RI

Conversion program

COLLABORATORS

	TITLE : RI	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Conversion program	October 9, 2022	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

RI

1.1 Overview of RI Eval Lib V2.5

```
                                    Overview

This library allows evaluation of complex expression strings, allowing the
programmer to provide expression handling within their programs. Thus
calculators (like this librarys' demo program) can be written with ease and
without having to know about how to actually evaluate an expression. This
library only works with integer values, you cannot use it to evaluate
floating point values.

This library supports the following operators:

    $ - The following number is hexadecimal
    % - The following number is binary
    + - Add two operands
    - - Subtract two operands
    * - Multiple two operands
    / - Divide two operands
    = - Test for equality between operands
    < - Test for less than
    > - Test for greater than
    ^ - Power operation
    & - Logically AND two operands
    | - Logically OR two operands
    {}  - The absolute decimal number inside the curly
        brackets is an address to read a value from.
        The value defaults to a longword in size but
        can be changed by using a '.W' or '.B' extension
        on the close curly bracket.

Brackets '()' can be used to group operands together inside an expression
and to change operator precedence when performing the evaluation.


Commands List:

    ConvToPostFix string$,buffer
    value=PFEvaluate string$
    PFRegisters address
```

```
    error=PFErrorType
    errstr$=PFErrorText
```

1.2 RI Eval Lib V2.5

```
Statement: ConvToPostFix
------------------------------------------------------------------------------
Modes : Amiga/Blitz
Syntax: ConvToPostFix string$,buffer

  This command is required if you wish to do an evaluation of a string. It
converts an expression expressed in infix form to postfix. For those who
do not understand this terminology, infix is an expression like '2+6',
where the operator is between the arguments - it is the normal method that
we use to express mathematical expressions. Postfix is when the operator
comes after the arguments of the expression, for example '2+6' in postfix
is '2 6 +'.

The librarys evaluation routine evaluates postfix expressions only, thus
this command is provided to allow you to convert an expression to postfix.

This string parameter is the string to convert, e.g. '2+6'. The buffer
parameter is an area of memory to receive the converted string. It should
be at least twice as large as the string being evaluated. After calling
this command the converted string can be retrieved with a command like:

    postfix$=peek$(buffer)
```

1.3 RI Eval Lib V2.5

```
Function: PFEvaluate
------------------------------------------------------------------------------
Modes : Amiga/Blitz
Syntax: value=PFEvaluate string$

  This command evaluates a postfix string and returns the result to you as a
longword value. Note that the string passed to this command must have been
generated by the ConvToPostFix command.

If this command fails, for whatever reason, the error type can be retrieved
by calling PFErrorType and PFErrorText. IF PFErrorType returns 0, then the
evaluation occured okay and the return value from this function can be
used, otherwise the return value should be discarded.

Note that the following tokens can be used in an expression to evaluate
registers:
    a0,a1,a2,a3,a4,a5,a6
    d0,d1,d2,d3,d4,d5,d6,d7
    PC

When these tokens are found in the expression, they are substituted with
the value of the token.  See PFRegisters for more information.
```

1.4 RI Eval Lib V2.5

```
Statement: PFRegisters
------------------------------------------------------------------------------
Modes : Amiga/Blitz
Syntax: PFRegisters address

  This command sets the address of the register structure that the library
should use when it encounters tokens that match the registers inside an
expression. The address passed to this routine should be a pointer to the
following newtype:

    Newtype.PFRegisterType
         d0.l[7]
       a0.l[6]
       pc.l
    End NewType

This command can be used to give the user of the routines a list of tokens
that can be assigned values by the calling program.
```

1.5 RI Eval Lib V2.5

```
Function: PFErrorType
------------------------------------------------------------------------------
Modes : Amiga/Blitz
Syntax: error=PFErrorType

  After calling PFEvaluate this routine should be called to make sure the
evaluation was performed without any errors. If an error occured, this
routine will return a non-zero value indicating the error that occured.
If a non-zero value is returned PFErrorText can be called to return a text
string describing the error (e.g. 'Divide by zero error').

Current errors are:

Error      Value   Description
============================================================
ERR_ARG              1     Need 2 arguments for expression
ERR_REG              2     Register structure not found - call PFRegister
ERR_DIVIDE0          3     Divide by 0 error
ERR_NORESULT         4     No result value - bad evaluation string
ERR_BADINDIRECT      5     No close bracket on indirect
ERR_BADINDIRECTSIZE  6     Invalid size for indirection
```

1.6 RI Eval Lib V2.5

```
Function: PFErrorText
------------------------------------------------------------------------------
Modes : Amiga/Blitz
Syntax: error=PFErrorText
```

Returns an error string for the last error (if one occurred). See PFErrorType for a list of errors.

1.7 RI Eval Lib V2.5

```
                .-------------------------------------------------------------------------
|                               RI Eval Lib V2.5                                  |
`-------------------------------------------------------------------------------'

                          ©1996 Red When Excited Ltd

          Undocumented commands added by Toby Zuijdveld 02/03/1999
                        mailto: hotcakes@abacus.net.au


                      Overview
                                                Command Index


                    ConvToPostFix

                    PFErrorText

                    PFErrorType

                    PFEvaluate

                    PFRegisters
                                                PFIndirectAddr returns last  ↩
                                                   indirect hookup


                    Examples
                                                Main Document
                                  Library Index
```

1.8 Example Programs

```
                                Example Programs

EXAMPLE 1 -  A font sensitive commodity calculator :

               Load Example 1
               Compile It!
```