1401 cross-development tools

1 Autocoder

Autocoder is very much like Tape Autocoder, `1401-AU-037`, described in `C24-3319`. Every statement allowed by `1401-AU-037` is accepted, but `CTL`, `DELET`, `INSER`, `PRINT` and `PUNCH` are ignored. `HEADR` is allowed and ignored in a macro file.

The listing is slightly different in that address constants, and the A- and B-address fields of instructions, are printed in decimal. There was a modification of `1401-AU-037`, by A. B. Platt from Endicott, that did this. As with `1401-AU-037`, negative addresses are replaced by their absolute values. Unlike the Platt modification, they are printed as negative in the decimal A- and B-address fields of the listing, to alert the reader to the possibility of a mistake. `1401-AU-037` and Autocoder *do not* replace negative addresses by 16000 less the amount of the negative address, e.g., -1 is *not* replaced by 15999.

The format of the symbol table is different, and it is printed after the listing instead of before.

Object files can be produced in the same format as decks or tapes produced by `1401-AU-037`, and several other formats as well.

1.1 Usage

The command line for Autocoder is

`autocoder` [*options*] [*input-file*]

where square brackets indicate optional items. The options are

`-a`	Code in columns 1–80 does not result in an error message.
`-b[ ]X[#]`	Select boot loader for "card deck" output: X = `I` means the same boot loader as provided by IBM in tape Autocoder, program `1401-AU-037`, where `#` is the core size selector: 0 ⇒ Boot, no clear, sequence numbers start at 3. 1 ⇒ 1400, 2 ⇒ 2000, 4 ⇒ 4000, 8 ⇒ 8000, `v` ⇒ 12000, `x` ⇒ 16000. The default is 16000. X = `N` means no boot loader. X = `B` means Van's favorite one-card boot without clearing core. X = `V` means Van's favorite two-card boot with clearing core.
`-d[ ]`*file-name*	Produce a "diagnostic-format object deck" in file *file-name*.

`-e[ ]X[ ][`*codes*`]`	Specify encoding. The default is `A`.
	X = A specifies Paul Pierce's primary (IBM A) encoding.
	X = H specifies Paul Pierce's alternative (IBM H) encoding.
	X = S specifies traditional SIMH encoding.
	X = I specifies Icelandic encoding.
	X = * specifies explicit encoding. The *codes* field must be 63 characters, in BCD order, 1 … GM. Blank is assumed to be first (BCD zero).
	X = ? prints the built-in encodings and stops.
	codes cannot appear if X is not *.
`-I[ ]`*path*	Specify a path to search for macros. The first path is always the current directory. Any number of `-I` options can appear. Paths are searched in the order specified. INCLD and CALL file names are the first three letters of the included or called macro. File names of inlined macros are all five letters. Macro file names are either all upper case or all lower case.
`-i`	Interleave "object deck" into listing, as would be produced by `1401-AU-037` on tape unit 3 if 1 had been punched in column 25 of the `CTL` card. Needs the `-o` and `-l` options.
`-l[ ]`*file-name*	Prepare a listing in file *file-name*.
`-L`	Store long literals once (an extension to `1401-AU-037`).
`-M`	MA is a macro in file `ma` or `MA`.
`-m[ ]`*ext*	Add *.ext* to the list of extensions to use when searching for macro files. Macros are searched in lower case first (including extensions), then upper case. Any number of `-m` options can be specified. The path loop is inside the extension loop. Macro files are always searched first without an extension.
`-o[ ]`*file-name*	Prepare an "object deck" in file *file-name*.
`-p[ ]`*number*	Set page length in lines to *number*. The default is 53.
	Vertical format control is the same as for Fortran, using the first column: 1 means skip to head of form, 0 means double space, blank means single space. A program `ansi.c` is provided to convert to using ASCII line-feed or form-feed characters.
`-r[ ]`*file-name*	Prepare relocatable output in file *file-name*. See sections 1.2.4 and 1.2.6.
`-s`	Dump the symbol and literal tables (for debugging).
`-t[ ]`*file-name*	Prepare an "object tape" in file *file-name*.
`-T[ ]`*letters*	Trace operation of components of the program (for debugging). `l` ⇒ Lexer, `p` ⇒ parser, `M` ⇒ macro processing, `P` ⇒ `PROCESS_LTORG`.
`-V`	Print version information and stop.
`-1440`	Enable 1440 op codes (see section 1.2.2).
`-X[ ]`*flags*	Set flags for extensions (sum them if necessary):
	1 ⇒ Quick `EX/XFR` (branch replaces `1040` in current "card").

2 ⇒ Quick `END /nnn080` replaces one of the "set word mark" instructions in the current "card").
4 ⇒ Queue "set word mark" instructions (saves them up to emit later, to allow filling more of columns 1–39).
8 ⇒ No reloader after `EX`/`XFR`.

`-h` or any other option not understood, results in printing this information and stopping.

If *input-file* does not appear, input is taken from standard input.

1.2 Extensions to Autocoder `1401-AU-037`

1.2.1 Diagnostic-format object file

The object file can be produced in the same format as diagnostic decks. That is, loading instructions are in columns 1–11, and the loaded field (one field per card) begins in column 12.

1.2.2 1440 instructions

For 1440 support, the following instructions change (or are provided), all using the `M` operation code:

Operation Mnemonic	First operand	Second operand	A address field	D	Description
`R`	unit number	B address	%G*unit*	`R`	Read a card
`PS`	unit number	B address	%G*unit*	`P`	Punch and stop
`P`	unit number	B address	%G*unit*	`G`	Punch and feed
`PSK`	unit number	amount	%G*unit*	`C`	Skip B punch columns
`W`	B address	none	%Y1	`W`	Print
`WS`	B address	none	%Y1	`S`	Print and suppress spacing
`WCP`	B address	none	&T0	`W`	Write to 1447 console
`RCP`	B address	none	&T0	`R`	Read from 1447 console

The second operand is required for `R`, `PS`, and `P` operation codes, and specifies the high-order character of the data field. It is required for the `PSK` operation code, and specifies the number of columns to skip (not the address of the number of columns to skip). The D modifier is provided automatically for all operation codes; it is not specified in the program. See `N24-0219` and `A24-3116`.

1.2.3 External or global labels

A label can be declared to be global by putting an asterisk in column 12.

A reference to a global label needs no special annotation.

1.2.4 Location counters

The Autocoder cross-assembler provides the capability for more than one location counter.

Ten location counters are available, numbered 0, . . . , 9. The number of them could be easily changed.

Location counter 0 is absolute. Other location counters can be absolute or relocatable. Once a location counter is established to be either absolute or relocatable, it cannot be changed.

An absolute ORG statement, or an ORG statement that specifies a global label or a label within a different location counter, is not allowed within a relocatable location counter. ORG statements with asterisk addresses or labels in the same location counter, with or without offsets, are allowed. Addresses within relocatable location counters begin at zero.

Relocatable location counters can be specified to be relocated at a multiple of 100 ± an offset, or to be relocated to an external symbol ± an offset.

To specify a location counter, write `LC` in the operation field, columns 16-20, and the location counter number (a constant, not a label) as the first operand, starting in column 21.

The location counter is absolute if there is no second operand. Location counter zero is absolute, and by default begins at 333. Other absolute location counters by default begin at zero.

If the second operand appears it must be `R` or `X`, and the location counter is relocatable.

If the second operand is `R`, an optional third operand, consisting of `X00` with an optional numeric offset preceded by a sign, specifies it is to be relocated to a multiple of 100, with the specified offset if any.

If the second operand is `X`, the third operand must consist of a label and an optional numeric offset preceded by a sign; it specifies that the location counter is to be relocated to the label, with the specified offset if any.

If no location counter is specified, or before any location counter is specified, location counter zero is in effect. If a location counter is specified, and then a different one is specified, the first location counter can be specified again. Code is assembled in the most-recently specified location counter, starting where assembly left off when the location counter was previously used.

If a location counter has been established to be relocatable and code has been generated in it, when it is re-established only the second operand can be specified, and it must be `R` or `X`.

A label on an `LC` command is ignored.

1.2.5 Examples of location counter specifications

```
....5...10...15...20...25...30...35...40...45...50...55...60...65...70..
               LC   0            The default location counter
               LC   1            A different absolute location counter
               LC   4,R          Relocatable to an arbitrary address
```

```
          LC   6,R,X00&6    Relocatable to a multiple of 100 + 6
          LC   8,X,OTHER&6  Relocatable to the external label
                            OTHER + 6
```

1.2.6 Command-line option for relocatable output

A command-line option `-r[ ]`*file-name* specifies the file upon which to write output of a relocatable assembly. The format of this file is different from the format of an "object deck" or an Autocoder-format tape:

Columns	Field Name	Contents of field
1	What	A to skip area of width FIELD(7:12) at ADR in LOC
		C to clear field of width FIELD(7:12) at ADR in LOC
		D for definition of LABEL as ADR in LOC
		E for END to LABEL + ADR in LOC or ADR in LOC
		F for a FIELD of LEN at ADR in LOC
		L for org LC to LABEL
		Q for A EQU B+LOC+IX and B is undefined, or A is global; field might be %xx
		R if ADR in LOC is a reference to LABEL with offset FIELD(7:12) in decimal and index IX
		X for EX or XFR to LABEL + ADR in LOC or ADR in LOC
		0 for org LC to X00+ADR
2:7	Label	Label
8:12	Adr	Decimal address
13:14	Loc	Location counter in decimal
15	R	R if ADR in LOC needs relocation
16	IX	Index in decimal, 0..3
17:21	Line	Sequence number in assembly listing produced using the `-l` command-line option
22:23	Len	Length of field, might be zero if WM from DA
24:77	Field	Usually an instruction or DC/DCW/DSA contents
78	WM	W If FIELD needs WM
79:80	FR(2)	Starting positions in FIELD needing relocation, 1,2,5 (two fields of one column each)
81:84	LCR(2)	Location counters to use where FR is nonzero (two fields of two columns each)
85:104	Text	Line(16:35) of source, for output to tape or "diagnostic-format deck"

If the `-r` option appears, undefined symbols are not diagnosed as errors. Address constants and the address fields of instructions that reference them appear in the listing as `###`, and their addresses are shown in the symbol table as `UNDEF`.

2 Linker

The linker is used to collect together several independently-assembled Autocoder program units, while satisfying external references and relocating relocatable location counters.

2.1 Linker control file syntax

Operation of the linker is controlled by specifications in a file.

The structure of the file is described by the following grammar. Terms in *italic* face are syntax rule names. Ellipses ... mean "zero or more repetitions of the previous item." Items described within square brackets are optional; the brackets are not part of the syntax. Each command begins on a new line. Lines cannot be continued. The maximum line length is 2047 characters, but this could easily be changed. Keywords are not case sensitive.

Lines are free format; no column requirements are imposed. Blank lines are ignored. Lines on which the first nonblank character is `*` or `!` are comments. Text on a `SEG` or `END` command after `!` is a comment.

control ::= *early* [*early* ...]
segment [*segment* ...]
END [*label*]

The *label* on the END statement is the entry to the program. It overrides an entry taken from an END statement in a program unit. If it does not appear, the entry is specified by the last END statement in a program unit that specified an entry.

early ::= ID [*deck-id*]
or SKIP *number*

The first five characters of *deck-id* are emitted into columns 76–80 of "card deck" and Autocoder-format "tape" records.

The SKIP command specifies to skip the first *number* of core locations, even if code has been allocated to them, in records after the first one on a bootable core-format "tape." This was put in place for the special purpose of skipping initialization of the phase name at locations 101–110 in the Fortran II compiler, `1401-FO-050`.

segment ::= *seg*
in [*in* ...]

seg ::= SEG *declared-name* [[+]*number*]

Start relocatable location counters in the segment named *declared-name* at the maximum of *number* (taken to be zero if *number* does not appear) and one more than the maximum absolute address in the segment, including global labels declared only by EQU.

or SEG *declared-name* −*number*

End relocatable location counters in the segment named *declared-name* at *number*.

or SEG *declared-name* **label*

Start relocatable location counters in the segment named *declared-name* at *label*, which must be declared to be a global label in some program unit.

or SEG *declared-name* *referenced-name* [*referenced-name* . . .]

Start relocatable location counters in the segment named *declared-name* at the maximum of the beginning addresses of relocatable location counters in all of the *referenced-name* segments. Each *referenced-name* shall appear as a *declared-name* in some other SEG command.

or SEG *declared-name* (*referenced-name* [*referenced-name* . . .])

Start relocatable location counters in the segment named *declared-name* at one more than the maximum of the ending addresses of absolute or relocatable location counters in all of the *referenced-name* segments. Each *referenced-name* shall appear as a *declared-name* in some other SEG command.

in ::= IN *file-name* [*file-name* . . .]

Include the contents of each *file-name*, in the order specified, in the segment. The contents of each *file-name* must have been produced using the `-r` command-line option of the Autocoder assembler.

or DATA *file-name* [*file-name* . . .]

Copy the contents of each *file-name*, in the order specified, to the output.

or SEQDATA *file-name* [*file-name* . . .]

Copy the contents of each *file-name*, in the order specified, to the output. Emit the first five characters of the *deck-id* from the ID command, if any, into columns 76–80. Emit a sequence number into columns 72–75 of output "card deck" files.

IN, DATA, and SEQDATA commands within each segment are processed in the order they appear.

2.2 Linker command line

The linker command line is

`link` [*options*] *command-file-name*

The options are

`-a`	Annotate outputs with segment and IN file names, but not DATA or SEQDATA file names. The annotation is executable, but does not load anything into core. For example, in a "card deck" file, the annotation would be in columns 1–39, and columns 40–71 would be `N000000N000000N000000N0000001040`.
`-A`	Annotate outputs with segment and IN file names, DATA file names, and SEQDATA file names. The annotation is executable, but does not load anything into core.
`-b[ ]X[#]`	Select boot loader for "card deck" output: X = `I` means the same boot loader as provided by IBM in tape Autocoder, program `1401-AU-037`, where `#` is the core size selector: 0 ⇒ Boot, no clear, sequence numbers start at 3. 1 ⇒ 1400, 2 ⇒ 2000, 4 ⇒ 4000, 8 ⇒ 8000, `v` ⇒ 12000, `x` ⇒ 16000. The default is 16000. X = `N` means no boot loader. X = `B` means Van's favorite one-card boot without clearing core. X = `V` means Van's favorite two-card boot with clearing core.
`-c[ ]`*file-name*	Specify the name of a file to receive a bootable core-image "tape." Each non-empty segment, and each EX or XFR statement within an Autocoder file, results in a separate record.
`-d[ ]`*file-name*	Specify the name of a file to receive a "diagnostic deck." This has one field per card, with load instructions in columns 1–11, and the loaded field (one field per "card") beginning in column 12. This is slightly different from the format of diagnostic "decks" provided by IBM, in that the sequence number is in columns 72–75 and the deck id is in columns 76–80, instead of having a sequence number in 79–80.
`-l`	List the control file.
`-n[ ]`*id*	Deck id for output, overridden by an ID command in the control file.
`-o[ ]`*file-name*	Specify the name of a file to receive an Autocoder-format "object deck."
`-S`	The symbol table list shows the absolute and relocatable address, location counter number, file name, and segment name, for each label. The default is to show only the label and its absolute address.

`-t[ ]`*file-name*	Specify the name of a file to receive an Autocoder-format “object tape.”
`-V`	Print version information and stop.
`-h`	or any other option not understood, results in printing this information and stopping.

2.3 Example of a linker control file

The following is part of a linker control file for a revision of program `1401-FO-050`, the Fortran II compiler.

```
! Linker control script for 1401 Fortran compiler
! 1401-FO-050, V4M0

ID V4M0
seg P0-1          ! Phases 0 and 1
  in LowCore.r Snapshot.r Monitor.r Param.r Startup.r
seg P2 *strtup    ! Phase 2,  starts at label STRTUP
  in Loader.r
seg P3 P2         ! Phase 3,  starts at beginning of P2
  in Scanner.r
seg P4 P3         ! Phase 4,  starts at beginning of P3
  in Sort1.r
seg P5 *strt04    ! Phase 5,  starts at label STRT04
  in Sort2.r
seg P6 *strt04    ! Phase 6,  starts at label STRT04
  in Sort3.r
seg P7 *strtup    ! Phase 7,  starts at label STRTUP
  in Gmark.r
seg P8 *strtup    ! Phase 8,  starts at label STRTUP
  in Squoze.r
seg P9 *strtup    ! Phase 9,  starts at label STRTUP
  in Dimen1.r
seg P10 *strt09   ! Phase 10, starts at label STRT09
  in Equiv1.r
seg P11 *strt10   ! Phase 11, starts at label STRT10
  in Equiv2.r
seg P12 *strtup   ! Phase 12, starts at label STRTUP
  in Dimen2.r
seg P13 *strtup   ! Phase 13, starts at label STRTUP
  in Varbl1.r
...
end strtup        ! All done, start at STRTUP in P0-1
```

3 Converting Autocoder tapes for use with SimH

SimH wants its tapes in one of a few formats, none of which are directly output by Autocoder, which simply outputs text.

The program `to_e11` copies a text file to one in SimH's E11 format. It depends upon a quirk of essentially all Fortran I/O libraries: E11 format has a little-endian four-byte count field before and after each record, which happens to be exactly what essentially all Fortran I/O libraries use for unformatted files on little-endian processors.

The `to_e11` command line is

`to_e11` *input-file-name output-file-name*

The program has no options.

4 Dumping SimH-format tapes

SimH can read and write four formats of tapes.

The program `tpdump` can dump tapes in two of those formats: SimH default format, or E11 format.

The command line for `tpdump` is

`tpdump` [*options*] *tape-file-name*

The options are

`-w`	Print word marks on a separate line
`-`*number*	Print *number* files, default 1
`-a`	Print all of each record, including blank lines, which are otherwise suppressed (except for the last one)
`-b`	use `b` for blank, default is blank
`-c`	use circumflex (ˆ) for blank, default is blank
`-e`	Use E11 format, i.e., don't require even-length records
`-h`	Use the Pierce `H` (Fortran) print arrangement (default Pierce `A`)
`-o`	Use the "old SimH" print arrangement (default Pierce `A`)
`-r[ ]`*number*	Print *number* characters per line, max 100, default 100