FUNCTIONAL SPECIFICATION FOR POPPY

William Clocksin
Department of Artificial Intelligence
University of Edinburgh
Edinburgh, Scotland

This is the documentation for POPPY, a POP-2 system for the PDP-11. The single-user RT-11 operating system must be used. POPPY contains some minor improvments over the standard POP-2. POPPY uses the ASCII character set and has convenient rules for how source text is to be typed. POPPY contains the features of standard POP-2 except for real numbers, sections, state-saving, and JUMPOUT. A future implementation containing real numbers, sections, and associated triples (properties) is planned. The system is capable of using arrays and records, but NEWARRAY and RECORDFNS are not yet available to the user. The system includes a garbage collector, which does not compact cells, but uses other appropriate techniques for optimising storage management. The system occupies 6.5K 16-bit words plus 4K words of reserved residence for the operating system. The RT-11 USR swap area is reserved for RT-11.

The present system appears to be free of bugs, but that's not a guarantee. It is possible to corrupt the system by supplying outrageously improper arguments to some functions which don't check for all that they should.

This document is, like Gaul, thrice divided. The first part contains the rules for preparing POPPY text. The second part is a list, in alphabetical order, of each system word (punctuation, function, operation, and variable) with a brief description of each. Most system words are defined as in standard POP-2, and all exceptions to standard POP-2 are noted. The third part describes the facilities for input, output, and error reporting. Appendix A gives a POPPY text listing of the file 'COMPAT.POP', which is compiled to provide more functions compatible with standard POP-2. Appendix B gives the POPPY text listing of 'NICERR.POP' and the file of error messages 'ERRORS.POX'.

The POPPY system is distributed to interested parties as an RT-11 save file plus some library programs written in POPPY.

1. RULES FOR TYPING POPPY TEXT

Character Set

The ASCII character set is used for both the internal and external representation of POPPY text.

Letters: Upper and lower case letters are distinguished.
Signs: $ # & * + - / : < = > ? @ \ ^ ! `
Special Signs: _ (underline) (backspace)
Separator: . ;
Comment Bracket: |
Bracket: ( ) [ ]
Decorator: %
String Quote: '
Word Quote: "
Dot: .
End-of-File Code: ASCII code 8:032 or 8:000

The remaining characters are non-printing control characters. Control characters are read by character repeaters and written by character consumers, but are treated as spaces by item repeaters. Typing certain characters may cause a mechanical action to be performed by the console (tab, newline, etc). Such characters are unchanged from the RT-11 definitions.

Numbers

Only integers in the range -32768 to +32767 are supported at present. Octal numbers are denoted by prefixing 8: to the number. Octal numbers are in the range 8:0 to 8:177777.

Words

Words read by an item repeater may consist solely of (a) letters, digits, and special signs; or (b) signs and special signs. In (a), the first character must be a letter. In any case, special signs cannot appear as the first character of a word. Words may be of any length. Words that identify standard variables are defined by the system in upper case letters. Words constructed by CONSWORD may contain any ASCII character.

Strings

Strings can be typed as any number of characters between string quotes (ASCII code 8:140). Strings may include any character. To include the string quote character in a string, two string quote characters are typed.

Comments

Comments consist of any number of characters between comment quotes (ASCII code 8:174). Comments are ignored by item repeaters. Comments are neither itemised nor interned.

File Specifications

A file specification is a string that specifies the name of a disc file. The string must consist of capital letters, and may have an optional colon or dot for specifying device names or an extension. The rules for forming file specifications are identical with the RT-11 definitions. If the device name is omitted, then device DK0: is assumed. If the extension is omitted, then extension POP is assumed.

2. STANDARD PUNCTUATION, FUNCTIONS, OPERATIONS, AND VARIABLES

Punctuation

"

Word quote.

.

The dot is used only as an alternative notation for function application.

, ;

Standard separators.

:

The colon is used for indicating a label.

( )

Parentheses are as in standard POP-2 except that the form (expression)(expression) is illegal.

(% %)

Encloses the values to be partially applied.

[ ]

Encloses a list constant.

[% %]

Encloses a list expression.

=>

The declaration arrow and print arrow are as standard POP-2 except that the print arrow lists the stack when used both at top level and when executed inside a function.

->

The assignment arrow is as standard POP-2.

Functions, Operations, and Variables

In the following list, functions and operations are given with their preferred arguments. The letter i denotes anything evaluating to any type of item; n denotes an expression evaluating to an integer; f denotes an expression evaluating to a function; l denotes an expression evaluating to a list; s denotes an expression evaluating to a string.

n + n

Integer addition, an operation of precedence 5.

n - n and -n
Integer subtraction and negation, operations of precedence 5 and 2, respectively.

n * n
Integer multiplication, an operation of precedence 4

n // n
Integer division, an operation of precedence 4, returns remainder and quotient. what is '/'

i = i and i /= i
Equality and non-equality, precedence 7

n < n n > n n =< n n >= n
Relational operators, precedence 7

n & n
Bitwise AND, an operation of precedence 4. As Standard POP-2 BOOLAND.

n ! n
Bitwise OR, an operation of precedence 5. As standard POP-2 BOOLOR.

\ n
Bitwise NOT, a unary operation of precedence 2. As standard POP-2 BOOLNOT.

i :: i
Infix CONS operator of precedence 2

l <> l
Infix list concatenation operator of precedence 2.

ADR(i)
A selector that returns an integer pointer to the item (a memory address).

i AND i
A syntax word that may be used in any expression. Parsed as an operation of precedence 8. The expression (i1 AND i2) is executed in a way equivalent to: (IF i1 THEN i2 ELSE FALSE CLOSE).

APPLIST(l,f)
Applies f to each element of l in turn. Presently available only by compiling 'COMPAT.POP'.

APPLY(f)
Applies the function f. Currenty available by compiling 'COMPAT.POP'. sp

ATOM(i)
A predicate returning 1 if i is not a pair.

CANCEL
Same as standard POP-2 except the following words must not be cancelled: NEWVARS_GAG, NEWVARS, ERRFUN, FREEWORD, GC_GAG, PROGLIST.

END

Closes a LAMBDA expression or a function definition.

ERASE

Discards the top item from the user stack. Currently available only by compiling 'COMPAT.POP'

ERRFUN(i,n)

Calls the error routine (initialised to SYSERR) with error number n and culprit i. The system always calls SETPOP after the execution of ERRFUN, even if ERRFUN is redefined by the user but called by the system.

EXIT

An abbreviation of RETURN CLOSE;. Currently available only by compiling 'COMPAT.POP'.

FALSE

A standard variable with value 0.

FNPROPS(f)

A doublet for selecting or updating the property component of a function f.

FNTOLIST(f)

Returns a dynamic list constructed from function f.

FORALL

A macro for specifying iteration. Presently available by compiling 'COMPAT.POP'.

FREEWORD

A variable whose value is the number of words of unused free space since the last garbage collection.

FROZVAL(i,f)

A doublet for selecting or updating the i-th frozen formal of the closure function f.

FUNCTION

Introduces a function definition.

GC_GAG

A variable which controls the printing of a message at the time of a garbage collection. Setting GC_GAG to true will inhibit printing of the message.

GOON

A keyword to force the compiler to exit. As in standard POP-2, except GOON is not needed at the end of the argument to POPVAL.

GOTO

Control passes to the label specified.

HD(i)

A doublet for selecting or updating the head of pair i, solidifying if i is a dynamic list.

IDENTFN()
The identity function of no argument and no result. Defined as: LAMBDA; END;. Available by compiling 'COMPAT.POP'.

IDENTPROPS(w)
A function returning an item that indicates how the word w is being used by the programmer. Same as standard POP-2. Available only by compiling 'COMPAT.POP'.

IF
Introduces an IF expression

INCHARITEM(f)
Returns an item repeater constructed from the character repeater f.

INFILE(s)
Returns a character repeater from the file specification s.

ISLIST(i)
A predicate returning 1 if and only if i is a list. ISLIST recognises the null list as a list.

ISFUNC(i)
A predicate returning 1 only if i is a function. Also recognises closure functions. Compile 'COMPAT.POP'.

ISNUMBER(i)
A predicate returning 1 only if i is a number. Compile 'COMPAT.POP'.

ISWORD(i)
A predicate returning 1 only if i is a word. Compile 'COMPAT.POP'.

ITEMREAD()
The item repeater which returns the next item from PROGLIST.

LAMBDA
Introduces the LAMBDA expression (function literal).

LENGTH(l)
Returns the number of elements of list l.

LISTREAD()
A repeater for returning the next list constant from PROGLIST.

LOOPIF
Introduces the LOOPIF expression.

MACRESULTS(l)
Specifies the result of a macro.

MACRO
Introduces a MACRO definition. May appear in a VARS declaration.

MAPLIST(l,f)
Applies f to each member of l, returning the list of results. Currently available only by compiling 'COMPAT.POP'.

MEM(n)
A doublet for accessing the PDP-11 address space. The MEM selector returns the contents of memory location n; The MEM updater updates the location. References to odd memory locations will cause a monitor trap. References to odd device locations are legal.

NEWSTRING(n)
Constructs and returns a string (8-bit components) of length n.

NEWVECTOR(n)
Constructs and returns a vector (16-bit components) of length n.

NEWVARS_GAG
A variable for controlling the printing of a message whenever an undeclared variable is recognised and automatically declared by the compiler. Setting NEWVARS_GAG to non-zero inhibits the message.

NIL
The atom NIL.

NL(n)
Prints a carriage return and n linefeed characters to the CUCHAROUT consumer.

NONMAC
A keyword that inhibits macro expansion.

NONOP
A keyword that inhibits parsing the next word as an operator.

NOT i
A unary operation of precedence 2 that reverses the truth value of its argument. Compiling 'COMPAT.POP' changes the definition of NOT to the standard POP-2 definition.

NULL(i)
A predicate that returns 1 if i is the null list or null dynamic list.

NUMBERREAD()
A repeater that returns the next (signed) number from PROGLIST.

OCTPR(n)
Prints its argument as an octal number to the CUCHAROUT device. Leading zeros are not suppressed.

OPERATION
Introduces an operation definition, or may appear in a VARS statement.

i OR i
A syntax word that may be used in any expression. Parsed as an operation of precedence 9. The expression (i1 OR i2) is executed in a way equivalent to: (IF i1 THEN TRUE ELSE i2 CLOSE).

OUTFILE(s)
Returns a character consumer as contructed from file specification s.

PARTAPPLY(f,l)
Returns the closure function constructed by partially applying f to the arguments listed in l. If f is a doublet, the resulting closure will be a doublet.

POPPIP(i1,i2)
A function that copies the contents of character repeater i1 into character consumer i2. If an argument is a file specification (a string), the the file specification will be automatically converted to the appropriate consumer or repeater. Available by compiling 'COMPAT.POP'.

POPVAL(1)
Evaluates the POP-2 text listed in l. Does not require COON as the last element of l.

PR(i)
Prints the item to the CUCHAROUT device in a standard form depending on the datatype of the item.

PRSTRING(s)
Prints the string s to the CUCHAROUT device. Omits the string quotes. Available only by compiling 'COMPAT.POP'.

PROGLIST
A variable whose value is FNTOLIST(INCHARITEM(CUCHARIN)). The compiler consumes POP-2 text from PROGLIST. PROGLIST is re-initialised to the above value at each SETPOP.

RENAME(s1,s2)
Renames the file named s1 to the new name s2.

RETURN
A syntax word that causes an exit from its enclosing function body.

REV(l)
Reverses the list l. Available only by compiling 'COMPAT.POP'.

SAMEDATA(i,i)
A predicate that returns 1 if its two arguments are of the same datatype. Can be used to test items that cannot be explicitly named. The Stack Mark, the value of TERMIN, and the item indicating no culprit for an error, all are of the same datatype, which is is known only to the system.

SETPOP()
A function of no arguments that resets the system by calling the setpop state. Typing Control-G does the same thing.

SP(n)
Prints n spaces (ASCII code 8:040) to the CUCHAROUT device.

SUBSCR(n,v)
A doublet for selecting or updating the n-th component of vector v.

SUBSTR(n,s)
A doublet for selecting or updating the n-th component of a string.

SYSERR
A variable that contains the system-defined function for reporting errors. ERRFUN is initialised to the value of SYSERR.

TERMIN
A variable whose value is the special item indicating end-of-file. This item is not the same as the end-of-file character code.

THEN
Introduces the consequent of an IF expression.

TL(1)
A doublet for selecting or updating the tail of a list. Solidifies dynamic lists if necessary.

TRUE
A variable whose value is 1.

UNDEF
A word indicating undefined. The value of UNDEF is UNDEF. All variables are initialised to UNDEF. Also initialised to UNDEF are the MEANING component of words, the FNPROPS component of functions defined by LAMBDA expressions and partial application, and probably others.

UPDATER(f)
A doublet that selects or updates the updater of a function.

VALOF(w)
a doublet that selects or updates the value component of word w.

VARS
Introduces a variable declaration.

3. FACILITIES FOR INPUT, OUTPUT, AND ERROR REPORTING

The function CHARIN consumes any character, including control characters, from the user's terminal. CHAROUT prints any character on the user's terminal.

POPMESS, as defined in standard POP-2, is not supplied. Instead, separate system functions are available for manipulating disc files. INFILE and OUTFILE construct character repeaters and consumers when given a file specification. Character consumers consume only characters, unlike standard POP-2. To close a file, consume the Control-Z character, 8:32. Characters consumers and repeaters are not designed to consume or repeat the item TERMIN. Only item repeaters return TERMIN when an end of file is encountered. Character repeaters return 8:32 on end of file.

When an error is detected during compilation or execution, the function named ERRFUN is executed. ERRFUN may be redefined and called by the user to provide customised error recovery. When the system calls ERRFUN, a SETPOP is always done after ERRFUN executes, even if ERRFUN is redefined by the user. If the user calls ERRFUN, then SETPOP is not automatically called. The variable SYSERR contains the system-defined value of ERRFUN. ERRFUN can be restored from SYSERR at any time. The value of SYSERR cannot be changed.

ERRFUN is a function of two arguments, a culprit c and an error number n. If the error has no culprit, then SAMEDATA(C,TERMIN) is true.

If SYSERR is executed, the error number and culprit (if any) will be printed on the user's terminal. If the error was detected during compilation of a function, the function name will be printed. For example,

```
Error 7
Culprit: ALPHA
Compiling function EXFN
```

Appendix B gives the text of a file that may be compiled to provide error messages. Compiling the file redefines ERRFUN to look in a file of error messages for the text associated with an error number. The files 'NICERR.POP' and 'ERRORS.POX' are supplied with the POPPY system.

```
VARS BOOLAND;

MACRO FORALL;
  VARS I J K L;
  ITEMREAD() -> I; ITEMREAD() -> J; ITEMREAD() -> K; ITEMREAD() -> L;
  MACRESULTS([%J,"-",K,"->",I,";"%]);
  MACRESULTS([%"LOOPIF","(",I,"+",K,"->",I,";",I,"=<",L,")","THEN"%])
END;

MACRO EXIT;  MACRESULTS([RETURN CLOSE ;])  END;

FUNCTION ISWORD;  .DATAWORD="WORD"  END;


FUNCTION ISFUNC;  .DATAWORD="FUNCTION" END;

FUNCTION APPLY F;  .F  END;

MACRO COMMENT;
  VARS C;
  LOOPIF (.ITEMREAD -> C; C/=";") THEN CLOSE
END;

FUNCTION ERASE I;  END;

FUNCTION ISNUMBER;  .DATAWORD="INTEGER"  END;

FUNCTION REV L => L1;
  NIL -> L1;
  LOOPIF L/=NIL THEN DEST(L) -> L;  L1.CONS -> L1  CLOSE
END;

CANCEL NOT;

FUNCTION NOT I;  IF I=0 THEN 1 ELSE 0 CLOSE  END;

NONOP & -> BOOLAND;

FUNCTION PRSTRING S;
  VARS I L;
  DATALENGTH(S) -> L;
  IF L>0 THEN FORALL I 1 1 L; CUCHAROUT(SUBSTR(I,S)) CLOSE CLOSE
END;

FUNCTION IDENTPROPS;
  VARS C G;
  MEM(.ADR)&8:177400 -> C;
  IF C<0 AND C&8:30000>0 THEN "SYNTAX"
  ELSEIF C>0 AND C&8:20000>0 THEN "MACRO"
  ELSEIF C&8:7400>0 THEN (C&8:7400)//256 ->C -> G; C
  ELSEIF C>0 AND C&8:40000>0 THEN 0
  ELSE UNDEF
  CLOSE
END;

FUNCTION APPLIST L F;
  LOOPIF NOT(L.NULL) THEN L.DEST -> L; .F; CLOSE;
```

```
END;

FUNCTION MAPLIST L1 F => L2;
 VARS L;
 IF L1.NULL THEN NIL->L2;RETURN CLOSE;
 [%UNDEF%]->L;
 L->L2;
  LOOPIF TRUE THEN
     L1.DEST -> L1;
     .F -> L.HD;
     IF NULL(L1) THEN EXIT;
     [%UNDEF%] -> L.TL;
     L.TL -> L;
  CLOSE
END;

FUNCTION POPPIP REP CON;
  VARS C;
  IF REP.DATAWORD="STRING" THEN INFILE(REP) -> REP CLOSE;
  IF CON.DATAWORD="STRING" THEN OUTFILE(CON) -> CON CLOSE;
  LOOPIF TRUE THEN
    REP() -> C;  CON(C);  IF C=8:32 THEN RETURN CLOSE
  CLOSE
END;

FUNCTION IDENTFN; END;
```

```
FUNCTION ERRFUN C N;
  VARS F I CH FOUND M J;
  PR("ERROR"); CHAROUT(8:72);
  INFILE('ERRORS.POX') -> CH;
  SP(2);
  INCHARITEM(CH) -> F;
  FALSE -> FOUND;
  LOOPIF (F() -> I; I/=TERMIN AND NOT(FOUND)) THEN
    IF I=N THEN
      TRUE -> FOUND;
      LOOPIF(CH()->J; J/=8:73) THEN CHAROUT(J) CLOSE;
    ELSE
      LOOPIF (CH()->J; J/=8:73) THEN CLOSE
    CLOSE
  CLOSE;
  IF NOT(FOUND) THEN PR(N); PR(' NO MESSAGE') CLOSE;
  SP(3);
  IF NOT(SAMEDATA(C,TERMIN)) THEN NL(1); PR("CULPRIT");SP(2); PR(C) CLOSE
END;
```

1 Improper opening bracket or missing separator;
2 Improper separator;
3 Attempt to apply a non-applicable item;
4 Improper closing bracket;
5 Improper place for a label to appear;
6 Label defined more than once;
7 Missing separator;
8 Improper use of assignment arrow;
9 Label not defined;
10 Improper word appears in a declaration;
11 Illegal name for a function;
12 Improper item found after a dot;
13 Missing word quote -- culprit was encountered instead;
14 Non-word within word quotes;
15 Improper argument for list-processing function;
20 The argument of UPDATER must be a function;
21 Improper item encountered during LISTREAD or NUMBERREAD;
22 CHAROUT requires a character argument;
23 CUCHAROUT requires a function argument;
24 Improper argument for NL or SP;
25 Division by zero is not allowed;
26 The GOON token must not appear on the CHARIN device;
27 User stack overflow;
28 Non-integer argument encountered when an integer was requested;
29 User stack underflow;
30 Undefined updater;
31 IF expressions are nested too deeply;
32 CLOSE encountered but no matching IF;
33 All permitted disc channels already in use;
34 Disc channel already open (SYSTEM ERROR);
35 File not present;
36 Disc read hardware fault;
37 File not open for read (SYSTEM ERROR);
38 Attempt to apply a repeater past the end of file;
39 Improper file specification;
40 ELSE or ELSEIF encountered outwith an IF expression;
42 The argument for FNTOLIST must be a function;
43 The argument to PARTAPPLY must be a function;
44 Improper argument for SUBSCR;
45 Improper argument for NEWVECTOR or NEWSTRING;
46 Improper argument for SUBSTR;
47 Do not attempt to cancel NONMAC;
48 Improper argument for VALOF or MEANING;
49 Out of user space. Type control-C.;
50 Overflow of compiler workspace. Type control-C.;
51 GC Stack overflow (SYSTEM ERROR). Type control-C.;
52 Improper argument for NONOP;
53 GC-LOOK Stack overflow (SYSTEM ERROR). Type control-c.;
54 Attempt to consume after file has been closed;
55 Out of disc space. FIle abandoned.;
56 DIsc hardware error for write;
57 CHannel not open for write (SYSTEM ERROR);
58 DATALENGTH BADKEY (DRYROT SYSTEM ERROR);
59 Improper argument for DATALENGTH;
60 Improper argument for FNPROPS;
61 Improper argument for FROZVAL;

*