Typical state assignment use is the following:

% jedi -p fsm.kiss2 | espresso > fsm.pla
% misII
misII> readpla -s fsm.pla
misII> source script
misII> printstats -f
misII> writeblif fsm.blif
misII> quit
%

ADDITIONAL COMMENTS
When reading in the encoded PLA after encoding, "readpla -s" often works better than "readpla". This option reads in the PLA in a collapsed single output single-level PLA form.

SEE ALSO
misII(1OCTTOOLS), espresso(1OCTTOOLS)


AUTHOR
Please direct any questions or comments to:
Bill Lin
Cory Hall
Dept. of EECS
University of California
Berkeley, CA 94720
email: billlin@ic.Berkeley.EDU
-c  uses a group based embedding algorithm.

-p  outputs in the PLA format.

-g  reads a general symbolic encoding format and performs general encoding.

INPUT FILE FORMAT

Below is a general input description.

```plaintext
.i 4
.o 4
.ilb c1 c2 c3 presentState
.ob nextState o1 light1 light2
.enum States 4 2 HG HY FG FY
.enum Colors 3 2 GREEN RED BLUE
.iotype Boolean Boolean Boolean States
.ootype States Boolean Colors Colors
0 - - HG HG 0 GREEN RED
- 0 - HG HG 0 GREEN RED
1 1 - HG HY 1 GREEN RED
- - 0 HY HY 0 YELLOW RED
- - 1 HY FG 1 YELLOW RED
1 0 - FG FG 0 RED GREEN
0 - - FG FY 1 RED GREEN
- 1 - FG FY 1 RED GREEN
- - 0 FY FY 0 RED YELLOW
- - 1 FY HG 1 RED YELLOW
```

The user can specify different symbolic types using the ".enum" command. Then, the type of the input and output variables can be declared using the ".iotype" and ".otype" commands, respectively. The syntax of the ".enum" command is: 
```
enum typename numberofsymbols codelength symbol1 ... symboln
```
A built-in type called "Boolean" can be used for binary variables. Binary variables of type "Boolean" are kept as they are. The ".i" command specifies the number of input variables, and the ".o" command specifies the number of output variables. The ".ilb" command is used to give the names of the input variables. The ".ob" command is used to give the names of the output variables. These two commands are optional. Below is a finite state machine description for the state assignment problem.

```plaintext
.i 3
.o 5
.s 4
0-- HG HG 00010
-0- HG HG 00010
11- HG HY 10010
```
NAME
jedi - A symbolic encoding program

SYNOPSIS
jedi [options] [input] [ > output]

DESCRIPTION
JEDI is a general symbolic encoding program intended for multi-level logic optimization by a logic synthesis system such as MISII(10CTTOOLS). A symbolic description is different a traditional Boolean logic description in that data types are defined in terms of symbolic values rather than simply 0 and 1. A special symbolic type is the Boolean set (0, 1). More general symbolic types can be defined. For example, one may wish to define a data type with the following values: (ADD, CMP, MUL, SUB). To convert a symbolic description to a form that can be implemented in hardware, the symbolic values must be given binary codes (eg. ADD -> 00, CMP -> 01, MUL -> 10, SUB -> 11). Depending on the choice of codes, the cost of the final implementation may vary dramatically (may be a factor of 2). Multiple symbolic data types and symbolic variables can be defined. JEDI can handle both input and output variables. The former is treated as an input encoding problem while the latter is treated as an output encoding problem. Variables of the same symbolic type are given the same encodings. The input format will be explained below.

JEDI can also be used for state assignment, which is a special case of symbolic encoding where the internal states of a finite state machine are expressed symbolically (eg. st0, st1, st2, st3). Here, only the state variables are encoded; the other binary variables are maintained. The input format for the state assignment problem is basically the same as those used by the state assignment programs KISS(10CTTOOLS), MUSTANG(10CTTOOLS), and NOVA(10CTTOOLS). This is so that these programs can be used interchangeably. Currently, this is the most commonly used application of JEDI. Since this is the case, JEDI assumes a state assignment problem unless otherwise specified.

If JEDI is used with the general symbolic format, the output is a fully encoded logic description in the Berkeley PLA format suitable for further optimization by the programs ESPRESSO(10CTTOOLS), and MISII(10CTTOOLS). The unused binary codes are automatically inserted as external don't cares. Therefore, the output of JEDI should be processed through the ESPRESSO program so that these don't care conditions can be properly exploited. Other logic optimization programs can be used if they accept the same logic interchange formats as the Berkeley logic tools.

If JEDI is used for state assignment, then it will by default produce an extended BLIF format suitable for further processing by the sequential logic synthesis system called SIS. An option can be specified to produce the PLA output format.

OPTIONS
-h prints out usage information.
-e r) performs random encoding, h) performs one-hot encoding, d) performs dynamic random encoding, s) performs straightforward mapping, i) encodes using the input dominant algorithm, o) encodes using the output dominant algorithm (default), c) encodes using a combination of input and output dominant algorithms, y) encodes using a modified output dominant algorithm.

-x expands the state codes using the invalid state codes as don't cares.