903 UTILITIES.

Copies of the standard 903 utility programs are kept in the folder 903UTILITIES.

C3A – C3A.903.

C3A is a general trace utility. The original tape, in 920 telecode, was found at the National Museum of Computing, labelled "C.34 TRACE PROG 920A T2" and this was converted to 903 telecode to make the file C3A.903.

C3A is distributed as T2 source (and in an obsolete 920 T2 syntax, still recognized by 903 T2).

C3A is controlled by parameters:

Location 105 should contain the entry point of the program to trace.
Location 106 should contain a code word indicating what is to be traced (the address bits can be added to specify a combination of registers to be traced):
 Function code 0 indicates trace all instructions
 Function code 1 indicates trace only jump instructions (7, 8, 9)
 Function code 2 indicates trace only conditional jump instructions (7, 9)
 Function code 4 indicates trace only condition jumps that are taken (i.e., the condition is met)
 Function code specifies nothing is to be punched
 Address 1 indicates punch previous and current contents of the S register
 Address 2 indicates punch content of the accumulator
 Address 4, ditto Q
 Address 8, ditto B
 Address 16, ditto current instruction
 Address 2048, report overflow on 1, 2, 13 and 14 instructions and continue
 Address 4096, report overflow and halt.
Location 107 onwards consists of successive three word entries for monitoring points. When the traced S register matches a monitoring point a specified region of memory in punched out.The first word specifies the monitored address and monitoring behaviour:
 Bit 18 indicates stop on reaching the monitoring point

Function code 0 specifies binary format
Function code 1 specifies fractional format
Function code 2 decimal format
Function code 4 instruction format.
The second word specifies the first location of the region to be punched and the third word the last location of the region. There can be at most 10 monitored addresses set up.

C3A should be entered at 8 with trace interrupts enabled for priority level 1. C3A runs the program whose address start address is contained in location 106 at level 4, producing trace and monitor output as required. When a stop is reached, execution an be restarted by entering C3A at 9.

Note the C3A uses 920 telecode and produced output to the punch not the teleprinter, unless overridden by a SELECT OUT TTY command.

C4 - C4(ISS1).BIN - "903 C4, ISSUE 1 Copy 502"

C4 checks a binary tape output by T22-23 with the contents of store. C4 sumchecks the whole of the tape output by T22 including the loader at its head. The program reports only those locations that will be loaded by the tape.

C4 is distributed as a binary tape for input by the initial instructions. After loading C4, load the sumchecked binary tape to be checked and enter at 8.

Messages are output on the teletype as follows:

L	Error in copy of loader at the head of the tape
S	Tape fails to sumcheck.

If a discrepancy between the contents of store and the tape is found, C4 displays the following information as three integers:

(i) The address being checked
(ii) The contents of the store location
(iii) The contents of the tape.

Note C4 cannot check tapes in ACD binary format, use the ACD VERIFY utility instead.

COPYTAPE - COPYTAPE(ISS4).BIN
- "903 COPY TAPE, ISS 4 Copy 502"

COPYTAPE is a utility to produce and check copies of a master paper tape.

Entry at 21 reads the master tape into store. Reading stops when the tape reader unloads or the user stops the computer. The reading may be slowed using TRACE INTERRUPT on level 1. If the tape is too long, the program loops emitting a two-tone hoot (which the Simulator cannot detect, so suggest you use a large STEP limit, e.g., 250000, to detect this).

Entry at 22 reads back in the master tape and checks it against the store version. Again a two-tone hoot indicates an error.

Entry at 23 punches an initial six feet of runout and start of copy. This should then be loaded in the reader and the program re-entered at 24 to commence producing and checking copies of the checked master.

DEMO7.DAT illustrates COPYTAPE.

DEMO8.DAT demonstrates the use of C3A a tracing program. The basic concept is that hardware trace interrupts are used to run C43 at priority level 1 monitoring the execution of a program at level 4. C3A is distributed as a T2 program so the first step is to assemble the test program, followed by loading T2 to read in C3A. Patches are used to set up parameters controlling C3A, then trace interrupts on level 1 are enabled and execution started by entering C3A at 8. The demonstration consists of several runs of a simple test program to illustrate all of C3A's features.

EDIT - EDIT(ISS2).BIN - "903 Edit Iss 2 Copy 259"

EDIT is a utility for editing a source program in SIR, ALGOL, FORTRAN, T2 etc, but not for editing a binary tape. It is distributed as a source code tape, for input by SIR.

EDIT may be run in any store module and at any interrupt level.

Entry points:
0; Read correction tape
1; Read first input tape
2; Read subsequent input tape
3; Re-entry after parity errors
4; Re-entry after stop instructions
5; Validate output tape.

The input tape consists of a succession of lines, each terminated by line feed (carriage return and erase are ignored).

The correction tape consists of a series of commands, each contained on one line, with the exception of the IB command). The first two alphanumeric characters on the line define the command. If the next character is a separator (space or tab) it is ignored. Runout is ignored. The remainder of the line up to but not including the terminating line feed comprise the command string.

The input tape also contains a number of lines. Initial Runout is not copied, but the number of blanks is counted and when a similar number of blanks is found in succession this is treated as the end of the tape. (A halt code is not a terminator for this tape.)

Commands

FL Find Line: copy the input until a new line beginning with the command string is matched. Separators on the input tape are output, but may be ignored in the match except where they must be present if in the command string. The last character output is the last character of the command string.

DL Delete up to Line. As for FL, but no output.

FC Find Characters in Succession. Runout in the command string is ignored. If the characters in the command string are c1, c2, ..., Cn, copy the input until C1 has been copied, the copy until C2 has been copied, etc. The last character output is Cn. It is not possible to match line feed or halt code.

DC Delete up to Characters in Succession. As for FC, but no output.

FE Find End of Line. The command string is ignored. The input is copied until a line feed is found, the line feed is not punched but a flag is set so that if followed by a FL, FC or FE command the newline sequence is punched, otherwise it is ignored.

DE Delete up to End of Line. As for FE but no output.

IS Insert String. Output the command string. Runout may be output.

IL Insert on New Line. As for IS but a newline sequence is output first.

IB Insert Block. Output the command string including line feed characters. The command string terminates when line feed is immediately followed by ^. The terminating ^ and any characters up to the next line feed are not output. ^ may be output as part of the command string provided it does not immediately follow line feed. Blank runout may be output in the block.

RE Remainder of tape. Copy the remainder of the input tape until a number of blanks equal to the initial runout is read, then punch this amount of runout.

SH Stop on Halt. The input is copied until a halt code is reached and the program halts. The halt code is not copied.

IH Insert Halt. A halt code is punched. The command string is ignored.

ST Stop. The program halts. The command string is ignored.

CO Comment. The command is ignored.

Commands are obeyed in sequence until the halt code at the end of the correction tape is reached. The program then stops.

Errors

If an error is detected the message ERROR n X is printed on the teletype where n is the error code and X the last character input.

n=1: unknown command.
n=2: missing command string for FL, DL, FC and DC commands.
n=3: failure of sum check when verifying edited tape.
n=4: halt code encountered within command string.

Parity errors give a dynamic stop but no output. Re-entry at 3 will correct the parity and continue.

DEMO1.DAT illustrates EDIT. The example is taken from the Elliott manual pages for EDIT. Note that the IB command inserts a line feed after the inserted block - this is probably not what Elliott's technical author intended, and instead IL should have been used to insert each line of the block separately.

MTINIT - MTINIT(ISS3).BIN - "903 MTINIT ISS3 Copy 502"

MTINIT initialises magnetic tapes for use by the ALGOL, FOTRAN and SIR magnetic tape i/o libraries. On entry at 32 the operator is prompted to enter a tape handler number, e.g., 0. The tape on that handler is then inspected. If the tape already has a header block, this is printed out, showing the tape's name. The operator is then prompted to enter a new name and a new header block is written.

The program may then be re-entered at 9 to initialize another tape.

DEMO9 illustrates MTINIT being used to initialize a blank tape with the name "SCRATCH", then re-initialized with the name "MYTAPE". Note that for the second step it displays the header block written by the first stage, to warn the operator that the tape is labelled.

MTINIT can also be driven by paper tape. This is shown in the second half of the demo.

Note on MTINIT input:

Blank, erase, carriage return: always ignored
Halt: causes wait; enter at 9 to continue
Lower case letters: treated as upper case
Space, Linefeed, tab: treated as separators up until first significant character, then treated as any other. Tab is treated as space.
Colon, comma: terminators. (Ignored until first significant character).
): program terminate if first significant character of an item.
/: Cancel - ignore anything already input and prompt for item again.

Tape names can contain any SIR internal code symbol.

Error messages are of the form *MTI n m where n is the magnetic tape handler status word and m is an error number.

Status bits

1	handler busy
2	handler in manual mode (offline)
3	missed data transfer
4	Parity error (can't arise in simulator)
5	Short block
6	Long block
7	Write permit ring fitted
8	At load point (beginning of tape)
9	At end of tape
10	Zero character (false end of block)
11	Instruction rejected as "do nothing" (generally indicates operator or hardware error).

Error numbers:

1	Error in opening file
2	Error in reading block
3	Error in writing block
4	Error in closing block
5	Incorrect handler number input (must be in range 0-3)
6	handler not available (e.g., in manual)
7	Incorrect file name
8	Incorrect numbers input
9	No write permit ring fitted

Generally entry at 9 after an error allows the program to continue, either repeating the failing action, or adopting some suitable default or corrective behaviour.

MTLIST – MTLIST(ISS3).900 - "903 MT LIST ISS 3 Copy 502"

MTLIST is a utility to examine the contents of a magnetic tape. The number of the handler to be used and the tape name are input then the program lists the header block. The program then reads instructions to list either the label blocks on the tape and/or specific groups of blocks.

DEMO10 illustrates use of MTLIST. The program is first assembled and then a new tape is opened by entry at 0; (32), with the handler number and tape file name input via the teleprinter. The tape file name is verified, then the L, instruction is given to list all label blocks. The tape then rewinds and the B,1,5: instruction is used to list all blocks, followed by a) to terminate the program. (The syntax follows the rules given for MTINIT). The program can also be driven from paper tape by entry at 1; analogously to MTINIT, but this is not included in the demonstration.

The input syntax and error reporting is the same as for MTINIT.

Note that header and label blocks are printed in a partially decoded form, whereas data blocks are listed as blocks of octal.

N.B., MTLIST fails if presented with a tape containing blocks longer than 2047 words. MSDUMP can produce such tapes if large regions of store are dumped as a single block).

MSDUMP - MSDUMP(ISS3).900 - "903 MSDUMP ISS3 Copy 502"

MSDUMP is used to dump specified areas of store onto a magnetic tape file suitable for re-input to the same area of store by the retrieval program MTCALL.

On entry at 0; the programs reads a handler number and tape file name that is verified against the tape mounted on the specified handler. This is followed by the input of a program name, followed by pairs of addresses specifying the regions of store to be written to tape. After dumping each program the file may be closed, or further dumps specified. Input syntax and error handling are as for MTINIT. MSDUMP and MSCALL are illustrated in DEMO11. MSDUMP is used to write a copy of the SIR assembler to tape and then later call it back to assemble further programs.

MTCALL - MTCALL(ISS3).900 - "903 MT CALL ISS 3 Copy 502"

MTCALL is used to call down named programs from a magnetic tape file. The program must have been written to the tape file by the MSDUMP utility.

The handler, file name and program name are input. The program then finds the program required on the file and loads it into store after checking that it will not overwrite itself (thus MTCALL can be used to organize program overlays on tape). TCALL syntax and error reporting is as for MTINIT, with two additional error numbers:

10	program required overlaps MTCALL
11	program name not on specified tape file

No continuation is possible after either error.

MONITOR – MONITOR(ISS1).900 - "903 MONITOR, ISS 1 Copy 502"

MONITOR is a program testing aid which gives the facility of a temporary hold-up at any specified point under test while the contents of the accumulator, the Q register and ay specified core locations are output for printing in any combination of integer, octal and instruction modes.

MONITOR is distributed as a tape for input by SIR.

On entry at 0; a parameter tape is read in by MONITOR to specify the points at which a hold-up is required and the locations and modes to be output. The program under test is entered and output occurs whenever a monitor point is reached.

There are two entry points:
 0; to read a parameter tape and set monitor points
 1; to cancel all monitors points and restore the test program to its original form.

A MONITOR parameter tape consists of a series of monitor point specifications of the form: <location>, <mode>. If the mode is not specified, then the most recent mode input applies, or if no mode has been set, then it default to 0. The mode must be a five bit number, interpreted as follows:

Bit	Set	Clear
1	layout using spaces	Layout using tabs
2	Output A & Q	No output of A & Q
4	Decimal output	No decimal output
8	Octal output	No octal output
16	Instruction output	No instruction output

Each monitor point can be followed by a list of individual store locations and/or regions of store to be printed.

A single location is specified as c1+|-c2+|-…+|-cm, where +|- indicates either plus or minus. If m=1 the address is directly defined. If m >= 2 the address is indirectly defined as follows: treat c1 as an address; take the contents of this address and add or subtract c2 and treat as an address; and so on up to cn. Note that the evaluation is done every time the monitor point is reached and the result may therefore vary from one time to the next.

A region is specified as a first and last address in the form: c1+|-c2+|-…+|-cm/d1+|-d2+|-…+|-dn. Either address may be directly or indirectly defined as for a single location. The order of the addresses is immaterial.

Each monitor point and each location or region to print must be on a line by itself. Blank lines are ignored. Input is terminated by halt code. There can be at most 10 monitor points.

MONITOR will detect and report the following errors in the parameter tape:

0	odd parity character
1	Unused characters
2	Any c or d >= 65536 in an address
3	Integer missing
4	Impermissible sequence of separators + - , /
5	Mode >= 32
6	Number of monitor points >= 11
7	Location >= 8191 monitor point.

Once the parameter tape is read in, MONITOR insert jumps in the program under test to enter MONITOR and print the specified data whenever the monitor point is reached. MONITOR then stops and the program can be run in the usual way and MONITOR will direct its output to the teletype.

MONITOR can be run in any store module, but addresses will be taken as relative to the module containing MONITOR. MONITOR can be run at any interrupt level, but is not re-entrant so multiple copies of MONITOR must be loaded if monitoring of programs at different levels is required, noting that output from the copies may be overlapped if MONITOR at one level is interrupted and then another MONITOR runs at the higher level.

DEMO9 in the 903SIR directory shows the use of MONITOR to track the execution of a program building an in memory database.

QBINOUT- QBINOUT(ISS1).900 - "QBINOUT T17, ISS 1 Copy 502"

QBINOUT is used to output a sumchecked binary version of a stored program for re-input by initial instructions at the top end of the first store module.

QBINOUT is distributed as source code tape suitable for assembly with SIR or T.2.

The program to be output may be held in any part of the store. When the binary version is loaded it will be placed immediately below location 8180. After reading all the words of the binary program, the initial instructions transfer control to location 8177. The programmer should ensure this contains a suitable instruction such as a dynamic stop, a routine to set up entry points in the low end of store followed by a dynamic stop or a jump to the start of the program.

If the data to be converted by QBINOUT is not held immediately below the initial instructions care must be taken to ensure that instructions to be converted are stored correctly. All locations are treated as constants by QBINOUT and therefore absolute addresses must be used when preparing data for conversion.

Two parameters must be placed in locations 39; and 40; of QBINOUT.

39;	-n where n is the number of words to be output
40;	+N where N is the address where the first word to be output is held.

Entry at 0; of QBINOUT will cause the program specified by the parameters to be converted to a sumchecked binary tape suitable for loading by initial instructions.

The first three words of the tape are:

```
0    8177
8    8182
-n
```

Each word is output as four characters in which the parity bit is always zero:

1. a marker: L (without parity)
2. bits 18-15
3. bits 14-8

4. bits 7-1.

QCHECK - QCHECK(ISS3).900
- "900 Q CHECK MASIR ISS 3, 17-12-70 Copy 259"

QCHECK is an interactive debugging aid. The program is distributed as a source code tape for assembly by SIR.

When QCHECK is entered at 0; it takes input from the teletype.

Input consists of one or two addresses followed by an operation code. Each of these elements should be typed on a separate line. If only one address is input, the operation applies to a single location, if two addresses are input they define a range of contiguous locations that will be operated on.

Addresses should be typed in the form +m or +m^n where n is the store module number and m is the address within this module.

All characters except <halt> are ignored before the fist <+>. After the first address all characters are ignored between linefeed and <+> or the first permissible operation code letter. After the second address all characters are ignored between linefeed and the first permissible operation code letter.

Output

i.	Output as integers	I
ii.	Output as pseudo instructions	O
iii.	Output as octal numbers	L
iv.	Output as fractions	F
v.	Output as binary numbers	B

Input

i.	Input signed integer	SI
ii.	Input a pseudo instruction	SO
iii.	Input an octal number	SL

Zero - the specified locations are zeroed	Z

Dynamic stop - the specified locations are replaced by dynamic stops. (If an attempt is made to replace more than 20 instructions by dynamic stops QCHECK loops, warbling the loud speaker)

Dynamic Stop	H
Clear Dynamic Stops	C

Trigger – control is transferred to the first address input.

Trigger	X

Drop and continue. Control is transferred to the first address input and the user program run on level 4. The program may then be interrupted on level 1. This will cause control to be transferred back to QCHECK. At this stage, the user may utilise any of the facilities of QCHECK. To return to the user program the operation code R is typed without any addresses. This cause control to be transferred to the next instruction of the user program and the program continues to run on level 4.

Drop and continue	D
Return	R

DEMO11.DAT in the 903SIR directory shows the use of QCHECK.

QCOPY – QCOPY(ISS2).BIN - "903 QCOPY, ISS 2 Copy 502"

QCOPY is a utility to copy and check large tapes.

To copy a tape enter at 21. The leading runout is copied and the counted. Tape is then copied until a trailer of the same size is found or the process is manually stopped.

To check the copy, load it and re-enter at 22. The copied tape will be read and checked against a sumcheck and count of blanks in the main body of the tape formed in the first step.

Erase is punched continuously is an error is encountered during checking.

T.2 – T2(ISS1).BIN - "903 T2, ISS 1 P/M MASTER Copy 502"
T.2 – T2(ISS2).900 - "903 T2 (RELOCATABLE), ISS 1 Copy 502"

T.2 is a primitive machine code assembler.

T.2 is distributed in two versions: a sumchecked binary suitable for loading by initial instructions and a source version suitable for input by T.2 or SIR.

T.2 input consists of an optional header, followed by a series of block directories followed in turn by a block of words all terminated by a halt code. (For compatibility with an earlier 920 version of T.2 a closing parenthesis may also be used to terminate a T.2 program).

Words can be written as instructions, integers of fractions. Instructions are written with either absolute or block relative addresses: e.g.,

```
     15 6144   absolute address
     /8 2;     jumps to location specified by the current
               block address plus 2.
     5 3;2     jumps to the location specified by the
               address of the second block in the program
               plus 2.
```

N.B. one and only one space is permitted between the function and the address.

For compatibility with an earlier 920 version of T.2 the letter B can be used as an alternative to the solidus symbol (/) to indication B modification and comma can be used in place of semicolon in block relative addresses.

Integers and fractions are written in the usual signed form, e.g., +123 -56, +.25, -.77.

Blocks are terminated by an asterisk symbol (*) on a line by itself as in:

```
     <block n>
     *
     <block n+1>
```

The block directories are introduced by a & symbol on a line by itself, followed by a sequence of signed integers specifying the start address for each block in the program, terminated by an asterisk (*) e.g.,

```
     &
     +32
     +1024
     *
```

The first block is numbered 1.

A header consists of a line starting with an equals symbol (=) then arbitrary text terminated by a runout character.

Runout is ignored everywhere, except when terminating a title. Carriage return and erase are ignored everywhere. Line feed is ignored if no meaningful character has been read since the last line feed or entry to T.2. In a directory, an extra line feed is an error.

Temporary directory

The temporary directory is a facility for editing programs or placing parameters in a program already translated.

The temporary directory is a signed address and is followed by a group of words to be stored in consecutive locations from that specified.

The address and words may be preceded by a title, except for this they must be the only items on a single tape, which should be translated at 8.

The signed address refers to the main directory already translated: e.g., +97;1 instructs T.2 to begin storing at address 97; of the first block in the main directory. The current block number becomes 1 and the current block address equal to the first block address.

The words are punched as for normal T.2, but they are only separated from the directory by line feed, not asterisk. Addresses may be relative to any block address already translated.

£ causes the computer to enter a dynamic stop.

The directory may also be punched as an absolute address, e.g., +113. The group of words is then treated as a separate block with the directory as the current block address. The block count is not preserved if an absolute directory is used.

Entry points

8 or 7749 or 0;	Translate first tape of a program.
9 or 7906 or 1;	Translate subsequent tape.
10 or 7850 02 2;	Translate subsequent tape starting a new block.

If errors are detected the character 01111o111 is punched continuously.

The standard version of T.2 uses locations 7740-8179.

DEMO2.DAT and DEMO3.DAT illustrate T.2. DEMO2 shows a trivial program being assembled and run using the binary version of T.2. DEMO3 shows the binary version being used to assemble the source code version at location 1024 onwards, and then this version of T.2 is used to assemble to same example program as in DEMO2.

T22-23 - T22_23(ISS3).BIN - "903 T22/23, ISS 3 Copy 259"

T22-23 is a utility to produce a sum-checked binary tape of the present store contents. The locations output may be defined by a data tape; otherwise the whole of the first store module from location 10 to location 7739 inclusive is output.

T22 is distributed as a sumchecked binary tape for input by the initial instructions. After loading it outputs the T23 loader to tape.

On entry at 8179, T22 inputs a data tape that specifies the areas to be output. The tape consists of pairs of absolute addresses, each address terminated by a newline. The data tape is terminated by a halt code. Each pair of addresses defines the first and last address of one block to be output. The addresses are signed integers. Any address up to 65535 may be specified but the program will stop if the address is outside of the range of the store actually fitted. The blocks may be declared in any order. To output one location punch its address twice.

On entry at 8178, locations 10 to 7739 inclusive are output.

Contents of output tape

On input T22 outputs a copy of T23. When T22 is entered this is followed by the blocks in the order specified on the data tape. When all blocks have been output a false directory and two checksums are output and the program waits.

All directories are defined by the addition of +128 to the first character. Apart from this the address is output as an 18 bit word.

A 18 bit word is output as three consecutive characters which are not parity checked:

```
     Bits 18-15
     Bits 14- 8
     Bits  7- 1
```

Consecutive locations are output as consecutive groups of three characters.

Checksums are formed during output. The first is formed by the addition of all words ignoring overflow. The second is the sum of all directories output, again ignoring overflow. They are output as consecutive 18-bit words as the final item of the output tape. If the sumcheck test fails on input, tape is output continuously.

T22-23 is used by DEMO10 in the directory 903SIR to produce a sumchecked binary tape for SIR Issue 6 in low store.