

NAME

`dat2ntl` - formats data prior to programming a PROM

SYNOPSIS

dat2ntl [`shift_count`] `xxxx.dat` `xxxx.ntl`

DESCRIPTION

The program *dat2ntl* has the following purposes.

- 1) To translate a data file of binary, decimal or hexadecimal numbers to a file which is in the format required by the DATA I/O PROM programmers.
- 2) To concatenate data from different files into a single file to be sent to the same PROM.
- 3) To split data words which are too wide for one PROM into multiple PROMs.
- 4) For picture and character generation.

Normally the user uses a shell script such as *assem* to invoke the *dat2ntl* program. However, in special cases the *dat2ntl* program can be invoked directly.

The output file produced by *dat2ntl* is in Intel hex format, `xxxx.ntl`, which is the format required by the PROM programmer.

The input file can consist of the following, separated by spaces, tabs, or carriage returns. Do NOT type the single quotes or the vertical bar!

Any integer no larger than 32 bits or one of the following statements.

```
# SET_ADDRESS = integer;   Interpreted by assembler as well.
# LOAD_ADDRESS = integer;  Only interpreted by dat2ntl.
# RIGHT_SHIFT = integer;
# MASK_COUNT = integer;    Number of bits to output.
# PACK_BYTES = integer;
# BASE = 'HEX' | 'DECIMAL' | 'BINARY';
```

The `#PACK_BYTES` command statement, if used, should appear before any integers. With an argument of two, integers are assumed to be 16 bits long; and the output stream consists of the low byte followed by the high byte for each integer.

The remaining statements can appear anywhere in the code, and each command will only affect the integers which follow it. A `#SET_ADDRESS` or `#LOAD_ADDRESS` statement has to appear before any data so that the starting address is well defined.

C style comments are allowed, i.e., between `/*` and `*/`.

All integers following the `BASE` statement are read in the base that has just been specified. The integer in a `#SET_ADDRESS` or `#LOAD_ADDRESS` statement, however, will be read in hex if the base is binary. Otherwise, it will be read in the base specified.

FORMATTING A FILE

To format a file of integers so that it can be sent to a PROM programmer, simply add a `#SET_ADDRESS` or `#LOAD_ADDRESS` statement at the beginning and send it through *dat2ntl*. By default, *dat2ntl* assumes the data is hex; but by inserting the `BASE` command, all following data can be interpreted as binary or decimal (except for the `ADDRESS` statements as noted above). Multiple `#SET_ADDRESS` or `#LOAD_ADDRESS` statements are permitted in a file so that data can be placed anywhere in the PROM.

The *dat2ntl* program will check to see if your address spaces overlap and issue a warning. Generally such address overlap has resulted from a mistake in the base of the `#SET_ADDRESS` directive.

CONCATENATING FILES

To concatenate data files together to be used in the same PROM (but at different addresses), simply concatenate the files before sending them through *dat2ntl*. This can be accomplished by

```
cat file1.dat file2.dat > final.dat
```

SPLITTING DATA WORDS

Often data words are too wide to fit on a single PROM so they have to be split onto two or more PROMs. To program multiple PROMs, a separate formatted file must be created for each PROM by executing *dat2ntl* once for each file.

To specify the bit length of your PROM, enter the `#MASK_COUNT` command into your data file (xxxx.dat) or, if you are using the assembler, into your assembler file (xxxx.as):

```
# MASK_COUNT = integer;
```

This command masks out everything but the lowest 'integer' bits of the data. The default mask count is eight bits. Now run *dat2ntl* once for each slice of the data which you wish to create with an argument specifying how much the data has to be shifted to the right in order to get the desired slice against the right edge of the word. The `#RIGHT_SHIFT` command could alternatively be used - this would have to be added to your data file (xxxx.dat).

For example, to convert a 16 bit data word into two eight bit words:

```
dat2ntl xxxx.dat byt0xxxx.ntl
dat2ntl 8 xxxx.dat byt1xxxx.ntl
```

PICTURE GENERATION

PROMs are often used as memory for characters or pictures to be displayed on a CRT. There is a library of characters and pictures in the directory `/mit/6.111/prom/picture_lib`. This library includes the characters A-Z and any other objects which have been entered by users. To use these characters, simply make a copy of the file, change the `#SET_ADDRESS` statements to specify the desired address, and then run the file through *dat2ntl*.

A new character is defined by setting `BASE` to `BINARY`, making a binary image of the character, and sending it through *dat2ntl*. When new characters of interest are defined, students are encouraged to submit them to be put in the library directory `/mit/6.111/prom/picture_lib`. Please include an explanation at the beginning of the file so that others can easily understand their intended use.

There is a shell script called `pview` which will change all 0's to blanks so that the characters can be better viewed.

INVERSE

The shell script *promprint* which calls `promprt` serves as the inverse of *dat2ntl* and will convert an `xxxx.ntl` file into an `xxxx.dat` file.

FILES

```
/mit/6.111/prom/picture_lib
ASCII Picture Library of dat files.
```

DAT2NTL(1)

DAT2NTL(1)

SEE ALSO

pview(1), assembler(1), exprin(1), exprou(1), expression(1), promprint(1)

BUGS