a copy of the incorrect line. The fault is located by printing of the form

3,6 A10/6

which indicates that the fault occurred immediately after the third terminator in the sixth new line (ignoring blank lines) after routine parameter A10 of routine 6 was set by labelling (not by an equation). Parameters of other routines are ignored in this context.

For locating a fault A0/m is taken to be the first printed line of routine m, and hence need not refer to the same line as program parameter A0/m. The first line of a routine will normally be the routine directive. If no routine is specified at the head of the program, then a heading R0 will be assumed, and the first line actually printed will be called 1A0/0.

Example

program	location of line
(R0 assumed)	0 A0/0
+6	1 A0/0
A16 = 0	2 A0/0
(blank line)	
121 6 0 4	3 A0/0
R3	0 A0/3
A166 = 3	1 A0/3
Z	2 A0/3
1)+6	0 A1/0

The items within a line (e.g., line 3A0/0) are located as follows :

121.	6	0	4 (Newline)
0,1A0/0	1,1A0/0	2,1A0/0	3,1A0/0

A phrase to identify the type of fault is printed on the same line as the location. Faults detected include attempts to store program in an address higher than J3, impermissible character in a format that has been recognised, overflow on forcing an exponent, and unrecognised format. Each of these is followed on the next line by a copy of the incorrect line. All characters not recognized in ABL are printed as α ; all compound characters as β . The symbols £ , π and | are all printed as | . & is printed as M; the letters I, O are printed as the figures 1, 0.

Attempts to set routine parameters more than once cause printing of all locations where the settings were repeated.

An expression that cannot be fully evaluated is evaluated as far as possible, and thenprinted out as a separate fault each time it appears in the program.

Compiling will cease if more than eight errors are found in any twenty-four consecutive lines of print out (not including blank lines). Then

TOO MANY ERRORS

is printed, and the run ends.

If any errors are found in the compiled program an E or ER directive will not be obeyed and ABL will print

ERRORS DO NOT ENTER

and end the run. However, an EX directive is obeyed regardless of errors.

It may be useful to know whether a run has ended during compiling or during execution of the program. During compiling, B3 is always in the range -127 to 0. The most likely value is -127. B3 is cleared before an E directive is obeyed, as indeed are B1 to B88. (B89 always contains the final value of *. After E, B90 contains J3; after ER and EX B90 is clear.)

SECTION 2

When a faulty item is encountered, zero will usually be stored in its place in the compiled program. However if the item is an expression that cannot be evaluated because of an unset parameter the expression is evaluated from the left up to the unset parameter and the resulting value is stored.

When the expression following an enter directive is faulted for any reason, the expression is given the value 10^6. If the enter directive is then obeyed it will cause a Supervisor monitor on ILLEGAL BLOCK, with the contents of the current instruction UNALLOCATED.

In other cases where an expression cannot be evaluated, it is given the value J36 or J36Y1. If the expression is put in the address part of an instruction, and the instruction subsequently obeyed, it is likely to cause a monitor on ILLEGAL BLOCK, since J36 is an address in the Supervisor working store.

SECTION 3 The following is a list of phrases, mostly self-explanatory, that ABL prints upon encountering a fault.

Note printed	Example
LABEL ALREADY SET AT	
EXPRESSION INDETERMINATE	
Ln NON-EXISTENT	
INSTRUCTION?	1A1,1,0,3
EXCESS COMMA	121,1,0,,3
IMPERMISSIBLE +	H ++4
NOT TERMINATED	A6H2
WRONG FORMAT	+1(?)
LABEL NOT ALLOWED	2) A1 = 1
Z IN RO	
SHIFT > 23 PLACES	2D29
AO ON FIXING	+1:0
IRREGULAR FUNCTION	721,1,0,0
OCTAL NUMBER CONTAINS 8/9	HJ9
* OUT OF RANGE	* = J3 + 1
PARAMETER OR ROUTINE NUMBER TOO BIG	A6000 = 1

Before listing any errors ABL prints ABL MONITORING.

SECTION 4 There are several special parameters, numbered from P100 upward, which are connected with ABL fault printing. They are set in the same way as ordinary preset parameters (see section 5.2), and are listed below.

P100 - Optional Printing

At the start of compiling ABL sets P100 to zero. Non-zero settings cause ABL to print various kinds of information during compiling. P100 is treated by ABL as made up of eight octal digits abcdefgh. Each octal digit controls the printing of one kind of information, as listed below. If the least significant bit of an octal digit is one, the information controlled by this digit will be printed - on a new line if the middle bit of the digit is one, on the same line if the middle bit is zero. If the most significant bit of b (see below) is 1, then the start and title of library routines are printed on a new line; otherwise the most significant bit is ignored. If the least significant bit is zero, the other two bits are ignored.

The kind of printing controlled by each octal digit is listed below. All printing is preceded by a space, except R.

OCTAL DIGIT	ENCOUNTERED BY COMPILER	PRINTED
a	* = expression	* = p
b	Rn	Rn * = q
,	Ln	Ln * = q, (title)
c	Z	Z * = q
d	(ignored)	
e	P111 = expression	p
f	E expression	Ep
g	ER expression	ERp
h	EX expression	EXp

p is the value of the expression, q the current setting of *.

P101 - Permitted total number of errors

At the start of compiling ABL sets P101 to 0.2. This puts no effective limit on the number of faults, since for each error met P101 is reduced by one, and compiling stops when P101 = 0. The run ends after printing

TOO MANY ERRORS

The program may set P101 = n, and counting from that point compiling will stop when n + 1 errors have been met. P101 = 0 may be useful for a developed program. For the purpose of counting errors, all parameters still not set when the E directive is met are combined into one error.

P102 - Entry despite errors

At the start of compiling, ABL sets P102 = 0.2. If errors have been found, an EX directive will still be obeyed, but E or ER will not. The program may set P102 as follows:-

P102 = 0	All E directives obeyed despite errors.
P102 = 0.3	No E directives obeyed after errors.
P102 = 0.1	E and ER obeyed, but EX not obeyed after errors.

P110 - Relocation of compiled program

At the start of compiling, ABL sets P110 = 0. This specifies the difference between * as evaluated in expressions ($*_1$) and * as indicating where items are to be stored ($*_2$). Thus

$$P110 = *_1 - *_2$$

Setting P110 = 0 permits the compiling of program into one set of store addresses for later execution in another set of store addresses. For example if a program is to be

executed while stored at addresses at J3, but compiled initially into store starting at J1, it would have at its start the directives

```
P110 = J2
*    = J3
```

The instructions

```
121  1  0  J3
1164 1  0  J1
```

would be necessary to rename block J1 as J3 before entering the program in J3.

P111 - Printing the values of expressions

When ABL meets the equation

P111 = expression

it evaluates the expression and sets P111 in the usual way. If the appropriate bit of P100 is set, the value of the expression is immediately printed out.

P115 - Change of input stream

The equation P115 = n causes ABL to start reading program from the programmers's input stream n. The rest of the line on which P115 occurs is ignored.

P120 - Re-entry to compiler

After ER or EX directives the compiler and parameters are retained in store and may be referred to directly. The compiler may be re-entered at the address specified by the value of P120 to read more program, all parameter settings being left unchanged. The value of * is stored by the compiler when the ER or EX directive is encountered and, when re-entered at P120, the compiler will read the next line after the line containing the enter directive and store the first item it finds at the address specified by the retained valve of *. Thus, the program is compiled into the same locations that it would be if the ER or EX directive were not there.

P121 - Preset parameter list

P121 is the start of the compiler's preset parameter list. Half word nP121 contains the value of Pn if Pn has been set. This list can only be referred to by a program entered by an ER or EX directive. (It should be noted that this is a list of alternate half words.)

P122 is the address of the start of the list which indicates whether a given preset parameter is set or not.

Counting 0 to 23 (most to least significant bits) bit one of half word nP122 is a one if Pn is unset, zero if set. This is also a list of alternate half words, and the other bits of these half words should not be disturbed. (See 5.3.1.)

P123 - Character count (See section 5.10, C directives)

After each C directive, P123 is set by the compiler to the number of characters stored (as a 24-bit integer) plus J4 if no carriage control character has been specified.

AMENDMENT 13

Program Errors detected by the Supervisor

FAULT	MONITOR PRINTING	DETECTED BY	MARK OR COUNT IN B91	TRAP NUMBER (IF ANY)
Local time expired	L TIME EXCEEDED	S	Bit 5	0
Division Overflow	DIV OVERFLOW	I	Bit 6	1
Exponent Overflow	EXP OVERFLOW	I	Bit 14	2
Page locked down	PAGE LOCKED DOWN	S	Bit 1	3
Number of blocks exceeded	EXCESS BLOCKS	S	2.0	4
Square root argument 0	SQRT -VE ARG	E	2.4	5
Logarithm argument	LOG -VE ARG	E	3.0	6
SPARE				7
Inverse trig. function	INVERSE TRIG OUT OF RANGE	E	4.0	8
Reading after Input Ended	INPUT ENDED	S	4.4	9
End of Magnetic tape	END TAPE	S	5.0	10
Variable length record error	V TAPE ERROR	E	5.4	11
Magnetic Tape failures	TAPE FAIL	S	6.0	12
Computer failures	COMPUTER FAIL	S	6.4	13
Unassigned Function	ILLEGAL FUNCTION	I	Bit 4	
Sacred Violation Instruction	SV INSTRUCTION	I	Bit 8	
Sacred Violation Operand	SV OPERAND	I	Bit 10	
Illegal block number	ILLEGAL BLOCK	S	9.6	
Band not reserved	BAND NOT RESERVED	S	10.2	
Computing time expired	C TIME EXCEEDED	S	Bit 2	
Execution time expired	E TIME EXCEEDED	S	Bit 3	
Input not defined	INPUT NOT DEFINED	S	11.6	
Output not defined	OUTPUT NOT DEFINED	S	12.2	
Output exceeded	OUTPUT EXCEEDED	S	12.6	
Tape not defined	TAPE NOT DEFINED	S	13.2	
Illegal search	ILLEGAL SEARCH	S	13.6	
No selected tape	NO TAPE SELECTED	S	14.2	

FAULT	MONITOR PRINTING	DETECTED BY	MARK OR COUNT IN B91
No mode defined or attempt to write when not permitted	WRONG TAPE MODE		
Number of decks exceeded	EXCESS DECKS	S	15.2
No trap set	TRAP UNSET	S	15.6
Number of branches exceeded	EXCESS BRANCHES	S	16.2

(I = Interrupt; E = Extracode; S = S.E.R.)

The last three of the above faults are not defined in the programming manual. They are :-

EXCESS DECKS : The program has called for a tape to be mounted by a 'mount' extracode when the program was using all the tape decks it had reserved in the job description.

TRAP UNSET : The extracode 1134 to enter a trap has been encountered before the extracode 1132 that sets the trap.

EXCESS BRANCHES : A new branch of the program has been asked for at a time when the program already had as many branches active as it had asked for with the 1103 extracode.

AMENDMENT 14.

The 1117 extracode ('end program')

The 1117 extracode prints on output 0 the number of instruction counter interrupts (units of 2048 instructions) that have occurred during the complete run, and also during compiling, together with the amount of store reserved by the job description, the amount actually in use at the end of the run and the number of blocks accepted on each input stream. The waiting time in seconds, and the number of blocks transferred on each magnetic tape are also printed. The number of blocks accumulated for each output stream is printed at the end of each stream. The operators are instructed to disengage and rewind all tapes used. The program is then cleared from the store.

For example a program ended by the 1117 extracode might print the following information on output 0.

```
INSTRUCTION 61          57
STORE 10          /8
1 DECKS 5          TAPE BLOCKS 1      HALT TIME
INPUT 0            1          BLOCKS
OUTPUT )           ANY 1        BLOCKS

ENEND OUTPUT       1 BLOCKS
```

INSTRUCTION 61 57

This means that 57 interrupts were used to compile the program and 61 to compile and execute it. Multiplication orders are counted as 2 and division orders as 4.

STORE 10 /8

The program asked for 10 blocks in the job description and 8 were in use when the 1117 extracode was encountered, including one for each of the input and output streams in the job description.

1 DECKS 5 TAPE BLOCKS 1 HALT TIME

One tape deck was reserved, 5 tape transfers have occured and the program was suspended by the Supervisor for a total of 1 second awaiting the completion of tape transfers.

INPUT 0 1 BLOCKS

One block was needed in the input well to hold the uncompiled program (and any data that followed the program's enter directive).

OUTPUT 0 ANY 1 BLOCKS

One block of lineprinter output was held in the output well. There is a similar line for each output stream.

END OUTPUT 1 BLOCKS

This is the end of output 0.

AMENDMENT 15

The following documents should be added to the list of references given in Appendix A (page 95)

CS 376 Introduction to Fortran for the Atlas computer

CS 377 Algol 60 report

CS 378 Reference manual for Atlas Algol (provisional)

CS 379 A primer of Algol 60 for Atlas

CS 383 Atlas user's description of the L.P. input scheme

CS 384 Summarized programming information

CS 390 Primer of Fortran programming

CS 395 In/Output for Algol on Atlas

CS 399 Peripherals for Atlas

R 70 Processing commercial data in Atlas

R 76 The compiler compiler

Atlas Extracodes (annotated programs for extracode 1200 and above)