905 FORTRAN.

905 FORTRAN is an implementation of FORTRAN IV (in contrast to the earlier FORTRAN II for the 903 computer) provided for the 905 computer.

The 905 FORTRAN system consists of a compiler and a run-time library. The compiler produces relocatable binary code suitable for use with the 900 LOADER (q.v.).

905 FORTRAN will run on 903/920B/ARCH9000 machines, but it generally needs 16K or more of store to accommodate reasonably sized programs and arrays.

The following description is based on MASD Library Book No. 306, Copy No. 5, Amendment 0 "905 FORTRAN".

Extensions and Restrictions.

905 FORTRAN implements ASA standard FORTRAN as defined in the USASI document X3-9-1966 with the following extensions and restrictions:

a) The following are extensions to ASA standard FORTRAN.
   1) Optionally, free format may be used for program and data
   2) Facilities for in-line machine code
   3) Some relaxation of ASA standard rules on the mixing of arithmetic modes
   4) Option of packed integer arrays

b) The following are restrictions present in 905 FORTRAN that are not present in ASA FORTRAN.
   5) Restrictions on the sequence of statements within a subprogram. The statements which make up a program unit must appear in the following sequence
      i. SUBROUTINE or FUNCTION (except in a main program)
      ii. Specification statements
      iii. DATA statements
      iv. Statement function definitions
      v. Executable statements, FORMAT statements and in-line machine code intermixed (in any order)
      vi. END statement

6) Restrictions on the sequence of items within an EQUIVALENCE group (set of parenthesized items in an EQUIVALENCE statement).
   i. If a group of equivalence items includes an item that is also in COMMON, that item must appear first in the equivalence group.
   ii. If the same name appears in more than one group, that name must appear at the beginning of the second and any subsequent group in which it occurs
7) Restrictions on names.
   i. The name of a COMMON block must not be the name of a FUNCTION statement.
   ii. Names beginning with the letter Q must be of the form QUnnnn, unless defined in a standard software manual.
8) Printing of formatted records.
   i. Standard FORTRAN specifies that the first character of a formatted record is not printed, but used for vertical format control. 905 FORTRAN prints this character.
9) No BLOCK DATA subprograms are permitted in 905 FORTRAN.

c) Compatibility between different FORTRAN implementations.
1) The use of 'A' format often causes compatibility problems. 905 FORTRAN makes it possible to store up to 3 characters per 18 bit word for each A format specifier.
2) Problems can often be overcome by only using one character per storage unit and avoiding arithmetic operations or tests on the stored characters.

Free and fixed format.

Fixed format programs must be laid out as if punched on cards using blanks to ensure items such as labels, continuation markers and statement start in appropriate columns as required by the standard.

Fixed format is assumed initially, but if the first character or the first line of a program unit is a character [, the rest of the line is taken as comment and the remainder of the program is input in free format.

In free format

1. comment lines are indicated by the symbol [followed by two or more blanks
2. continuation lines by the symbol $ followed by two or more blanks
3. two or more spaces should follow every statement number
4. unnumbered statements may be input without any spaces before the first character of the statement.

Efficiency considerations.

Use of the packed arrays option will reduce the space required for holding such arrays, but this is in violation with the ASA standard.

Each subprogram carries some 20-30 instructions of overhead in both space and time; it is therefore inefficient to break programs up into a large number of small units. On the other hand in an 8K store there is a limit to the size of program unit that can be handled by the loader, so some compromise is needed.

Similar arguments apply to statement functions, which carry an overhead of 18-25 words.

Integer constants in real expressions are converted to reals at runtime, resulting in space and time overheads.

The assigned GOTO is faster than the computed GOTO.

Arrays made elements of named COMMON can be larger than 8192 words and straddle module boundaries, unlike those in blank COMMON or local to main and sub-programs.

Input / output.

The standard run-time package for 905 FORTRAN uses 1 as the device number for paper tape input and output, and 3 for the teleprinter.

End of record is considered to be represented by a newline sequence.

Free format.

A format list may consist of the single word FREE, which indicates free format for use on input.

```
      READ(1, 7) I, J, X, Z
   7  FORMAT (FREE)
```

will cause four numbers to be read from the data type and assigned to I, J, X and Z with the correct value type.

Mixed mode arithmetic.

905 FORTRAN permits an integer element to be combined with a real or double precision element in an add, subtract. Multiply or divide operation.

'A' format specification on input.

A A descriptor will only transfer one word, and a word may only contain at the most three characters. On input Aw with w>3, the first w-3 characters are skipped and only the last 3 stored. If w<3, the right most w-3 characters appear as blanks (0). On output, if w>3, the first w-3 characters will be output as blanks.

Hollerith constants.

The maximum number of characters that can be stored in a variable by a data statement containing a Hollerith constant depends on the type of variable:

One word integers	3 characters
Two word integers	6 characters
Real variables	6 characters
Double precision variables	12 characters
Complex variables	12 characters

Use of MASIR/SIR coding within FORTRAN text.

A code section may be a complete subprogram, or may be part of a program unit, the remainder of which is FORTRAN text. In the latter case, the machine code instructions are preceded by the directive CODE written as a statement on a line by itself and terminated by the directive FORTRAN written on a newline.

Machine code instructions are written in a form similar to 900 SIR coding. Instructions are written one per line. Instructions may be labelled, but only with identifiers of the form Qn, where n represents a number consisting of 1-5 digits. It is treated as if it were a FORTRAN statement number n.

1. Constant. Integer, fractional and octal constants may be used. These are automatically allocated local work space by the compiler.
2. Variable or Array name.
    i. If the variable is located and local, its module relative address is put in the operand field. Otherwise the name is classed as integer or real according to FORTRAN implicit type and allocated local work space.
    ii. If the name is for a local array, the address is that of the first element.
    iii. Any such address may be followed by a positive or negative increment.
    iv. If the variable or array name is in COMMON, or a formal parameter of a subprogram, the address of a local data location containing the module relative address of the item is placed in the instruction.
3. Absolute addresses (i.e., an unsigned integer).

Program units in machine code.

The 900 LOADER (q.v.) allows independently compiled MASIR blocks to be incorporated into an object program. The following rules apply to MASIR program units called from within FORTRAN texts:

1. On entry a correct module relative link will have been planted in the first word of the block and a jump made to the second word. The accumulator contains in bits 17-14 the module number of the call minus the module number of the block.

2. Following the call are the addresses of the operands, relative to the module of the call. A parameter containing a direct address has bit 18 = 1, and single level of indirection is required if bit 18 is 0.
3. The CALLG macro should be used to call any further subroutines.
4. It is not possible to access COMMON storage, except by passing addresses of items in COMMON as parameters.
5. Return should be made to the location following the last parameter of the call.

Compiler operation.

The compiler is designed to process independent program units, which it converts to relocatable form for presentation to the 900 loader.

Options.

Options are expressed as an octal number. A default option of 00 is assumed.

01	syntax check only
02	data map required
04	output data map to punch (as opposed to teleprinter)
10	pack integer arrays.

Data map.

A data map takes the form:

```
DATA MAP : xxxxxx
yyyyyy aaa b
yyyyyy aaaa t bbbbbbb
```

where

```
xxxxxx is the unit name
yyyyyy is the symbolic name of an item
aaaa   is the relative address of the item, and
t      is the address type
bbbbbb is the block name (in the case of instances of COMMON).
```

0 undefined
1 COMMON
3 local data
4 indirect address in local data

Labels are listed in the form

LABELS

nnnnnn aaaa

where nnnnnn is the statement number and aaa is the relative address of the statement.

External references are listed in the form:

EXTERNAL
ppppppp
bbbbbb ssss

where ppppppp is the name of an external procedure, bbbbbb the name of a COMMON block and ssss is the size of the block.

Operating Instructions.

Compilation.

1. Input the 905 FORTRAN compiler tape by initial instructions.
2. Enter at location 16. A ← symbol will be given as a prompt.
3. Specify the required option in the form Onn.
4. Load a FORTRAN source tape in the reader and type M (or R) to compile the program. Program units will be read until a halt code is read.
5. To process further programs repeat from step 3).

The paper tape resulting from the compilation step should be wound back to front for reading in using the 900 LOADER.

Loading.

1. Load the 900 LOADER by initial instructions.

2. Enter at location 16. A ← symbol is displayed as a prompt.
3. Type options into the loader in the form Onn.
4. Load the first relocatable binary tape into the reader and type L. The tape is read until the reader unloads.
5. Load subsequent tapes into the reader as required.
6. Re-enter at location 16 and type option O3L to load the first library tape ("905 FORTRAN LIBRARY VOL. 1"). If output to paper tape is required use 017L instead.
7. Load the second library tape ("905 FORTRAN LIBRARY VOL. 2").
8. When all programs are loaded re-enter at 16.
9. Type M. If there are unlocated labels these will be listed. If all labels are located GO is output.

   If the loading was direct into store the program may be started by typing M again.

   If the loading process produced paper tape, this should now be complete; type M, runout the tape and tear off from the punch.

   If there were unlocated labels return to either step 3) or 6) to load further tapes, or type M to run the program disregarding the missing labels (OV will be output to indicate "override").

   If there is a main program execution will be entered at this point. Otherwise the program will be entered at the location of the first unit to be loaded.

   If a sum-checked binary tape was produced, this should be loaded by initial instructions and entered at location 16, then typing M. If an octal number is typed before the M it will be held in the accumulator on entry.
10. Type C is continuation required after a halt code on inputting data, PAUSE or run-time error message.

Error Messages.

Error reports have the general form:

```
ttt nnnn llll
```

where:

ttt is the error type code (see following list)
nnnn is the last statement number encountered, and
llll is the count of non-blank lines since this statement number.

ARD array declaratory error
ASS ASSIGN statement syntax error
CHx character x expected
CMN name usage in COMMON
CN1 illegal constant formation
CN2 invalid complex constant
DA1 DATA statement syntax error
DA2 DATA statement list error
DO1 DO statement error
DO2 DO loops not nested
EQS illegal subscript within EQUIVALENCE
EX1 replacement operator usage
EX2 unmatched parenthesis within parameter expression
EX3 relational usage
EX4 illegal operator or operand/operator combination
EX5 array or function usage
EX6 Incorrect unary usage
EX7 illegal type association
EX8 Too many open parenthesis
EX9 too many closed parenthesis
FOR FORMAT statement unnumbered
FUN FUNCTION without arguments
GTO GOTO statement error
HOL Hollerith constant error
LIF logical IF error
NU1 name missing
NU2 variable missing
NU3 invalid procedure name
NU4 integer variable expected, not found
NU5 integer variable or constant expected, not found
NU6 array name error
PBN procedure name = block name
RET RETURN statement in main program
RW1 READ/WRITE format reference error
RW2 input/output list name error
RW3 input/output list syntax error
SBX subscript expression syntax error
SN1 invalid statement number definition
SN2 invalid statement number reference
SSQ statement sequence error

STF statement function name error
STM improper termination of statement
STY statement type error
TYS type statement syntax error

WARNINGS

WD1 unterminated DO loop
WD2 illegal DO termination
WEX exponent underflow in REAL constant
WF1 improper zero in FORMAT
WF2 parenthesis nested too deep
WF3 improper scale factor
WF4 scale factor not followed by conversion format
WF5 decimal point missing from conversion format
WF6 no digit following decimal point
WHC Hollerith constant count error
WN1 doubly defined statement number
WN2 statement number usage error
WN3 no path to this statement
WN4 numbered END statement
WQ1 Formal parameter of multiple COMMON in EQUIVALENCE
WQ2 COMMON base extended back by EQUIVALENCE
WQ3 special EQUIVALENCE rules contravened
WS2 logical constant spelling
WX2 procedure call parameters disagreement

MACHINE CODE ERRORS

X1 illegal first character
X2 function code exceeds 31
X3 invalid operator
X4 invalid character
X5 field too long

YLO invalid logical operator
YXM Exponentiation mode error
YY free format line conversion error
ZZ compiler workspace full

RUN TIME ERRORS

first address = call address, second address = operand address

EC1 overflow on conversion to integer
EDI integer dividend = -131072
EDZ attempted division by zero
EML logarithm of negative or zero argument

EMM exponentiation of negative real argument by negative real exponent
EOI integer overflow
EOO exponent overflow
ESN square root of negative number
EZZ exponentiation by zero

INPUT-OUTPUT ERRORS

EO1 free format specified with WRITE operation, or logical item in list for free format output
EO2 type disagreement between format and list
EO3 initial character of format is not left parenthesis
EO4 illegal character in format
EO5 format syntax error
EO6 unmatched parenthesis in format, or parenthesis level greater than 2
EO7 improper use for device type
EO8 improper QIO call
WO1 illegal character in data
WO2 integer out of range
WO3 exponent out of range

CONTROL ERROR REPORTS

QFP EIF invalid parameter code
QFP EMS invalid mode setting
QCG ERR index out of range in computed go to.