

## NAME

exprin, exprout, expression, expressdat - create and convert data for use in PROM lookup tables

## SYNOPSIS

```
exprin >xxxx.exp
exprout <xxxx.exp >xxxx.dat
expression xxxx
expressdat xxxx
```

## DESCRIPTION

The two programs `exprin` and `exprout` together form a "friendly" system for generating PROM data for an expression of a single input variable.

Applications might include:

- 1) A table lookup for trigonometric values. This would be useful in games which need to transfer from polar to Cartesian coordinates.
- 2) A table lookup for logarithmic values. This would be useful for logarithmic multiplication.

To create `xxxx.ntl` which is ready to be sent to the PROM programmer use the shell script `expression` :

```
expression xxxx
```

To create `xxxx.dat` use the shell script `expressdat` :

```
expressdat xxxx
```

This is helpful when you want to concatenate several expressions into a single PROM. Use

```
cat a.dat b.dat > final.dat
```

and then edit `final.dat` to insert the appropriate `#SET_ADDRESS` command.

On line help is available for expression and expressdat.

The shell script expression consists of the three programs `exprin`, `exprout`, and `dat2ntl` piped together. The script expressdat omits the `dat2ntl` program. `Exprin` and `exprout` are described below; `dat2ntl` is described in another man page.

The first program, `exprin`, is simply an interactive guide for creating a file to be used by `exprout` shown below. It may be created and edited using an editor instead of using

exprin.

```
NUMBER_OF_STEPS = 314;
START_ADDRESS = 0;
INPUT_INITIAL_VALUE = 0;
STEP_SIZE = .01;
128 + 127 * SIN(INPUT);
```

Example file created by `exprin` and used by `exprout`

The expression must obey the following rules.

An expression can be of arbitrary size.  
It must be in infix form.

It may contain the following binary operators:

+ - \* /

and the following unitary functions:

sin, cos, tan, asin, acos, atan, sinh,  
cosh, tanh, log, exp, abs, and sqrt.

Parentheses can be used in the usual manner.

It may contain any real number and the single variable:  
INPUT.

The variable INPUT takes NUMBER\_OF\_STEPS steps starting at the initial value INPUT\_INITIAL\_VALUE. Each step increments INPUT by the value of STEP\_SIZE. NUMBER\_OF\_STEPS outputs will be created to be sent to the PROM programmer starting at the address START\_ADDRESS.

The output will be rounded to the nearest integer.

The parser is not case sensitive. All numbers are interpreted as decimal. Spaces are ignored.

The output of `exprout` is in the standard form used by the program `dat2ntl`

## FILES

## SEE ALSO

`dat2ntl(.)`

## BUGS