U.1. READIT.

Purpose This is a program for producing a listing of a a loader or core-dear or similar. It will not list tapes designed for input in 920A format.

Format The program is provided in AbRLB format for loading via initial instructions. It occupies 232 consecutive previously empty locations in module 0, and of course, 8135-8179.

Method of Use

After loading, a '?' is printed, after which the tape to be read should be loaded into the reader and an option number in the range 1-7 typed. Each bit of the number specifies an option:

Bit 1	2^0	Print addresses in decimal
Bit 2	2^1	Print addresses in octal (in parentheses)
Bit 3	2^2	Print locations (in parentheses).

The first character on the tape is taken to be the marker character. The program stops if the marker deviates from this or if it ~~reaches~~ reaches the end of the program.

U.2. PUNCHII.

Purpose This is a program for producing tapes suitable for loading via initial instructions. The absolute listing may use BFN format, integers (signed) or octal groups preceded by '&'. See Appendix 1 for restrictions on initial instruction format.

Format The program is supplied in AldRLB for loading with initial instructions. It occupies consecutive previously empty locations in module 0.

Method of Use

After loading the program, the tape will shoot through the reader. A prepunched tape should be loaded. The binary tape will be punched in real-time. When sufficient tape has been read, the program stops and clears itself from store.

Character set: 0-9, +, -, &, /, (H) are significant
(CR), (ERASE), (RUNOUT) are ignored

All other characters are treated as separators.

N.B. Tapes must be in ISO code. There is no ← or $$ facility.

There should be no spaces between / and initial

The tape will normally begin with:

```
0 8179
8 8182
/15 n.
```

If the 3rd word read is negative, the program will use it as a counter to decide when to stop reading/punching.

INSTRUCTIONS FOR D.3

1.READING STORES

```
R <ADDRESS>          READ STORE AS BFN.
A <ADDRESS>          READ STORE AS BF&N
I <ADDRESS>          READ STORE AS AN INTEGER.
O <ADDRESS>          READ STORE AS OCTAL GROUP.
\ <ADDRESS>          READ STORE AS /ABC GROUP.
```

2.SETTING STORES

```
<WORD> <ADDRESS>     SETS ADDRESS TO WORD GIVEN.
```

3.LISTING STORES

```
L<START ADDR.><END ADDR.>  LISTS AREA SPECIFIED.
```

4.SEARCHING FOR A WORD

```
S<WORD><START ADDR><END ADDR>
                     SEARCHES AREA  GIVEN FORWORD GIVEN.
```

5.CONTROL FUNCTIONS

```
*X  (0≤X<8)          SETS OPTION, VIZ:
                       BIT 1  LIST ADDRESSES
                       BIT 2  LIST  ADDR. IN OCTAL.
                       BIT 3  LIST ZERO STORES.
M <WORD>             MASKS WORD FOR SEARCH, E.G. M /15 0
                     MATCHES INSTRUCTION ONLY.
T <ADDRESS>          TRIGGER TO ADDRESS GIVEN.
X                    = T 8131
```

ACCEPTABLE FORMATS

```
<WORD>   BFN      E.G. /15 8191  OR 15 6144
         BF&N     E.G. /15 &17777  OR 15 &14000
         BF;+N    E.G.  5 ;+0  5 ;-3
         INTEGER  E.G. +131071  OR -5
         OCTAL    E.G.  &777777  OR  &374000
         \ABC     E.G.  \ABC   OR  \[]↑

<ADDRESS> N      E.G.  8177  OR 8225
          &N     E.G.  &17761
          ;+N    E.G.  ;+1  OR  ;-5
          CR     JUST HIT RETURN TO GET NEXT ADDRESS.
```

U.3 STORE.

Purpose This program is for looking at, listing and modifying any number of store locations. It may be used as a means of loading an absolute binary listing of a program into store.

Distribution It is distributed in AldRLB for loading via initial instructions, and occupies ~~780~~ 690 consecutive previously empty store in module 0.

Method of Use

The following facilities are available after a '?':

1. Reading Individual stores

   Typing R, I, O or A followed by an address will cause the contents of that address to be listed. The format will be as follows:

   R: EFN
   I: Integer.
   O: Octal Groups.
   A: BF&N (Instruction in Decimal, Address in Octal)

2. Setting Stores

   A statement beginning with 0-9, &, +, -, /, will be deemed to be setting a location to the value given. The format is:

   /15 8191 9
   word address.

   or: -1 8
   word address.

3. Listing Areas of Store

Type 'L' followed by two addresses. The format may be varied by typing * followed by a number in the range 0 to 7. The three binary bits of the number will be interpreted as follows:

Bit 1: Print Addresses
Bit 2: Addresses to be given in octal.
Bit 3: Print zero stores.

Note that the stores occupied by STORE are listed as zero. The option also affects the output format from the 'SEARCH' instruction.

4. Searching for a given instruction

Type 'S' followed by an instruction and the first and last addresses between which you wish the search to be carried out. The first occurrence of the word only is printed. It may be that you only wish to search for a particular address reference, irrespective of which instruction is associated with it. This may be done by setting a mask. Type 'M' followed by a word with the bits set that you want checked. E.g. for addresses only to be checked, type 'M +8191', or for instructions only, type 'M 115 0'.
'M -1' obviously resets the mask to normal

5. Clearing Store

Type 'C' followed by first and last addresses to clear an area of store. The program is automatically disabled from clearing itself.

6. Triggering

To jump to a certain address, type 'T' followed by the address. The end of the U.3 tape contains a core-clear for U.3, so to remove U.3 from store, type 'T 8181'. Exceptionally 'T 7' will reset STORE variables to their presumed values.

7. Exiting to a new program

Typing 'X' is equivalent to typing 'T 8191'.

Implied Addresses

If no address is specified before the next Carriage Return, then STORE assumes that a reference to the next store after the current one is required.

Relative Addresses

The last address referred to is taken to be the current address, so R j-1 would result in the previous address being listed. R j+1 is equivalent to an implied address (see above). The relative address is particularly useful in the 'Search' instruction. E.g. to search module 0 for a dynamic stop, type:

'S 8 j+0 2 8179'

Significant Characters

All characters apart from +, −, 0-9, /, &, are treated as separators. Each number should be separated from the next by one or more separators. Letters A-Z, *, are only significant after '?'.

Instruction Summary

R (address)	Read address. (BFN)
I (address)	Read address (Integer)
O (address)	Read address (Octal)
A (address)	Read address (BFN)
(Number or Instruction)(Address)	Set address to word given.
L (first address)(last address)	List from first to last.
* X (0 ≤ X ≤ 7)	Set option bits.
S (word)(first address)(last address)	Search for word between first & last.
M (word)	Set Mask to given word.
C (first address)(last address)	Clear from first to last.
T (address)	Trigger to address given.
X	Exit (= 8 9161)

Please note that this is an (Ald)RLB tape. After loading by init. instructions, leave the tape in the reader. Then, when you have finished type 'X', it will then wipe itself out of store.

U.4 MAP

Purpose This program lists the areas of store that are non-zero, in a variety of formats, on the teletype.

Format It is distributed in AldRLB format and occupies 206 previously empty stores in module 0.

Method & Use After loading, a '?' is printed, requesting an option in the range 1-7. The bits of the digit have the following meanings:

Bit 1 (2^0) Output addresses in decimal.
Bit 2 (2^1) " " - octal
Bit 3 (2^2) Output both store modules.

Note that as the AldRLB loader occupies 7 and 8135-8179, these stores will always be listed as full. MAP does not list the stores that it occupies itself. After the listing is complete, it deletes itself.

ALDCOPY. Iss.2

This is a program for copying tapes of all formats. It is distributed as a SCB tape which self-triggers at 8. It has three entry points selected by a character from the teletype

L – Load a master tape and store.
C – Check a tape with tape in store
P – Punch a new copy.

Store used. The first 143 stores of store module 0 are used for the program. The tape to be copied is stored in the remainder of module 0 and module 1. Characters are stored 2 per word of store, so capacity is limited to 32482. 8-bit characters or 270 feet of paper. *

Notes on Use

Normal method of use is to load the master tape by typing *L. Note that the tape will shoot through the reader. Retrigger at 8 to get a * back and then load the master tape into the reader again. Type *C to check the master tape. If it is happy with what it has read, it will print "OK" and a * again. If any character disagrees with that in store it will print a foot of runout, the character expected and the character actually read, and another foot of run.out. If the check is successful, type *P to punch a new copy of the tape. The new copy can be checked by typing *C as before.

* Will copy all standard programs except Algol 16K (LG).

U.6. AldRLB Generator.

(Iss.B4)
(Available)

Purpose

This program is for generating relocatable tapes which can live in any blank area of store in module 0, from conventional SCB tapes written in 920B or 903 T.23 format.

Format

The tape is supplied in absolute T.23 format, and occupies 32–1599 in module 0.

Method of Use

There are two entry points, depending on whether a self-triggering loader is required (33) or one that does a dynamic stop. (See below for issue B6)

A locater and loader are punched (see Appendix), after the necessary information has been entered, followed by the body of the program. Both AldRLB output and T.23 input are sum-checked. Finally, if the read sum-check is successful, a core clear is output which will remove the program from store. It should be noted that AldRLB always corrupts addresses 7, 8135–8179.

Data Input

The program prints "How many words long is the program?" after which the maximum length of the program should be specified. If in doubt, make it too long, rather than too short.

The program then prints "What range of addresses should be considered relative?". You should normally type the range of addresses occurring in your program, i.e. the numbers given by SIR under 'FIRST' 'LAST'. It is strongly recommended that the program is patched to 4096 in order that the minimum of confusion should arise between addresses and

constants. Even so, there may be constants in the program which will need special treatment, for example Ø 4096 would be punched as Ø Ø; but so would +4096 which is identical, in binary terms. The message "What range should be queried?" allows you to specify a first and last address for an area which needs specifying manually. If there are no problem areas, type 'Ø to Ø'.

The program then punches the locater and loader, and begins to read tape (or store.*). When it finds an address that needs querying, it prints the contents and waits for you to type:

R if it should be treated as relative, e.g. 8 Ø;
A if it should be treated as absolute, e.g. 15 2048.

If you are not sure, you can ask for it to be printed in a different format:

& for Octal
£ for £ABC
+ for an integer.

Follow this with A or R as necessary.

Finally (if the option bit is set *) a core-clear is punched. If this option is used, the body of the program should have the dynamic stop replaced with a jump to 8181. It is not possible to use this facility with a program that descends to level 4.

Iss. B6. (not available yet)

This differs slightly from Iss. B4. The first message printed is "OPTION?" after which a number should be entered, the bits of which will be interpreted as follows:

* B6 only.

Bit 1	Take source from tape (otherwise take from store)
Bit 2	Punch loader with self-trigger (otherwise dynamic stop)
Bit 3	Punch core-clear afterwards.
Bit 4	Load core-clear immediately after program, but do not obey until 8 8162 is reached in program.

Note that Iss.B6 self-triggers at 32. If option bit 1 is not set, the next message asks for the area of store to be dumped.

AldRLB.

RESTRAINTS ON THE BODY OF THE PROGRAM.

It is recommended that the program is assembled by a patch to ↑4096 as this will cause the least ambiguity for short programs. The first address of the program should always be the start address.

Check that there are no constants in the address range, and especially watch for negative integers, £ABC groups and octal groups.

U.7 UNIDUMP-B.

Purpose This tape is for dumping the entire contents of one or both modules of the store to paper tape.

Method of Distribution It is distributed in AldRLB format for loading via initial instructions. It occupies 248 previously empty locations in module 0. If this area is not available, the program will stop.

Method of Operation The program is entirely automatic in operation, except that you must type 1 or 2 on the teletype to specify if you want both store modules punched.

Output Format The output is punched in standard T.23 format, preceded by one or two core-clears. The version of T.23 which is punched has a dynamic stop (7 8171) in 8171. Note that more than 4 consecutive blank stores are punched as a skip (hence the core clears) and that U.7 doesn't punch itself.

After Use U.7 clears itself from store after use. However, addresses 7 and 8135-8179 will have been corrupted.

N.B. Unidump-B doesn't require T.23 to be in store before use, as it loads it into 8135-8179 itself. If 8135-8179 contains important (non-T.23) program matter, U.8 must be used.

C H Tozer

18/12/77.

U.7 Tape Format

1. AldRLB Locater
2. AldRLB Loader
3. U.7 in AldRLB
4. Core-clear 8192-16383
5. Core-clear 0-8179
6. T.23 (Trigger at 7)
7. Bit 8 (to terminate T.23 load)
8. AldRLB Clearer.

U.8 UNIDUMP-C.

Purpose Unidump-B cannot dump programs when 8135-8179 contain important program matter. Such programs are (e.g) the 920A SIR assemblers, and Fortran Tape 2, and 16K Basic. This tape (which has to be used in 2 stages) can dump a program located anywhere in store, and it only corrupts address 7.

Format It is supplied in AldRLB format, and occupies 225 consecutive previously blank locations in module 0.

Method of Use

Because this program has been designed to only corrupt one store (7) it is rather complicated to use.

1) Load Prog. to be dumped.
2) Load U.8
3) When it stops, type 1 or 2 for no. of store modules.
4) When it has finished copying loaders/core-clears etc, reload the program to be dumped. (without core clear!)
5) Jump to 7. A complete tape is now produced of the entire non-zero store*, excluding Unidump-C itself. (* 8 – 8179.)

Format of Output

The output consists of
a) 1 or 2 core-clears
b) T.23 (trigger at 8181)
c) Body of prog. 8-8135
d) 8135-8179 in initial instruction format.

CHT 25/6/77.

Purpose

This program dumps areas of store as specified by a data tape. It is designed to overcome some of the disadvantages of T.22/23, namely:

a) All characters except 0-9 are ignored.
b) Data tape is read in completely before punching begins
c) Core-clears can be specified.
d) The trigger address for the T.23 loader is specified on the data tape.
e) Skips (large blank areas) are not punched. *
f) Input may be from the teletype or the reader.

Format

It is provided in ABRLB and occupies 410 consecutive previously empty stores in module 0.

Method of Use

When a '?' appears on the teletype, the input device should be specified (1 or 3). The remainder of the data consists of:

a) An option: =0 — no core clears
=1 — clear 8-8179
=2 — clear 8-8179 and 8192-16[illegible]

b) The trigger address for T.23 (=8171 for dynamic stop, =33 for Edit etc)

c) Pairs of addresses (First, Last) There should not be more than 10 pairs)

d) A stopcode.

* This is a mixed blessing - see over.

The areas of store that you have specified will be punched in T.23 format. If you make a data error, you will get a message of the form En, as follows:

E1	(First > Last)
E2	(Too many groups)
E3	(Address out of range)
E4	(Device or option > 3)

Note that the addresses specified must lie in the range ($8 \leq X \leq 16383$) and must not include 8135 - 8191. *
U.9 does not dump itself.
The program may be retriggered at 7 at any time. If a restart is made before data entry is complete, the program returns to stage 1. If a restart is made after punching has begun, a new copy will be punched, without the need for repeating data entry.

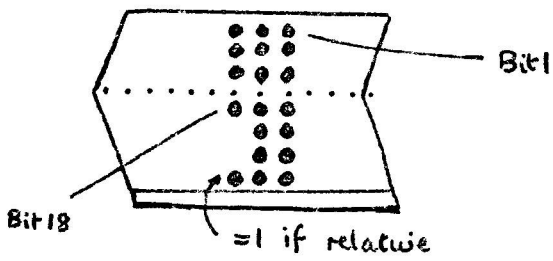
* 8135 - 8191 will be dumped by Iss 2 if FIRST < 8135 and FIRST ≥ 8191.

* As blank areas are <u>not</u> punched, there is a danger that stores that must be set to zero at the loading stage will not be so set. Hence a core-clear should be used whenever possible.

AldRLB Format.

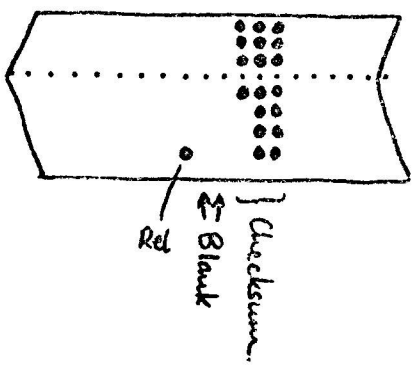
AldRLB tapes consist of four parts. First, the locater (which occupies 8159-8179) looks for n consecutive p blank stores. If it fails to find them it stops. The first address of the area found is always put in address 7, which will consequently always be corrupted. On exit to 8181, 7 contains 0 FIRST.

The loader occupies 8135-8179 (the same area as T.23) It self-triggers after loading and proceeds to read tape in a similar manner to T.23. However the format is different:



(The 8th hole is not used)

The tape is terminated with and for this reason, 0 0; is not an acceptable AldRLB word.



Finally, the final core-clear is punched. If this is used, the final ~~core-clear~~ dynamic stop should be replaced by a jump to 8181.

Location 7 is set to 0 START by the locater, and to 8 START by the loader. Hence programs may always be re-entered by a jump to 7.